Machine Learning Applications in Search Algorithms for Gravitational Waves from Compact Binary Mergers

Von der QUEST-Leibniz-Forschungsschule
der Gottfried Wilhelm Leibniz Universität Hannover

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
Dr. rer. nat.

genehmigte Dissertation von

M.Sc. Marlin Benedikt Schäfer

2023

Referent:	Prof. Bruce Allen
Korreferenten:	Prof. Bernd Brügmann
	Dr. Francesco Salemi

Promotionskommission:	Prof. Domenico Giulini
	Prof. Bruce Allen
	Prof. Bernd Brügmann

Tag der Promotion: 16.12.2022

Abstract

Gravitational waves from compact binary mergers are now routinely observed by Earth-bound detectors. These observations enable exciting new science, as they have opened a new window to the Universe. However, extracting gravitational-wave signals from the noisy detector data is a challenging problem. The most sensitive search algorithms for compact binary mergers use matched filtering, an algorithm that compares the data with a set of expected template signals. As detectors are upgraded and more sophisticated signal models become available, the number of required templates will increase, which can make some sources computationally prohibitive to search for. The computational cost is of particular concern when low-latency alerts should be issued to maximize the time for electromagnetic follow-up observations. One potential solution to reduce computational requirements that has started to be explored in the last decade is machine learning. However, different proposed deep learning searches target varying parameter spaces and use metrics that are not always comparable to existing literature. Consequently, a clear picture of the capabilities of machine learning searches has been sorely missing. In this thesis, we closely examine the sensitivity of various deep learning gravitational-wave search algorithms and introduce new methods to detect signals from binary black hole and binary neutron star mergers at previously untested statistical confidence levels. By using the sensitive distance as our core metric, we allow for a direct comparison of our algorithms to state-of-the-art search pipelines. As part of this thesis, we organized a global mock data challenge to create a benchmark for machine learning search algorithms targeting compact binaries. This way, the tools developed in this thesis are made available to the greater community by publishing them as open source software. Our studies show that, depending on the parameter space, deep learning gravitational-wave search algorithms are already competitive with current production search pipelines. We also find that strategies developed for traditional searches can be effectively adapted to their machine learning counterparts. In regions where matched filtering becomes computationally expensive, available deep learning algorithms are also limited in their capability. We find reduced sensitivity to long duration signals compared to the excellent results for short-duration binary black hole signals.

Keywords: gravitational waves, compact binary mergers, deep learning, gravitational-wave search algorithms

Contents

Abstract **iii**

Contents **viii**

List of Figures **x**

List of Tables **xi**

List of Acronyms **xiii**

1 Introduction **1**

2 Chapter Descriptions and Authorship Clarifications **5**

3 Foundations **9**

3.1 Gravitational Waves 10
3.1.1 Linearized Gravity 12
3.1.2 Post-Newtonian Formalism 19
3.1.3 Waveform Models 23
3.2 Data Analysis for Compact Binary Coalescence Signals 26
3.2.1 Noisy Detector Data 28
3.2.2 Matched Filtering 37
3.2.3 Search Algorithm and Significance of Detections 43
3.3 Deep Learning 46
3.3.1 Neural Networks 48
3.3.2 Training Neural Networks 54
3.3.3 Convolutional Neural Networks 65
3.3.4 Special Layers and Concepts 70
3.4 Machine Learning in Gravitational-Wave Astronomy 75
3.4.1 Data Quality 75
3.4.2 Gravitational-Wave Searches 76
3.4.3 Parameter Estimation 80

4 Detection of Gravitational-Wave Signals from Binary Neutron Star Mergers using Machine Learning 83
4.1 Introduction . . . 83
4.2 Methods . . . 84
4.3 Results . . . 88
4.4 Conclusions . . . 89

5 Gravitational-wave Merger Forecasting: Scenarios for the Early Detection and Localization of Compact-binary Mergers with Ground-based Observatories 93
5.1 Introduction . . . 93
5.2 Methods . . . 94
5.3 Results . . . 95
5.4 Conclusions . . . 99

6 Training Strategies for Deep Learning Gravitational-Wave Searches 101
6.0 Abstract . . . 102
6.1 Introduction . . . 102
6.2 Methods . . . 105
6.2.1 General setup . . . 105
6.2.2 Network performance . . . 109
6.2.3 Training strategies . . . 113
6.2.4 Matched-filter baseline . . . 114
6.3 Results . . . 114
6.3.1 Sensitivities . . . 114
6.3.2 Training strategies . . . 123
6.4 Conclusions . . . 126
6.5 Appendix: Efficiency curve examples . . . 128

7 From One to Many: A Deep Learning Coincident Gravitational-Wave Search 131
7.0 Abstract . . . 132
7.1 Introduction . . . 132
7.2 Coincident Search from Independent Single-Detector Networks 135
7.2.1 Architecture . . . 135
7.2.2 Data Sets and Training . . . 136
7.2.3 Single Detector Events . . . 138
7.2.4 Coincident Events . . . 138
7.2.5 Background Estimation . . . 139
7.2.6 Sensitivity . . . 140

7.2.7 Matched Filtering . . . 141
7.2.8 Evaluation and Comparison to Matched Filtering . . . 142
7.3 Two Detector Network . . . 145
7.3.1 Architecture . . . 146
7.3.2 Data Sets and Training . . . 146
7.3.3 Coincident Events . . . 148
7.3.4 Background Estimation . . . 148
7.3.5 Evaluation and Comparison to Matched Filtering . . . 149
7.4 Conclusions . . . 149

8 MLGWSC-1: The first Machine Learning Gravitational-Wave Search Mock Data Challenge 153
8.0 Abstract . . . 154
8.1 Introduction . . . 155
8.2 Methods . . . 158
8.2.1 Challenge Resources . . . 158
8.2.2 Test Data . . . 158
8.2.3 Evaluation . . . 164
8.2.4 Submission Requirements . . . 167
8.3 Submissions . . . 169
8.3.1 MFCNN . . . 170
8.3.2 PyCBC . . . 171
8.3.3 CNN-Coinc . . . 172
8.3.4 TPI FSU Jena . . . 173
8.3.5 Virgo-AUTh . . . 174
8.3.6 cWB . . . 175
8.4 Data release . . . 176
8.5 Results and discussion . . . 177
8.5.1 Sensitivities . . . 178
8.5.2 Found and missed injections . . . 185
8.5.3 Runtimes . . . 186
8.6 Conclusions . . . 188

9 Internship at Bosch 195
9.1 Basics of Object Detection . . . 196
9.2 Self-Supervised Pre-Training . . . 200
9.3 Research at Bosch . . . 203
9.3.1 Introduction . . . 204
9.3.2 Methods . . . 206
9.3.3 Experiments . . . 209
9.3.4 Conclusions . . . 213

10 Conclusions and Outlook **215**

Bibliography **219**

A Acknowledgments **269**

B List of Publications **271**

C Curriculum Vitae **273**

List of Figures

3.1 Effect of a gravitational wave on a ring of test-masses 15
3.2 Plus polarization waveform linearized gravity 18
3.3 Diagram of a laser interferometer gravitational-wave detector . 30
3.4 Gravitational wave angular power distributions 33
3.5 Gravitational wave signal parameters 34
3.6 Noise amplitude spectrum 36
3.7 Raw detector data 39
3.8 Whitened detector data 40
3.9 Bandpassed and whitened detector data 43
3.10 McCulloch-Pitts-neuron 49
3.11 Perceptron and Multi-Layer Perceptron 50
3.12 Activation functions 51
3.13 Single artificial neuron 52
3.14 Neural network 53
3.15 Gradient 58
3.16 Over- and underfitting 63
3.17 Edge detection convolution 67
3.18 Convolutional layer and its options 69
3.19 Pooling layers 71
3.20 Dropout 72
3.21 Residual block 74

4.1 Multi-rate sampling 85
4.2 Architecture 86
4.3 Sensitivity 88
4.4 True positive rate 90

5.1 "Design" era detection ranges, localizations, distances, and merger rates 96
5.2 "A+" era detection ranges, localizations, distances, and merger rates 97

5.3 "Voyager" era detection ranges, localizations, distances, and merger rates 98

6.1 Sample output 112
6.2 Network efficiency at different signal-to-noise ratios 117
6.3 Network efficiency evolution fixed low strategy with Softmax . 118
6.4 Network efficiency evolution fixed low strategy with unbounded Softmax replacement 120
6.5 Sensitive distance plot 121
6.6 Sensitive distance of different floating point precisions 122
6.7 Network efficiency with Softmax overview 125
6.8 Network efficiency with unbounded Softmax replacement overview 127
6.9 Efficiency evolution "Accuracy relative" using Softmax 129
6.10 Efficiency evolution "Accuracy relative" using unbounded Softmax replacement 129
6.11 Efficiency evolution "Fixed 30" using Softmax 130
6.12 Efficiency evolution "Fixed 30" using unbounded Softmax replacement 130

7.1 Found and missed injections 143
7.2 Sensitive distances as function of false-alarm rate of coincidence searches 145
7.3 Coincidence network architecture 147
7.4 Sensitivity of different coincidence strategies 150

8.1 Parameter space 160
8.2 Sensitive distances 179
8.3 Sensitive distance calculated on foreground 191
8.4 Found-missed chirp mass versus distance 192
8.5 Found-missed template duration versus mass ratio 193
8.6 Found-missed effective precessing spin versus inclination . . . 193

9.1 Feature pyramid network 200
9.2 Selective object Contrastive learning 203
9.3 SSDoCo framework 207

List of Tables

4.1 Parameter distributions 87

6.1 Network architecture 106
6.2 Test set injection parameters 109
6.3 Training strategy overview 115

7.1 Network architecture 136
7.2 Test set injection parameters 140

8.1 Shared injection distributions 160
8.2 Hardware specifications 167
8.3 Core software stack 168
8.4 Results 180

9.1 Transformations 208

List of Acronyms

A

AI Artificial Intelligence
ASD Amplitude Spectral Density

B

BBH Binary Black Hole
BH Black Hole
BNS Binary Neutron Star

C

CBC Compact Binary Coalescence
CE Cosmic Explorer
CNN Convolutional Neural Network
CPU Central Processing Unit
CV Computer Vision
CW Continuous gravitational Wave
cWB coherent Wave Burst (search)

E

ELU Exponential Linear Unit
EM Electromagnetic (radiation)
EMRI Extreme Mass-Ratio Inspiral
EOB Effective One Body
ESA European Space Agency
ET Einstein Telescope

F

FAP False-Alarm Probability
FAR False-Alarm Rate
FPN Feature Pyramid Network

G

GPU Graphics Processing Unit
GR General Relativity
GW Gravitational Wave
GWOSC Gravitational Wave Open Science Center

H

HM Higher Modes

I

ILSVRC Imagenet Large Scale Visual Recognition Challenge
IMR Inspiral-Merger-Ringdown
IoU Intersection over Union

L

LAMR Log Average Miss Rate
LIGO Laser Interferometer Gravitational-wave Observatory
LISA Laser Interferometer Space Antenna
LVC LIGO-Virgo Collaboration
LVK LIGO-Virgo-Kagra collaboration

M

mAP mean Average Precision

ML Machine Learning
MLGWSC-1 Machine Learning Gravitational-Wave Search mock data Challenge One
MLP Multi-Layer Perceptron
MSE Mean Squared Error

N

NASA National Aeronautics and Space Administration
NLP Natural Language Processing
NN Neural Network
NR Numerical Relativity
NS Neutron Star
NSBH Neutron Star Black Hole (binary)

O

O1 Observing run One
O2 Observing run Two
O3 Observing run Three
O3a first half of the third observing run
O3b second half of the third observing run
O4 Observing run Four

P

PM Post-Minkoskian (approximation)
PN Post-Newtonian (approximation)
PSD Power Spectral Density
pSNR peak Signal-to-Noise Ratio
PTA Pulsar Timing Array

R

ReLU Rectified Linear Unit
RNN Recurrent Neural Network
ROC Receiver Operating Characteristic
RPN Region Proposal Network

S

SGD Stochastic Gradient Descent
SMBH Supermassive Black Hole
SN Super Nova
SNR Signal-to-Noise Ratio
SoCo Selective object Contrastive learning
SSD Single Shot multibox Detector
SSDoCo Single Shot Detector object Contrastive learning
SSL Self-Supervised Learning
SVM Support Vector Machine

T

TT Transverse-Traceless gauge

U

USR Unbounded Softmax Replacement

W

WD White Dwarf

Y

YOLO You Only Look Once (object detection network)

Chapter 1

Introduction

The Universe is a vast space holding many secrets. Each new observational channel and instrument has brought with it new discoveries and peeled back the curtain on fundamental physics further. With the advent of telescopes humanity discovered more intricate details about the solar system [1]. Radio telescopes led to the discovery of the cosmic microwave background [2]. Infrared telescopes, like the recently launched James Webb Space Telescope [3], allow us to study the distant past of the Universe. X-ray observations have improved the understanding of stars and especially supernovae [4]. Gamma ray observations unveiled a new type of signal known as gamma ray bursts [5], the sources of which have been a mystery for a long time.

The newest tool in the pocket of astronomers are kilometer scale laser interferometric gravitational-wave (GW) detectors [6–9]. They allow to probe the Universe using gravity as a messenger medium, which opens up a completely new view into the cosmos. GWs were first hypothesized to exist by Einstein as a direct consequence of his theory of general relativity (GR) [10] but were believed to be too weak to ever be observed directly. Although various attempts were made [11] and indirect evidence of their existence was found in the 1980s [12], a direct detection took almost 100 years after their initial theoretical description. On September 14th, 2015 the advanced laser interferometer gravitational-wave observatory (LIGO) detectors picked up the first confirmed direct detection of a GW [13]. The signal was emitted from a merging system of two black holes (BHs).

In the 7 years following the first detection new observations have become a common occurrence and almost 100 sources have been confirmed to date [14, 15]. This plethora of signals has allowed for new tests of fundamental physics [16–21] and insights into the contents of the Universe [22, 23]. In 2017 the first binary neutron star (BNS) merger was observed [24]. It was accompanied by an electromagnetic counterpart [25–27], first picked

up as a gamma ray burst, thus confirming that at least some gamma ray bursts originate from BNS mergers. In 2019 two signals from binary systems whose component masses are consistent with neutron star–black hole binaries (NSBH)s were detected [28], completing the set of expected detectable compact binaries.

To detect these signals, many technological breakthroughs both on the instrumental and data analysis side were required [6, 29–32]. The 4 km long arms of the interferometers change their length by $\approx 10^{-18}$ m [13, 33]. Consequently, a lot of noise sources can have a stronger effect on the detector readout than GWs [6]. These noise sources need to be supressed as much as possible by the instrument design. However, some fundamental limits [34] exist and, therefore, GW signals are not clearly visible in the detector output. Extracting weak signals from the noise floor requires sophisticated data analysis methods. While instrumental development is paramount to GW detection, this thesis touches only briefly on challenges in this field and focuses mainly on analyzing the resulting data.

Today the most sensitive data analysis methods to detect GWs from compact binary mergers rely on a process known as matched filtering [30, 35]. Matched filtering is the optimal discriminator between stationary Gaussian noise that contains a known signal and pure stationary Gaussian noise [30]. It compares the known signal with the data and checks how well the two match. There also exist search methods that only loosely model the source and are, therefore, more sensitive to signals of unknown shape and origin [36–39].

Matched filtering assumes that the signal in the data is known exactly which is not true in a realistic search scenario. As a consequence, one needs to filter for a whole set of possible signals. This set is known as the template bank and it discretizes the continuous parameter space of potential sources. The computational cost of a matched filter search scales linearly with the number of templates in the template bank, as the data has to be filtered against each template individually. However, the size of the template bank scales exponentially with the number of parameters used to describe the sources. Therefore, current searches make some simplifying assumptions that reduce the dimensionality of the parameter space but also limit the modeled physics. The discrete nature of the template bank and the simplifications used to construct it can cause some signals to be missed [40–44]. Additionally, the number of templates has to be increased as detector sensitivity in the future is expected to improve at low frequencies relative to high frequencies [8, 45–47]. With this relative increase in sensitivity, the early part of the signal is weighted more strongly and discrepancies between signal and template accumulate over time, thus requiring a denser coverage of the

parameter space. To reduce the computational burden, include more physically relevant effects, or increase the parameter space which we are sensitive to, more efficient search algorithms are desirable.

Machine learning (ML) methods are one possible avenue to reduce computational costs that have started to be explored in the last decade [48]. Especially methods based on deep learning have recently gathered significant interest. Deep learning is a field of computer science where artificial neurons are connected to networks and collectively trained to solve a given task. It has been successfully applied to numerous other problems and is often computationally more efficient than alternative methods [49–51]. The hope is that ML can generalize to unseen regions of parameter space and new signals without increasing the computational burden too much. Furthermore, they may be capable of improving on matched filtering based searches when the detector noise is not stationary or Gaussian. They could also outperform matched filter searches due to the discreteness of the template banks [52–54].

Today most of these goals are still out of reach for current algorithms and works are often at a proof of principle level. Many studies target limited parameter spaces that are efficiently searched by existing search algorithms [54–57]. ML search algorithms for long duration low mass BNS or NSBH signals, where current searches are expensive to run, are still rarely explored due to problems with processing large inputs. Furthermore, studies often use Gaussian noise for training and evaluation. Consequently, it is difficult to judge the real world applicability of these algorithms. This is further complicated by the common usage of metrics which are inspired by deep learning literature rather than existing metrics developed for traditional GW searches.

The work discussed in this thesis aims at pushing deep learning searches beyond the proof of principle stage. All studies presented here estimate sensitive distances, a metric commonly used for state-of-the-art GW detection pipelines, at previously untested false-alarm rates (FARs). To enable these tests, several new methods are developed which significantly improve the performance over other machine learning search algorithms at astrophysically relevant FARs. This work allows for a clear comparison between advanced deep learning GW search algorithms and existing production search pipelines. Such comparisons had previously been difficult, due to differing metrics. From these, it is inferred that the presented solutions are already competitive for some parameter regions. For other regions, where machine learning searches are currently still outperformed by traditional algorithms, our analyses reveal the main problems that need to be addressed to elevate these novel algorithms to state-of-the-art performance.

More specifically, chapter 4 presents a new deep learning model designed for BNS detection. It performs significantly better at low FARs than previ-

ous deep learning searches but also finds a gap in performance compared to PyCBC Live [58], a state-of-the-art low-latency production search pipeline. Chapter 6 re-analyzes an existing deep learning algorithm and introduces a modification to prevent a collapse of the sensitivity at low FARs. It also investigates the impact of different training strategies on the final performance of the network. Chapter 7 adapts the coincidence analysis used in production searches [38, 59, 60] to a deep learning search algorithm trained for a single detector. This trivial extension of the deep learning search algorithm reduces the FARs at which it can be tested by several orders of magnitude at a negligible computational cost. Chapter 8 presents the results of a mock data challenge led by myself that makes the tools and experiences gained in previous studies available to the global community. It compares several contributions from international groups to production level searches and assesses the current state of the field.

Chapter 2

Chapter Descriptions and Authorship Clarifications

This thesis is based on a set of publications.

- Chapter 4 – a summary of [61] – contains a novel method to search for BNS mergers using a deep neural network.
- Chapter 5 – a summary of [62] – evaluates the capability of a state-of-the-art analysis for generating pre-merger alerts and sky-localizations for GWs from BNS mergers.
- Chapter 6 – a reprint of [63] – contains a reanalysis of an existing deep learning search for binary black hole (BBH) signals at low FARs, extends the method to work with even lower FARs, and tests the influence of different training strategies on the detection performance.
- Chapter 7 – a reprint of [64] – tests the applicability of single detector deep learning search algorithms in a coincidence multi-detector search.
- Chapter 8 – a reprint of [65] – contains the results of a global mock data challenge that assesses the current capability of ML search algorithms for BBH signals and compares them to state-of-the-art pipelines.

The works [61], [63], and [64] are published in Physical Review D, the work [62] is published in Astrophysical Journal Letters, and the work [65] is accepted by Physical Review D. Below I will give a more detailed summary of each chapter in this thesis, including those not based on the works listed above, and will clarify my contribution to each work. A full list of publications I was involved in can be found in appendix B.

Chapter 3 gives an overview of the foundations required to understand the content of this thesis. Section 3.1 discusses the theory of GW generation and the resulting waveforms. It summarizes the treatment from the approximation of linearized gravity up to complex modern waveform models. Section 3.2 treats various aspects used in current day GW detection, touching on the subjects of GW detectors and their noise characteristics, signal detection algorithms, as well as the significance of detections. Section 3.3 provides an introduction to deep learning. It discusses the mathematical foundations of neural networks and how they can be trained. Section 3.4 gives an overview of recent works relevant to the field of ML based GW data analysis. The foundations discussed in chapter 3 are also touched upon to various degrees of depth in chapters 4, 6, 7, and 8. However, chapter 3 provides more detail and background, as well as tying the works into the greater context of existing research. It was entirely written by myself with some help in proof reading.

In chapter 4 I summarize the results of [61] and present the differences to my master thesis [66], which the paper is based on. The study introduces a procedure to reduce the amount of data that needs to be processed for the detection of GW signals from BNS mergers. This procedure is then utilized to build a novel multi-detector search algorithm that uses deep neural networks. The resulting algorithm significantly improves on the performance of a previous deep learning based algorithm for the most commonly expected signal strengths and at low FARs. It also stresses the importance of testing machine learning based algorithms at FARs ≤ 1 per month for the application in production analyses. The topic was suggested by my master supervisors F. Ohme and A. H. Nitz, who proposed to use multiple sampling rates. My contribution was to create training and testing data, optimize the neural network architecture, define the process of sampling the data at multiple rates, and write the implementation. The paper was written by myself with close guidance from A. H. Nitz and F. Ohme.

Chapter 5 summarizes the work done in [62]. The paper investigates the prospects of detecting GW signals from compact binary coalescences before the merger for current and planned detector networks. It also discusses how well these pre-merger detections can be localized on the sky, to allow for prompt electromagnetic follow-up observations. It finds that future GW observatories will be capable of providing several minutes of early warning for sources localized to $< 100\text{deg}^2$ in the sky. The research was led by A. H. Nitz, who also suggested the topic. I provided the code to perform a high-level analysis of the detectability of signals based on a network signal-to-noise ratio threshold. The code calculates the network signal-to-noise ratio for post-Newtonian waveforms truncated at different high-frequency

cutoffs. This also allowed for the creation of a suitable template bank for a full analysis on mock data. The draft of the paper was written by A. H. Nitz and T. Dal Canton. I contributed to the body of the publication by proof reading the draft.

Chapter 6 is a reprint of [63]. It reproduces the results of [56] and extends them to lower FARs. From this baseline, the study investigates the influence on the sensitivity of different training strategies for deep learning GW search algorithms. It is found that the particular strategy has little influence, as long as sufficiently difficult examples are used. Furthermore, a new output statistic for the networks is presented which avoids numerical instabilities that have made it previously impossible to test the network at production level FARs. The idea for the study was developed by myself in collaboration with O. Zelenka and in close correspondence with A. H. Nitz, F. Ohme, and B. Brügmann. The paper draft was written by myself with some sections being contributed by O. Zelenka. A. H. Nitz, F. Ohme, and B. Brügmann helped with revisions of the draft and made comments to improve the evaluation.

Chapter 7 is a reprint of [64]. The study tests the applicability of deep learning GW search algorithms trained on a single detector in a coincidence analysis. It compares the results from a time-coincidence analysis of the deep learning results with a state-of-the-art matched filter based production search. We find that the application works seamlessly but falls short in sensitivity compared to matched filtering due to the inability of comparing signal parameters across multiple detectors. We also highlight the usefulness of our approach in probing deep learning searches at FARs < 100 per year. The paper, furthermore, presents a combined ranking statistic based on the single detector network output presented in [63]. The study was entirely proposed by myself, with some input on the details from A. H. Nitz. Accordingly, the draft of the paper was written by myself with minor revisions by A. H. Nitz.

Chapter 8 is a reprint of [65]. It discusses the first machine learning gravitational-wave search mock data challenge and its results. Several research groups around the world were asked to submit GW search algorithms which were subsequently evaluated on common data sets using common metrics. The goal of the challenge was the evaluation of different machine learning based algorithms to create a reference and objective comparison between the different submissions, as well as to state-of-the-art production searches. Due to the open source policy of the challenge, its resources are intended to be a base of comparison also for future algorithms. Furthermore, the most promising future research areas for machine learning search algorithms were identified from the results. The project was suggested by A. H. Nitz. The organization of the challenge, including meetings, communication, and the development of the public codebase, was primarily in my responsibility.

O. Zelenka helped with PyTorch implementations and provided tutorials. Details of the challenge, including the parameters of the data sets, were discussed with the organization team consisting of O. Zelenka, A. H. Nitz, B. Brügmann, F. Ohme, and myself as well as the scientific advisers E. Cuoco, E. A. Huerta, and C. Messenger. The initial paper draft was written by myself, with revisions being made by all authors. An exception to this are the descriptions of the submissions, which were written by the groups.

Chapter 9 is a summary of a voluntary internship I did at Bosch Hildesheim, to gain insights into the non-academic development process of ML algorithms. It is separate from the rest of the thesis, as it discussed self-supervised learning algorithms for object detection applications. For this reason, it includes a short overview of the required foundations of object detection and self-supervised learning in the context of computer vision. Afterward, my research at Bosch is summarized, which tried to test an existing self-supervised learning framework and develop a new one. Both approaches turned out to yield negative results, as we could not demonstrate an improvement over randomly initialized networks. The entire chapter was written by myself with some help in proof reading.

Chapter 10 concludes this thesis and gives an outlook into possible further research topics.

Chapter 3

Foundations

Contents

3.1 Gravitational Waves 10

3.1.1 Linearized Gravity 12

3.1.2 Post-Newtonian Formalism 19

3.1.3 Waveform Models 23

3.2 Data Analysis for Compact Binary Coalescence Signals 26

3.2.1 Noisy Detector Data 28

3.2.2 Matched Filtering 37

3.2.3 Search Algorithm and Significance of Detections . 43

3.3 Deep Learning 46

3.3.1 Neural Networks 48

3.3.2 Training Neural Networks 54

3.3.3 Convolutional Neural Networks 65

3.3.4 Special Layers and Concepts 70

3.4 Machine Learning in Gravitational-Wave Astronomy 75

3.4.1 Data Quality 75

3.4.2 Gravitational-Wave Searches 76

3.4.3 Parameter Estimation 80

3.1 Gravitational Waves

Gravitational waves (GWs) are a form of radiation, which use spacetime itself as a propagation medium. They are a direct consequence of Albert Einstein's general theory of relativity (GR) [67], as he found in 1916 [10]. Their effect can be straight forwardly derived assuming only a small deviation from the flat spacetime metric, as will be done in subsection 3.1.1. Sources of GWs are accelerated masses, or more accurately masses with a non-vanishing second time derivative of their mass-quadrupole moment.

The strength of GWs depend on the acceleration within the source and the involved masses. The larger the mass and the higher the acceleration, the bigger the amplitude of the resulting wave. For this reason, the sources that can be most easily detected are very heavy objects that experience extreme acceleration. These conditions are fulfilled, for instance, by two very compact astronomical objects, like black holes (BHs) or neutron stars (NS), that rapidly orbit each other.

Over time, a binary system of compact objects loses energy, due to the emission of gravitational radiation. This causes the two bodies to slowly come closer together until they merge. GW signals emitted by systems of this kind are known as *compact binary coalescence* (CBC) signals and they are usually classified into three categories; binary neutron stars (BNS), binary black holes (BBHs), and NS-BH-systems (NSBH). At the time of writing this thesis, all $\mathcal{O}(100)$ detected GWs are believed to belong to one of these classes [14, 15].

CBC-signals are commonly described to be composed of three stages; the *inspiral*, the *merger*, and the *ringdown*. During the initial phase, the two bodies have a large separation and relativistic effects are small. The resulting GWs can be well described by analytic approximations, an overview of which is given in subsection 3.1.1 and subsection 3.1.2. This phase is known as the inspiral, as the two objects are slowly spiraling towards each other, due to the orbital energy carried away by the emitted GWs. During the final few orbits relativistic effects have a non-negligible effect on the orbital dynamics and the approximations made during the inspiral phase are not valid anymore. Because the two bodies merge in this phase, it is called the merger. For accurate descriptions of the GW during this phase one has to resort to numerical relativity [68]. Their discussion goes beyond the scope of a brief introduction and I refer the interested reader to section 14.3 of [69] for an introduction to the topic and to [70–73] for deeper reviews. Once the binary has merged, the resulting body is a perturbed compact object. This perturbation will then be radiated off during the ringdown phase [74]

which for BHs can be described analytically and yields exponentially damped sinusoids [74]. When the remnant is some kind of NS the ringdown is affected by the mass distribution and can be a lot more complicated [75]. I will not discuss the ringdown in this work, but will point the interested reader to [76].

Other kinds of GW sources are expected to exist but have not yet been directly detected. The most promising candidates for future detection include continuous gravitational waves (CW) [77–79], supernovae (SN) [80], and extreme mass ratio inspirals (EMRI)[69, 81]. CWs are emitted by a rapidly spinning NS whose shape slightly deviates from that of a perfect sphere. If the deformation is not rotationally symmetric around the rotation axis of the star, it causes the second time derivative of the quadrupole moment to be non-zero. Due to the extreme stability of the rotational frequency of observed NS [82, 83] and the low amount of energy lost due to emitted GWs, the signal is expected to have a very small amplitude, but be extremely long lasting and almost monochromatic [84]. SN can emit GWs due to the rapid evolution of their mass distribution and the scales of energy which are released during their explosion. The exact mechanisms that lead to the emission of gravitational radiation are manifold and can only be modeled numerically [69]. EMRIs are binary systems, where one object is of stellar mass, such as a NS, stellar mass BH, or white dwarf (WD), while the other body is a supermassive black hole (SMBH). In this setup, the mass ratio between the two bodies is on the order $\geq 10^5$ and most established methods to calculate the emitted GWs break down [69]. However, we do expect to be able to detect EMRIs with future, space-born detectors [81]. While these signals are expected to exist, this thesis will exclusively treat CBC signals.

Even the strongest GWs interact only very weakly with matter and other forms of energy [68]. For this reason they travel almost unaffected through the Universe. This is a major advantage over electromagnetic (EM) radiation, which is shielded by matter, and is especially important when studying the very early stages of the Universe, where it was opaque to EM radiation [85]. Additionally, the most common sources observed today are very compact objects which usually do not emit a lot of EM radiation. Studying them through GWs allows us to detect them nonetheless and make statements about their population [22], constrain the percentage of dark matter that can be explained by BHs [23], and test GR [16, 17].

Subsection 3.1.1 closely follows sections 2.1.1 of [66]. Subsection 3.1.2 is oriented along chapter 5 of [68] and [86]. Throughout this section I will use the Einstein summation convention, denote 4-dimensional spacetime indices by Greek letters, and purely spatial indices by Latin letters. The convention $\eta^{\mu\nu} = \text{diag}(-,+,+,+)$ is used for the flat special relativistic Minkowski-metric.

3.1.1 Linearized Gravity

In GR gravity is described as a property of spacetime. Instead of being a force, gravity is the effect of following shortest paths in a curved spacetime. The curvature is governed by the Einstein equation

$$\mathcal{G}_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}, \tag{3.1}$$

where $\mathcal{G}_{\mu\nu}$ is the Einstein tensor, G is the gravitational constant, c is the speed of light in vacuum, and $T_{\mu\nu}$ is the energy-momentum tensor. $\mathcal{G}_{\mu\nu}$ is constructed entirely from constants and a combination of the spacetime metric $g_{\mu\nu}$ and its first and second derivatives. Overall, (3.1) specifies a set of coupled, non-linear, second order, partial differential equations, where $T_{\mu\nu}$ acts as the source of the curvature. To find trajectories of test-particles in this theory, one needs to specify the matter-, energy-, and stress-contents of the universe in the energy-momentum tensor $T_{\mu\nu}$. Afterward, the set of differential equations have to be solved to find the metric $g_{\mu\nu}$. From the metric one can then find (local) trajectories by solving the geodesic equation

$$\ddot{x}^\rho + \Gamma^\rho_{\mu\nu} \dot{x}^\mu \dot{x}^\nu = 0, \tag{3.2}$$

where $\Gamma^\rho_{\mu\nu}$ are Christoffel-symbols and $\dot{x}^\mu$ is the derivative of x^μ by the proper time.

As stated earlier, GWs are waves that use spacetime as their medium. In mathematical terms, we are looking for a wave-like solution $g_{\mu\nu}$ to the Einstein equation (3.1). Their existence in GR should not be surprising, as one of the core concepts of the theory is that nothing travels faster than light in vacuum. As a result the curvature changes induced by moving masses will also have to obey this speed limit. Analogously, in electrodynamics electric charges act as a source of the electric field, which travels at the speed of light. When they are accelerated, one finds wave-like solutions to the field equations. Therefore, a waveform solution to the Einstein equation seems plausible from a physical standpoint.

To find these solutions we assume small and slowly varying deviations from a flat spacetime. In mathematical terms we are only considering metrics

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu}, \tag{3.3}$$

with $||h_{\mu\nu}|| \ll 1$ and $||\partial_{\sigma_1} \dots \partial_{\sigma_n} h_{\mu\nu}|| \in \mathcal{O}(h_{\mu\nu}) =: \mathcal{O}(h)$. Inserting (3.3) into (3.1) and keeping only terms linear in $h_{\mu\nu}$ and its derivatives, i.e. discarding any terms of order $\mathcal{O}(h^2)$ and above, yields the linearized Einstein equation

$$\mathcal{G}_{\mu\nu} = \frac{1}{2}(\partial_{\alpha\mu} h^\alpha_\nu + \partial^\alpha_\nu h_{\mu\alpha} - \partial_{\mu\nu} h - \Box h_{\mu\nu} - \eta_{\mu\nu} \partial^{\sigma\alpha} h_{\sigma\alpha} + \eta_{\mu\nu} \Box h) = \frac{8\pi G}{c^4} T_{\mu\nu}, \tag{3.4}$$

where $h \coloneqq \eta^{\mu\nu} h_{\mu\nu}$ and $\Box \coloneqq \eta^{\mu\nu}\partial_{\mu\nu}$.

The linearized Einstein equation is invariant under coordinate transformations $x'_\mu = x_\mu + \xi_\mu(x)$, when $||\partial_\mu \xi_\nu|| \in \mathcal{O}(h)$ [68]. We can use this property[1] to choose a gauge condition that simplifies the equation. To simplify notation we can define $\bar{h}_{\mu\nu} \coloneqq h_{\mu\nu} - \frac{1}{2}\eta_{\mu\nu}h$. We can then use the DeDonder gauge [68]

$$\partial^\alpha \bar{h}_{\alpha\mu} = 0, \tag{3.5}$$

by choosing $\Box\xi_\mu = \partial^\alpha \bar{h}_{\alpha\mu}$, which is known to always have a solution [68]. In this gauge equation (3.4) reduces to

$$\Box \bar{h}_{\mu\nu} = -\frac{16\pi G}{c^4} T_{\mu\nu}. \tag{3.6}$$

This equation is a well known wave equation [88], which, again, is known to always have a solution. However, the DeDonder gauge doesn't fix the coordinate system completely. It only used $\Box\xi_\mu = \partial^\alpha \bar{h}_{\alpha\mu}$, which was set to zero. Other transformations with $\Box\xi_\mu = 0$ are still allowed. This freedom can be used to further constrain the degrees of freedom of (3.6). It allows us to set the trace of the metric perturbation $h = 0$. As a consequence $\bar{h}_{\mu\nu} = h_{\mu\nu}$. It also allows us to set the components $h_{0\mu} = 0 = h_{3\mu}$, by choosing the coordinate system such that the wave-vector of the solution points in the x_3-direction [89]. In this gauge, solutions must be of the form

$$h^{\mathrm{TT}}_{\mu\nu} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & h_+ & h_\times & 0 \\ 0 & h_\times & -h_+ & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{3.7}$$

By construction in this gauge the metric is traceless and transverse in nature, i.e. $k^\mu h_{\mu\nu} = 0$ where $\vec{k}$ is the wave-vector [89]. For this reason it is known as the *transverse-traceless gauge* (TT). The superscript TT specifies that the metric is evaluated in the TT gauge.

A complete gauge, such as the TT gauge, selects one specific coordinate system. It is usually chosen to simplify calculations, but may not be observable. To understand the coordinate system chosen by the gauge conditions underlying the TT gauge, we can check the trajectories of test particles which are initially at rest ($\dot{x}^i = 0$) in this frame. In this case, the geodesic equation (3.2) simplifies to [68]

$$\ddot{x}^i = -\Gamma^i_{00}\left(\dot{x}^i\right)^2 = -\frac{1}{2}(2\partial_0 h_{0i} - \partial_i h_{00})\left(\dot{x}^i\right)^2 \overset{\mathrm{TT}}{=} 0. \tag{3.8}$$

[1] See [87] for a more mathematical justification of this gauge freedom.

Therefore, particles at rest remain at rest and are not accelerated by the passing of a GW. This means that the reference frame selected by the TT gauge changes with the GW.

However, the coordinate system of the TT gauge is not the frame we observe. Instead, we observe from a central point and live in a locally flat spacetime. Mathematically this can be expressed by constructing a local inertial frame at a point chosen as the origin and describing everything in terms of that frame [68]. To calculate the change in the distance of any point from the origin, we can use the equation for the geodesic deviation, which reduces to

$$\ddot{x}^i = -R_{i0j0}x^j \tag{3.9}$$

in linearized gravity and this frame of reference [89]. $R_{\mu\nu\sigma\rho}$ is the Riemann tensor, which is invariant under coordinate transformations in linearized gravity [68]. We can, therefore, choose any gauge we like to calculate it. In the TT gauge the required components are given by $R_{i0j0} = -\frac{1}{2c^2}\ddot{h}_{ij}^{\mathrm{TT}}$. Inserting this expression into (3.9) one can directly integrate the equation to get [89]

$$x^i(\tau) = x^j(0)\left(\delta_{ij} + \frac{1}{2}h_{ij}^{\mathrm{TT}}(\tau)\right). \tag{3.10}$$

In this frame we can then observe the effect of a passing GW on a ring of freely falling test-masses in the x-y-plane. To obtain a solution to (3.6) we assume to be in vacuum, i.e. $T_{\mu\nu} = 0$. One special solution to this equation is the plane wave $h_{ij}^{\mathrm{TT}} = A_{ij}\sin(\omega\tau)$. The effect of such a solution on the ring of test-masses for the two individual polarizations as obtained from (3.10) is depicted in Figure 3.1. The figure justifies the names "plus" and "cross" given to the two polarizations h_+ and $h_\times$, respectively.

While the observable effects of a GW can be studied by assuming vacuum solutions, their production cannot. To do so, the general solution of equation (3.6) has to be considered, which is given by

$$\bar{h}_{\mu\nu}(t,\vec{x}) = \frac{4G}{c^4}\int \mathrm{d}^3x' \, \frac{T_{\mu\nu}(t - ||\vec{x}-\vec{x}'||/c, \vec{x}')}{||\vec{x}-\vec{x}'||}. \tag{3.11}$$

As the approximations underlying linearized gravity are only valid when there are at most small deviations from flat spacetime, we consider (3.11) only far from the source of radiation. At large distances to the source the scale within the source is negligible and as such $||\vec{x}-\vec{x}'|| \approx ||\vec{x}|| =: r$. Using this approximation (3.11) simplifies to

$$\bar{h}_{\mu\nu}(t,\vec{x}) = \frac{4G}{c^4}\frac{1}{r}\int \mathrm{d}^3x' \, T_{\mu\nu}(t - r/c, \vec{x}'). \tag{3.12}$$

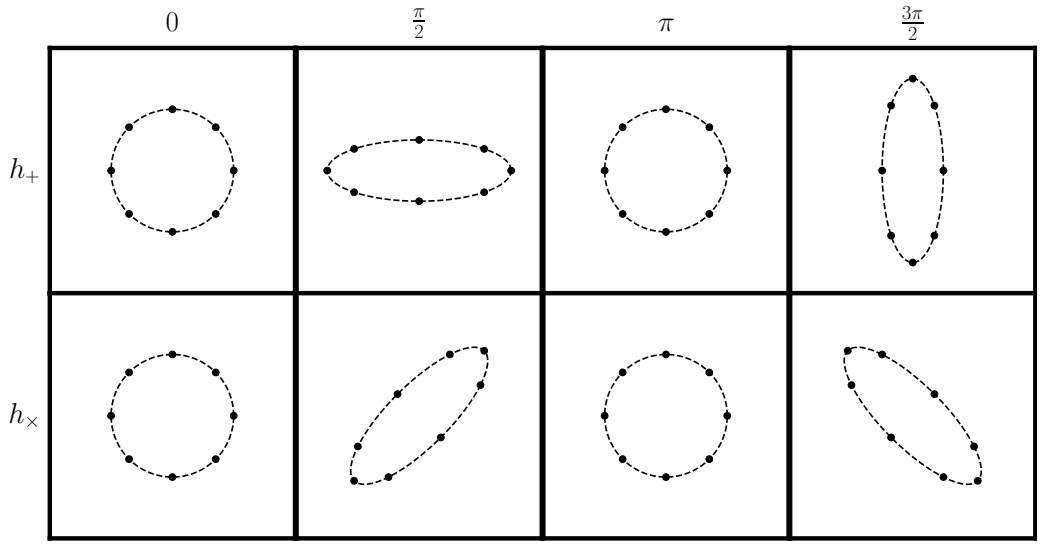


Figure 3.1: The effect of a GW on a ring of freely falling test masses in the plane perpendicular to the propagation direction of the GW. The two rows show the effect of the "plus" and "cross" polarization, respectively. The columns show different phases of the wave. The labels give the value of $\omega\tau$.

Projecting the equation into the TT gauge and using that the energy-momentum tensor is divergence free, one finds [68]

$$h_{ij}^{\mathrm{TT}}(t,\vec{x}) = \frac{2G}{c^4}\frac{1}{r}\ddot{I}_{ij}^{\mathrm{TT}}(t-r/c), \tag{3.13}$$

where $\ddot{I}_{ij}^{\mathrm{TT}}$ is the second time derivative of the projection into the TT gauge of the second mass moment

$$\ddot{I}^{ij} = c^2\partial_0^2 \int \mathrm{d}^3x' \; x'^i x'^j T^{00}. \tag{3.14}$$

The projection of this quantity is given by [68]

$$\ddot{I}^{\mathrm{TT}} = \begin{pmatrix} \left(\ddot{I}_{11} - \ddot{I}_{22}\right)/2 & \ddot{I}_{12} & 0 \\ \ddot{I}_{21} & -\left(\ddot{I}_{11} - \ddot{I}_{22}\right)/2 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{3.15}$$

The above equations can be used to approximate the gravitational radiation emitted by any mass distribution. Because this work considers only CBC sources, we are interested in the GWs sent out by two point particles orbiting each other. Due to the initial assumptions of linearized gravity, we

are restricted to a separation of the two objects where the curvature of the background spacetime caused by the other object is small. In turn, however, this means that the orbital dynamics are governed by Newtonian dynamics and the problem reduces to the Kepler problem [68]. One solution to this problem are circular orbits, which can be described using an effective one body formalism with the reduced mass $\mu = \frac{m_1 m_2}{m_1+m_2}$. From this setup the second time derivative of the second mass moment is calculated to be

$$\ddot{I}^{ij} = 2\mu R^2 {\omega_s}^2 \begin{pmatrix} \cos(2\omega_s t) & \sin(2\omega_s t) & 0 \\ \sin(2\omega_s t) & -\cos(2\omega_s t) & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{3.16}$$

where R is the orbital separation of the two bodies and ω_s is their orbital frequency. In combination with (3.13) and (3.15) this can be used to find the expressions for the polarizations of the GWs. Interestingly, the frequency of GWs originating from a binary system in circular orbits is exactly twice the orbital frequency. For eccentric orbits this property is lost [68].

The expressions obtained from the system described by (3.16) are given in the frame of reference used to solve the Kepler problem. This means that we obtained the waves emitted in the x^3 direction from the center of mass. However, we are interested in the radiation emitted in all directions, as we cannot a priori know the orientation of a source with respect to our detectors. To resolve this issue, one needs to do one final projection which yields the waveform functions [68]

$$\begin{aligned} h_+ &= \frac{4}{r}\frac{G}{c^4}\mu R^2 {\omega_s}^2 \left(\frac{1+\cos^2(\iota)}{2}\right)\cos(2\omega_s t + 2\Phi) \\ h_\times &= \frac{4}{r}\frac{G}{c^4}\mu R^2 {\omega_s}^2 \cos(\iota)\sin(2\omega_s t + 2\Phi), \end{aligned} \tag{3.17}$$

where ι is the inclination of the system with respect to the line of sight and Φ is the orbital phase of the two objects at $t = 0$.

Equation (3.10) showed that GWs can change the distance between particles. Therefore, the particles obtain kinetic energy from a passing GW and stress can be induced in rigid bodies [68]. As a consequence, GWs must carry energy in order to pass it on. However, according to (3.1), if GWs carry energy, they must act as a source of curvature themselves and the assumption (3.3) of GWs being a small perturbation in a flat background is not sufficient anymore. Instead one needs to consider a more general metric $g_{\mu\nu} = \hat{g}_{\mu\nu} + h_{\mu\nu}$.

To obtain the energy carried by a GW the background $\hat{g}_{\mu\nu}$ must be separated from the GW $h_{\mu\nu}$. To achieve this we impose the condition that the

background curvature fluctuates at most slowly compared to the rapidly oscillating $h_{\mu\nu}$. In a rather lengthy calculation one can expand the Ricci tensor in terms of $h_{\mu\nu}$ to second order. Taking the time average over multiple cycles of the GW will then filter out its contribution to the curvature and leaves the background curvature behind. However, since terms quadratic in $h_{\mu\nu}$ may have low frequency components, when the wave-vectors $\vec{k}_1$ and $\vec{k}_2$ are of similar magnitude but opposite sign, one can find that the GW does have an influence on the background curvature. Since the background changes very slowly compared to the GW, we can again view it as locally flat. Under these assumptions one can then write down an expression for the effective energy-momentum tensor of a GW

$$t_{\mu\nu} = \frac{c^4}{32\pi G}\langle \partial_\mu \left(h^{\mathrm{TT}}\right)^{\sigma\alpha} \partial_\nu h^{\mathrm{TT}}_{\sigma\alpha} \rangle. \tag{3.18}$$

A more thorough discussion of this derivation can be found in 1.4.2 of [68], where most of the above was paraphrased from, and 35.13 and 35.15 of [89].

Due to conservation of energy, the energy carried by the GW must be taken away from the source. The main form of energy stored in the binary system studied above is the orbital energy. In turn, this means that the assumption of a constant circular orbit that led to the equations (3.17) cannot hold. Instead one needs to consider the energy lost by gravitational radiation to calculate the phase of the GW.

The total energy carried away from the system is known as the luminosity and can be obtained by integrating the outwards pointing energy flux over a sphere with infinite radius. If we consider a binary system with large separation, where the orbital dynamics are well approximated by Newtonian theory, one finds

$$L_{\mathrm{GW}} = \frac{32}{5}\frac{c^5}{G}\left(\frac{G\omega_s M_c}{c^3}\right)^{10/3} \tag{3.19}$$

for the luminosity L_{GW}, where

$$M_c \coloneqq \frac{(m_1 m_2)^{3/5}}{(m_1 + m_2)^{1/5}} \tag{3.20}$$

is the chirp mass.

By setting the change in orbital energy equal to the luminosity and using Kepler's third law one can obtain a differential equation of the orbital frequency. This can be solved to get [68]

$$f_{\mathrm{GW}}(\tau) = \frac{1}{\pi}\left(\frac{5}{256}\frac{1}{\tau}\right)^{3/8}\left(\frac{GM_c}{c^3}\right)^{-5/8}, \tag{3.21}$$

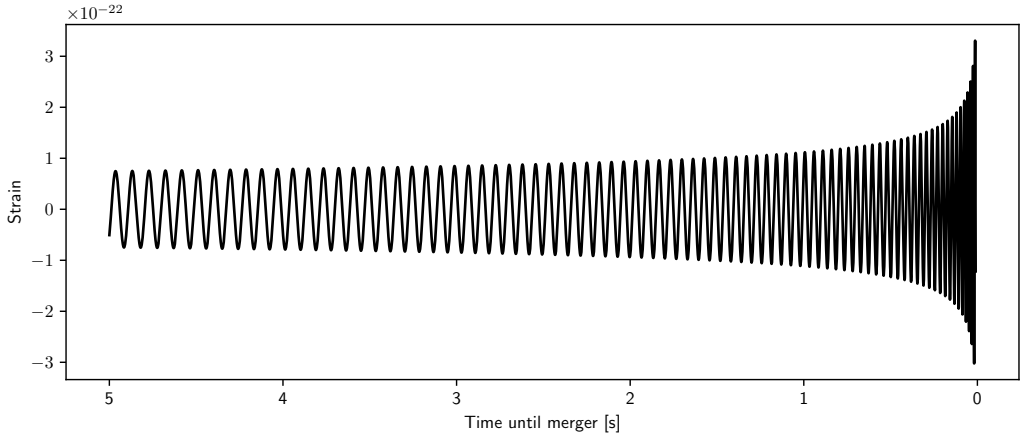


Figure 3.2: The strain of the plus polarization of a GW as calculated by (3.23). The parameters of the waveform are $m_1 = 35\,\mathrm{M}_\odot$, $m_2 = 30\,\mathrm{M}_\odot$, $r = 1000\,\mathrm{Mpc}$, and $\iota, \Phi_0, \theta = 0$.

where τ is the time until the system merges. Integration yields the orbital phase

$$\Phi(\tau) = -2\left(\frac{5GM_c}{c^3}\right)^{-5/8} \tau^{5/8} + \Phi_0, \tag{3.22}$$

with Φ_0 the phase at coalescence. To obtain the waveform, this expression could be inserted into the second mass moment, replacing $\omega_s t$. However, in the approximations that were used to derive the waveforms (3.17) we assumed an almost flat background. As a consequence, the dynamics cannot be highly relativistic and the frequency will change slowly. Therefore, derivatives of the separation R and the frequency $\omega_s = \pi f_{\mathrm{GW}}$ can be ignored to first order and we can approximate the waveforms by using Kepler's third law to replace R by $\sqrt[3]{\frac{GM}{{\omega_s}^2}}$, ω_s in the prefactor of (3.17) by $\pi f_{\mathrm{GW}}(\tau)$, and $2\omega_s t + 2\Phi$ in the argument of the cosine by $\Phi(\tau)$. This yields

$$\begin{aligned} h_+(\tau) &= \frac{1}{r}\left(\frac{GM_c}{c^2}\right)^{5/4}\left(\frac{5}{c\tau}\right)^{1/4}\left(\frac{1+\cos^2(\iota)}{2}\right)\cos(\Phi(\tau)) \\ h_\times(\tau) &= \frac{1}{r}\left(\frac{GM_c}{c^2}\right)^{5/4}\left(\frac{5}{c\tau}\right)^{1/4}\cos(\iota)\sin(\Phi(\tau)). \end{aligned} \tag{3.23}$$

In Figure 3.2 the h_+ polarization from (3.23) is plotted for m_1 =35 $\mathrm{M}_\odot$, m_2 =30 $\mathrm{M}_\odot$, r =1000 Mpc.

3.1.2 Post-Newtonian Formalism

In the previous subsection the main approximation made to derive the gravitational radiation is the assumption of a flat background when describing the source dynamics. In other terms, we assumed that the orbital dynamics are governed by Newtonian dynamics and the background curvature is independent of the speed at which the bodies are orbiting. However, in a gravitationally bound system, the orbital velocity depends on the separation of the two bodies. More specifically, by the virial theorem one finds $(v/c)^2 \sim R_S/r$, where R_S is the Schwarzschild radius and r is the separation of the two objects [68]. Therefore, the core assumption does not hold.

To treat gravitationally bound systems, one needs to consider a generic metric. However, solving the full Einstein equations is difficult and often impossible. One option to simplify problems are post-Newtonian (PN) approximations [89], where the full equations are expanded in a small parameter ϵ. For binary systems, we use $\epsilon \sim (v/c)$ and demand $|T^{ij}|/T^{00} = \mathcal{O}(\epsilon^2)$, i.e. the source is at most weakly stressed [68]. Requiring invariance of the equations under time-reversal the expansion works out to

$$\begin{aligned} g_{00} &= -1 + g_{00}^{(2)} + g_{00}^{(4)} + g_{00}^{(6)} + \dots \\ g_{0j} &= \qquad\quad g_{0j}^{(3)} + g_{0j}^{(5)} + \dots \\ g_{ij} &= \delta_{ij} + g_{ij}^{(2)} + g_{ij}^{(4)} + \dots, \end{aligned} \tag{3.24}$$

where $g^{(n)}$ denotes terms $\sim \epsilon^n$ [68]. To work consistently, when g_{00} is expanded to order n, g_{0i} must be expanded to order $n-1$ and g_{ij} to order $n-2$ [68]. Similarly, the energy-momentum tensor also needs to be expanded

$$\begin{aligned} T^{00} &= T^{(0)\,00} + T^{(2)\,00} + \dots \\ T^{0j} &= T^{(1)\,0j} + T^{(3)\,0j} + \dots \\ T^{ij} &= T^{(2)\,ij} + T^{(4)\,ij} + \dots. \end{aligned} \tag{3.25}$$

To obtain approximations these expressions can be inserted into the Einstein equation (3.1) and terms of the same order in ϵ can be equated. A solution is known to be of n-th PN-order when terms up to order ϵ^{2n} are kept. Therefore, equations of $x.5$ PN-order exist.

To 1 PN order explicit solutions depending on the energy-momentum tensor are given in section 5.1.4 of [68], under the usage of the DeDonder gauge which in the full theory is given by

$$\partial_\mu\left(\sqrt{-g}g^{\mu\nu}\right) = 0, \tag{3.26}$$

where g is the determinant of $g_{\mu\nu}$. For the computation one also needs to consider that for $v/c \ll 1$ the time derivatives are smaller than the spatial derivatives by a factor $\mathcal{O}(\epsilon)$ and hence the flat spacetime d'Alembert operator is given by

$$\Box = -\frac{1}{c^2}\frac{\partial^2}{\partial t^2} + \nabla^2 = \left(1 + \mathcal{O}\left(\epsilon^2\right)\right)\nabla^2, \tag{3.27}$$

where $\nabla^2 = \delta^{ij}\partial_i\partial_j$. From this it is immediately clear that these solutions can only be valid close to the source, as they have to be instantaneous potentials [68].

For further computations it is beneficial to cast the Einstein equations into their relaxed form

$$\Box k^{\mu\nu} = \frac{16\pi G}{c^4}\tau^{\mu\nu}, \tag{3.28}$$

where $\Box$ is the flat spacetime d'Alembert operator,

$$k^{\mu\nu} = \sqrt{-g}g^{\mu\nu} - \eta^{\mu\nu}, \tag{3.29}$$

and

$$\tau^{\mu\nu} = -gT^{\mu\nu} + \frac{c^4}{16\pi G}\Lambda^{\mu\nu}. \tag{3.30}$$

The tensor $\Lambda^{\mu\nu}$ captures the curvature contributions of the deviation from flat spacetime to the energy-momentum tensor [86] and an explicit expression is given in (5.74) and (5.75) of [68]. For (3.28) to be an exact equivalent to the Einstein equations (3.1), one also needs to enforce the DeDonder gauge which is given by $\partial_\nu k^{\mu\nu} = 0$ in this formulation.

To obtain a PN expansion, we can then write the metric $k^{\mu\nu}$ in a formal expansion of $v/c \approx 1/c$

$$k^{\mu\nu} = \sum_{n=2}^{\infty} \frac{1}{c^n} k_n^{\mu\nu}, \tag{3.31}$$

where v/c is the small parameter and only written for bookkeeping. As before, the right hand side of (3.28) must be expanded in the same manner [68]

$$\tau^{\mu\nu} = \sum_{n=-2}^{\infty} \frac{1}{c^n} \tau_n^{\mu\nu}. \tag{3.32}$$

Inserting these expressions into the relaxed Einstein equation, using (3.27), and equating terms of the same order in $1/c$ we obtain the recursive equation

$$\nabla^2 k_n^{\mu\nu} = 16\pi G \tau_{n-4}^{\mu\nu} + \partial_t^2 k_{n-2}^{\mu\nu}. \tag{3.33}$$

When solutions k_0 and k_1 are known, the right hand side of the above equation is fully determined and one needs to invert the Laplace operator ∇^2 to obtain higher order PN solutions.

This inversion, however, is not trivial, as the usual Poisson-integral diverges for high PN-orders [68]. Instead, one can find a particular solution for some finite PN-order by multiplying the right hand side by r^B, for B negative and large enough in magnitude. One can then consider the result in the complex plane and study the pole for $B \uparrow 0$, which is well defined and omits a solution. For the general solution the homogeneous solution has to be added. See 5.3.2 of [68] and 5.2 of [86] for more detail. Importantly, these solutions depend on the energy-momentum tensor of the source, but are only valid close to the source.

To get the radiation of the far field, we can consider the Post-Minkowskian (PM) expansion. Since we are considering the region outside of the source, we are looking for vacuum solutions, i.e. $T^{\mu\nu} = 0$. In this case, the relaxed Einstein equation (3.28) simplifies to

$$\Box k^{\mu\nu} = \Lambda^{\mu\nu}. \tag{3.34}$$

Outside the source the curvature will be small and we can expand in $R_s/r \sim G/r$. In a slight abuse of notation we write

$$k^{\mu\nu} = \sum_{n=1}^{\infty} G^n k_n^{\mu\nu} \tag{3.35}$$

$$\Lambda^{\mu\nu} = N^{\mu\nu}\left[k,k\right] + M^{\mu\nu}\left[k,k,k\right] + \dots, \tag{3.36}$$

where $N^{\mu\nu}$ and $M^{\mu\nu}$ are tensors of quadratic and cubic order in G, respectively. Equating terms of the same order in G leads to

$$\Box k_n^{\mu\nu} = \Lambda_n^{\mu\nu}\left[k_1, \dots, k_{n-1}\right], \tag{3.37}$$

where $\Lambda_n^{\mu\nu}$ is the sum of all tensors $N^{\mu\nu}$, $M^{\mu\nu}$, $\dots$ of order n in G.

In (3.37) we find another recursive relation, where higher order terms are determined from lower order ones. So once a solution $k_1^{\mu\nu}$ is known, in principle all higher order solutions can be determined.

To find a solution to $k^{\mu\nu}$ we observe that $\Lambda_1 = 0$ and write $k_1^{\mu\nu}$ in a multipole expansion. Once the gauge condition is enforced, one can obtain solutions that depend on six families of multipole moments. See (5.95) to (5.101) in [68] for explicit expressions. However, these multipole moments are still arbitrary functions, as they know nothing about the source of the radiation yet.

Iterating k_1 to find higher order solutions is again not trivial. In principle the right hand side of (3.37) is determined by all lower order solutions $k_1, \cdots, k_{n-1}$. So the task is in inverting the d'Alembert operator $\Box$. Notice that since the expansion is not in terms of v/c anymore the d'Alembert operator must be inverted, rather than the Laplace operator. The solution to the problem is very similar to the solution used for the PN-expansion. We find a particular solution by truncating the multipole expansion of $k_1^{\mu\nu}$ at the desired order and multiply $\Lambda_n^{\mu\nu}$ by r^B, with B now positive and sufficiently large to cancel all factors $1/r$ in the multipole expansion. We then use the analytic continuation again and add the homogeneous solution to obtain the most general one. See 5.3.1 of [68] or 4.1 of [86] for more detail.

To determine the multipole moments of the PM solutions, one observes that the PN and PM solutions have an overlapping region of validity [86]. Therefore, one uses the PN solutions, which are determined by the energy-momentum tensor of the source, to fix the multipole moments of the PM solutions. To do so, the PN potentials are written as a multipole expansion and the PM solitions are re-rexpanded in terms of powers of $1/c$. Finally, terms of the same order are equated. The explicit expressions are given in (85) - (90) of [86].

With this, the metric in the far region has in principle been determined. To find the actual waveform emitted by some physical system, one can follow a very similar approach to the one used in subsection 3.1.1. A binary system is well approximated by a sum of two delta functions[2] in the energy-momentum tensor. Using this energy-momentum tensor, one can find the metric in the near region of the source to determine the orbital energy E. On the other hand, it can be used to determine the metric in the far region, which can in turn be integrated to obtain the luminosity L_{GW} of the source. Under the assumption that the system loses energy only through gravitational radiation, we can use the equation

$$-\frac{\mathrm{d}E}{\mathrm{d}t} = L_{\mathrm{GW}} \tag{3.38}$$

to obtain the orbital phase of the binary system.

For the case of linearized gravity discussed in subsection 3.1.1 the orbital energy was known from Newtonian dynamics. In the PN approximation it needs to be calculated. To do so, one inserts the PN metric into the geodesic equation (3.2) to obtain the equations of motion of the two orbiting bodies. From this one can find the orbital energy. Explicit expressions can be found

[2]Using delta functions will actually lead to divergencies at high PN-order. For this reason some regularization has to be applied. See section 8 of [86] for details.

in [86], equation (115) gives the generic metric to 3 PN-order, (168) gives the equation of motion of a binary system to 3.5 PN-order, (170) gives the orbital energy to 3 PN-order, and (194) gives the energy to 3 PN-order assuming circular orbits.

The luminosity can be directly computed from the metric by integration. See section 3.5 of [68] or section 6 and section 10.2 of [86] for details. Explicit expressions at 3.5 PN-order are given in (5.257) of [68] and (231) of [86]. The resulting phase to 3.5 PN-order is given in (235) of [86].

To obtain the waveform, one can project the metric $g_{ij} - \delta_{ij}$ into the TT-frame. The metric g_{ij} is the desired PN-approximation, the derivation of which was discussed above. The polarizations can be read of. Explicit expressions were derived in [90, 91] and can be found in (237) - (241) of [86]. To get the actual waveform, the time dependent phase to highest known PN-order should be inserted.

The discussion above is meant as a high-level overview of how PN-waveforms can be obtained. A more detailed overview, including mathematical details, is given in [86]. Notably, this discussion excluded spin effects of the binary system and its constituents. Discussions including spin can be found for instance in [92–96].

3.1.3 Waveform Models

The waveforms discussed in subsections 3.1.1 and 3.1.2 are, by construction, only valid during the inspiral, even though high order PN waveforms remain accurate until a few cycles before the merger of the two objects [86, 97]. However, the emitted energy scales to high power with the orbital frequency of the two merging objects, as can be seen from (3.19) to linear order and from (231) of [86] to 3.5 PN-order. Therefore, the evolution of the waveform close to merger can be of high importance for detection.

This subsection gives an overview of the three most prominently used waveform-model families and their variants. All of them model the entire orbital evolution from inspiral, over merger, to ringdown (IMR), in the parameter regions where they are valid. To be able to represent the merger phase, all of them rely on numerical relativity (NR) simulations, which solve the Einstein equations numerically. These simulations are the most accurate tools available to solve the highly non-linear interactions but running them requires enormous amounts of computational power. Generating a waveform encompassing the final few orbits and the merger for a single binary system using NR can take days or even weeks, depending on the initial conditions and type of binary system [71]. Therefore, it is often not viable to use the resulting waveforms in data analysis methods discussed in section 3.2 di-

rectly. The waveform models below are constructed to overcome this issue by combining the accurate NR waveforms with analytical approximations.

Instead of writing the GW polarizations as individual functions, one can also define a single complex strain

$$H \coloneqq h_+ + ih_\times, \tag{3.39}$$

which can be decomposed using spherical harmonics [44, 98]

$$H = \sum_{l\geq 2} \sum_{m=-l}^{l} Y_{lm}^{-2}(\iota, \Phi_0) h_{lm}(\tau, r, \kappa), \tag{3.40}$$

where

$$h_{lm}(\tau, r, \kappa) = A_{lm}(\tau, r, \kappa) e^{-i\Phi_{lm}(\tau,\kappa)}. \tag{3.41}$$

κ represent all parameters that describe the source itself and which do not depend on the location at which one observes the radiation. Waveform models usually describe the amplitude evolution A_{lm} and the phase evolution Φ_{lm}. If a waveform model is capable of modeling these quantities for $(l, m) \neq (2, \pm 2)$, one speaks of a model that is capable of simulating higher order modes (HM).

Effective One-Body Waveform-Family

Effective one body (EOB) waveforms were one of the first GW approximants that could model not only the inspiral but also the merger and ringdown. The formalism was introduced in [99] and maps the two body problem of a BBH system onto that of a test particle moving in an effective external metric. To that end, a Hamiltonian is constructed which is supplemented with a radiation-reaction force. It takes the PN approximation as a starting-point to express the Hamiltonian and then expresses it as well as the radiation-reaction force in a resummed form, where the functions are non-polynomial [100]. An introduction to the resummation is given in 14.1 of [69]. This resummation can then be extended by adding free parameters that model non-perturbative effects [92, 101, 102]. The EOB formalism models the inspiral and merger by fitting the free parameters to NR and then attaches an analytical model for the ringdown.

The first full IMR waveform from the EOB family was derived in [31]. Initially it modeled only non-spinning BBHs but was later extended to the spinning case in [103, 104]. The early models were solely based on analytic calculations. However, with the breakthrough in NR in 2005 [69, 105–107] accurate ground-truths became available, which could be used to constrain the free parameters that extend the EOB models [100, 108]. These models

were incrementally improved [109, 110] and extended to include higher order modes, aligned spins, and precession [111–114]. To keep track of the capability of these models, a naming scheme was introduced. Waveforms of this family carry an "EOB" in their name, which is usually followed by "NR" to signify that the model coefficients were tuned to NR simulations. A leading "S" informs the user about the capability of the model to represent (anti-)aligned spins. Afterwards a version number is tagged onto the string. The first model named this way is known as "SEOBNRv1" and was introduced in [115], "SEOBNRv2" is the model described in [116, 117], "SEOBNRv3" the one described in [114, 117], and "SEOBNRv4" in [118]. At the time of writing, the EOB model that encompasses the most orbital dynamics is "SEOBNRv4PHM", which can account for higher order modes and precession at the same time [119]. The EOB models are formulated as a set of differential equations, that need to be solved for given parameters, which can be prohibitively slow depending on the application [117]. They are also formulated in the time domain, wheras data analysis usually requires the waveform in the frequency domain. For these reasons, EOB models are often sped up by building reduced order models in the frequency domain [120, 121].

Phenom Waveform-Family

The phenomelogical – short Phenom – waveform-family was introduced in [32]. At the core, their method matches the analytic PN-waveform that describes the inspiral with the more accurate NR-waveform during merger to produce a finite amount of hybrid waveforms. Afterward, the hybrid waveforms are fit to a phenomenological parametrized model which can in turn be mapped to the physical parameters of the binary system [97]. The resulting model is fast to evaluate, highly accurate, and usually native to the frequency domain.

Since the initial paper, which focused on non-spinning BBHs in a mass range of $30\,\mathrm{M}_\odot$ to $130\,\mathrm{M}_\odot$, many different versions of the waveform model have been released. The work of [122] introduced a Phenom-waveform model that is capable of modeling BBHs, the spins of which are (anti-)aligned with the orbital angular momentum. Through [123, 124] waveform models that can take precession effects of the emitting binary into account were introduced and dubbed PhenomP, where the "P" signifies the ability to represent precessing systems. The PhenomP model was used during data analysis of the first detected GW [13, 117, 125]. The authors of [126] introduced a model that can account for higher multipoles ("PhenomHM"), which was combined with an updated version of the precessing waveform model [127] to obtain a waveform model that can represent both effects [128] ("PhenomPv3HM").

The underlying model was also revised multiple times over the years. The evolving non-precessing approximants were called "PhenomB" [122], "PhenomC" [97], and "PhenomD" [129, 130]. The most recent version, skipping many letters in between, is the "PhenomX" waveform model [131], which has a version that allows for the computation of higher order multipoles called "IMRPhenomXHM" [132], and a version that includes both higher order multipoles and precession effects, known as "IMRPhenomXPHM" [133]. The prefix "IMR" signifies that the waveform models the full inspiral-merger-ringdown. The postfix "HM" signifies that higher order multipoles can be computed.

Numerical Relativity Surrogate Waveform-Family

By now hundreds of NR simulations are available that span large parts of the expected parameter space for BBH mergers [134, 135]. The resulting waveforms are the most accurate predictions of the emitted radiation of a merging binary system that are available to us. NR surrogate models leverage the existing catalogs of NR simulations to interpolate between them. They are typically more accurate than the approximate methods described above but are only valid in the limited parameter regions that are covered by the underlying simulations [136].

The first surrogate model was introduced in [137] and used non-spinning waveforms produced by the Spectral Einstein Code [138, 139]. The model was later extended to include precession [140] and more parameters [141, 142]. Currently the model named "NRSur7dq4" [143] is used in state-of-the-art analyses [14], where "7dq4" signifies that the model is valid for a 7 dimensional parameter space (6 spins + mass ratio) up to a mass ratio q of 4.

3.2 Data Analysis for Compact Binary Coalescence Signals

The observable changes that GWs induce in detectors are very small, as we will see in section 3.2.1. Consequently, detecting them directly is a difficult problem that stood unsolved for nearly a century.

The first indirect detection of GWs was made in 1982 when Joseph H. Taylor and Joel M. Weisberg closely studied the pulsar PSR 1913+16 [12]. Said pulsar was discovered in 1974 and was the first one that was part of a binary system [144]. It became known as the Hulse-Taylor pulsar, named after the researchers who discovered it. A pulsar is a rotating NS that emits

a beam of radiation. If the beam continuously sweeps across a detector, it appears as a periodic signal. The frequency of these pulses is known to be extremely stable [82, 83] and, consequently, allows to trace the orbit of pulsars in binaries very accurately. As was discussed in section 3.1, a binary of compact objects emits considerable amounts of energy as gravitational radiation and subsequently slowly spirals together. The change in orbit for the Hulse-Taylor pulsar due to energy lost in GWs was calculated and then checked against measurements [12, 145]. The observed change in the orbital period matches the theoretical predictions from GR to an astonishing accuracy.

Although this indirect detection was strong evidence for the existence of GWs, it was no direct detection. It could not be ruled out that some other unaccounted-for effect caused the change in the orbital period. To confirm the observation, a direct detection of the physical effects of GWs was sought.

Initial efforts toward a direct detection were made by Joseph Weber in the 1960s [11, 68]. He tried to measure resonances in an aluminum bar induced by the gravitational radiation. In 1969 he claimed to have detected a GW [146, 147] but his findings were not reproducible and subsequently dismissed [148]. For details on resonant bar detectors and their response to GWs see chapter 8 of [68].

The first confident detection of a GW was made by the LIGO-Virgo collaboration (LVC) on September 14, 2015 [13]. Since then, $\mathcal{O}(100)$ observations were reported by several groups analyzing the public data [149] that has been recorded in three observation periods [14, 15, 150–152]. The data was recorded by three earth bound detectors LIGO Livingston, LIGO Hanford [6], and Virgo [7]. Toward the end of the last observing period, a fourth detector – KAGRA [8] – joined the network. All of them are kilometer scale laser interferometers, of which a brief introduction will be given in subsection 3.2.1.

A lot of new insights into the Universe can be drawn from these observations. One of the most prominent studies are tests of GR [16, 17, 153, 154]. So far, no evidence for a deviation from GR has been found. Other insights that can be obtained from GW observations include constraints on the equation of state of NS [18, 19], the Hubble-constant [20, 21], the population of BHs [22], and the fraction of dark matter explainable by BHs [23, 155, 156]. They can also help explain how BHs and other systems form [157]. The list of possible applications goes on and keeps growing as our understanding of the Universe improves.

Nonetheless, to make any of these claims, GWs must first be identified and characterized in the detector data. In subsection 3.2.1, we will see that noise dominates the raw output data of the detectors. Sophisticated data analysis methods were developed to extract the weak signals from the raw

data. The LIGO-Virgo-KAGRA collaboration (LVK) deploys a suite of different algorithms to detect potential signals in the data [14], which can be classified into two categories based on their underlying search strategy. GstLAL [59, 158–160], MBTA [161, 162], and PyCBC [30, 58, 60, 163, 164] are search pipelines that filter the data against a set of pre-computed waveforms. They are often called modeled searches and I will describe their core concept "matched filtering" in subsection 3.2.2. Coherent wave burst (cWB) [38, 165–167] is a loosely modeled search pipeline that makes minimal assumptions about the source and looks for coherent excess power in multiple detectors to detect transient events.

When accurate models of the signal exist, modeled searches are more sensitive than loosely modeled searches. However, modeled searches can only target a fixed range of parameters of binary systems, which has to be chosen before the data is searched. Since the computational cost of these searches scales with the searched parameter space, one has to make compromises between search space and runtime. As a consequence, signals from sources with unexpected or unlikely parameters may be missed. The modeled searches that are used in the studies that have identified all known signals to date [14, 15, 151, 152] assume circular orbits, aligned spins, and only search the mass range from $1\,\mathrm{M}_\odot$ to $758\,\mathrm{M}_\odot$. They also put constraints on the mass ratio and assume that higher order modes have a negligible effect on the observed waveform. Several specialized searches targeting eccentric systems [168], large higher order mode contributions [43], or sub-solar mass systems [169–173] have been carried out, none of which have found any new signals. Most importantly to this thesis, machine learning search algorithms are starting to be explored to reduce computational costs and widen the parameter space that can be searched [48, 55, 57, 61, 65]. A more detailed discussion of these approaches is given in section 3.4 and chapters 4, 6, 7, and 8.

3.2.1 Noisy Detector Data

At the time of writing, five laser interferometer GW detectors are operational. These include the LIGO detectors [6] in Livingston (Louisiana, USA) and Hanford (Washington, USA), the Virgo detector [7] in Cascina (Italy), the KAGRA detector [8] in Kamioka (Gifu, Japan), and the GEO 600 detector [9, 174, 175] in Sarstedt (Germany).

The LIGO and Virgo detectors are already sensitive enough to regularly detect GWs during operation [14]. KAGRA was only taken into service recently and has not yet reached a level of sensitivity where detections are likely [176]. It is expected to reach its sensitivity targets during the fourth

observing run (O4) [8]. GEO 600 is unlikely to ever reach a sensitivity where it can regularly detect GWs, due to the size of the detector. It is, therefore, used as a testbed for new technologies that transition to the other detectors once they have proven to yield improvements [177].

At their core all of these instruments are highly advanced Michelson interferometers, which split a laser beam into two orthogonal beams. Each beam travels along the arm of the interferometer until it is reflected back by a mirror at the end of the arm. When the beams recombine at the beamsplitter, they interfere and the output power depends on the difference of the distance the light has traveled in both arms. Due to the short wavelengths of light, this allows to measure very small changes in the differential arm lengths [178].

As discussed in section 3.1, GWs cause a physical displacement of test masses with respect to some reference point. In the GW detectors, the mirrors at the end of the arms act as test masses and the beamsplitter as the point of reference. Therefore, the Michelson interferometer is sensitive to GWs. Under the assumption that the arm length L of the interferometer is small compared to the wavelength of the GW, from equation (3.10) one finds to first order that the change in arm length ΔL is given by

$$L + \Delta L = L\left(1 + \frac{h_+}{2}\right), \tag{3.42}$$

when only the + polarization is considered and the source is optimally located. Therefore, in order to amplify the effect of a GW on the detector, one can increase the arm length L. Another option to increase the measured signal is to increase the power that circulates in the arms or install a Fabry-Perot cavity [68]. For these reasons, the LIGO and Virgo detectors have arm lengths of 4 km and 3 km, respectively, and circulate several hundred kW of laser power in the Fabry-Perot cavities of their arms. KAGRA also has an arm length of 3 km but is built underground and cryogenically cools its mirrors. GEO 600 has an arm length of 600 m. Figure 3.3 shows a high level overview of current detectors.

To obtain the length deviation, one can in principle use equation (3.10) as discussed above. However, that equation is in a frame of reference in which the effect of GWs on test masses can be easily described and where the x^3 direction points from the source to the detector. The frame of reference of the detector, on the other hand, defines the x- and y-axis as the arms of the detector and the z-axis such that it is orthogonal to the x-y-plane pointing away from the center of the earth. To translate the motion in one frame to the motion in the other frame, one can use a transformation based on

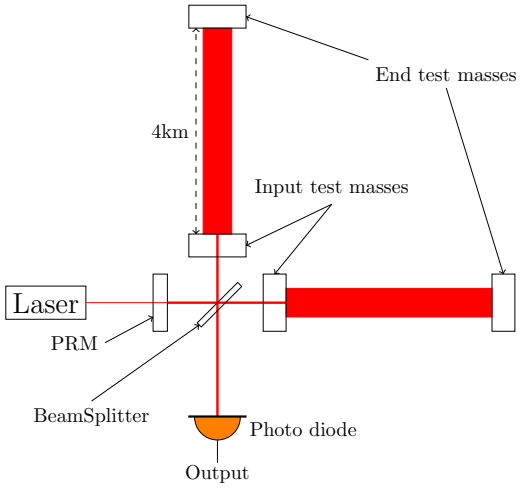


Figure 3.3: The figure shows a simplified diagram of the interferometric GW detectors. The diagram is not to scale and only highlights the beam path. Any operations for stabilization, mode cleaning, and so on are not shown. The laser beam travels from the laser through the power recycling mirror (PRM) to the beam splitter, where it is split into equal parts to the two arms. In each arm, the beam passes an input test mass, which is a mirror, travels 4 km, and is reflected at the end test mass. On its way back, a large portion of the light is again reflected by the input test mass to increase the light travel time in the arms. Together the two test masses essentially create a large Fabry-Perot cavity. Once the light passes the input test mass, it recombines and interferes with the light from the other arm. A photodiode captures the beam at the output port (south) to produce the readout. The part going back to the laser is reflected at the power recycling mirror to increase the laser power in the interferometer. This figure was adapted from Figure 1 in [6].

Euler-rotations. Carrying out these calculations leads to [179]

$$\begin{aligned} h = \; & F_+(\theta,\varphi)(\cos(2\psi)h_+ - \sin(2\psi)h_\times) \\ & + F_\times(\theta,\varphi)(\sin(2\psi)h_+ + \cos(2\psi)h_\times), \end{aligned} \tag{3.43}$$

where θ, φ, and ψ are the Euler-angles known as declination, right ascension, and polarization angle, respectively, and

$$\begin{aligned} F_+(\theta,\varphi) &= \frac{1}{4}(3+\cos(2\theta))\sin(2\varphi) \\ &= \frac{1}{2}\left(1+\cos^2(\theta)\right)\sin(2\varphi), \\ F_\times(\theta,\varphi) &= \cos(\theta)\cos(2\varphi). \end{aligned} \tag{3.44}$$

In total, the GWs the detectors measure depend on 15 parameters for BBH signals and 17 for BNS systems. An overview of these parameters is given in Figure 3.5. They affect the measured signal in different ways:

m_1, m_2 The component masses of the two objects. Usually it is defined that $m_1 \geq m_2$. To leading order they are the only parameters that influence the frequency evolution of the signal and they act mainly through the mass combination known as the chirp mass. The larger the total mass of the system, the lower the frequency at which the two objects merge.

$\vec{\chi}_1, \vec{\chi}_2$ The three-dimensional spin vectors of the individual objects. They often are only measured in a mass-weighted form known as the effective spin χ_e [123] and the effective precession spin χ_p [180]. As long as the spins are (anti-)aligned with the orbital angular momentum, they affect the time scale at which the objects merge. The more aligned the spin, the longer the time scale. If they are not aligned with the orbital angular momentum they cause precession effects, that become visible as a beating in the signal.

Λ_1, Λ_2 Tidal deformabilities of the two objects. They quantify the amount of matter distortion due to the gravitational gradient in the close proximity to a highly compact object. Subsequently they are 0 for BHs and are only considered for BNS and NSBH mergers. The larger the tidal deformability, the stiffer the NS, and the longer the duration of the inspiral. However, the influence of matter effects are expected to be very small and they have not been measured yet [14].

r The distance from the source to the detector. It is inversely proportional to the amplitude of the waveform measured in the detector.

$\boldsymbol{\theta, \varphi}$ The declination θ and the right ascension φ in the reference frame of the detector. They affect the amplitude of the measured signal through (3.43) to the point where the detectors can be blind to specific polarizations from given locations in the sky. Figure 3.4a shows the average influence of the sky position on the strength of the detected signal.

$\boldsymbol{\Psi}$ The polarization angle. It is the angle by which the frame of the GW has to be rotated around the propagation axis to line up with the frame of the detector. For a single detector it can be absorbed into a redefinition of the two polarizations of the wave and cannot be resolved. See Figure 3.5 for a visualization of this angle.

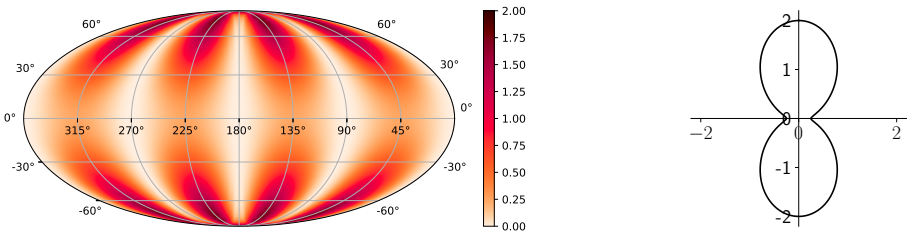
$\boldsymbol{\iota}$ The inclination of the orbital plane of the source with respect to the line of sight of the detector. It influences the radiated power and, therefore, the amplitude of the signal. A plot of the power distribution is shown in Figure 3.4b.

$\boldsymbol{\Phi_0}$ The coalescence phase, i.e. the orbital position of the two objects at merger. It introduces an overall phase-shift in the measured signal.

$\boldsymbol{t_0}$ The time of coalescence, which is often defined as the time at which the merger signal arrives at the center of the earth. Controls the time at which the system merges and time shifts the waveform in the detector data.

Equation (3.43) describes the signal the detector would pick up in an ideal environment. However, as we all know the world around us is not an ideal environment, as it is noisy. To get a sense of scale at which noise sources have to be considered, one can estimate the length change induced by a GW on an interferometric detector with arms at a right angle and 4 km length. Using (3.23) for a non-spinning, face-on system ($\iota = 0$) at a distance r of 1 Gpc with masses similar to those of the first observed GW directly overhead the detector yields a length deviation of $\approx 10^{-18}$ m. At these length scales almost any noise source is non-negligible. The most important noise sources are the following [6, 7, 148]:

Seismic noise Seismic noise consists of small vibrations of the earth, caused by anything from passing cars to earthquakes. It is most notable at small frequencies. Along with the gravity gradient noise (see below) and control noise [182, 183] it is the main reason why earth bound detectors are not sensitive to sub-Hertz GWs (compare Figure 3.6a). To isolate the detector from most of the seismic noise, the mirrors are suspended in a triple pendulum setup and vertically damped by springs. A combination of active and passive isolation is deployed.



(a) Sky location (b) Inclination

Figure 3.4: **(a) Sky location:** The influence of the sky-position on the GW intensity observed by the detector averaged over all source orientations. It is known as the antenna power function and we observe that there are favorable and unfavorable sky-locations for the detectors. This figure was inspired by Figure 2 of [179]. **(b) Inclination:** The multiplicative factor of the radiated power of a GW. The angle to the x-axis is the inclination of the system, the distance to the origin gives the factor by which the radiated power from the source intrinsic parameters has to be multiplied. This shows that face-on systems ($\iota = \pi/2$ or $\iota = 3\pi/2$) are seen with the highest intensity, while edge-on systems ($\iota = 0$ or $\iota = \pi$) are observed with the lowest intensity. However, it also shows that there is no dead-spot and some power is radiated in all directions. This figure was adapted from Figure 3.7 of [68]

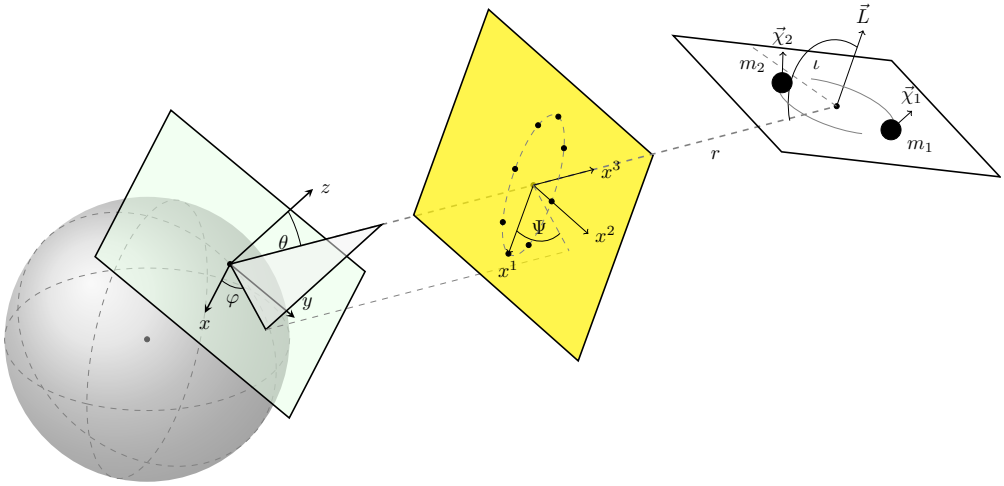


Figure 3.5: A diagram showing the parameters the measured signal of a BBH merger depends on. The three differently colored planes indicate the three different coordinate systems used in the calculation of the GW effect. The white plane belongs to the source. It is aligned with the orbital plane. The yellow frame shows the preferred frame of reference for a GW, where its propagation direction is aligned with the x^3-axis. The deformed ellipsoid on the plane indicates the movement of a ring of test masses in this frame (compare Figure 3.1). The green plane shows the frame of the detector, where the x- and y-axes are aligned with the detector arms and z points away from the center of the earth, which is depicted as the gray sphere. The angles show the transformations that are associated with them. The distance r is highlighted by the dashed gray line connecting the detector and the source. Not shown are the time of coalescence t_0 and the coalescence phase Φ_0. $\vec{L}$ shows the orbital angular momentum. The figure is not to scale and only meant to show the meaning of the different parameters. It was inspired by Figure 9.2 of [181].

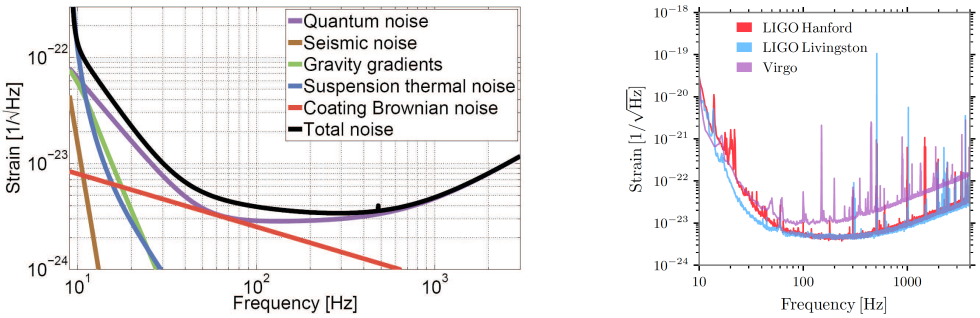
Thermal noise Thermal noise has many sources. Any part that has a finite temperature vibrates and adds noise into the system. The most important sources are the Brownian motion of the mirror coatings and the thermal noise in the suspension used to isolate the mirrors from seismic noise. Additionally, due to the high laser-power required in the detectors, the mirrors have a temperature gradient that adds additional noise. The coating Brownian noise has a non-negligible impact in the most sensitive regions of the detector (compare Figure 3.6a), while the other thermal noise sources usually enter at low frequencies. To reduce thermal noise, KAGRA operates at cryogenic temperatures [8].

Quantum noise Quantum noise has two origins. First, there is shot noise. This is the noise in the measured output power, due to the discrete nature of individual photons in the laser beam. Second, there is radiation pressure noise. It is caused by the laser beam exerting a pressure on the mirror. Due to the discrete nature of the photons, this pressure fluctuates. Combined, quantum noise is the limiting factor for the sensitivity in the most sensitive frequency range as well as at high frequencies. To reduce shot noise, one can increase the laser-power. However, with greater laser power, the radiation pressure noise increases and thermal effects become more prominent. Another option to reduce shot noise is the usage of squeezed light [184–186]. Squeezing can be used to shift noise from phase to amplitude, thus reducing shot noise, or vice versa to reduce radiation pressure noise by utilizing the Heisenberg uncertainty principle [34].

Gravity gradient noise Gravity gradient noise is caused by seismic waves that induce density fluctuations in the earth. These density fluctuations causes the gravitational field to change over time and influence the detector. Because these shifts in density are relatively slow processes, this type of noise is large at low frequencies and is a limiting factor. Since the gravitational field cannot be shielded, the only way to reduce gravity gradient noise is to select locations for the detectors where density fluctuations are small. One can also monitor the changes in the gravitational field and subtract their influence afterward.

To characterize the strength of the noise, one can calculate the *power spectral density* (PSD). The PSD gives the power present in the detector due to noise as a function of the frequency. On a finite stretch of pure noise data $n(t)$ with duration T the one-sided PSD $S_n(f)$ can be calculated as [68]

$$S_n(f) = \frac{2}{T}\langle|\tilde{n}(f)|^2\rangle, \tag{3.45}$$



(a) The noise budget of the advanced LIGO detectors.

(b) Power spectrum measured during O3.

Figure 3.6: The theoretical noise budget of the detector (a) and the measured amplitude spectrum (b) of advanced LIGO. The left figure shows the contribution of the different noise sources to the overall noise of the detector. It was taken from [6]. The right figure shows measured ASDs for the three main detectors during O3. The curves are representative of the detector sensitivity during the observational period and the figure was taken from [14].

where $\tilde{n}(f)$ is the Fourier transform of $n(t)$ and the average $\langle\cdot\rangle$ is performed over different realizations of the noise. Its frequency resolution Δf is given by $1/T$ and, by matching the units, one can infer that the PSD is given in units of 1/Hz. In practice, the data is often chopped into many overlapping shorter pieces, a window is applied to each piece to reduce the impact of required periodicity of the Fourier transform, and the resulting PSDs are averaged. This algorithm is known as Welch's method [187].

Figure 3.6b shows an estimate of the *amplitude spectral density* (ASD) during the third observing run (O3) for the three detectors that collected the majority of the data. The ASD is the square root of the PSD. Comparing it to the theoretical limit from Figure 3.6a one finds that the measured noise contains high spikes at certain frequencies. Most of these originate from known sources, such as the power grid frequency or violin modes of the mirror suspensions [125]. However, due to their narrow frequency range the effect on the detectability of GWs from CBC sources is marginal.

One noise characteristic that is not evident in the PSDs are short duration non-Gaussian noise transients known as *glitches*. They can mimic the frequency evolution of real GW events and a lot of effort is being invested to reject them [188–191]. These glitches are by no means rare with an average glitch rate of $0.29\,\mathrm{min}^{-1}$ to $1.17\,\mathrm{min}^{-1}$ [14]. The cause for some of these glitches is known and can be traced back for instance to light scattering or computer glitches; others have yet uncharacterized origins [190].

Several upgrades and extensions to the current network of ground based GW detectors are planned. To extend the network in the short term future, it is planned to construct a third observatory of the advanced LIGO specifications in India [192, 193]. It is scheduled to start operations around the year 2024 [194]. The next major upgrades to the LIGO detectors are known as "A+". They are supposed to be incremental enhancements involving better light squeezing technology, bigger test masses, and improved materials, among other adjustments [45]. Beyond 2025 upgrades are still being planned but may include substantially increased laser power, a change of the laser frequency, and cryogenic operation [45]. One such proposal is named LIGO Voyager. Currently two new ground based detectors are being considered for constructions and are planned to become operational in the mid 2030s. These are the Einstein Telescope (ET) in Europe [46] and Cosmic Explorer (CE) in the United States [47]. ET is planned to be a triangular interferometric detector built underground. Its special shape allows to eliminate any signal from the data and be left with a pure noise channel. CE plans to use an arm length of 40 km.

To eliminate the low frequency limitations of earth bound observatories, two projects are developing space based detectors. The European Space Agency (ESA), the National Aeronautics and Space Administration (NASA), and the LISA Consortium are working on the Laser Interferometer Space Antenna (LISA) [81, 195, 196]. LISA will be a constellation of three satellites on a heliocentric orbit following the earth. It will use time delay interferometry with an arm length of 2.5 Gm and is expected to detect GWs in the sub-Hertz regime [81]. After a widely successful test of the technology known as LISA Pathfinder [197–199], the LISA mission is now scheduled to launch in 2034 [195]. A second proposed space based observatory is the TianQin detector [200]. It, too, will consist of three satellites but its arm length is reduced to 10^5 m and it will orbit the earth.

The final detection scheme I will briefly discuss are Pulsar Timing Arrays (PTA)s [201, 202]. PTAs use the extreme timing accuracy of millisecond pulsars to measure large scale deviations of their arrival times over long periods. A GW in the nHz regime will cause these deviations to be correlated and be recoverable. They potentially allow to detect for instance the GW background or other low frequency GW sources. So far no signal consistent with expected GWs has been recovered with high confidence [203, 204].

3.2.2 Matched Filtering

As discussed in the previous subsection 3.2.1, the output of the detectors is contaminated with large amounts of noise. To achieve optimal sensitivity,

sophisticated data analysis techniques have to be used. This subsection will discuss a technique known as *matched filtering*, that uses a model of the expected signal to maximize the amount of information that can be extracted from the data.

The output of the detector is a time series $d(t)$ that is the sum of the noise $n(t)$ and potentially an additive signal $h(t)$

$$d(t) = n(t) + h(t). \tag{3.46}$$

In the following discussion we will assume that the noise $n(t)$ is stationary and Gaussian with zero mean. For real data this is only an approximation, as the noise characteristics slowly drift over time and glitches are short duration, non-Gaussian noise transients. However, the time scale of the noise drift is slow compared to the usual duration that CBC signals are observable for [205].

If one knows the signal that is hidden in the data, one can compare the data to the signal. To do so, one can convolve the expected signal, which is called the *template*, with the data

$$\begin{aligned}(h * d)(t) &= \int_{-\infty}^{\infty} \mathrm{d}\tau \; h(\tau) d(t-\tau) \\ &= \int_{-\infty}^{\infty} \mathrm{d}\tau \; h(\tau) h(t-\tau) + \int_{-\infty}^{\infty} \mathrm{d}\tau \; h(\tau) n(t-\tau).\end{aligned} \tag{3.47}$$

In this calculation, the first integral in the second line is positive definite when the template and the signal are aligned correctly, i.e. when $t = 0$. Therefore, equation (3.47) will yield larger values in the presence of the signal h than in its absence. This is the core concept behind matched filtering.

Figure 3.7 shows a short time slice of data collected during LIGOs first observing run (O1) and the ASD calculated from the detector data d on a 32 s interval around the shown slice. The ASD shows that the detector noise is dominated by low frequencies, which becomes evident by the slow oscillations with large amplitudes in the left panel of Figure 3.7. However, this also tells us that signals at low frequencies are much more likely to be overshadowed by noise. Therefore, one can downweigh frequencies where the detector is particularly noisy.

In practice this process is called *whitening*, as it attempts to scale the power at each frequency to be 1. Noise which has a flat PSD is known as white noise. This can be achieved by dividing the Fourier transformed data $\tilde{d}(f)$ by the ASD

$$\hat{d}(t) = \int \mathrm{d}f \; \frac{\tilde{d}(f)}{\sqrt{S_n(f)}} e^{i2\pi f t}. \tag{3.48}$$

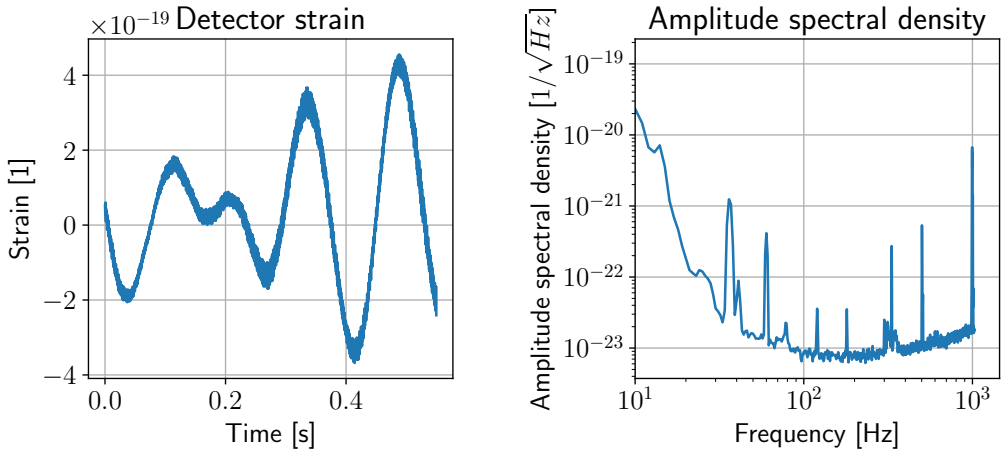


Figure 3.7: A small slice of raw detector data from the Gravitational Wave Open Science Center (GWOSC) (left) and the corresponding ASD (right). The ASD was calculated from 32 s around the shown data slice using Welch's method with a window duration of 1 s and a step size of 0.5 s between the windows [149].

This operation in the time domain is a convolution of the data d with a filter $F^{-1}\left(1/\sqrt{S_n(f)}\right)$, where the function F^{-1} represents the inverse Fourier transform. Due to high peaks in the ASD at some frequencies, the filter has non-negligible power even at times far away from zero. Since the Fourier transform assumes the data to be periodic, the long duration of the filter correlates independent data points and corrupts the data. To avoid this corruption the filter is truncated in the time domain to a finite duration, by applying a window that tapers to zero. The corruption is, thereby, limited to half the truncated duration of the filter in the beginning and end of the whitened data. The corrupted parts of the data are subsequently cropped. By truncating the whitening filter in the time domain, the sharp frequency lines of the ASD are broadened which is usually not a problem for CBC signal detection.

Figure 3.8 shows the same data as Figure 3.7 but with whitening applied to it. The ASD is now almost flat and has a value of ≈ 1 at each frequency. Looking closely, one can start to make out a waveform around 0.4 s from the start of the data. Cropping to remove edge effects was applied outside of the shown window.

Using the whitened data and an accordingly whitened template in (3.47) maximizes the difference between the value obtained when filtering pure noise and noise with the template added into it [68]. It is mathematically proven

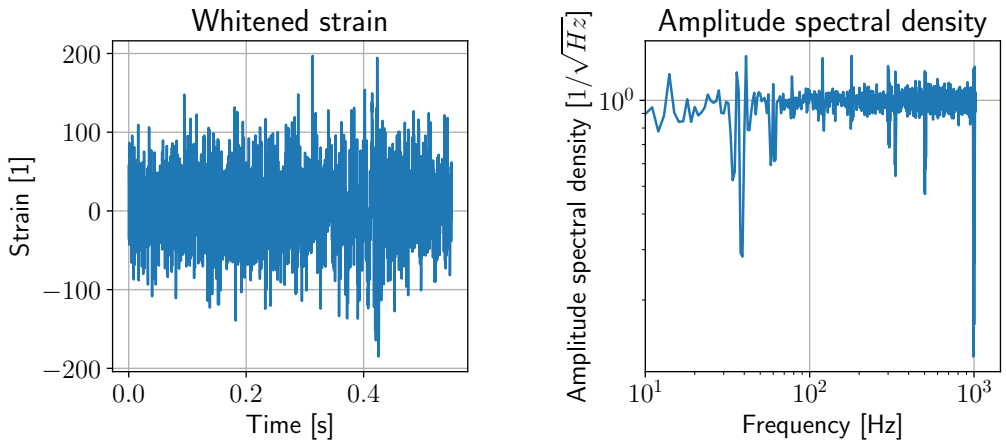


Figure 3.8: The data shown in Figure 3.7 whitened by the associated PSD (left) and the ASD calculated on the whitened data (right). The entire 32 s of data were whitened. Therefore, edge effects are clipped from the shown data slice.

to be the optimal filter for stationary Gaussian noise [30]. Since the data and template are transformed to the Fourier domain to whiten them and since the convolution operation can be expressed by a multiplication in the Fourier domain, the matched filter is given by [60, 68]

$$\rho(t) = \frac{\langle h|d\rangle}{\sqrt{\langle h|h\rangle}}, \tag{3.49}$$

where

$$\langle a|b\rangle(t) = 4\mathrm{Re}\left[\int_0^\infty \mathrm{d}f\ \frac{\tilde{a}^*(f)\tilde{b}(f)}{S_n(f)} e^{i2\pi ft}\right]. \tag{3.50}$$

The factor $\sqrt{\langle h|h\rangle}$ is used for normalization and the factor 4 comes from using the one-sided PSD as well as taking the integral of the symmetric argument from 0 instead of $-\infty$. The function $\mathrm{Re}\,[\cdot]$ extracts the real part of a complex number. The function $\rho(t)$ is the *signal-to-noise ratio* (SNR). When the SNR exceeds a threshold, which is determined by search configurations, the search has potentially identified the signal.

So far it was assumed that the signal in the data is known. In reality this assumption does not hold, as we do not even know if a signal is present and if one is present, what kind of system emitted it. However, matched filtering is highly sensitive to the phase of the signal. This means that when the phases of the template and the actual signal deviate by even a moderate amount, the SNR drops significantly. For this reason, one usually constructs

a *template bank*, that covers a specified region of possible parameters. It is constructed by requiring that the *match* between two neighboring templates does not fall below a given limit. The match is the inner product (3.50) of the two signals, maximized over the phase and coalescence time, and normalized by the norm induced by the inner product (3.50) of both templates. Usually matches ≥ 0.95 are required [14, 206].

The data has to be filtered against every template in the bank. Therefore, the computational cost scales linearly with the number of templates. However, the number of templates scales exponentially with the number of degrees of freedom in the system. For this reason, it is desirable to reduce the dimensions of the parameter space as much as possible. To do so, searches make a series of assumptions and optimizations. First, search template banks are usually constructed using only the dominant 22-mode. This means that equation (3.40) essentially reduces to

$$H = h_+ + ih_\times = A(\tau, r, \iota, \Phi_0, \kappa)e^{-i\Phi(\tau,\kappa)}. \tag{3.51}$$

Second, by using

$$h_+ = \frac{H + H^*}{2}, \qquad h_\times = \frac{H - H^*}{2i} \tag{3.52}$$

in (3.43) one finds that the measured strain in the detector is of the form [30]

$$h = \mathrm{Re}\left[\frac{A(\tau,\kappa)}{D_{\mathrm{eff}}(r,\theta,\phi,\psi,\iota)} \exp\left(-i(2\Delta\Phi(\theta,\phi,\psi,\iota,\Phi_0) + 2\Phi(\tau,\kappa))\right)\right]. \tag{3.53}$$

D_{eff} is known as the effective distance and quotes the distance at which the same source could be seen with the same amplitude, assuming an optimal orientation and a location directly overhead the detector. An explicit expression for D_{eff} can be found in [30]. κ represents all source internal parameters, i.e. the masses, spins, and tidal deformabilities. From (3.53) we find that the source location and orientation with respect to the detector introduce a time independent phase shift and scale the amplitude. The amplitude scaling can be combined with the distance r to form a single parameter; the effective distance. Observing a source in one location has the same amplitude evolution as observing the same source at the optimal location and orientation, just at a farther distance. The phase shift on the other hand can be combined with the coalescence phase, to form a second effective parameter. So instead of being concerned with the 4 parameters θ, ϕ, ψ, and ι, all of them can be absorbed into the distance r and coalescence phase Φ_0. The distance, on the other hand, sets a normalized scale for the SNR and, therefore, does not need to be included in the template bank [30]. The coalescence phase can also be eliminated from the template bank, by maximizing the SNR over it.

If the projection onto the real axis in (3.50) is removed, the SNR time series becomes complex. An overall phase shift in the template rotates the SNR-vector in the complex plane, but does not change its length. So by taking the absolute value of the SNR time series, one maximizes its value over the coalescence phase, irrespective of the phase value. This is equivalent to using the total SNR by combining the SNR of filtering with the template h and its phase shifted counterpart ih [30, 60, 207]

$$\rho_{\Phi_{0,\max}}(t) = \sqrt{\frac{\langle h|d\rangle^2 + \langle ih|d\rangle^2}{\langle h|h\rangle}}. \tag{3.54}$$

Third, template banks usually assume non-precessing systems [14, 15]. This reduces the dimensionality of the parameter space even further, as the original 6 spin parameters are reduced to a total of 2 values that need to be covered. Finally, the inverse Fourier transform in (3.50) is equivalent to doing a convolution of the data with a template. So it efficiently changes the coalescence time t_0 of the template and produces the SNR value for all possible values. All these optimizations lead to a four dimensional space that needs to be covered by the template bank; m_1, m_2, χ_1^3, χ_2^3.

Current standard searches are limited to systems with aligned spin and a mass range from $\approx 1\,\mathrm{M}_\odot$ to $\approx 500\,\mathrm{M}_\odot$, depending on the implementation [14, 206]. Lower masses require denser template banks, as the systems are within the sensitive band of the detectors for a larger number of cycles. This allows small phase errors to accumulate to the point where match requirements are exceeded [208]. The high-mass region is rather limited by the short duration, which makes it difficult to distinguish astrophysical signals from glitches [209].

Although using matched filtering alone is sufficient to detect GWs, it is often stated that the first detected GW is visible in the data with the naked eye. While one can barely make something out in Figure 3.8, we can make use of the fact that we know the frequencies of the signal. Figure 3.9 shows the data after whitening and applying an additional bandpass filter for the range 20 Hz to 256 Hz. The signal is now clearly visible in the time domain data.

Since matched filtering is mathematically proven to be the optimal descriminator between pure noise and noise plus signal [30], it is used as a target for the deep learning detection algorithms discussed in this thesis.

This section discussed detection algorithms based on matched filtering. Due to heavy optimization, these algorithms are not designed to make statements about many of the interesting parameters of the source. The task of extracting all parameters that influence the waveform from the data is called

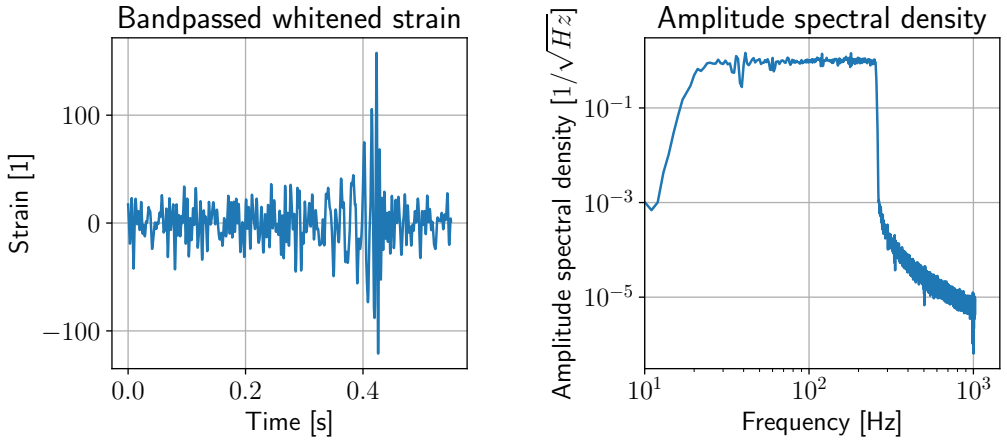


Figure 3.9: The whitened data from Figure 3.8 with a highpass filter of 20 Hz and a lowpass filter of 256 Hz applied (left). The corresponding ASD is shown on the right. The signal is now clearly visible. The data for all of these figures is the Hanford data around GW150914 [13].

parameter estimation. These algorithms explore the likelihood surface and allow to give estimates of the parameters of the source and quantify them with error bars. Since the likelihood usually has to be estimated for millions of points, these algorithms are computationally very expensive and can take days or even weeks to converge. For a deeper explanation see for instance [117, 125, 210].

3.2.3 Search Algorithm and Significance of Detections

The matched filter of the previous subsection produces a SNR time series for every template. These are numerical values where larger values usually imply a greater certainty that a GW is present in the data. Values of this kind are known as *ranking statistic*. The question a detection pipeline has to answer is: What value of the ranking statistic is required for us to be confident that we have detected a GW? This subsection will go into how this question is answered. I will follow the order of steps used by the PyCBC offline analysis [60] but note that other modeled pipelines conceptually work the same way and only differ in details [59, 161].

The first step a search pipeline has to perform is to select the data that should be analyzed. The noise characteristics of the detectors change over time [125]. During some periods, the data quality drops to a level where analysis would lead to excessive rates of false positive detections. For this reason, these parts of the data are identified by high-level quality checks [211]

and excised from the analysis data. Easily identifiable loud glitches are also cropped from the data [211]. The overall shape of the stationary component of the noise is estimated by calculating the average PSD for each detector and an appropriate template bank is constructed [60].

Afterward, the data from all detectors are filtered against the template bank to produce SNR time series. From there, candidate events are identified by applying a simple threshold to the SNR time series. The resulting points are subsequently clustered for each template, as especially loud signals usually exceed the threshold for multiple samples. The cluster window duration is a free parameter, which has to be small compared to the expected time between signals and subsequent glitches.

If the noise in the detector were stationary and purely Gaussian, one could take the SNR values of the candidate events as a ranking statistic. However, as discussed in subsection 3.2.1 the noise is frequently contaminated with glitches. Those that have not been removed from the data can lead to large SNR values, even when the underlying process that produced the transient has little to no resemblance to the signal one is searching for. Therefore, the SNR is usually augmented with additional statistics to form the ranking statistic. The goal of these additional statistics is to reduce the value of the ranking statistic for glitches, i.e. down weighting their influence.

The most common adaptation is to to check the data for consistency with the template. PyCBC and MBTA use a χ^2-test that checks if the evolution of the potential signal matches the expected signal evolution in different time- and frequency-bands [60, 161, 189]. GstLAL uses a ξ^2-test that checks the evolution of the SNR time series against the expected evolution [59]. More recently, efforts were made to include the short term fluctuations of the PSD into the ranking statistic [150, 212]. This reduces the impact of a less stationary detector on the trigger rate.

Once the triggers and their corresponding ranking statistics have been determined for each detector individually, they are checked for coincidences between multiple observatories. A signal from an astrophysical source has to be present in all detectors and arrival times may not differ by more than the time of flight difference between the detectors. Additionally, coincident triggers must be generated from the same template and the amplitude and phase evolution must be consistent with the time delay between the detectors [163]. These criteria already eliminate many false detections caused by glitches or other noise processes. For surviving triggers, the single detector ranking statistics are combined into a coincident ranking statistic. In recent works the coincident ranking statistic was further altered by incorporating information on the expected noise and signal trigger rates for different source parameters [163].

As a final step, the search algorithm has to relate the numerical value of the coincident ranking statistic to a measure of how confident we are to have detected a GW. In other words, the analysis must check how often a false detection is made at a given ranking statistic value. The number of false positive detections per unit time is known as the *false-alarm rate* (FAR). It is estimated by analyzing constructed background data. The background data is constructed by time shifting the single detector triggers of one detector by a duration longer than the time of flight difference between the detectors. This way, any coincident trigger cannot be of astrophysical origin. By applying multiple different time shifts, the theoretical amount of data which can only contain false positives can be extended to millions of years [60]. To determine the FAR at a ranking statistic, the number of false positives with a coincident ranking statistic larger than the threshold are counted and divided by the theoretical duration of the analyzed data:

$$\mathcal{F}_\rho = \frac{N_\rho}{T}. \tag{3.55}$$

Here $\mathcal{F}_\rho$ is the FAR for the coincident ranking statistic ρ, N_ρ is the number of false positives in the background with a ranking statistic $\geq \rho$, and T is the duration of the analyzed background.

This thesis is mainly concerned with the comparison of machine learning based search algorithms against matched-filter based searches. For this comparison, one can analyze a simulated population of sources with both algorithms and study how many signals are recovered by either search. However, for a fair comparison we want to assess the sensitivities at the same useful astrophysical FAR. Furthermore, to allow for an objective comparison between different works, the sensitivity must be normalized to the population. In the extreme case, the number of detected sources can always be driven to zero by choosing a population of signals injected into the noise where all sources are excessively far away. For these reasons, all works discussed in this thesis estimate the sensitive distance of the search for sensible populations. I will briefly introduce it here.

Let $\Phi(\vec{x}, \lambda)$ be the probability distribution that describes the injected population, where $\vec{x}$ is the location in space and λ are the remaining source parameters. Let, further, $\epsilon(\vec{x}, \lambda; \mathcal{F})$ be the fraction of injections being recovered by the search with parameters $\vec{x}$ and λ with FAR $\leq \mathcal{F}$. Then the expectation value of the volume from which sources are detected is given by [60]

$$\langle V(\mathcal{F}) \rangle = \int \mathrm{d}^3x \, \mathrm{d}\lambda \, \frac{\mathrm{d}^3 V(\vec{x})}{\mathrm{d}x^3} \Phi(\vec{x}, \lambda) \epsilon(\vec{x}, \lambda; \mathcal{F}), \tag{3.56}$$

where $\frac{\mathrm{d}^3 V(\vec{x})}{\mathrm{d}x^3}$ is the differential volume element. This volume is usually measured empirically, by applying the search to data with many known injections drawn from the distribution Φ. The integral can then be estimated by Monte-Carlo integration. If the prior is uniform in volume, the sensitive volume is proportional to counting the recovered injections. This counting statistic then needs to be normalized to the prior volume and the number of injections. If the prior volume is a sphere with radius r_{max} where injections are distributed uniformly in volume, the Monte-Carlo approximation of the sensitive volume is given by [60]

$$\langle V(\mathcal{F}) \rangle \approx V(r_{\mathrm{max}}) \frac{N_{I,\mathcal{F}}}{N_I}, \tag{3.57}$$

where $N_{I,\mathcal{F}}$ is the number of injections recovered from the data with FAR $\leq \mathcal{F}$, N_I is the total number of injections in the data, and $V(r_{\mathrm{max}})$ is the volume of a sphere with radius r_{max}. When a different distribution is used to sample the injections, the formula has to be reweighted or the integral has to be computed. The sensitive distance is defined as the radius of a sphere with volume $\langle V(\mathcal{F}) \rangle$. As such, it is the average distance from which the search can detect sources and not an upper limit.

3.3 Deep Learning

Artificial Intelligence (AI), Machine Learning (ML), and deep learning are terms often used interchangeably and to describe a wide range of topics. The beginning of this section tries to clarify the meaning of these terms in the context of this work before discussing the topic of deep learning in more detail.

AI is the broadest of the three topics and encompasses the others [213]. While it lacks an agreed-upon definition, the term broadly describes the research dedicated to creating machines that enact some form of intelligence [214]. However, this only shifts the burden of definition to the term "intelligence". In his excellent book on the history of AI Nilsson describes intelligence as the "quality that enables an entity to function appropriately and with foresight in its environment" [214]. Taking these definitions as a basis, AI research tries to create algorithms or machines that make decisions based on their environment to maximize their chance of success to achieve some goal.

ML is part of the research field on AI and aims to create algorithms that extract general features from example data. These learned features are then to be used to make predictions about samples that were not part of

the example data. It specifically requires algorithms to improve based on past experiences [213]. Usually, a general purpose framework is adapted to specific problems only through a prior training phase, where the algorithm is exposed to example data. This means that the learning algorithm does not have to be changed to adapt to different data.

An example for a class of AI algorithms that do not fall under the category of ML algorithms are knowledge based approaches. These try to encode knowledge in a formal language and infer decisions by querying the resulting database. [213]. The Cyc project is one of the most well known approaches to a knowledge based AI and makes use of the CycL language [215]. However, creating a database that is both suitably large and complex is very labor intensive and often more fragile than utilizing ML algorithms. For these reasons ML is currently the most widespread approach to AI.

Deep learning refers to ML algorithms which utilize a specific kind of framework; deep neural networks (NNs). While the concept of NNs will be introduced in detail below, in short they are directed graphs that apply simple functions at every node, where the weights of the edges are optimized during the training phase based on the example data. These graphs can be structured into layers and a NN is called deep, when the network is composed of many subsequent layers [213].

Besides deep learning, there are many other kinds of ML algorithms. Some examples include random forests [216], support vector machines (SVMs) [217], and Gaussian process regression. Even simple regression algorithms, like linear regression, can be seen as a kind of ML algorithms [218]. For a short overview of these different types see for instance chapter 1 and for a more detailed discussion chapters 5 to 7 of [218].

All ML algorithms can broadly be classified into three categories based on the *training* mechanism, where training is the process of improving the algorithm based on example data. These three classes are *supervised learning*, *unsupervised learning*, and *reinforcement learning* [213, 219]. In supervised learning tasks the desired output y – often called *label* or *target* – for example data x is known [218] and the algorithm tries to approximate the conditional probability distribution $p(y \mid x)$ [213]. Regression algorithms are a good example for supervised learning. In unsupervised learning only the example data x is known and the ML algorithm is tasked with finding the underlying probability distribution $p(x)$ [213]. Many clustering algorithms are a prime example for unsupervised learning. Reinforcement learning is completely detached from the previous two classes. Here the algorithm observes the state of its environment and has to decide on an action it wants to take. It is then rewarded or penalized based on the action it has chosen [218, 220]. The algorithm learns by trial and error, without human intervention. This kind of

learning is especially popular in the field of robotics and computer games [51, 213, 221–223]. The above mentioned classifications of algorithms are by no means rigid. A single algorithm may also be trained by a combination of the above mentioned strategies. There are also other means of classifying ML algorithms into different categories [218].

Today ML algorithms are used anywhere from recommendation systems in the entertainment industry [224] to advanced medical diagnostics [225]. Especially NNs have gained a lot of traction in recent years and improved state-of-the-art performance for ML algorithms for a wide variety of tasks such as image recognition [49, 226, 227], autonomous driving [222], playing board- [228] and computer-games [51], speech recognition [229], or audio synthesis [230]. ML algorithms have also been applied in many scientific fields [231], some of which are the prediction of protein structure used in pharmaceutical studies [232, 233], improvements to material composition and synthesis [234], and event reconstruction at the Large Hadron Collider [235]. Machine learning has also been explored as an option for many tasks in GW data analysis and GW astronomy and an overview is given in section 3.4.

This work considers only neural networks as a means of machine learning. For this reason the basic concepts as well as a few advanced techniques will be briefly introduced in the following subsections. For a more thorough introduction I recommend [213] for a deep dive into the topic, [218] for a more applied approach, and [236] for a quick and easy to understand overview.

3.3.1 Neural Networks

The start of the research on neural networks based on the theory of computation and mathematics can be traced back to a study conducted by McCulloch and Pitts in 1943 [237]. They tried to come up with a model description of the human brain, where they modeled neurons by utilizing propositional logic. By making the simplifying assumptions that neurons have an arbitrary number of binary inputs, a single binary output that is one (i.e. "fires") only if a pre-defined number of inputs is active, and that individual inputs may prevent any output, they were able to build all logic gates and combine them to do complex computations [238]. While the original work exclusively used propositional logic, their idea of the neuron can suggestively be expressed by the functional form

$$n : \{0,1\}^m \times \{-1,1\}^m \times \mathbb{Z} \to \{0,1\}\,;\ (\vec{x},\vec{w},b) \mapsto H(\vec{x}\cdot\vec{w}+b), \qquad (3.58)$$

where $\vec{x}$ are the inputs to the neuron, $\vec{w}$ are known as the weights, b is known as the bias, ans H is the Heaviside function. See Figure 3.10 for a

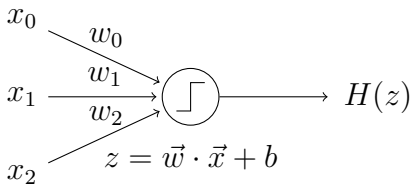


Figure 3.10: Depiction of the neuron defined in (3.58). The inputs x_0, x_1, x_2 are either 0 or 1. The weights w_0, w_1, w_2 are either -1 or 1. The bias b is a whole number. H is the Heaviside function.

depiction. This definition of the neuron is not equivalent to the idea proposed by McCulloch and Pitts but conveys the central findings of their study and leads to a more coherent picture of subsequent developments.

To form basic logic gates, values for the parameters $\vec{w}$ and b have to be chosen. The "and"-operation $A \wedge B$ is obtained by setting $\vec{w} = (1,1)^T,\ b = -1$, whereas the "not"-operation $\neg A$ can be represented by $w = -1,\ b = 1$. These simple gates can then be connected to form complex computational graphs and compute any computable function. However, notice that the parameters of these networks have to be hand-picked.

In 1957 Rosenblatt introduced the Perceptron [239]. Instead of restricting individual neurons to have binary inputs and discrete weights, he allowed all arguments to be real numbers

$$n : \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R} \to \{0,1\};\ (\vec{x}, \vec{w}, b) \mapsto H(\vec{w} \cdot \vec{x} + b). \tag{3.59}$$

More importantly, however, he proposed an algorithm based on the Hebbian principal that allowed the parameters of the network to be optimized from example data. The Hebbian principal states that directly connected neurons that often fire together form stronger bonds; "Neurons that fire together, wire together." [218] The Perceptron is a collection of these neurons, where every neuron is connected to all inputs. The number of binary outputs, therefore, is equivalent to the number of neurons in the Perceptron. The left panel of Figure 3.11 shows an example of a perceptron.

During training the output of the Perceptron is compared to the target value. If the predicted output of any neuron does not match the target value, the weights connected to the inputs that would have pushed the output to its correct state are increased [218]. The actual change applied to the weight connecting input i with output neuron j is given by

$$dw_{ij} = \eta(y_j - \hat{y}_j)x_i. \tag{3.60}$$

Here y_j is the j-th target output, $\hat{y}_j$ is the corresponding predicted output, x_i is the i-th input, and η is a special parameter called the *learning rate* [218].

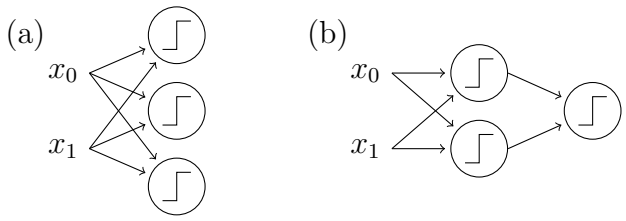


Figure 3.11: An example of a perceptron (a) and a MLP (b). The arrows indicate connections between the input vector $\vec{x} = (x_0, x_1)^T$ and the individual neurons, where each connection is assign a weight. The step inside each neuron represent the Heaviside function that is applied to the weighted sum of the inputs.

Notice that in a Perceptron the neurons are only connected to the inputs and not other neurons.

Although the Perceptron had great success at the time, it was quickly shown that the design had its limitations. One of the greatest criticisms was the inability to represent the XOR-gate, i.e. there exists no Perceptron such that $n((0,0), \vec{w}, b) = 0$, $n((0,1), \vec{w}, b) = 1$, $n((1,0), \vec{w}, b) = 1$, and $n((1,1), \vec{w}, b) = 0$ [213]. To resolve this issue, the outputs of a first set of neurons have to be used as input to a subsequent neuron. Connecting multiple neurons in succession was hence called a multi-layer Perceptron (MLP) and the name is still sometimes used today to refer to deep NNs. Panel (b) of Figure 3.11 shows a simple MLP that can represent an XOR-gate, when the weights and biases are chosen correctly.

The MLP demonstrates that a greater depth of the network, i.e. a greater number of neurons between the input and output, can allow the network to represent more complex functions than an individual neuron is capable of. This behavior should not surprise given the early work of McCulloch and Pitts [237], as modern computers fundamentally are a complex network of simple logic gates. The concept of greater depth leading to the ability to represent more complex concepts is also true for modern NNs [240], hence the term "deep learning".

The problem with deeper networks is the training process. The original algorithm proposed by Rosenblatt becomes ineffective and the weights and biases for deeper networks could not be set efficiently. A solution to the problem was only found in 1986 [218], when Rumelhart, Hinton, and Williams introduced the backpropagation algorithm [241]. I will discuss this algorithm in more detail in subsection 3.3.2, but in short it is an extension of the updating procedure given in equation (3.60) that utilizes the chain-rule to efficiently calculate derivatives with respect to all weights and biases. The

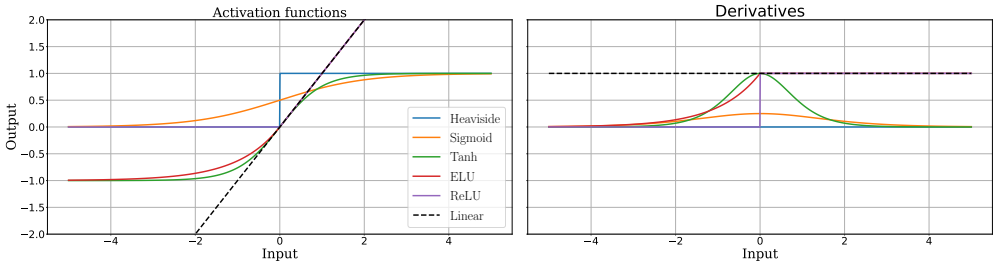


Figure 3.12: A selection of different commonly used activation functions on the left and their derivatives on the right.

resulting gradient can then be used to update all parameters of the network.

The use of the backpropagation algorithm required to switch out the Heaviside function previously used in neurons to a function that has a non-zero derivative. Otherwise the gradient would vanish everywhere and no weight update could be applied. Today this function is called the *activation function*

$$a : \mathbb{R} \to \mathbb{R};\ z \mapsto a(z). \tag{3.61}$$

Furthermore, if the resulting NN is supposed to be able to represent non-linear functions, the activation function has to be non-linear as well. The original paper [241] used the sigmoid function

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{3.62}$$

Today popular choices for the activation function include the hyperbolic tan function, the Exponential Linear Unit (ELU) [242], and the Rectified Linear Unit (ReLU) [243, 244]. See Figure 3.12 for an overview of these different activations and their derivatives.

With this, today's definition of a neuron is given by

$$n : \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R};\ (\vec{x}, \vec{w}, b) \mapsto a(\vec{w} \cdot \vec{x} + b). \tag{3.63}$$

A diagram of a modern neuron is shown in Figure 3.13. In general a NN consists of arbitrary connections of these kinds of neurons, that allow for the representation of more complex functions.

However, when defining a NN it is often convenient to take a more structured approach. The individual neurons can be combined into layers. Each neuron in the layer is then required to have the same inputs and the same activation function. As such a layer of neurons is a function

$$\mathcal{L} : \mathbb{R}^m \times \mathbb{R}^{l \times m} \times \mathbb{R}^l \to \mathbb{R}^l;\ \left(\vec{x}, W, \vec{b}\right) \mapsto a\left(W \cdot \vec{x} + \vec{b}\right), \tag{3.64}$$

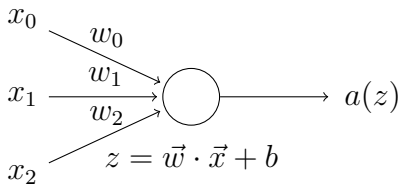


Figure 3.13: Depiction of a single neuron. It takes a three dimensional input $\vec{x} = (x_0, x_1, x_2)^T$ and has four parameters; the weights $\vec{w} = (w_0, w_1, w_2)^T$ and the bias b. Its output is the activation a applied to $z = \vec{w} \cdot \vec{x} + b$.

where W is the weight-matrix, consisting of the weights of each neuron stacked in the rows, and $\vec{b}$ summarizing the biases of the different neurons. The activation function a is understood to operate component-wise on its argument.

To form a NN, these layers can then be connected. In its most simple form, a NN is a set of nested layers:

$$\mathcal{N}(\vec{x}, \theta) = \mathcal{L}_n(\mathcal{L}_{n-1}(\dots \mathcal{L}_1(\vec{x}, \theta_1), \dots \theta_{n-1}), \theta_n). \tag{3.65}$$

The values $\theta_i = \left(W_i, \vec{b}_i\right)$ are collectively called the parameters of the layer and $\theta = (\theta_1, \dots, \theta_n)$ are the parameters of the network.

Equation (3.65) defines a *feed forward neural network*. Information flows only in one direction, from the input x, through the intermediate layers $\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_{n-1}$, to the output $\mathcal{L}_n$. No feedback connections are allowed and the resulting graph is acyclic [213]. When the graph does contain cycles, the resulting network is known as a *recurrent neural network* (RNN) [245, 246]. These have traditionally had many applications in different kinds of natural language processing (NLP) [247] and are viewed as resembling biological brains more closely. They were even considered as a central idea in the original works by McCulloch and Pitts [237, 238]. However, feed forward NNs tend to be easier to optimize than RNNs and variants of feed forward NNs have started to perform better than state-of-the-art RNNs [248, 249]. For these reasons my works have exclusively considered feed forward NNs. I point the interested reader to chapter 10 of [213] and chapters 15 and 16 of [218] for more detail. When talking about NNs from here on out I will exclusively refer to feed forward NNs unless otherwise stated.

A NN can generally be structured into three parts, as illustrated in Figure 3.14. The first part is the *input layer* that passes its inputs unchanged. It has not been stated explicitly in the discussion above and is represented by the input vector $\vec{x}$ in equation (3.65). Its shape is, therefore, dictated by the example data. The final layer $\mathcal{L}_n$ is the *output layer* and its shape is

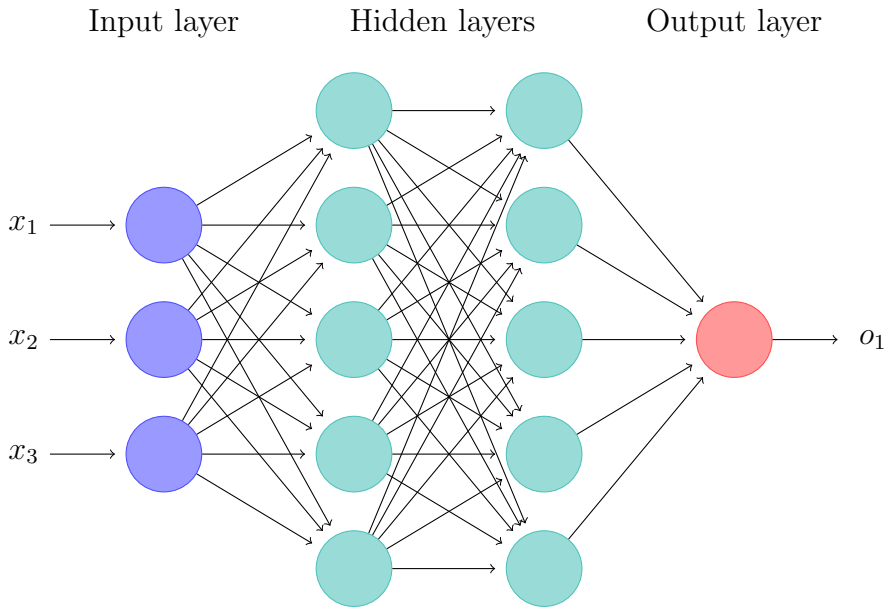


Figure 3.14: A fully connected NN with two hidden layers of equal size (green). It has an input layer with three neurons (blue) that accepts the input-vector $\vec{x} = (x_1, x_2, x_3)^T$. Its output layer has a single neuron (red) and outputs o_1.

determined by the desired output of the network. All intermediate layers $\mathcal{L}_1$ to $\mathcal{L}_{n-1}$ in equation (3.65) are called *hidden layers*. The name stems from the fact that the use of these layers is not dictated by the data but fully determined by the learning algorithm [213]. The state is basically hidden to the outside observer, i.e. one can generally not determine the structure of the NN just from considering outputs on example data.

The statement that the design of the hidden layers cannot be obtained from example evaluations of the network follows directly from the *universal approximation theorem*. The theorem states that a feed forward NN with a linear output layer and at least one hidden layer of sufficient size with a non-linear activation function[3] can approximate any finite dimensional Borel measurable function to arbitrary precision [213, 250, 251]. In practical terms this means that one can always find a NN $\mathcal{N}(\vec{x}, \theta)$ with appropriate parameters θ that can approximate a function $f(\vec{x})$. Since only a single hidden layer is required, this also means that any NN $\widehat{\mathcal{N}}(\vec{x}, \hat{\theta})$ can be approximated by a NN $\mathcal{N}(\vec{x}, \theta)$ with just a single hidden layer.

[3]The activation function has to also be "squashing". The common sigmoid activation for example fulfills all required conditions for the universal approximation theorem [213].

One caveat of the universal approximation theorem is the size of the hidden layer. It is generally not possible to know how large it has to be to approximate any function and greater depth usually leads to better performance. Furthermore, while it is true that effectively most functions can be approximated by some NN, there is no NN that is the best for all tasks. This statement can be made even sharper: All ML algorithms perform equally well when their performance over all possible data generating distributions is averaged [213]. It is known as the *no free lunch theorem* and was proven by Wolpert in 1996 [252]. However, it does not state that a particular algorithm cannot outperform others for any particular task. It simply means that it is impossible to design an algorithm that is optimal for all tasks.

To conclude this subsection I want to point out that equations (3.63), (3.64), and (3.65) have only considered vector-valued inputs and functions. This is not required but simplifies notation. In subsection 3.3.3 layers with different inputs and outputs will be introduced.

3.3.2 Training Neural Networks

The task of training a NN $\mathcal{N}$ is to use example data in order to find a set of parameters θ such that $\mathcal{N}(\cdot, \theta)$ suitably approximates a target function f. This simple statement already entails the three critical components that are required for training. First we need data sampled from the input domain of f. If the algorithm is supposed to be trained in a supervised manner, the corresponding outputs $f(\vec{x})$ are also required. Second, we need to quantify what it means for $\mathcal{N}$ to be a good approximation to f, i.e. we require a performance measure or error-function. Finally, we require a procedure to find parameters θ that minimize the error between $\mathcal{N}$ and f [213]. This subsection will discuss these three aspects and will also touch on a few pitfalls that one may encounter.

The data used to train the NN is called the *training set*. It is a collection of discrete examples which are used to optimize the parameters θ. As my work has focused solely on supervised learning tasks, for the remainder of this section I will assume that these examples contain both the input as well as the label. While a lot of what is said in this section remains true also for unsupervised learning algorithms, I refer the reader to [213] for more detail.

While the training set is used to optimize the parameters of the NN, we are only indirectly interested in its performance on this set. If we cared only about the predictions of the network on the training set we could simply create a lookup-table of the examples and would obtain optimal performance. Instead of remembering only the training set, we want the NN to learn the underlying function f. To measure how good this approximation is, we

have to measure the performance of the NN on a second set that is sampled independently from the training set. This second set is called the *test set* and the error obtained on it is known as the *generalization error* or simply *test error* [213].

To be able to make statistical statements based on mathematical proofs about the ability to generalize learned features from the training set to the test set, the test set should be sampled not only independently from the training set but also be distributed identically to the training set. Collectively these conditions are called the *i.i.d conditions*. In practice it is most important that the test set is as close to the real application scenario as possible. In my work the training set often consists of discrete samples, while the test set is a continuous time series. This means that both sets are fundamentally not distributed identically. However, both sets are generated with very similar assumptions and thus it is plausible that the NN will be able to generalize. In fact a core result of our studies is that we are able to optimize our NNs on the discrete training sets and obtain low, but marginally larger, generalization errors on the continuous test sets.

In order to measure the generalization error a performance measure is required. This performance measure is problem specific but may not always be computationally efficient to calculate or even tractable. Furthermore, for the optimization algorithms that will be discussed below it is vital that the error function is differentiable. Therefore, instead of optimizing the true performance measure one often optimizes a substitute error function instead, in the hope that the two are correlated [213]. This error function in deep learning is known as the *loss* or *cost-function* and the name comes from the fact that it penalizes errors. If instead it rewards good performance, it is called the *fitness function* [218]. In deep learning the use of a loss function is more prominent. Mathematically, the loss is a function

$$L : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R};\ (y, \hat{y}) \mapsto L(y, \hat{y}), \tag{3.66}$$

where y is the label and $\hat{y}$ is the predicted output of the network. The lower the loss-values the closer the network is supposed to approximate the function. When the loss is calculated for the entire training or test set it is averaged over all examples.

For certain purposes the loss may also depend directly on other variables, most notably the parameters of the network. We ignore this case here as it is not relevant to my work, but note that including the parameters can force certain behavior of the network. For example, one can push the weights of the network to be numerically small, which can have desirable properties [213].

While the loss can in general be freely chosen to suite the task that should be solved, it is often at least partially inspired by a maximum likelihood prin-

ciple. For instance, if we assume the labels y of the input data x to be drawn from a normal distribution, it is sensible to train the network to estimate the mean of that distribution. Assuming further that the variance σ^2 of the underlying distribution $y = f(x)$ is fixed, the network that approximates the data generating process best will be the one that maximizes the likelihood

$$\begin{aligned}\sum_{i=1}^{M} \log\left[p(y_i \mid x_i, \theta)\right] &= \sum_{i=1}^{M} \log\left[N\left(y_i, \hat{y}_i, \sigma^2\right)\right] \\ &= -M \log\left[\sigma\right] - \frac{M}{2}\log\left[2\pi\right] - \frac{1}{2\sigma^2}\sum_{i=1}^{M} ||\hat{y}_i - y_i||^2, \quad (3.67)\end{aligned}$$

where $\hat{y}_i = \mathcal{N}(x_i, \theta)$, $N(y_i, \hat{y}_i, \sigma^2)$ is the normal distribution with mean $\hat{y}_i$ and variance σ^2 evaluated at the point y_i, and (x_i, y_i) are the examples from the training set. Maximizing equation (3.67) by tuning only the parameters θ is equivalent to minimizing

$$\text{MSE} = \frac{1}{M}\sum_{i=1}^{M} ||\hat{y}_i - y_i||^2, \quad (3.68)$$

which is known as the *mean squared error* (MSE) [213]. This function is commonly used as a loss in regression tasks and I have utilized it in many of my works.

Another common task in machine learning is binary classification. Defining a loss function inspired by a maximum likelihood principle in that case is not trivial, as the conditional probability, too, is binary and as such is not continuous. In this case one often uses a sigmoid activation function (see equation (3.62)) on the output layer of the network and interprets the output as a probability. In this case the negative log-likelihood divided by M is given by [213]

$$\text{Binary Cross-Entropy} = -\frac{1}{M}\sum_{i=1}^{M}\left(y_i \log\left[\hat{y}_i\right] + (1 - y_i)\log\left[1 - \hat{y}_i\right]\right), \quad (3.69)$$

which is known as the *cross-entropy loss* function or *binary cross-entropy loss* function [218]. In case the output layer has n neurons and the labels are one-hot encoded, the loss can be generalized to the *categorical cross-entropy*

$$\text{Categorical Cross-Entropy} = -\frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{n} y_{i,j}\log\left[\hat{y}_{i,j}\right]. \quad (3.70)$$

A one-hot encoded vector has a single entry with the value of 1, whereas the remaining entries are 0. Alongside the MSE, the categorical cross-entropy was the most used loss function in my works.

The third component required to train a NN is an *optimizer* that adjusts the parameters θ. The goal of the optimizer is to minimize the loss on the training set. NNs as introduced in subsection 3.3.1 are generally highly complex, non-linear functions that require non-convex optimization. Therefore, it is usually not possible to analytically find the optimal parameters for the network. However, if we consider a fixed example or set of examples from the training set, the loss indirectly becomes a function that depends only on the parameters of the network θ. Minima of that function are extremal points and have a vanishing gradient. Furthermore, the gradient points in the direction of larger values and so one lowers the loss by taking a sufficiently small step in the opposite direction of the gradient (see Figure 3.15). This concept is known as *gradient descent* and its usage requires the existence of a derivative of the loss function [213]. With

$$J(\theta) = \frac{1}{M} \sum_{i=1}^{M} L(y_i, \hat{y}_i = \mathcal{N}(x_i, \theta)) \tag{3.71}$$

the parameters of the network are updated as [213]

$$\theta \to \theta - \eta \nabla_\theta J(\theta), \tag{3.72}$$

where η is the learning rate that controls how far the parameters are moved along the error surface of the loss function in the opposite direction of the gradient. Small values of η lead to very accurate optimization but potentially require many steps to reach an optimal value. Large values on the other hand lead to quick improvements but risks overshooting minima [213]. Therefore, the learning rate usually has to be adjusted by trial and error to be as large as possible while still reaching low values of the loss.

The gradient in equation (3.72) is calculated on the entire training set. To take multiple steps, the calculations have to be repeated on the entire training set. One such pass on the training set is called an *epoch*. Since the entire training set has to be used to calculate a single step, gradient descent scales linearly with the size of that set. As it can be shown that the generalization error falls when the size of the training set increases, NNs are usually trained on very large sets [213]. This can make the use of full gradient descent computationally prohibitive.

A cheaper alternative to gradient descent is *stochastic gradient descent* (SGD). Instead of using the entire training set, a random subset, known as a

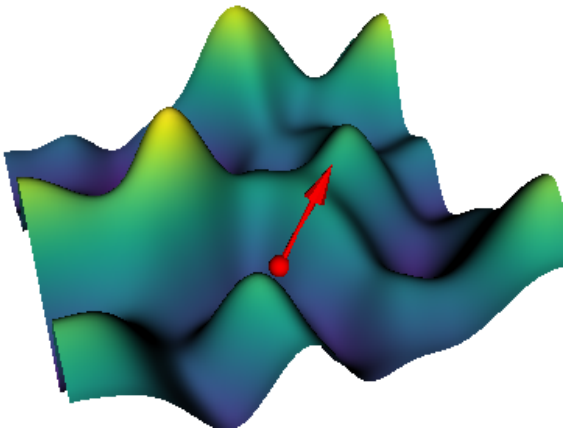


Figure 3.15: Visualization of the gradient of a cost function for a two dimensional parameter space. The height on the vertical axis and the color represent the value of the cost function. The gradient is evaluated at the location of the red dot and is represented by the red arrow. It points in the direction of the largest positive incline. Moving a small step in the oposite direction will reduce the value of the cost function, which is the idea behind gradient descent.

mini-batch, is selected. The gradient is subsequently approximated using only the examples from the mini-batch. This procedure has multiple advantages. First, it reduces the computational cost of a single update to the network parameters and forces it to be constant with regards to the size of the training set. Updating the network more often with an approximation of the gradient has proven to reduce wall-clock training time significantly [213]. Second, the error between the approximate version and the full gradient scales as $1/\sqrt{n}$, where n is the size of the mini-batch [213]. It, therefore, scales slower than the cost of calculating the gradient and one can balance computational efficiency against the quality of the gradient approximation. Third, using a random subset reduces redundant calculations. In the extreme case, where all examples in the training set are the same, calculating the gradient on the entire training set is equivalent to calculating it on a single example. In a realistic case many examples from the training set may have similar effects on the gradient and applying the update to the parameters earlier leads to faster convergence [213]. Finally, the stochastic nature introduces randomness into the parameter update steps taken. This may be beneficial when the algorithm finds a local minimum of the loss function, which gradient descent would be unable to get out of. However, this randomness also prevents the algorithm

to settle into true minima. In practice, this is a minor issue when mini-batch sizes are not too small and the advantages outweigh the drawbacks [218]. A more sophisticated version of SGD known as "Adam" will be discussed at the end of this subsection as it was the optimizer I primarily used in my works.

SGD is not mathematically guaranteed to arrive even at a local minimum, but practical application have shown that it is capable of reaching sufficiently small values in most cases [213]. To improve optimization further, the whole training set can be shuffled and used for another iteration of SGD. Since the entire training set is reused in this case, one full pass on it is also called an epoch.

While SGD describes how to change the network parameters once the gradient is known, it does not provide means to obtain it. This step was the major roadblock for deep learning which was resolved with the introduction of the *backpropagation* algorithm, or simply *backprop*, by Rumelhart, Hinton, and Williams in 1986 [241]. At its core, backprop is a simple application of the chain rule of derivatives. However, it also utilizes the structured nature of the NNs to save redundant calculations and thus enables an efficient computation of the gradient. Due to its importance to the field of deep learning, I will introduce the algorithm in the following paragraphs in a bit more detail.

The goal of backpropagation is to efficiently calculate $\nabla_\theta J(\theta)$. For simplicity we assume J to be of the form introduced in (3.71), i.e. the loss function depends on the parameters θ only through the network. In this case we can focus on calculating the gradient of the per-example loss $L(y_i, \hat{y}_i)$. For simplicity we, furthermore, assume that the output of the network $\hat{y}$ is a vector. If it is not, we apply a reshaping operation that orders the outputs to be of vector form. For readability we also drop the example index i and understand that the following calculations are done for a single example. By the chain rule we then find [213]

$$(\nabla_\theta L(y, \hat{y}))_i = \frac{\partial \hat{y}^j}{\partial \theta_i} \frac{\partial L(y, \hat{y})}{\partial \hat{y}^j} = (\nabla_\theta \hat{y} \cdot \nabla_{\hat{y}} L(y, \hat{y}))_i, \tag{3.73}$$

which uses the Einstein summation convention. The rightmost part in equation (3.73) $\nabla_{\hat{y}} L(y, \hat{y})$ is the gradient of the loss function with respect to the network output. This can be calculated analytically once and is then evaluated for each example simply by inserting the corresponding network output. The other part is the Jacobian of the network with respect to its parameters

$$\nabla_\theta \hat{y} = \nabla_\theta \mathcal{N}(x, \theta). \tag{3.74}$$

We assume the network to be a simple chain of layers as defined in (3.64). Once the backpropagation algorithm has been developed for this case, extending it to branching networks is trivial. For easier notation it is helpful to

define the recursive relation $z_i := W_i a_{i-1}(z_{i-1}) + \vec{b}_i$, with the stopping condition $z_0 = x$ and a_0 being the identity mapping. Let $\nabla_{\theta_i}\hat{y}$ be the gradient of $\hat{y}$ only with respect to the parameters of layer i. With this we find

$$\begin{aligned}\nabla_{\theta_i}\mathcal{N}(x,\theta) = \nabla_{\theta_i} a_n(z_n) &\overset{i\leq n}{=} (\nabla_{\theta_i} z_n)\cdot \nabla_z a_n\Big|_{z=z_n} \\ &= W_n \nabla_{\theta_i} a_{n-1}(z_{n-1})\cdot \nabla_z a_n\Big|_{z=z_n}, \end{aligned} \tag{3.75}$$

which is a recursive relation for $\nabla_{\theta_i} a_n(z_n)$ that stops on layer i. n is the number of layers of the network. Note that while the calculations above suggest that W_i have to be matrices and z_i have to be vectors, the equations hold for tensors of arbitrary dimension. A few things about equation (3.75) are noteworthy. First, a_i are scalar functions that are applied component wise. To calculate the gradient $\nabla_z a_i$, the one-dimensional derivative $\partial_z a_i$ simply has to be evaluated at the different values of z_i. So the derivative for each activation function a has to be computed analytically in advance only once and can then be used to rapidly calculate the gradients $\nabla_z a_i$. Second, since (3.75) is a recursive equation, the gradient at layer $n-i$ depends on all gradients from layer $n-i+1$ to n. This is where the name backpropagation comes from, as the gradients are propagated back through the network. The recursion stops on layer i

$$\nabla_{\theta_i} a_i(z_i) = (\nabla_{\theta_i} W_i) a_{i-1}(z_{i-1}) + \nabla_{\theta_i}\vec{b}_i. \tag{3.76}$$

For practical implementations, the backprop algorithm is comprised of two stages. First, the input data is passed through the network and the activations z_i are stored for every layer. This operation is known as the *forward pass*. The second step uses the data from the forward pass to iteratively compute the gradient using equations (3.73), (3.75), and (3.76). This is referred to as the *backward pass*.

In principle the calculation of the gradient could be done sequentially. In real applications this is often impractical, since GPUs allow for high degrees of parallelization. Therefore, many, if not all, examples from a mini-batch are evaluated at the same time. Since the backprop algorithm requires to store the output of all intermediate layers, the memory requirements during training scale linearly with the mini-batch size. This was a limiting factor for many of our works and usually required us to use small mini-batches. Furthermore, a computational graph is usually built which describes the operations and their order that need to be taken to compute the output of the NN given its input. Each node in this graph then needs to implement a backprop-method that returns the gradient of that node given any of its inputs and an inbound gradient.

As an example consider the matrix multiplication operation $C = AB$. Its backprop-method needs to return GB^T when it is called for input A and inbound gradient G and A^TG if it is called for input B. For a more thorough discussion of how the computational graphs are built I refer the reader to chapter 6 of [213], where the above example was taken from.

With the definition of a training and test set, the loss function, and SGD in combination with backprop we have developed the most important tools to train a NN. However, during training multiple problems can arise. I will now discuss the most common ones and how to characterize them.

Common problems during training

One of the most challenging problems of deep learning is the training setup. How does one choose a network structure that is capable of efficiently solving a given problem? The structure of a NN is most commonly referred to as the *architecture* of the network and designing it is often more of an art than a science. Certain conditions, like translation invariance of the problem or hardware resource limitations, may inform or constrain it, but most of the time, the architecture is found empirically.

Another part of the training setup is the optimizer. It, too, can greatly influence how efficiently a NN learns. Even the simple SGD optimizer discussed above has a free parameter, the learning rate η. Setting it too large or too small may have detrimental results on the ability of the NN to learn. More complex optimizers, such as Adam [253], often have a larger number of tunable parameters and setting them correctly can be challenging. To find good settings for the optimizer, one usually has to resort to trial and error as well.

Both the architecture and the parameters of the optimizer have in common that they are set outside of the training loop. They can, therefore, not be updated automatically. Parameters that control the behavior of the learning algorithm are known as *hyperparameters* [213].

To optimize the hyperparameters, one often tests different settings, trains the algorithm, and compares their generalization errors. The settings that yield the best performance are then used. As we are interested in the generalization error, the training set cannot be used for this calculation. On the other hand, using the test set to optimize the hyperparameters would be dangerous, as this would introduce a bias. We would optimize the learning algorithm to perform well on the test set, reducing the ability of said set to measure how well the algorithm can generalize beyond previously observed examples. For this reason a third set is introduced: the *validation set*. With this set, hyperparameters are optimized by choosing specific settings, train-

ing the resulting algorithm using the training set, and calculating the error on the validation set. The errors for different hyperparameters are subsequently compared and the model with the best performance is chosen. Finally, the generalization error is calculated on the test set only once on the chosen model to estimate how well the algorithm will perform. The error calculated on the validation set is also called the *validation error*, to differentiate it from the generalization error [213].

When a NN is trained for long periods of time, one can often observe that the network starts to "remember" the samples of the training set. To spot this, one can compare the error or loss calculated on the training set to the one calculated on the validation set after every epoch. When the network starts to remember individual samples it stops to generalize and the validation error will grow. This process is known as *overfitting* [218].

To understand overfitting, we can consider polynomial regression. Let us imagine that our training set consists of N samples with data x_i and labels y_i. Let us further assume that the underlying distribution of our data is the polynomial $p(x) = \sum_{j=0}^{M} a_j x^j$ of degree $M < N$, i.e. $y_i = \sum_{j=0}^{M} a_j x_i^j$. If we use a polynomial $\hat{p}(x) = \sum_{j=0}^{M} \hat{a}_j x^j$ of degree M to fit our training data, we will drive the MSE of the training set to 0 when $\hat{a}_j = a_j \; \forall j \in \{0, 1, \cdots, M\}$. However, we can also fit all samples from the training set identically when we use a polynomial of any degree $\geq N$ [213]. Importantly, this fit does not necessarily have to find the coefficients

$$\hat{a}_j = \begin{cases} a_j, & j \leq M \\ 0, & \text{otherwise} \end{cases} \tag{3.77}$$

but may find a different solution. See panels (b) and (c) of Figure 3.16 for an example. In the case of overfitting the model is too complex for the task that has to be solved [213]. When the new model of degree $\geq N$ is evaluated on new points, it will most likely make incorrect predictions; it has "memorized" the training samples but is incapable of generalizing to new samples.

There are two options to reduce overfitting. One can either decrease the model complexity or increase the number of samples in the training set. As data could be generated at little additional cost for most of the studies in this work, we have usually chosen to take the latter approach [213, 218].

Instead of the model being too complex for the problem, the opposite may also happen. When the model is inherently not capable of representing the underlying data distribution, it is *underfitting* the data. In the example of polynomial regression above, any polynomial of degree $< M$ would most likely be incapable of matching the training data. Panel (a) of Figure 3.16 shows an example [213].

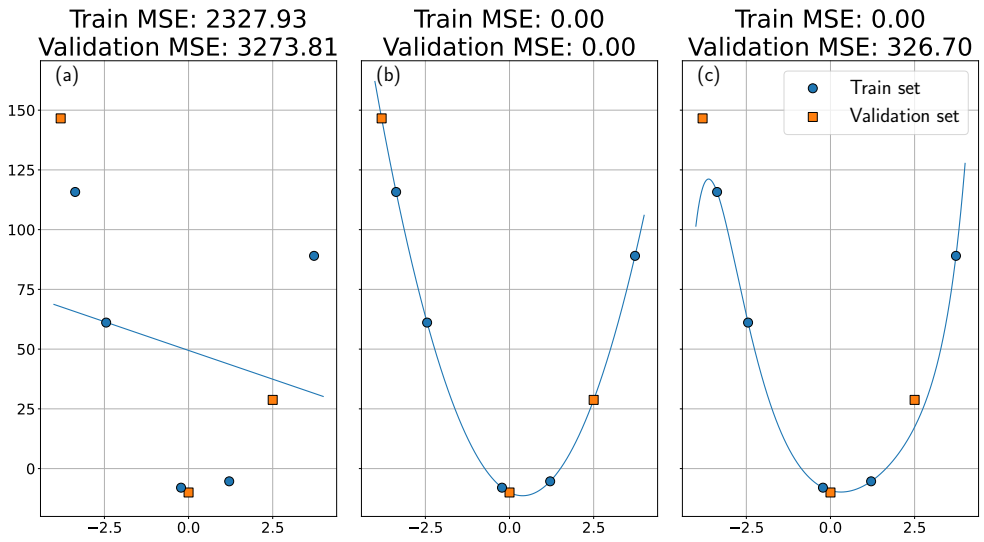


Figure 3.16: Polynomials of different degrees fitting the same training data using a least squares fit. The training data is generated from the second degree polynomial shown in panel (b). All panels list the MSE of the fit against the training and validation data on top. In panel (a) a linear model is used to fit the data. It is not complex enough and underfits the data. Panel (b) is a second degree polynomial that optimally fits the data. It minimizes the MSE on both the training and validation set. Panel (c) fits a 9th degree polynomial to the training set. It overfits and cannot generalize well to the unseen validation data. This figure was inspired by figure 5.2 of [213].

Spotting underfitting is often a lot more challenging than spotting overfitting, because it keeps the error calculated on the training set large. However, the same effect happens, when the network has not yet converged to a good local minimum of the loss. Furthermore, one rarely knows a priory what a good value for the converged loss would be. Upper limits can often be derived, but those usually cover the network not being able to optimize at all, rather than it failing to suitably approximate the true data distribution. Once underfitting has been identified though, one can usually increase the number of trainable parameters of the NN to resolve the issue. Other times finding a more suitable architecture or different data representation is required.

Another common problem of deep neural networks are vanishing or exploding gradients [213, 218, 240]. Deeper layers tend to have smaller gradients, which causes a stagnation in their optimization [218]. To a certain degree studies have found that the main cause of vanishing gradients were the combination of the network initialization, i.e. the initial parameters of the network, and the activation functions [254]. But even with improved network initialization and suitable activation functions, training can become difficult beyond a certain depth [240]. The opposite can also happen, where gradients grow exponentially. However, this is mostly a problem in RNNs and can be combated by clipping the norm of the update step [213].

Further failure modes for the training of deep NNs are discussed in chapter 8.2 of [213].

The Adam Optimizer

All of the works centered around deep learning that are discussed in this thesis make use of the Adam optimizer. For this reason, it is introduced here. Adam stands for "adaptive moment" estimation and it computes individual adaptive learning rates for different parameters from estimates of the first and second moments of the gradients [253]. It can be seen as a variant to the RMSProp algorithm [213, 255], which extends the AdaGrad algorithm [256].

The core idea of AdaGrad is to scale the learning rate of individual parameters based on the scale of the gradient. In regions where the loss surface is shallow one wants to traverse it fast to reach more optimal values. In regions where the surface has a steep gradient, smaller steps should be taken to not overshoot a potential minimum. To do this, the algorithm aggregates the squared gradients. On each SGD step the current gradient is then divided by the square root of the aggregated squared gradients.

RMSProp uses an exponential moving average to aggregate the squared gradients. This reduces the memory of gradients from the extreme past and

helps to keep the learning rates large enough to converge quickly once a locally convex bowl has been found [213].

Adam uses two exponential moving averages. Instead of just keeping track of the squared gradient, i.e. the second moment of the gradient, it also keeps track of the first moment of the gradient, i.e. its mean. Additionally, it removes biases from the moments. The full algorithm is given in algorithm 1 of [253]. The core calculations of the algorithm in the language used in this thesis are [218]

$$
\begin{aligned}
&1.\ m \leftarrow \beta_1 m - (1-\beta_1)\nabla_\theta J(\theta) \\
&2.\ s \leftarrow \beta_2 s + (1-\beta_2)\nabla_\theta J(\theta) \odot \nabla_\theta J(\theta) \\
&3.\ \hat{m} \leftarrow \frac{m}{1-\beta_1^t} \\
&4.\ \hat{s} \leftarrow \frac{s}{1-\beta_2^t} \\
&5.\ \theta \leftarrow \theta + \eta \hat{m} \oslash \sqrt{\hat{s}+\epsilon},
\end{aligned}
\tag{3.78}
$$

where β_1, β_2, and ϵ are hyperparameters of the algorithm, t counts the iterations, and $\odot$ and $\oslash$ are element wise multiplication and division operators, respectively. The default values suggested in [253] are $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^-8$, and $\eta = 0.001$.

Steps 1 and 2 are the exponential moving averages of the first and second moment of the gradient, respectively. The values for m and s are both initialized as arrays filled with zeros. This initialization would introduce a bias, which is removed by the operations in steps 3 and 4. Finally, step 5 applies the gradient update, which scales the learning rate by the inverse square root of the second moment of the gradients. The gradient updates are also only propagated using the running mean of step 3.

3.3.3 Convolutional Neural Networks

The layers discussed in section 3.3.1 connect every input to every neuron. They are known as fully connected or dense layers. A fully connected layer with N inputs and M outputs has a weight matrix of dimension $M \times N$. For large inputs and outputs this can quickly become very straining on both memory and compute power. Furthermore, both the input- and output-shape of the layers must be set at the beginning and cannot change.

All of these problems are being addressed by the convolutional layer which was invented in 1989 [257]. Instead of connecting all inputs to all outputs, it connects only parts of the input to each output neuron. Furthermore, the weights are shared between different connections. In mathematical terms,

the layer performs a discrete convolution of a kernel with the input data; hence the name "convolutional layer". For 1 dimensional data $\vec{x}$, weights $\vec{w}$, and bias b the layer can be written as

$$\mathcal{L}_{\text{conv}}(x,\theta)_i = a\left(\sum_j x_j \hat{w}_{i-j} + b\right), \tag{3.79}$$

where

$$\hat{w}_i = \begin{cases} w_i, & 0 \leq i < \dim(\vec{x}) \\ 0, & \text{otherwise} \end{cases}. \tag{3.80}$$

The weights $\vec{w}$ in the above equation are often called *kernel*, and when defining a NN architecture with convolutional layers one often sets the *kernel size*, i.e. the dimension of the kernel.

Depending on the problem, convolutional layers have several advantages over fully connected layers. First, the introduction of sparse connections. This means that not all outputs are connected to all inputs, which reduces the theoretical computational time from $\mathcal{O}(M \times N)$ to $\mathcal{O}(K \times N)$, where $K = \dim(\vec{w})$. Second, the introduction of shared weights, which reduces memory requirements, as only K rather than $M \times N$ numbers have to be stored for the weights. Third, the introduction of equivariance to translation. This is a direct consequence from sharing parameters and translating them over the input, as it allows to search for the same learned features in multiple locations of the input data. For instance, if a kernel has been optimized to detect edges in an image, it will be able to detect these edges in all parts of the image. A fully connected layer, on the other hand, would need to learn this filter at every location. While this is in principle possible, and one can reduce the convolutional layer to a special case of a fully connected layer when the input size is known[4], it is not quite so simple in real applications. If we want to learn to detect a specific edge in all parts of the image, a fully connected layer needs sufficiently many examples of such edges at every location of the image. A convolutional layer might learn the filter even if the edges are only presented in one specific region of the images from the training set [213]. Additionally, convolutional layers can process inputs of arbitrary size, as the parameters of the layer do not depend on the input.

A single kernel will learn a single filter. For example, the two dimensional filter in the top branch of Figure 3.17 detects vertical edges. It will, however,

[4]The weight vector can be zero padded to have the same dimensionality as the input vector. One then cyclicly changes the position of the weight vector in the zero padding and uses the different resulting vectors as rows for the weight matrix of a fully connected layer. See section 9.4 of [213] or [66] for more details.

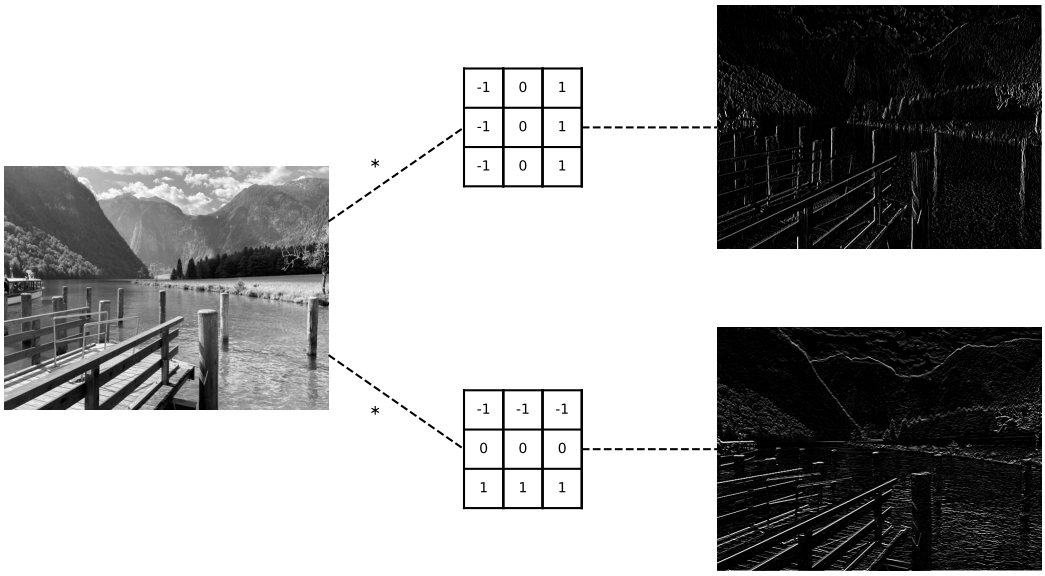


Figure 3.17: An image convolved with a vertical (top) and horizontal (bottom) edge-detection kernel. The original image is shown on the left. The processed images are shown on the right. The $*$ is the symbol for the convolution operation. Brighter pixels in the output images signify larger values, i.e. a larger overlap between the kernel and the image. On the top branch one can clearly spot the bright vertical edges of the posts, which are not visible in the lower branch. On the other hand, one can clearly spot the sharp edges at the top of the posts in the lower branch but not in the upper branch.

not be able to also detect horizontal edges. For this reason, a convolutional layer usually has multiple kernels of the same dimensions. It optimizes them in parallel to learn multiple different filters. The outputs of the convolutions with the different kernels are then stacked. The output of a single kernel convolution is known as a *feature map* and the index dimension of the different feature maps is known as the *channel* dimension. The name feature map originates from the observation that the activation at a certain location is particularly large, when the kernel strongly overlaps with a specific feature in the input data [258]. Naming different feature maps different channels, stems from RGB-images, which have three distinct channels; one for red (R), one for green (G), and one for blue (B) [213].

As was the case for NNs consisting only of fully connected layers, greater depth of convolutional neural networks (CNNs) allow them to detect features of greater complexity. For instance, if the first layer learns one filter for vertical edges and one for horizontal edges, a subsequent layer may use the

information from both channels to infer the angle of an edge. For this to be possible, it has to combine the information. Therefore, a single kernel of a one dimensional convolution as defined in equation (3.79) spans all channels. If the input to a one dimensional convolutional layer has n samples and c channels, a single kernel W will be of dimension $k \times c$, where k is the kernel size. See Figure 3.18a for a visualization.

The definition of the convolution operation in equation (3.79) shifts the kernel by a single sample for each step. However, one can just as easily implement a convolution-like operation that has a larger step size. This step size is known as the *stride* and exchanges resolution for computational efficiency, as fewer computations need to be carried out [213]. The standard convolutional layer has a stride of 1. Figure 3.18b shows a strided convolution, i.e. a convolution with stride > 1.

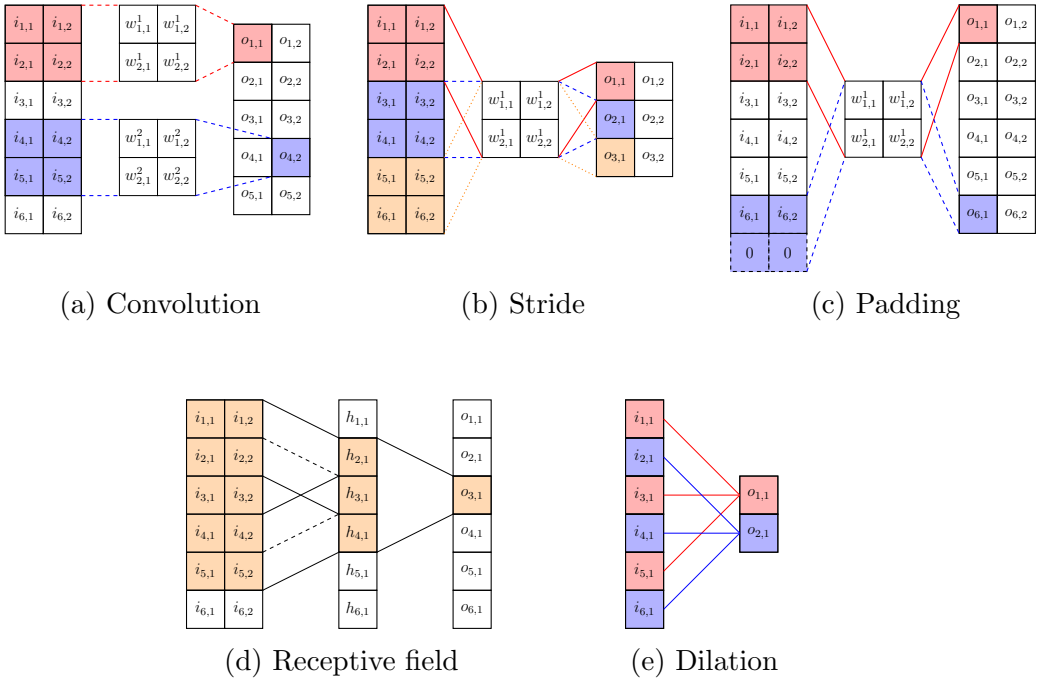
When a convolutional layer with kernel size k and stride s is applied to input data with n samples, the number of output samples n_c by default is given by

$$n_c = \lfloor \frac{n-k}{s} \rfloor + 1, \tag{3.81}$$

where $\lfloor \cdot \rfloor$ is the flooring operation. For $k > 1$ or $s > 1$, we find $n_c < n$. So the output is smaller than the input when a convolutional layer is applied. To avoid this shrinking, one often applies *padding* to the data. To pad the data one commonly uses zeros or mirrors the data at the edges. See Figure 3.18c for a visualization of zero-padding.

By design, individual output neurons of a convolutional layers have only access to a small portion of their input data. This in turn limits the scale of features they may detect. To avoid this issue, one can stack multiple convolutional layers. Figure 3.18d shows two stacked convolutional layers, which both have a kernel size of 3. Each output of the final convolutional layer is connected to 3 neurons on the first layer. Each of those neurons, in turn, is connected to three input samples. As such, the output neurons have indirect access to 5 samples from the input. This increases the amount of data they are receptive to. For this reason, the input samples each output of a convolutional layer is directly or indirectly connected to is known as its *receptive field*.

Instead of connecting only subsequent samples from the input, a kernel may also skip a few samples in between. This concept is known as *dilation*. It can be used to increase the scale of features a convolutional layer can detect, at the cost of resolution. The number of neurons that are skipped minus one is known as the *dilation rate*. When a convolutional layer has a dilation rate of n, every n-th input sample is connected. Figure 3.18e shows how a convolutional layer with a dilation rate of 2.



(a) Convolution (b) Stride (c) Padding

(d) Receptive field (e) Dilation

Figure 3.18: **(a) Convolution:** Depiction of a 1 dimensional convolution with multiple kernels. Each kernel has the same size and spans all channels of the input. The different channels are shown as columns. Different kernels produce different channels in the output. The output in this example is calculated as $o_{a,b} = \sum_{m=1}^{2} \sum_{n=1}^{2} i_{m+a-1,n} \cdot w_{m,n}^{b}$. **(b) Stride:** A convolution with a stride of 2. Instead of shifting the kernel by one sample, it is shifted by multiple. The different colors relate the outputs with their inputs. **(c) Padding:** The input of the convolutional layer is padded with zeros, such that the output has the same number of samples. **(d) Receptive field:** The figure shows the neurons of two stacked convolutional layers, where padding is applied such that the input and output have the same number of samples. Both convolutional layers have a kernel size of 3. The colored squares highlight the neurons which the output $o_{3,1}$ depends on. The connecting lines for every neuron show the region of the previous layer they are directly connected to. By stacking the convolutional layers, the single output neuron $o_{3,1}$ is influenced by almost the entire input data. The part of the input a single neuron is influenced by is called its receptive field. **(e) Dilation:** Shown is a convolutional layer with a kernel that is not connected but skips a few of its input. The kernel in the figure uses only every second output and it, thus, has a dilation rate of 2. A dilation rate of n means that only every n-th neuron from the input is considered.

The first major application of CNNs was the LeNet-5 [259], which was used for handwriting recognition [213, 218]. Since then CNNs have been the major driving force in computer vision tasks such as image classification [49, 260–262] and object detection [50, 218, 262–265]. They have also found applications in audio processing [230], NLP [247], and many scientific fields [231]. All of the works which utilized deep learning and that are part of this thesis have made use of CNNs.

Convolutional layers can be easily extended to higher dimensions. When the input data has n dimensions with c channels, the kernel will still span all channels but be limited in size in the data dimensions. It will then be moved across the input in all dimensions to fully cover it.

3.3.4 Special Layers and Concepts

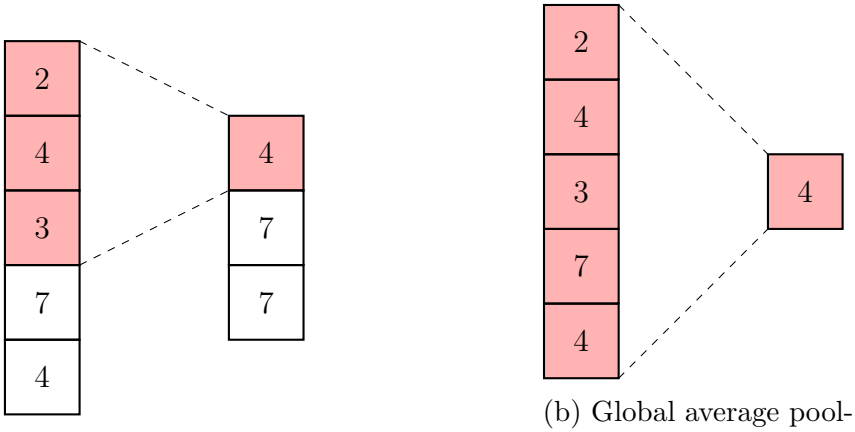
This subsection covers a few deep learning concepts that were used in different works discussed in this thesis. They go beyond fully connected and convolutional layers and are largely non-essential to gain an understanding of the works. I will, therefore, discuss them only very briefly. The interested reader may check the different sources provided below to learn more.

Pooling

One type of layer that is often used in CNNs are *pooling layers*. Like convolutional layers, these layers take a confined part of the input and aggregate it. The region they summarize is then shifted over the entire input. However, in contrast to dense or convolutional layers, they do usually not have any trainable parameters. They also usually operate on every channel individually and do not combine them like convolutional layers.

The two most common pooling layers are max-pooling [266, 267] and average pooling [257]. As their name suggests, they summarize a region of the input by its maximum or average value, respectively. See Figure 3.19a for an example of pooling.

Another form of pooling divides the entire input into a fixed number of bins and summarizes each bin. This form of pooling is useful, when a variable size input has to be mapped to a fixed size. This is often the case in image classifiers, where the input image is analyzed by a fully convolutional NN that produces a feature map. This feature map is then passed to a fully connected classifier to extract a prediction for the classes one is interested in. See for instance the RoIPooling operation in [268] or RoIAlign in [269]. At the extreme, all inputs are pooled together. This operation is commonly called global pooling.



(a) Max pooling

(b) Global average pooling

Figure 3.19: Two kinds of pooling layers. Panel (a) shows a normal max pooling layer, that pools 3 input neurons and has a stride of 1. Panel (b) shows global average pooling, where all input neurons are pooled together.

The main effect of pooling layers is the introduction of an invariance to the exact location of features in the network input. By summarizing the activations in a given region, a large value anywhere within this region usually has a large value of the summary as consequence. Another effect is the downsampling of data throughout the network. This reduces the computational cost to evaluate the network, as fewer paths have to be calculated, at the expense of resolution.

Dropout

In 2014 dropout layers were introduced [270]. They aim to reduce overfitting during training by randomly setting the output of some neurons on hidden layers to zero, effectively erasing them from the network for one training step. See Figure 3.20 for a visualization of dropout. The rate of setting outputs to zero is known as the dropout rate and is the only hyperparameter of this layer. Erasing individual outputs has multiple beneficial effects.

First, the network is trained as a kind of ensemble of multiple networks. The different networks of the ensemble all share their architecture and weights with the network where no units are dropped, but some paths are removed. As the dropped units change for every mini-batch, each ensemble member usually is trained directly only for few iterations. Due to the shared weights, however, they are all updated simultaneously and profit from every step. During inference, no dropout is applied and as a consequence, each member of the ensemble is effectively evaluated. This can then be understood as the

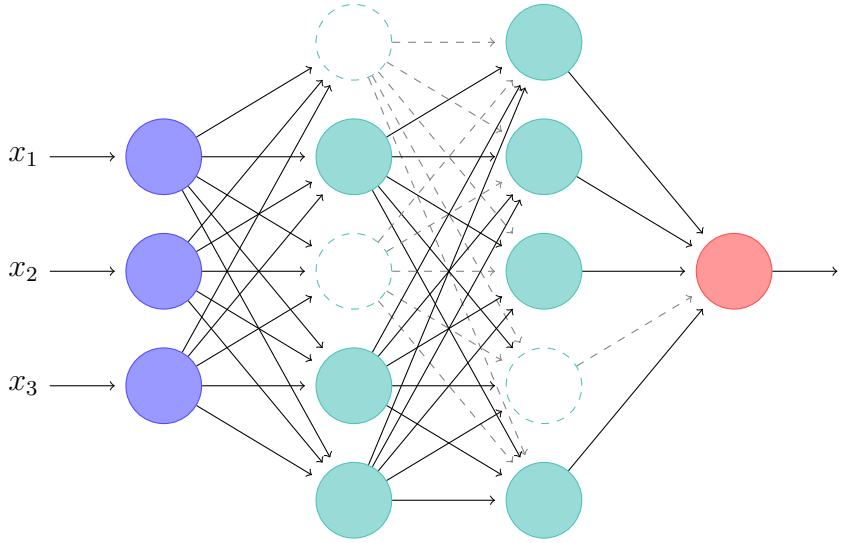


Figure 3.20: Visualization of NN with dropout. Dashed neurons have been dropped from the graph by setting the connected dashed weights to zero.

ensemble of all possible dropped units "voting" for the correct output [213].

Second, dropping different hidden units forces the network to become resistant to the removal of individual hidden units. It, thus, has to either learn redundancy or to classify based on different features. In a sense, dropout applies noise on the feature level. For instance, if a neuron has learned to detect a nose in a face and this neuron is dropped, the network is forced to learn to classify a face even when a nose is not present [213].

Batch Normalization

When training a NN the weights are usually updated by equation (3.72). The gradient $\nabla_\theta J(\theta)$ is comprised of partial derivatives ∂_{θ_i}, which assume that all other parameters are kept constant. However, in deep NNs changing the weights on one layer will have a direct influence on the inputs to the second layer. This means that higher order effects are disregarded[5] and the output can change a lot more than one would initially expect. In other words, the distribution of inputs to deeper layers changes as parameter updates are applied to earlier layers.

To counteract this problem, the authors of [271] introduced a layer called *Batch Normalization*. The core idea is to keep the mean and standard deviation of all or most layer outputs constant. At its core, batch normalization

[5]Compare equation (8.34) of [213].

calculates

$$\mu = \frac{1}{M}\sum_{i=1}^{M} x_i \tag{3.82}$$

$$\sigma^2 = \frac{1}{M}\sum_{i=1}^{M} (x_i - \mu)^2 \tag{3.83}$$

for every mini-batch. It then shifts the inputs by

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}, \tag{3.84}$$

where ϵ is a small constant used for numerical stability. This centers the activations of the previous layer to have a mean of 0 and a standard deviation of 1. Crucially, the gradient takes these calculation to correct the mean and variance into account. Were this not the case, the optimization algorithm could drive the mean or the variance to infinity [213, 271].

By applying the transformation of equation (3.84) one limits the ability of the network to represent certain distributions. For instance, if the Sigmoid activation (3.62) is considered, one would constrain it to the linear regime [271]. To allow the network to set the batch normalization layer to act as identity, the output is defined as

$$y_i = \gamma\hat{x}_i + \beta. \tag{3.85}$$

The parameters β and γ are of the same dimension as the x_i and are learned during optimization. While it may seem like a null-operation to first set the mean to 0 and the standard deviation to 1 and then introduce parameters β and γ to change the same parameters to different values, it does have a positive effect on the learning dynamics [213].

The calculation of the mean and standard deviation described above are only done during training. For inference a running mean of these values is kept.

Residual Blocks

The submission that won the 2015 ImageNet large scale visual recognition challenge (ILSVRC) [227] made heavy use of a concept known as *residual connections* [240]. These residual connections allowed for their network to be effectively trained even when more than 100 stacked layers were used [240]. They found that the more layers could be added before training stopped being effective the larger the performance of the network. This highlights further

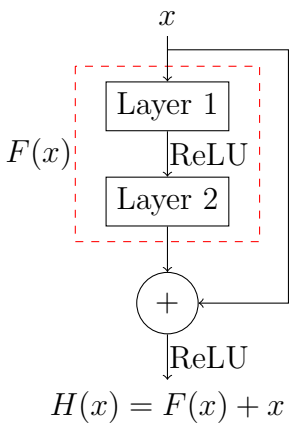


Figure 3.21: General structure of a residual block. The input x is passed through one or multiple layers in the residual block $F(x)$ and the output is added onto its input. This figure was adapted from figure 2 in [240].

the general notion that deeper networks usually perform better. Although introduced in 2015, their architecture is still commonly used for many state-of-the-art works today [262, 272–274].

Residual blocks contain one or multiple layers that are setup such that the blocks do not change the shape of their inputs. Their outputs are then added onto the input and passed on. See Figure 3.21 for a visualization. This change reformulates the learning problem. Rather than optimizing the layer block to learn a function $H(x)$, one optimizes the layer block to learn the residual function $F(x) = H(x) - x$. While this does not change the functions that the block can represent, it changes the mapping the block learns if all parameters are driven to 0. If the block does not use residual connections and is, hence, optimized to learn $H(x)$ directly, setting all parameters to 0 results in the zero mapping. The residual block instead would drive $F(x)$ to the zero mapping, such that $H(x)$ becomes the identity. Therefore, residual blocks hypothesize that an identity mapping should be the default behavior of a layer block, if it can extract no further information. This seems plausible, as one would expect that when two networks are compared, where the only difference between them is the depth, that the deeper network should not perform worse than the shallower one. In the worst case it should be able to set all of its layers at a greater depth than the shallower network to be the identity mapping [240].

Another benefit of residual blocks is that they allow for an easy passing of the gradient to layers further up in the stack. This is because the derivative

of the residual block is given by

$$\frac{d}{dx}(F(x)+x) = \frac{d}{dx}F(x) + 1. \tag{3.86}$$

So even if the derivative $\frac{d}{dx}F(x)$ is very small, i.e. the block is close to an optimal configuration or has not started to improve yet, the additive term allows the gradient to skip past the block and propagate upwards in the network [218, 240]. For this reason the residual blocks are also called skip connections.

3.4 Machine Learning in Gravitational-Wave Astronomy

Machine learning is a computational tool that has started to gain renewed interest in the early 2010s. It is only natural for scientists to evaluate the usefulness of new tools to their own area of research. However, there are reasons beyond the academic curiousness that justify a thorough investigation of the capability of modern ML algorithms to solve some of the problems in GW data analysis. As previously discussed, since the first observation of a GW the rate of detections has rapidly increased [15] and is expected to grow faster as detectors are upgraded [194, 275]. This necessitates the use of highly optimized data analysis algorithms to process the data in real time and produce accurate and reliable alerts. Additionally, it is desirable to produce accurate sky-maps of the expected origin of each signal to allow for prompt EM follow-up and to be able to extract more information from some mergers [62]. ML algorithms are known for their capability to discover patterns in data and their computational efficiency has enabled many advances in other scientific fields such as computer vision [49, 213]. This makes them a great contender to solve many of the above mentioned problems.

Many applications of ML to GW data analysis have already been studied. These include the identification and classification of glitches in the detector output, partial and full GW search pipelines, and parameter estimation algorithms. Below I will give an overview of recent developments in the field and will mainly focus on algorithms relevant to CBC signals. A great and more general overview of the field can be found in [48].

3.4.1 Data Quality

One of the first GW research fields that has made use of ML algorithms is the classification of data quality of the detector output [276–279]. As previously

discussed, the detector output contains many non-Gaussian noise transients known as glitches. Finding and classifying these glitches is important to identify their source and to reduce the number of false alarms. ML can be useful to identify different categories of glitches or predict them from recordings of external sensors that monitor noise sources, such as ground motion or EM interference [280], which couple highly non-linearly into the detector output.

The citizen science project GravitySpy [190] was started in 2016 and asks volunteers to classify time-frequency representations of different glitches. It leverages the Zooniverse platform and combines the use of ML algorithms with human categorization to classify glitches into known and unknown categories. The resulting data sets are then used to train machine learning models [281]. Importantly, the input data is a direct product of the detector output and does not take into account auxiliary data channels.

A different approach is taken by the iDQ pipeline, which tries to predict the presence of glitches only from auxiliary sensor data in real time [282]. It monitors $\mathcal{O}(10^3)$ auxiliary channels to produce a classification into "glitch" or "no-glitch" for every time step. The underlying ML algorithms are continually updated to account for non-stationarity of the noise. iDQ has been in use throughout the first three observing runs [14] and contributed to the rapid release of GW170817, which coincided with a glitch [48].

Other projects trying to identify glitches exist [283–285], many of which rely on the time-frequency representation produced by the Omicron software package [191].

3.4.2 Gravitational-Wave Searches

More important for this thesis, there exists a wide variety of ML based GW search algorithms. It is currently a very active area of research with several groups around the world providing rapid improvements over initial algorithms. The earliest works based on random forests use data products from other search algorithms or hand crafted features to identify signals [53, 286].

The first proof of principle using deep learning to directly detect BBH signals in time series data was proposed by George et al. in 2016 [55]. They used a 3-layer CNN to process 1 s of whitened time series data sampled at 8 kHz from a single detector and classified it into the two categories "signal" and "noise". Their network was trained on non-spinning BBH waveforms with masses between $5\,\mathrm{M}_\odot$ and $75\,\mathrm{M}_\odot$. Due to these simplifications, the parameter space consisted only of the two dimensional $m_1 - m_2$-plane. For this restricted parameter space they demonstrated the ability of their algorithm to be competitive in signal recovery to matched filtering for high false-alarm

probabilities (FAPs). As a difference to FARs, FAPs, in the context of this work, are not derived on long duration data but from individual samples that either do or do not contain a signal. The FAP is then the number of false positives divided by the number of true negatives. Independently, Gabbard et al. developed a similar algorithm [56] and verified the findings of [55]. They also extended it to lower FAPs. Both works, however, were limited by testing the signal recovery only on discrete samples, where each sample consisted of either a well aligned waveform submerged in noise or pure noise.

The first study using deep learning to detect BNS signals was published by Krastev in 2019 [287]. He used a network architecture similar to that of [55] to process 10 s sampled at 4 kHz to distinguish pure noise, BBH signals, and BNS signals. He was able to reproduce the performance on BBH signals from [55, 56], but his method was significantly less sensitive to BNS signals. He also quoted performance figures only as a function of FAP derived on discrete samples rather than in terms of FAR derived on continuous data. The study was later extended to cover real detector noise and produce point estimates for the source parameters [288]. In an independent work Schäfer et al. proposed a novel NN architecture tailored toward the detection of BNS signals that allowed processing of 32 s of data [61]. To enable processing so much data, they introduced a multi-rate sampling approach, that reduced the size of the data by a factor of 9 while preserving all relevant information. This allowed them to be substantially more sensitive than [287] to low SNR signals. However, their approach does not generalize well to high SNR signals and is incapable of detecting BBH signals. They also tested the algorithm on a continuous data set down to a FAR of 0.3 per month, thus providing a direct grounds of comparison to state-of-the-art matched filter search pipelines. It showed that there is a significant performance gap especially for low FARs. Furthermore, they analyzed data from two detectors in a single network. This approach is briefly summarized in chapter 4 of this thesis.

The studies discussed so far made use of a CNN with a few fully connected layers to generate the classification output. This architectural choice requires the use of a sliding window approach, when the networks should be applied to data of duration longer than the input the network was trained on. Gebhard et al. introduced a fully convolutional architecture in [289], which allows for the network to be applied to input of arbitrary sizes. It also produces outputs with a higher time resolution than partially convolutional networks. The authors provide criticism of the evaluation procedure used in previous works, which quote performance metrics in terms of FAPs instead of FARs. They then provide extensive studies of their own approach and test it in terms of FARs. They also investigate the timing accuracy of their network to the location of the merger. Their architecture was picked up and further

improved by Wei et al. in [57]. This improved architecture was tested on real data from O2 and O3 and is capable of detecting real GW events in the data at a FAR of 2.7 per day. Both of these networks were also designed to process data from multiple detectors.

Another approach that has started to be explored is building NNs inspired by matched filtering. Wang et al. create a convolutional layer from a reduced template bank of whitened waveforms and use it to perform a matched filter operation in the time domain [290]. Their network takes an estimate of the PSD into account and uses a CNN to process the resulting SNR time series. They manage to detect all GW events from O1 at a high FAR. The group also contributed to the mock data challenge discussed in chapter 8, where their approach is evaluated at low FARs. Yan et al. notice that matched filtering searches are formally equivalent to a particular NN architecture, that can be hand crafted [54]. However, they highlight that matched filtering is not optimal when the signal is not known exactly, as it is not mathematically guaranteed that it minimizes the FAR for a given true positive rate. As a consequence, they initialize their network with a given discrete template bank and fine tune it in the hope of finding a better detection criterion. They claim that their approach can consistently outperform matched filtering. However, they test a limited mass range of $40\,\mathrm{M}_\odot$ to $50\,\mathrm{M}_\odot$, where they use up to 10 000 templates, use only data from a single detector, and maximize over the raw SNR of all templates. The use of a possibly over-dense template bank, as well as the lack of coincidence and signal consistency tests artificially increase the FAR of any detection. Nonetheless, they highlight important deficiencies of matched filtering and their work demonstrates a possible avenue for deep learning to go beyond the capability of existing methods.

The previously discussed algorithms have all made use of time series data. Most ML advances, on the other hand, originate in computer vision, which processes two dimensional images. For this reason, some studies have looked at a time-frequency representation of the data to make use of such concepts. Wei et al. utilize a ResNet50 [240] pre-trained on image classification data to create early warnings for BNS mergers from spectrograms of the data [291]. The network is fine tuned on injected GW data and is resilient to glitches. The network is capable of detecting GW170817 10 s before the merger. Aveiro et al. use the object detection network YOLOv5 [292] to accurately locate GW mergers in these time-frequency plots. While their work is still at a proof of concept stage, this domain transform highlights an under utilized aspect of machine learning, where rapid development in other fields can improve the capability of GW detection.

Some works try to utilize the computational efficiency of NNs along a similar route as Wei et al. [291]. They try to produce early warnings for

BNS signals to increase the probability of successful prompt EM follow-up observations. Baltus et al. [293, 294] use a CNN similar in nature to the early works by George et al. [55] and Gabbard et al. [56] to analyze time series data. In the optimal case of strong signals they expect an early warning of up to 100 s. Yu et al. [295] use the detector output as well as auxiliary noise channels to reduce non-linear noise couplings and increase their sensitivity to low-frequency signals. Their network is trained to produce early warnings for BNS and NSBH signals. They claim similar early warning capabilities as Baltus et al. Chapter 5 of this thesis summarizes a qualitative analysis of early warning capabilities of an existing state-of-the-art matched filtering based search pipeline. Such studies are important on their own but are also imperative as a point of comparison to machine learning algorithms. An advantage of the study presented in chapter 5 over the works by Baltus et al. and Yu et al. is the capability of producing a sky-location estimate.

Besides direct searches, many studies are looking into ways to improve existing search pipelines. They mainly try to achieve this goal by adjusting the ranking statistic. Jadhav et al. [296] introduce MLStat, a deep learning algorithm that processes time-frequency representations of the data obtained from a continuous wavelet transform to differentiate between CBC signals and glitches. The model is a pre-trained InceptionV3 [297] image classifier which is fine tuned on parts of the GravitySpy data set [190]. They use the output of the CBC class as a probability to re-weight the coincident ranking statistic and report an increase in the sensitive volume of the PyCBC search [60] by up to 30%. Instead of creating a new metric to adjust the ranking statistic, McIsaac et al. [298] use deep learning to improve the existing χ^2 tests. They train a NN to optimize hyperparameters of a χ^2 test to improve signal recovery and glitch rejection for high mass signals. They quote an improvement in sensitivity of up to 11% to high total mass signals and are confident that such an automatic tuning of hyperparameters can also be used to improve other signal consistency tests. Choudhary et al. [299] present a NN that aims to distinguish CBC signals from blip glitches. While they do not specify its use in altering the ranking statistic, they claim it to be superior in classification to classical χ^2 tests. They introduce a sine-Gaussian projection, which produces a time-frequency representation of the data and use this as input to their network. The loosely modeled coherent wave burst (cWB) search [37, 38] has recently introduced a ML based enhancement to their pipeline [300]. The original algorithm produces vetos based on summary statistics generated by the search to reduce the impact of non-Gaussian noise transients. These vetos are a binary decision between noise-like events and signal-like events. The ML improvement uses a learning algorithm known as XGBoost [301] to create an ensemble of decision trees. A weighted average

of all ensemble outputs is then processed by a Sigmoid activation to produce a continuous output. As input the decision trees use a subset of 14 summary statistics generated by the search. By using the Sigmoid output as a modification to the original ranking statistic and by eliminating all other vetos, the authors of [300] find an improvement of up to 26% in sensitivity.

Other applications of ML to GW searches exist. Some notable works include the application to CW searches [302–305], EMRI searches [306], and the search for novel signals by anomaly detection [307, 308]. The GWSkyNet project uses public data from alerts of GW events intended for other astronomers for EM follow-up to distinguish between astrophysical events and noise artifacts [309, 310]. The aim of the project is to better inform other astronomers about which alerts are most valuable for follow-up observations.

With this plethora of different search algorithms an objective comparison among different approaches and to state-of-the-art methods is desirable. However, this task is complicated by differing data sets and the usage of different evaluation metrics. For instance, in chapter 4 we find that our approach is substantially more sensitive to quite BNS systems than the work presented in [287] but is still far away from the sensitivity of PyCBC Live, a state-of-the-art low-latency GW search pipeline. For this reason, chapters 6 and 7, among other contributions, re-analyze the early work by Gabbard et al. [56] and compare them to PyCBC [60], both in the single- and multi-detector case. Chapter 8 describes an attempt of creating a reference data set and evaluation metrics to allow for the important quantification of ML based search algorithm performance. It is one of the goals of this thesis to create an environment that allows for an objective evaluation of ML algorithms for CBC detection to enable targeted development and quick adoption of good methods into production searches.

3.4.3 Parameter Estimation

Another important aspect of GW data analysis is the estimation of source parameters including error estimates. In the last three years major progress has been made in using ML to rapidly produce posterior estimates. This is of special interest as traditional parameter estimation can take days or even weeks, depending on the source, to finish analyzing a single event.

As parameter estimation is not the main focus of this thesis, I will only highlight a few of the important works of the recent past. The inital work by Chua et al. [311] is capable of producing posteriors in one or two dimensions by producing histograms or parameters for a Gaussian mixture distribution. Gabbard et al. [312] use a conditional variational autoencode network to predict the full 15 dimensional posterior of BBH directly from the whitened

strain data. A variational autoencoder creates a latent representation of the input data in terms of parametrized probability distributions using an encoder. The distribution is then sampled and each sample is processed by a decoder, which in turn produces a second set of parametrized distributions. A single sample is drawn from this second set of distributions for every sample from the latent distributions. In this way a posterior is built up. Green et al. [313–315] use normalizing flows to produce posteriors. The idea of a normalizing flow is to find the transformation from a simple distribution, like a Gaussian distribution, to the target distribution. In their case the target distribution is the GW posterior for the parameters. They also include knowledge about symmetries of the parameters in their algorithm to simplify the task. All of these algorithms have the advantage that they do not need true posteriors as targets. Instead they learn by sampling the prior distribution during the training process. This means that training data is drawn from the prior and the networks only require the true parameter as label. Chatterjee et al. [316] also use a normalizing flow for rapid sky-location estimation. It can produce accurate estimates of the source location in milliseconds. Other than previous deep learning methods it does so not only for BBH, but also for BNS and NSBH sources.

A completely different approach was taken by Williams et al. [317]. They create an algorithm named NESSAI, which is a drop in replacement for the samplers used in many state-of-the-art parameter estimation codes. Their network is also based on a normalizing flow and predicts the contours of iso-likelihood surfaces based on live-points in a nested sampling algorithm. The normalizing flow then allows them to efficiently sample the contour, resulting in an increase in the speed of nested sampling by a factor of 1.4 over the nested sampler DYNESTY [318].

Chapter 4

Detection of Gravitational-Wave Signals from Binary Neutron Star Mergers using Machine Learning

This chapter briefly summarizes the work done in [61], which in turn is based on my master thesis [66]. It highlights the improvements of [61] over [66]. It is discussed in this thesis, as the detection of long duration GW signals from BNS mergers is a major challenge in machine learning based search algorithms.

4.1 Introduction

Long duration BNS signals are computationally expensive to search for using matched filtering. This is an effect of the required high density of templates in the bank at low masses. On the other hand, detecting BNS signals with as low a latency as possible is important to maximize EM observation time of potential counterparts.

One possible approach to try to reduce computational demands of the search is to employ advanced ML methods, that shift the computational cost to the training phase. To prove they are capable of rapid and accurate detection, these algorithms must be evaluated at FARs that are at a level comparable to online production search pipelines [58]. In this study, we develop a deep learning search algorithm for BNS signals. We compare it

to the PyCBC Live pipeline which has been used in the online analysis of O2 and O3. We also compare our algorithm to another deep learning based search that targets BNS mergers.

The core problem of developing deep learning based BNS search algorithms is the duration the signal spends in the sensitive bands of the detectors. This long duration in combination with a merger at high frequencies results in a large number of samples that have to be processed. When the size of the input to a NN grows, so must typically the size of the network [213]. This makes training unfeasible due to excessive hardware requirements. Our solution to this problem is to use the knowledge of the frequency evolution of BNS signals to re-sample different parts of the data at different rates.

4.2 Methods

To reduce the number of samples in the input to the NN we sample the data at different rates. During the early inspiral the frequency is low and evolves slowly. As a consequence, a low sample rate is sufficient to resolve the signal for a long duration. Only close to the merger are high frequencies involved and a high sample rate is necessary. Informed by this signal evolution, we re-sample the first 16 s of the input data at a rate of 128 Hz. The next 8 s are sampled at a rate of 256 Hz. We continue doubling the sample rate and halving the duration of data we sample until the final second of our 32 s input data. The final second is split into two 0.5 s parts, each sampled at 4096 Hz. This procedure reduces the number of samples by a factor of 9, while the SNR that is lost due to the early truncation at low frequencies is only $\approx 2\%$. In [66] we had re-sampled 64 s by starting with a sample rate of 64 Hz for the first 32 s part. However, we found that for some signals the maximum frequency in that time period exceeded the limit of 32 Hz set by the sampling rate. See Figure 4.1 for a visualization of the re-sampling process.

The architecture used in [61] is highly adjusted to the multi-rate sampled data. Each of the 7 different parts of the input data is processed by a different input to the network. After each sample rate has been processed individually, pairs of two are combined and processed further. This structure cascades down until only a single branch remains. This single branch is processed by a few more layers to produce two outputs: An estimate of the optimal SNR contained in the input and a p-score that is a value between 0 and 1 signifying the confidence of the network that a signal is present in the input. A high-level overview of the architecture is given in Figure 4.2. For more details see [61, 66]. The architecture is the same as the one presented in [66], adjusted to the reduced number of multi-rate sampled parts.

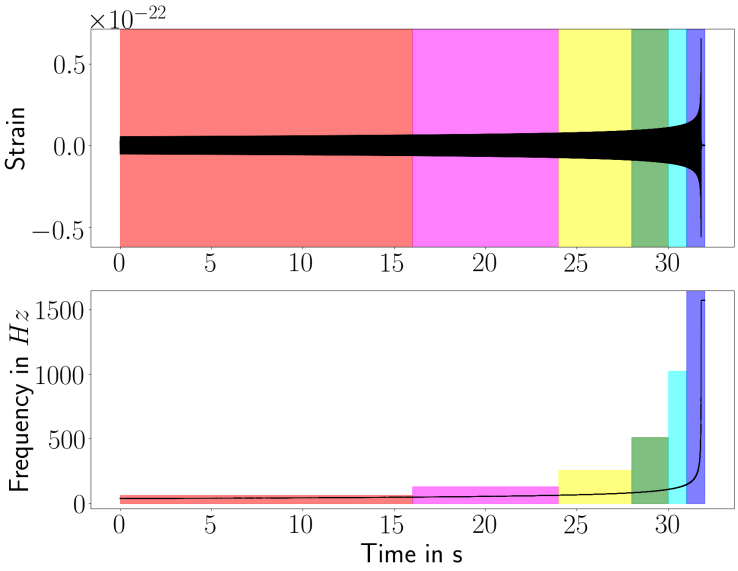


Figure 4.1: The top panel shows the strain evolution of an example GW from a BNS merger in black. The bottom panel shows the corresponding frequency evolution in black. The colored boxes represent parts of the signal which we sample at different rates. The height of these boxes in the bottom panel represents the Nyquist-frequency of the sample rate which is used for each part. To fully resolve the signal, the black curve must stay inside the colored boxes of the bottom panel at all times. Figure and caption were taken from [61].

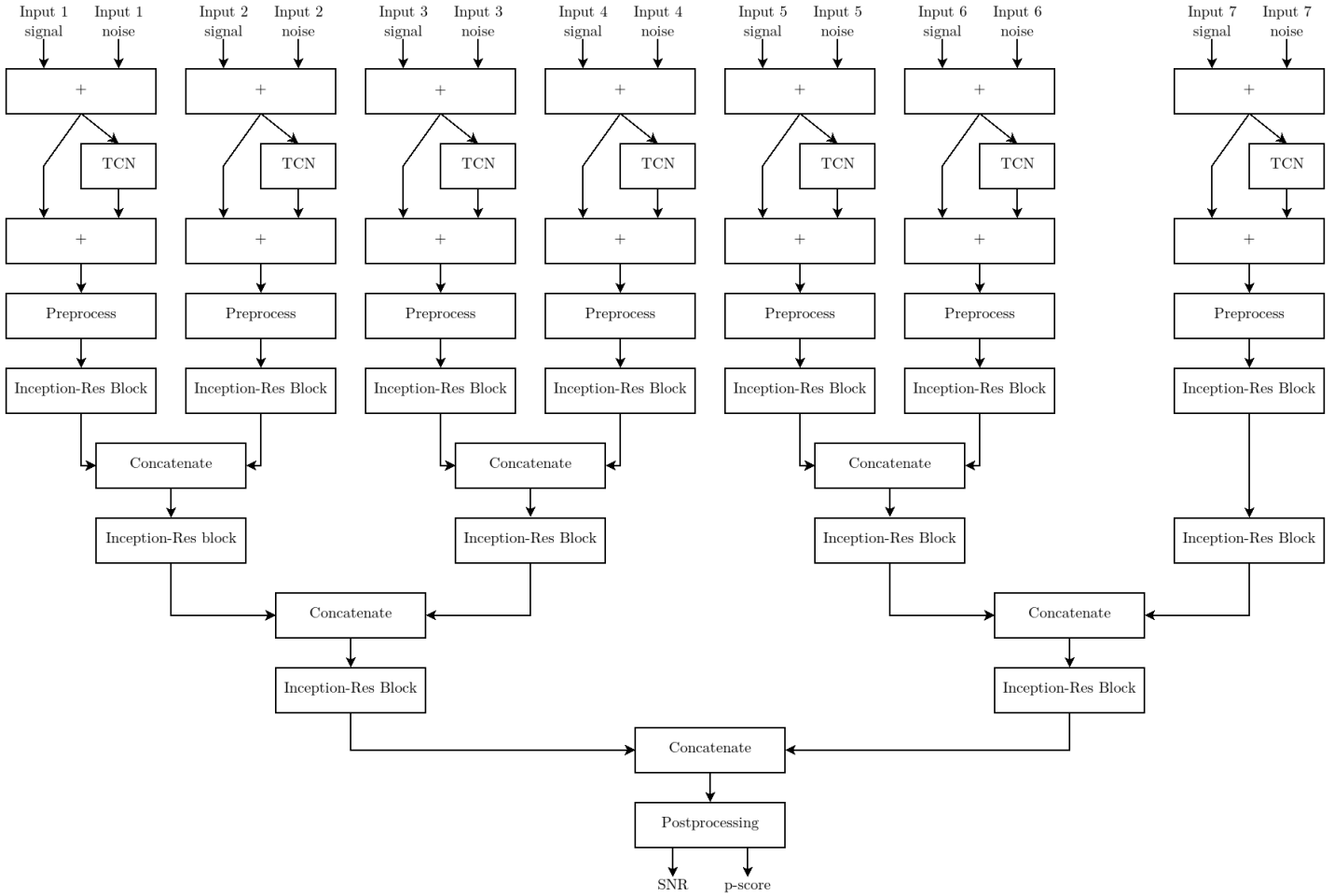


Figure 4.2: A high level overview of the architecture presented in [61]. Details on every block can be found in [66]. The network takes signal and noise inputs 1 to 7, where each number corresponds to a different part of the re-sampled raw data. It outputs an estimate of the SNR contained in the input and a p-score, which rates how likely the data is to contain a BNS signal. Figure and caption were taken from [61].

parameter	uniform distribution
component masses	$m_1, m_2 \in (1.2, 1.6)\,M_\odot$
spins	0
coalescence phase	$\Phi_0 \in (0, 2\pi)$
polarization	$\Psi \in (0, 2\pi)$
inclination	$\cos\iota \in (-1, 1)$
declination	$\sin\theta \in (-1, 1)$
right ascension	$\varphi \in (-\pi, \pi)$
distance	$r^2 \in (0^2, 400^2)\,\mathrm{Mpc}^2$

Table 4.1: The astrophysically motivated distribution of parameters used to generate injections. These are used to estimate the FAR and sensitivity of the search algorithm specified in [61]. Table and caption were taken from [61].

Training and validation data were created by the same process. Noise is simulated from the advanced LIGO design sensitivity curve in its zero detuned high-power configuration [319]. Signals are generated using the TaylorF2 waveform approximant [86, 320, 321], with all parameters except for the distance r drawn from Table 4.1. The distance is set indirectly by fixing the optimal SNR to a value uniformly drawn between 8 and 15. For each sample in the training set, we generate 96 s of data, whiten it by the PSD model, and crop the resulting data to 32 s. The exact position of the merger time in the final data is varied by ± 0.25 s. We create noise for the LIGO-Hanford and LIGO-Livingston detectors and inject the projected waveforms into both. The test set consists of $\approx$ 101 days of continuous data split into multiple files. Injections are generated using the same waveform model and the distributions of Table 4.1. They are spaced by 180 s to 220 s. Our work in [61] corrects an error from [66] where the PSD was sampled too coarsely. This reduced the SNR of the signals below the expected value.

To apply the network to data of duration longer than the input, i.e. 32 s, a sliding window is used. Each window is whitened and re-sampled as described above. The window uses a step size of 0.25 s, matching the variation of the merger time in the training data. The outputs of the network are time series of SNR estimates and p-scores. We apply thresholds of SNR 4 and p-score 0.1 and cluster points exceeding the threshold if they are within 1 s of each other. Afterward, we compare the time of the maximum value of each cluster with the injections to determine true and false positive detections. We accept something as true positive, if it is within 3 s of an injection.

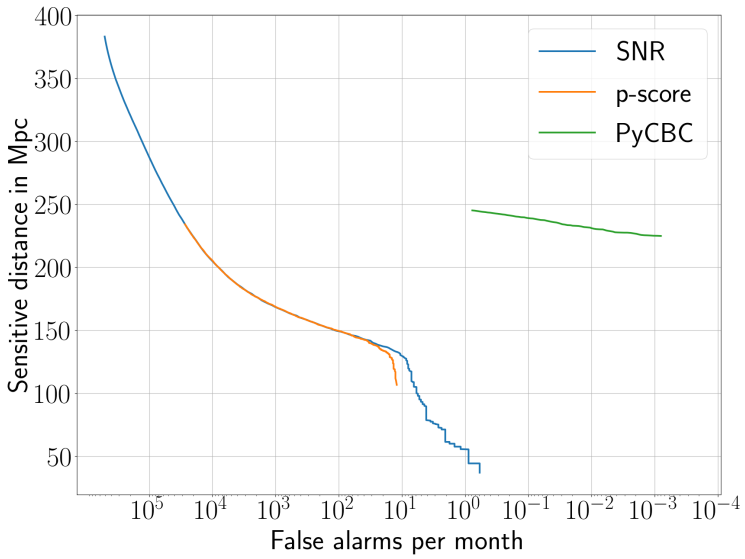


Figure 4.3: The sensitive distance as a function of the FAR. The blue curve shows the sensitive distance when the SNR is used to classify events. The yellow curve shows the sensitive distance when the p-score is used. The green curve is generated from the data found in [163] by counting all signals at a higher injection SNR than the corresponding FAR. We are able to resolve a small overlap-region between the two different searches but find that the sensitivity of our search drops close to zero for FARs below 10 per month. At high FARs both outputs of our network perform equally well, for low FARs the SNR shows superior performance. Figure and caption were taken from [61].

4.3 Results

The sensitivity of our proposed algorithm is summarized in Figure 4.3. Our testing procedure allows us to calculate sensitivities down to a FAR of 0.3 per month. We find that the SNR output collapses to zero sensitivity below a FAR of 0.6 per month, while the p-score collapses already at a FAR of 12 per month. This is an improvement over [66], which was incapable of resolving FARs below 30 per month. The algorithm shows non negligible sensitivity down to a FAR of 10 per month in the SNR output and 20 per month in the p-score output, where it reaches a sensitive distance of $\approx 130\,\mathrm{Mpc}$. We also find that at low FARs the SNR output is more sensitive than the p-score output, which stands in contrast to our previous findings in [66].

We compare our analysis to PyCBC Live [58]. To estimate the sensitivity of PyCBC Live, we use Figure 1 from [163] to obtain a ranking statistic $\mathcal{R}$ as a function of FAR. We then assume that all injections with optimal network SNR $> \mathcal{R}$ are found by PyCBC Live. The green curve in Figure 4.3 shows the resulting sensitivity curve. We find that PyCBC Live achieves about twice the sensitive distance measured for our algorithm at a FAR lower by about one order of magnitude. We also measure the latency introduced by our algorithm to compare it to the latency of PyCBC live. Ignoring pre-processing, our algorithm is capable of producing alerts in real time and introduces an average latency of 10.2 s. Restricting PyCBC Live to the parameter region of Table 4.1 results in a template bank containing 1960 templates per detector, which can be used to filter the data for both detectors on a single CPU core in real time. This analysis also introduces a latency of $\mathcal{O}(10)$ seconds, making it at least as computationally efficient as our deep learning alternative.

We also compare our search algorithms to another deep learning based BNS signal detection pipeline published as [287]. The original pre-print [322] was revised before publication as [287] shortly before we published our study. For this reason we compare our approach to both versions. The pre-print study [322] gave signal strength in terms a peak signal-to-noise ratio (pSNR), which we estimate to be related to optimal SNR by SNR = 41.2pSNR. Both the pre-print as well as the published version operate only on data from a single detector, whereas our algorithm uses data from two detectors. For this reason, we also scale the SNR from [322] and [287] by a factor of $\sqrt{2}$ to estimate the network SNR. The comparison in terms of true positive rate can be found in Figure 4.4 at different FARs. We find that our approach is about 4 times as sensitive as the one presented in [287] in the SNR region our network was trained on, which is marked by the gray area in the plot. However, our algorithm falls off rapidly for loud signals and only saturates at a sensitivity of 100% for SNRs > 46.65.

4.4 Conclusions

We introduced a novel procedure of making long duration time domain data accessible to NNs, by sampling the data at multiple rates. This multi-rate sampling was used to train and evaluate a deep learning based search algorithm for GWs from BNS mergers. We compared it to the state-of-the-art matched filter based low-latency search pipeline PyCBC Live and found that it is neither computationally more efficient nor as sensitive. This shows that more work is required to build competitive deep learning search algorithms for complex signals.

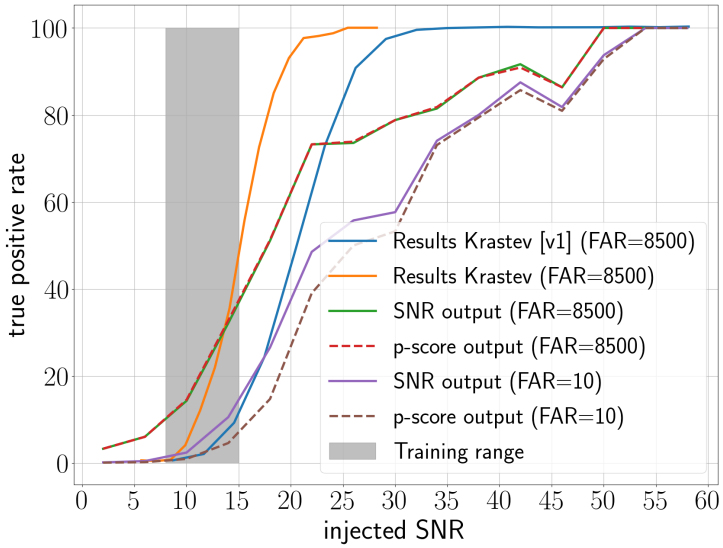


Figure 4.4: To compare our search to the work of [287] we plot their true positive rate at a fixed FAR of 8500 per month in yellow and our true positive rate at the same FAR in green and red. On the x-axis we track the injected optimal network SNR. The blue curve shows the data from [322], where the results were given in terms of pSNR. We use the conversion SNR $= 41.2 \cdot \sqrt{2} \cdot$ pSNR. To obtain these curves we bin the injected signals by their optimal injection SNR and a bin size of 4. For high SNRs some bins are empty. Empty bins are interpolated linearly from the remaining data. The area marked gray highlights the region covered by the training set. We find that our search performs better for low SNRs but is less sensitive for strong signals. We also show the true positive rate of our search at a FAR of 10 in purple and brown. Within the training range we find that our search closely matches the true positive rate of [322] at a higher FAR. Figure and caption were taken from [61].

We also compare our analysis to another deep learning BNS search algorithm. Our algorithm was approximately four times as sensitive to signals with SNR ≤ 15 but could not generalize well to louder signals.

Finally, we proposed an evaluation scheme that produces results that are comparable to existing pipelines and is normalized to the injected population of GW sources. We used this scheme to test our algorithm down to a FAR of 0.3 per month. This analysis showed that the sensitivity of our deep learning based algorithm drops to zero for FARs that are low compared to FARs at which deep learning algorithms are usually tested. Following [289], we argued that using FAPs, which are derived on discrete samples does not translate well to performance measures based on the more physically relevant FARs, as clustering effects are disregarded for FAPs.

Chapter 5

Gravitational-wave Merger Forecasting: Scenarios for the Early Detection and Localization of Compact-binary Mergers with Ground-based Observatories

This chapter summarizes the work published as [62]. It discusses a classical low-latency search for pre-merger detection of BNS systems. It is covered in this thesis, as machine learning is often stated to be a tool to decrease latency. Having a reference point of existing algorithms that are capable of pre-merger detection is important to determine research areas where machine learning algorithms may be of practical use.

5.1 Introduction

GW170817 [24] was the first detected GW event emitted by a BNS merger. It was accompanied by an EM counterpart observed in multiple frequency bands [27]. The earliest EM signal was a gamma ray burst observed by Fermi-GBM and INTEGRAL [323–325] about 1.7 s after the GW merger. The first optical observations began only about 11 hours after the merger [25]. These EM observations provided new insight into the nuclear equation of state [18, 19, 326–329], the Hubble constant [20, 330, 331], the phenomenon of kilonova (see [332] and references therein), and the central engine of short gamma-ray

bursts [62, 333–335].

If early observations of the optical band would have been available, different kilonova emission models could have been differentiated [336]. It is also hypothesisized that pre-merger EM emissions exist [337–342], which could be constrained by pre-merger observations. However, early or even pre-merger EM observation of CBC mergers are constrained by two factors: The latency between GW signal detection and telescopes being alerted, as well as the ability of EM instruments being able to cover the sky area where the source is estimated to be located.

This study assesses the possibility of generating pre-merger alerts for different eras of detectors and quantifies the localization error associated with them. By providing pre-merger alerts, the latency is automatically reduced. Quantifying the localization error allows for EM observation strategies to be optimized and coordinated. To do so, the evolution of the pre-merger alert capabilities of ground based detectors over the next $\mathcal{O}(10)$ years are evaluated. Additionally, the capabilities of the current low-latency PyCBC Live [58, 343] analysis to generate early warnings is tested.

5.2 Methods

Our study covers the five current and planned ground based GW observatories of the coming decade LIGO-Hanford (H), LIGO-Livingston (L) [6], LIGO-India (I) [192], Virgo (V) [7], and KAGRA (K) [8]. For this network of detectors, we consider three different eras: "Design", "A+", and "Voyager". The "Design" era expects the four detectors LIGO-Hanford, LIGO-Livingston, Virgo, and KAGRA to be operational at their 2021-2022 design sensitivity [194]. Starting with the "A+" era, we expect LIGO-India to be operational and matching the sensitivity of both LIGO-Hanford and LIGO-Livingston. We use the design PSD of the planned upgrades to the LIGO instruments from 2024-2026 for the three LIGO detectors [344]. For Virgo and KAGRA we conservatively assume that their sensitivity will not improve beyond their "Design" era. The "Voyager" era assumes the PSD of the three LIGO detectors to match the "Voyager" plans [45]. A follow-up study covered the third generation detectors ET and CW [345].

To assess the capability of pre-merger detection in the different configurations discussed in [62], we simulated a population of $\mathcal{O}(10^5)$ BNS mergers. This population is distributed uniformly in volume and isotropically in sky location and binary orientation. The masses are chosen to be a reference binary with $1.4 - 1.4\,\mathrm{M}_\odot$, but the results can be generalized to arbitrary masses (see Section 4 in [62]). The signals are subsequently injected into simulated,

Gaussian noise, colored by the PSD of different detectors and eras.

We pursue two different criteria for signal detection. To gauge the general capability of pre-merger detection, our first method assumes the detection of a signal if the combined optimal network SNR > 10. This choice is consistent with the threshold for confidently detected mergers in [212, 346]. To determine the pre-merger detection capabilities, we calculate the network SNR as a function of time before merger. Once the network SNR exceeds the threshold, it is counted as detected and we generate a posterior of the sky location using Baystar [347] for every time step. The second method involves a full search using PyCBC Live in a low-latency configuration at a FAR of 1 per year. To adapt it to pre-merger detection, the template bank is a combination of several template banks using different pre-merger truncation times. The truncation of the different banks are chosen such that they cover 5% increments of the total expected SNR. This second analysis allows us to test how capable existing analysis methods are at pre-merger detection. Sky localization is again performed using Baystar [347].

5.3 Results

The core results of the study are given in Figures 5.1 to 5.3. They show the search sensitivity, localization error accuracy, and detection rate estimates as functions of time of detection before merger. The two columns compare the network of currently operational detectors with the planned detector network for each era. The times on the horizontal axis do not include the latency of the analysis or any other processing. They merely reflect the upper frequency cutoff at which the two analysis methods can detect them.

From the plots, we find that the PyCBC Live analysis is comparable to the idealized search and even outperforms it at a FAR of 1 per month. This shows that current analysis methods are fully capable of pre-merger detections at the calculated rates. Assuming a merger rate density of $1000\,\mathrm{Gpc^{-3}yr^{-1}}$, we find that one source per year can be expected to be detected with a 90% confidence localization of < 100 deg^2 and an early warning of 18 s, 54 s, and 195 s for the "Design", "A+", and "Voyager" era, respectively. If the confidence region is reduced to 50%, the early warning increases to 34 s, 104 s, and 335 s. Alternatively, the pre-merger warning can be kept constant to increase the expected number of detections to $4-6$ sources per year. This strategy would more than double the number of detected EM counterparts, assuming that 50% of the counterparts lie outside the credible region.

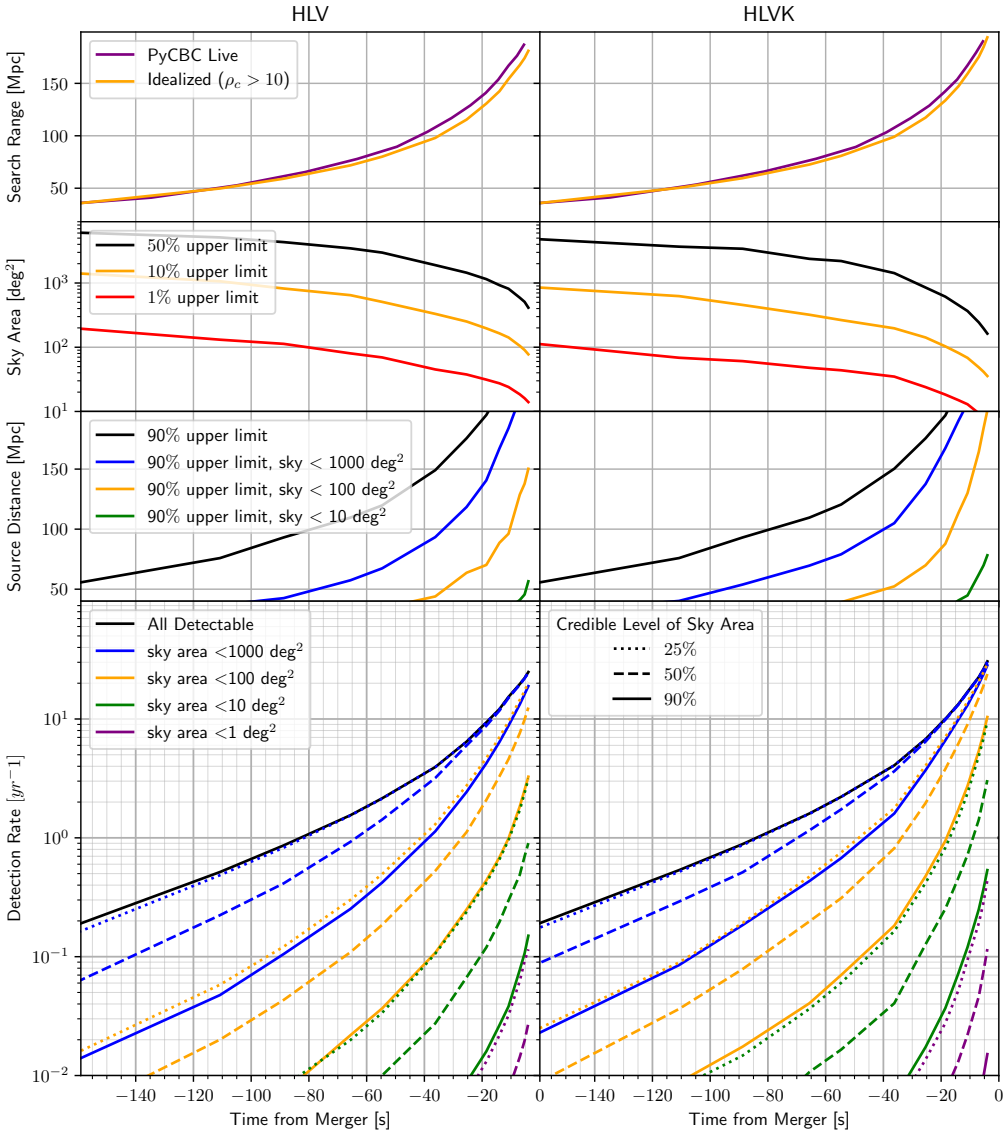


Figure 5.1: "Design" era (2021-2022) detection and localization for the HLV network (left) and the full gravitational-wave detector network (right) as a function of time before merger for a fiducial 1.4-1.4$M_\odot$ BNS merger. (Top) The sky-averaged detection range for the idealized search and PyCBC Live operating at a false alarm rate of once per year. (Middle) The upper limit on the localization sky area and source distance, respectively, for detectable sources. Sky areas are quoted at the 90% credible level. (Bottom) The detection rate of all sources (black) and those that also have a sky localization less than 1000 deg^2 (blue), 100 deg^2 (orange), 10 deg^2 (green), or 1 deg^2 at a 90% (solid), 50% (dashed), and 25% credible level (dotted). Figure and caption are taken from [62].

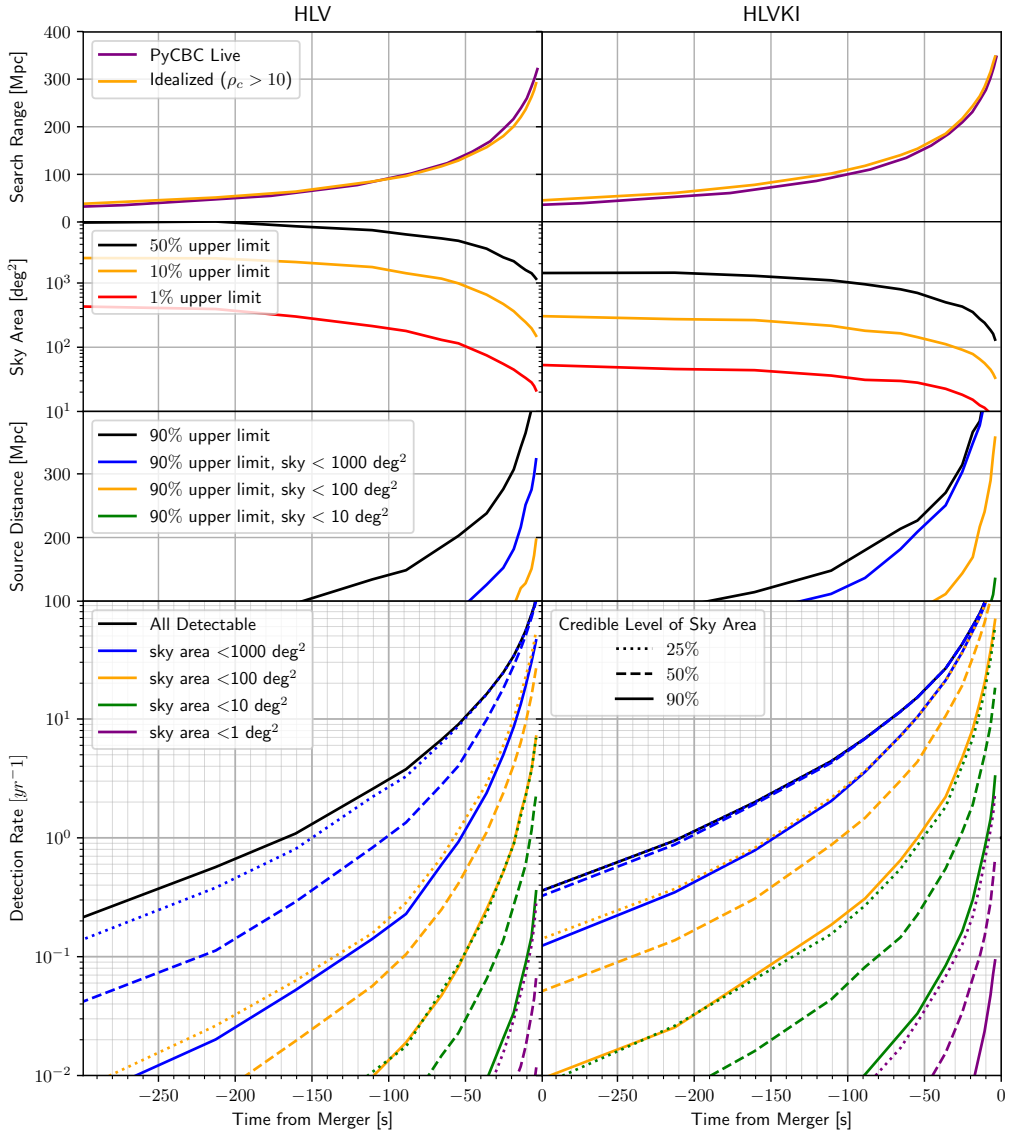


Figure 5.2: "A+" era (2024-2026) detection and localization for the HLV network (left) and the full gravitational-wave detector network (right) as a function of time before merger for a fiducial 1.4-1.4$M_\odot$ BNS merger. (Top) The sky-averaged detection range for the idealized search and PyCBC Live operating at a false alarm rate of once per year. (Middle) The upper limit on the localization sky area and source distance, respectively, for detectable sources. Sky areas are quoted at the 90% credible level. (Bottom) The detection rate of all sources (black) and those that also have a sky localization less than 1000 deg^2 (blue), 100 deg^2 (orange), 10 deg^2 (green), or 1 deg^2 at a 90% (solid), 50% (dashed), and 25% credible level (dotted). Figure and caption are taken from [62].

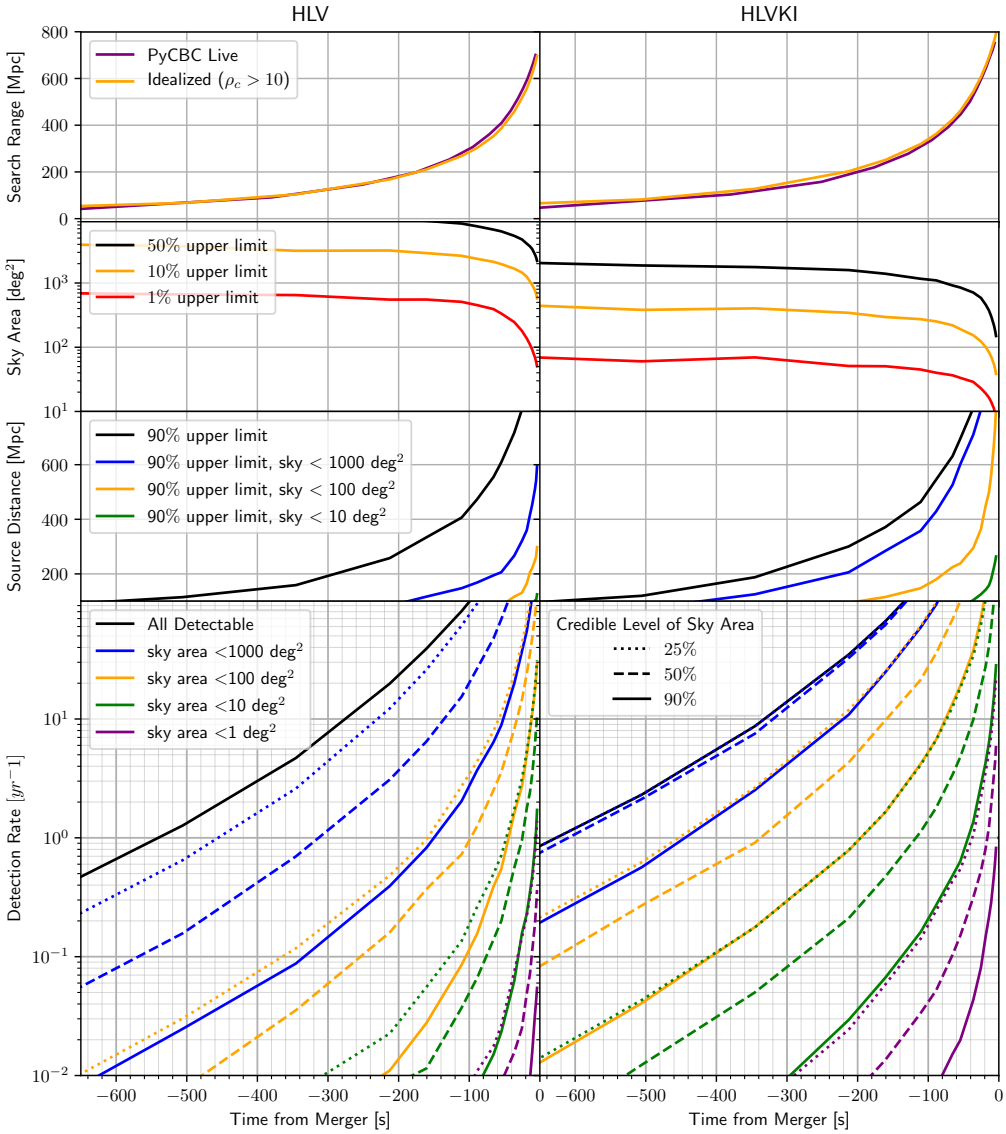


Figure 5.3: "Voyager" era (late 2020's) detection and localization for the HLV network (left) and the full gravitational-wave detector network (right) as a function of time before merger for a fiducial 1.4-1.4$M_\odot$ BNS merger. (Top) The sky-averaged detection range for the idealized search and PyCBC Live operating at a false alarm rate of once per year. (Middle) The upper limit on the localization sky area and source distance, respectively, for detectable sources. Sky areas are quoted at the 90% credible level. (Bottom) The detection rate of all sources (black) and those that also have a sky localization less than 1000 deg^2 (blue), 100 deg^2 (orange), 10 deg^2 (green), or 1 deg^2 at a 90% (solid), 50% (dashed), and 25% credible level (dotted). Figure and caption are taken from [62].

5.4 Conclusions

We have shown that for different detector network configurations in the coming decade the possible pre-warning time for a BNS merger may increase from $\mathcal{O}(10)$ to $\mathcal{O}(100)$ seconds. However, this amount of pre-warning is still insufficient for many observatories for re-pointing and tiling a 100 deg^2 area [348]. Notable exceptions include Swift [349], ZTF [350, 351], MASTER [352], and CTA [353]. Observatories that are not capable of re-point and tile the sky region in the provided early warning time may still benefit by adjusting their observing configurations [354].

We hope that this roadmap provides grounds for the observing community to plan continued and automated observations with existing instruments, as well as further motivation to build new instruments with different observation bands. This includes concepts such as the Transient Astrophysics Probe [355]. If these hopes are met, it seems plausible that the first BNS detection with simultaneous observation of the prompt EM emissions can be made within the next decade.

Chapter 6

Training Strategies for Deep Learning Gravitational-Wave Searches

Contents

6.0 Abstract 102

6.1 Introduction 102

6.2 Methods 105

6.2.1 General setup 105

6.2.2 Network performance 109

6.2.3 Training strategies 113

6.2.4 Matched-filter baseline 114

6.3 Results 114

6.3.1 Sensitivities 114

6.3.2 Training strategies 123

6.4 Conclusions 126

6.5 Appendix: Efficiency curve examples 128

This chapter is essentially a full reprint of [63] with minor edits for formatting. It discusses the influence of different training strategies on deep learning GW search algorithms and introduces a novel way of treating the network output.

6.0 Abstract

Compact binary systems emit gravitational radiation which is potentially detectable by current Earth bound detectors. Extracting these signals from the instruments' background noise is a complex problem and the computational cost of most current searches depends on the complexity of the source model. Deep learning may be capable of finding signals where current algorithms hit computational limits. Here we restrict our analysis to signals from non-spinning binary black holes and systematically test different strategies by which training data is presented to the networks. To assess the impact of the training strategies, we re-analyze the first published networks and directly compare them to an equivalent matched-filter search. We find that the deep learning algorithms can generalize low signal-to-noise ratio (SNR) signals to high SNR ones but not vice versa. As such, it is not beneficial to provide high SNR signals during training, and fastest convergence is achieved when low SNR samples are provided early on. During testing we found that the networks are sometimes unable to recover any signals when a false alarm probability $< 10^{-3}$ is required. We resolve this restriction by applying a modification we call unbounded Softmax replacement (USR) after training. With this alteration we find that the machine learning search retains $\geq 91.5\%$ of the sensitivity of the matched-filter search down to a false-alarm rate of 1 per month.

6.1 Introduction

The direct detection of a gravitational wave (GW) on September 14, 2015 [13] started the era of GW astronomy. After the analysis of two and a half observing runs, tens of GWs have been confirmed [206, 356]. GW170817 [24] was the first GW event also seen in the electromagnetic spectrum [26, 27, 323, 324].

The latency between a GW and its reported detection is a vital aspect of multi-messenger missions. Lowering the delay between data aggregation and signal detection allows to maximize the electromagnetic observation time and reduces the risk that early emissions are being missed.

To extract GW signals from the instrument data, a well-established technique known as *matched filtering* is used in many search algorithms. It convolves *templates*, i.e. pre-calculated models of the expected signals, with the measured data [36, 58, 158, 161, 357]. When one of these templates matches the data to a given degree and the data quality is high enough, these searches report a candidate detection.

Matched filtering is known to be optimal in stationary Gaussian noise when accurate models of the waveform exist [30]. However, it can be computationally limiting when many templates are required. This is the case when effects such as higher-order modes [41], precession [40] or eccentricity [358] are considered. Furthermore, signals which are not covered by the filter bank may be missed entirely. While there are unmodeled searches that detect coincident excess power in different detectors [37–39], they are less sensitive in regions where accurate models exist.

Recently, new deep learning based searches have started to be explored [55, 56, 61, 287, 302, 359]. Summaries of the current state of the field are given in [48, 360]. The pioneering works by George et al. [55] and Gabbard et al. [56] demonstrated that deep neural networks are capable of detecting GWs from two merging black holes (BBH). The networks have also proven to generalize to signals with previously unseen parameters [55, 361]. It was shown that these algorithms can distinguish data containing a GW from pure noise as well as matched filtering with a false-alarm probability (FAP) down to 10^{-3}. That means, the networks were tested down to a level at which about 1 in 1000 pure noise samples was falsely classified as containing a signal.

The authors of [289] find that the FAPs determined by the original studies do not directly translate to false-alarm rates (FARs) on continuous data streams. For FARs, the appropriate question to ask is how many false signals does the network identify per time interval of continuous data, as opposed to how many uncorrelated data chunks are falsely identified as containing a signal. The effects of clustering subsequent outputs when the network is applied via a sliding window have to be accounted for. Comparing deep learning searches to traditional matched-filter searches is, therefore, not trivial because matched-filter searches typically operate at FARs that are orders of magnitude smaller than what has been tested for early neural networks. In [61] we suggested a standardized testing procedure which produces statistics which are comparable to traditional search algorithms to resolve these issues.

In this paper we reanalyze and extend the results given in the initial papers [55, 56]. Our motivation is twofold. First, we want to verify and test the performance of the networks quoted in those papers. Specifically, we apply the testing procedure outlined in [61]. Second, we discuss how the GW data is prepared for and presented to the network. The form of data preparation is often taken as a given, while comparatively more work is invested in finding a network structure that suits the problem. We carefully examine the influence of different choices of data presentation and training strategies on the ability to detect signals given a fixed network.

Here we focus on signal detection. The problem of deep learning param-

eter estimation is another vital and active area of research. Multiple groups have made advancements in this field [55, 311–313, 317, 362].

We use the network presented by Gabbard et al. [56] for most of our studies. It is trained on simulated data containing GWs from BBHs with individual black hole masses ranging from $10\,M_\odot$ to $50\,M_\odot$. The search is restricted to a single detector. This restriction reduces the parameter space to the two component masses, the orbital phase, the distance to the source and the time of coalescence.

The network classifies segments of 1 s duration sampled at 2048 Hz into the two categories "noise + signal" and "noise" by returning a value between 0 and 1 we call "p-score". A larger p-score corresponds to a higher confidence of the network that the input contains a signal.

To optimize the training strategy, we focus on the difference between *curriculum learning* [363] and *fixed interval training*. Fixed interval training uses a single training set, i.e. a single, fixed range of signal-to-noise ratios (SNRs). Curriculum learning lowers the SNR of the training signals progressively, thus increasing the complexity with time. We evaluate different variants of both strategies. In total 15 different approaches are tested.

Each strategy is applied to 50 randomly initialized networks. We do this to guard against favorable initializations. All tests are done with two different implementations, to further increase robustness of our results. The different implementations use the two core libraries Tensorflow [364] and PyTorch [365], respectively.

We find that most training strategies are capable of closely reproducing the results given in [56]. We do not see a significant difference in performance between curriculum learning and fixed interval training strategies. However, networks that had access to lower SNR signals during training generally outperformed those that only saw high SNR signals. We find that networks trained on fainter signals can generalize to loud ones, while the opposite is not the case.

Further analysis of the networks showed that the efficiency, which is the fraction of correctly classified input samples containing a signal at a given FAP, drops to zero beyond a FAP of 10^{-3} when the training is carried out for long enough. This drop is caused by numerical instabilities in the final activation and the comparatively low penalty of false positives. We propose a simple modification that does not require retraining of the network to push this problem to significantly lower FAPs. We call this modification unbounded Softmax replacement (USR).

We evaluate 3 different networks of each training strategy on a month of simulated data. The networks are applied using a sliding window with step size of 0.1 s. We follow the procedure outlined in [61] to analyze the

results. Our evaluation of the base line network is limited by $\mathcal{O}(10^3)$ false alarms estimated with perfect confidence to contain a signal. By applying the USR modification we are able to eliminate this restriction and can calculate sensitivities down to a FAR of 1 per month. For comparison, we construct a template bank and use it to do a matched-filter search on the same data used to evaluate the networks. We find that the machine learning search retains at least 91.5% of the sensitivity of a matched-filter search for all tested FARs and most strategies.

All code required to reproduce our analysis is public and can be found at [366].

The contents of this paper are structured as follows. In section 6.2 we describe the architecture, data sets, training strategies, and evaluation methods. We apply these in section 6.3 and describe our findings. In particular we describe the USR modification which allows the networks to be tested at low FAPs. We conclude in section 6.4.

6.2 Methods

6.2.1 General setup

We focus our studies on the network presented by Gabbard et al. in [56]. They used a convolutional neural network with 6 stacked convolutional layers followed by 3 fully connected layers. All but the last layer use an exponential linear unit (ELU) as activation function.

The architecture is altered in two details compared to the original version of [56]. We added a batch normalization layer before the first convolutional layer to take care of *input normalization*. Input normalization scales all inputs to have a mean-value of 0 and a variance of 1. This is standard practice in contemporary deep learning and has been proven to help the network train efficiently [271]. The second modification is a reduction of the pool sizes. This change was required because we lowered the sample rate of the data from 8192 Hz to 2048 Hz. We decided to lower the sample rate for multiple reasons. First of all, the detector sensitivity drops sharply above 1 kHz. Thus little to no SNR is lost by disregarding higher frequencies. For this reason current searches are often limited to the same frequency band as well [58, 367, 368]. Second, signals within our training set merge at much lower frequencies and do not exceed 1 kHz. Finally, as we will show in section 6.3, our training converged to the same state as previous works. We are thus confident that this reduction in the sample rate has no negative impact on the network's ability to detect signals. A reduction of the size of

Table 6.1: The modified neural network from [56] as used in this study. The given shapes correspond to the tensor shapes in the TensorFlow version of the code, i.e. data length × number of channels. PyTorch swaps these dimensions. The order of the layers is given by reading the column "layer type" from top to bottom and left to right. Layers are grouped by their influence on the output shape and by trainable weights.

layer type	kernel size	output shape
Input + BatchNorm1d		2048×1
Conv1D + ELU	64	1985×8
Conv1D	32	1954×8
MaxPool1D + ELU	4	488×8
Conv1D + ELU	32	457×16
Conv1D	16	442×16
MaxPool1D + ELU	3	147×16
Conv1D + ELU	16	132×32
Conv1D	16	117×32
MaxPool1D + ELU	2	58×32
Flatten		1856
Dense + Dropout + ELU		64
Dense + Dropout + ELU		64
Dense + Softmax		2

the input to a neural network usually also helps with training. The resulting network setup is depicted in Table 6.1.

All studies presented in subsection 6.2.2 and subsection 6.2.3 were carried out using the network from George et al.[1] [55] as well. However, with our particular training setup, every metric showed performance similar to the network from [56]. We present only results using the network based on the work of Gabbard et al.

Each GW signal is defined by the component masses m_1, m_2 and a phase ϕ_0. Two masses are drawn independently from a uniform distribution between $10\,\mathrm{M}_\odot$ and $50\,\mathrm{M}_\odot$ and the higher and lower values are assigned to m_1 and m_2, respectively, to enforce the condition $m_1 \geq m_2$. Phases are uniformly drawn from the interval $[0, 2\pi]$. We generate signals with 5 different phases for each pair of masses (m_1, m_2). The amplitude, and therefore the distance of the source, is determined by the target SNR we have chosen. We fix the sky position to be overhead the LIGO Hanford detector [6] and the

[1] We adjusted the network from George et. al. too, by using batch normalization for input normalization and reducing the sample rate of the input to 2048 Hz.

inclination as well as the polarization to 0, because in the case of nonprecessing signals and assuming a single detector, any variation in those parameters can be fully absorbed by modifications of the amplitude and phase of the signal.

The waveforms are generated with a sample-rate of 2048 Hz, a lower-frequency cutoff of 20 Hz, and using the model `SEOBNRv4_opt` [369] (optimized version of `SEOBNRv4` [118]). It is common practice to shift the location of the maximum amplitude by some small time within each training sample. This procedure allows the network to be less sensitive to the exact alignment of the waveform within its input. To achieve this behavior, we shift the position of the merger by a time uniformly drawn from $-0.1\,\mathrm{s}$ to $0.1\,\mathrm{s}$ before projecting onto the Hanford detector. After the projection the signals are *whitened* using the analytic model of LIGO's design sensitivity at its zero detuned high power configuration [319], i.e., we divide the Fourier transformed signal by the square root of the power spectral density (PSD) associated with the power of the background noise at different frequencies, and transform back to the time domain. Whitening the data reduces the power at frequencies where the detector is known to be less sensitive. Next, the waveforms are scaled to an optimal SNR of 1. The optimal SNR ρ_{opt} is defined by

$$\rho_{\mathrm{opt}}{}^2 = 4\mathrm{Re}\left[\int df \frac{\tilde{h}(f)\tilde{h}^*(f)}{S_n(f)}\right], \tag{6.1}$$

where $\tilde{h}$ is the Fourier transform of the time domain signal, before it was whitened, $\tilde{h}^*$ is its complex conjugate, S_n is the PSD and Re extracts the real part of the complex number. Finally, we extract a time slice such that the original, not shifted merger time is located 0.7 s from the start of the window.

All noise is simulated from the same PSD used to whiten the signals. After generation, the noise is whitened by the PSD used to create it in the same way the signals are whitened. We choose to explicitly whiten the colored noise to take into account any artifacts the process may introduce. This also eliminates sources of errors and is in principle extendable to real noise.

The whitened signals and noise samples are combined during training. This allows us to rescale the signals at runtime to a desired strength. Since during generation all signals are scaled to SNR 1, rescaling is achieved by a multiplication of the signal with the target SNR.

We have briefly tested training on frequency domain data. This was motivated by studies such as [291, 302]. While these studies analyze longer duration signals, there is no conceptual problem to using the frequency representation of short BBH waveforms. To accommodate the complex valued

frequency representation we changed the input layer in Table 6.1 to a shape of 1025×2 and inserted the real and imaginary parts as different channels. With this being our only modification to the architecture, the network was able to differentiate data containing a signal from pure noise but found up to 65% fewer signals at low SNR. We suspect that with greater effort in finding an optimized architecture, one could regain the performance of the network on time domain data.

We have also explored training on raw data, i.e. data where the computationally expensive whitening is skipped. Even after several hundred epochs the network was not able to distinguish data containing signals from pure noise, irrespective of using the time or frequency domain representation of the data.

Our training set contains 20 000 unique combinations of component masses each of which is used to generate 5 waveforms with random coalescence phases. Therefore, it contains 100 000 individual signals. We generate 200 000 independent noise samples, 100 000 of which are used in combination with the signals. The remaining 100 000 noise samples are used as pure noise. Our training set, therefore, contains 200 000 independent samples.

The validation set is assembled in the same way as the training set. It too contains 100 000 samples of the "signal"-class and 100 000 samples of the "noise"-class for a total of 200 000 samples. The validation set was chosen to be of equal size to the training set due to its influence when curriculum training strategies are used. The conditions for when the complexity of the training set is increased are evaluated on this set.

We use a third data set to calculate relevant metrics of the network during training. This third set is required, as the validation set directly influences the training for curriculum strategies. Metrics determined on the validation set may, therefore, be biased. We call this third set the efficiency set and describe its usage in subsection 6.2.2. It contains 10 000 unique signals and 400 000 independent noise samples.

Finally, we evaluate the performance of the network on a test set. This test set contains a month of simulated Gaussian noise with injections separated by a time uniformly distributed in the interval $[16, 22]$ s. The injection parameters are drawn from the distributions shown in Table 6.2. Noise is generated using the same PSD used for the training set. A month of data corresponds to ~ 26 million correlated samples.

Each network is trained for 200 epochs, i.e. 200 full passes on the training set. We found this to be a sufficient number of training cycles for most of the networks to converge to a stable performance on the validation set. We use the default implementations of the Adam optimizer with a learning rate of 10^{-5}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$ [253]. As loss we use a variant of

Table 6.2: Injection parameters for the data set used to determine the FAR and sensitive volume of the different networks.

Parameter	Uniform distribution
Component masses	$m_1, m_2 \in (10, 50)\,\mathrm{M}_\odot$
Spins	0
Coalescence phase	$\Phi_0 \in (0, 2\pi)$
Polarization	$\Psi \in (0, 2\pi)$
Inclination	$\cos\iota \in (-1, 1)$
Declination	$\sin\theta \in (-1, 1)$
Right ascension	$\varphi \in (-\pi, \pi)$
Distance	$\mathrm{d}^2 \in (500^2, 7000^2)\,\mathrm{Mpc}^2$

the binary crossentropy that is designed to stay finite,

$$L(\boldsymbol{y}_\mathrm{t}, \boldsymbol{y}_\mathrm{p}) = -\frac{1}{N_\mathrm{b}} \sum_{i=1}^{N_\mathrm{b}} \boldsymbol{y}_{\mathrm{t},i} \cdot \log\left(\epsilon + (1 - 2\epsilon)\boldsymbol{y}_{\mathrm{p},i}\right). \tag{6.2}$$

Here $\boldsymbol{y}_\mathrm{t}$ is either $(1, 0)^T$ for data containing a signal or $(0, 1)^T$ for pure noise, $\boldsymbol{y}_\mathrm{p}$ is the prediction of the network, $N_\mathrm{b} = 32$ is the mini-batchsize, and $\epsilon = 10^{-6}$.

6.2.2 Network performance

A common metric when training neural networks is the *accuracy*, which is the ratio of correctly classified samples over the total number of samples. This approach weighs false-negatives and false-positives equally.

GW searches assign a statistical significance to each event. This is usually given as the FAR of the search at the ranking statistic threshold associated with the candidate event. For the network we use the p-score as ranking statistic. The more false positives a search produces at a given ranking statistic, the less significant each event becomes. Therefore, false-positives severely limit the ability of the search to recover true events. Low latency searches do not distribute any event candidates publicly with a FAR greater than ~ 1 per month [58]. For searches which operate on archival data, low FARs are needed to assign a probability for the signal to be of astrophysical origin, based on the expected astrophysical rate of comparable events [206, 356].

For these reasons we monitor the *efficiency* of the network rather than the accuracy. The efficiency is the true-positive probability at a fixed false-positive probability, i.e., a fixed FAP. To do so, we sort the p-score outputs

of the network on the noise from the efficiency set and use the x-th largest as a threshold, where we choose

$$x = \lfloor N_n \cdot \mathrm{FAP} \rfloor \tag{6.3}$$

Here, N_n is the total number of noise samples used and $\lfloor \cdot \rfloor$ denotes the flooring operation. We then evaluate the signals from the efficiency set scaled to SNRs $3, 6, 9, 12, 15, 18, 21, 24, 27$ and 30 and count the samples that exceed the threshold. The efficiency is then given by

$$\text{efficiency} = \frac{N_{s>t}}{N_s}, \tag{6.4}$$

with $N_{s>t}$ being the number of signals assigned a p-score larger than the threshold and N_s the total number of signals. To get a better understanding of the efficiency as a function of the signal strength, we also calculate the efficiencies at each of the SNRs individually. In this work, a FAP of 10^{-4} is used for all efficiency calculations.

For each of the strategies we discuss in subsection 6.2.3 the network is trained 50 times from scratch. The parameters of the networks are initially random for each run. The final performance of a single network may depend on these initial values. Training each network-strategy combination multiple times and averaging over their efficiencies reduces the influence of the network initialization, thus yielding greater insight into the impact of the training strategy.

After training has completed for all 50 networks, we choose 3 networks for which we calculate the sensitive volume and the false-alarm rate on a month of simulated data. The networks are chosen by the following scheme. We select the epoch of the maximum efficiency of all networks. At this epoch we pick the best and the worst performing networks, where ranking is based on the efficiency. The last network is chosen to be the one which has the efficiency closest to the average efficiency over all 50 runs at the chosen epoch.

The sensitivity and FAR calculation follows the procedure outlined in [61]. As suggested in [55, 56], the network is applied to time series data of duration longer than the input window via a sliding window. We choose a step size of 0.1 s to ensure the correct alignment of the merger time within the input window for at least one step. Each window is whitened individually using the same method and noise model applied to the training set.

To reduce the computational cost, the data are sliced into the input windows and preprocessed only once. We store this sliced data and apply the different networks to it. This allows us to evaluate the entire month of data in about 1 h on a single NVIDIA RTX 2070 SUPER.

The network outputs a value between 0 and 1 for every slice. A value of 1 corresponds to the network being confident that it has seen a signal. We use this output as ranking statistic. Outputs that exceed a threshold, which we call trigger-threshold, are clustered by their time. Within each cluster the first time where the output becomes maximal is picked. The combination of this time and the corresponding network output is called an event.

The list of events is compared to the known injection times. If the event is separated from the closest injection by more than some maximum time it is called a false positive. Otherwise we consider it a true positive. From these we can calculate the FAR as well as the sensitive volume as detailed in [61]. The FAR is given by

$$\mathrm{FAR} = \frac{N_f}{T_o} \ , \tag{6.5}$$

where N_f is the number of false positives and T_o is the duration of the analyzed data. When the injections are distributed uniformly in volume the sensitive volume of the search is given by

$$V\left(\mathrm{FAR}\right) = V\left(d_{\max}\right) \frac{N_t\left(\mathrm{FAR}\right)}{N_i}, \tag{6.6}$$

where $d_{\max}$ is the maximum distance at which sources are injected, $V\left(d_{\max}\right)$ is the volume of a sphere with radius $d_{\max}$, N_i is the total number of injections and $N_t\left(\mathrm{FAR}\right)$ is the number of true positives at a given FAR. The FAR can be adjusted by considering only events above a given threshold. To convert the sensitive volume to a distance we calculate the radius of a sphere of the given volume.

We use a p-score of 0.1 as our trigger-threshold. Triggers are said to belong to a cluster if they are within 0.2 s of the cluster bounds. An event is called a true-positive if there was an injection within 0.3 s of the reported event time. Otherwise it is a false-positive. We chose the cluster boundary time as twice the step size to allow for modest smoothing of the network output, while keeping it short compared to the average duration of a signal ($\mathcal{O}\left(1\,\mathrm{s}\right)$). The maximum separation between an event and the corresponding injection was chosen to be larger than the cluster boundaries but still small compared to the average signal duration. None of these parameters were optimized.

Figure 6.1 shows example output from one of the networks. The top panel shows the raw input with the injected waveform overlayed in black. The injection time is marked with a red vertical line and the grey lines highlight $\pm 0.3\,\mathrm{s}$ where events are true positives. The bottom panel shows the network output for the corresponding time. The black vertical lines show the events returned by the search.

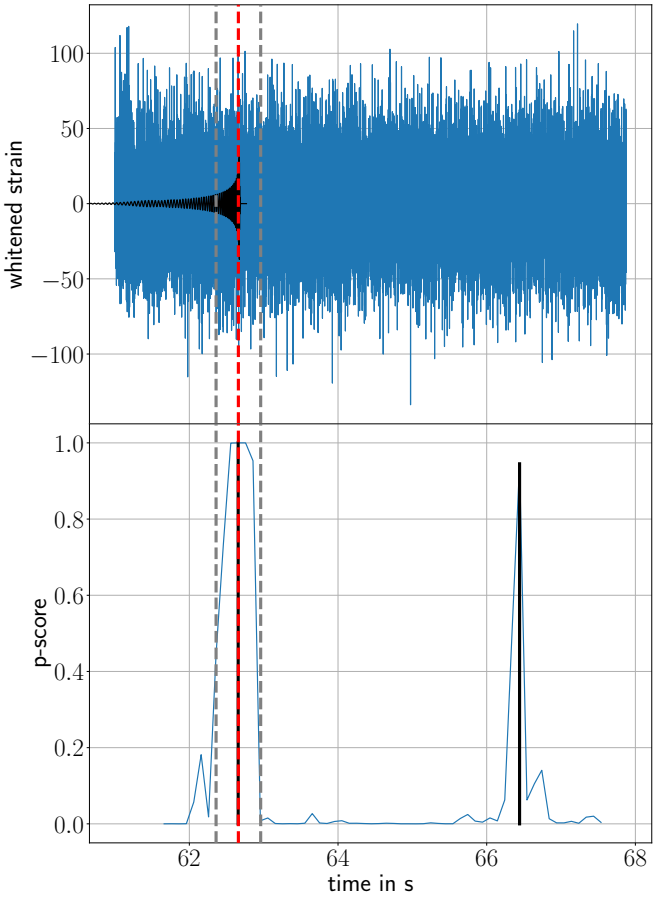


Figure 6.1: A sample output from the network on long duration data. The top panel shows the whitened input data. The injected signal is overlayed in black. The red vertical line signifies the time of the injection, i.e. the time that would ideally be returned by the search algorithm. The vertical grey lines mark the interval within which a returned event is classified as a true positive. The bottom panel shows the output of the network corresponding to the input. The vertical red and grey lines, again, show the true injection time and the allowed interval for true positives respectively. The black vertical lines mark the events returned by the search. Their height is the p-score attributed to the event. While the first event is a true positive, the second event is a false-positive originating from noise.

6.2.3 Training strategies

The two initial publications by George et al. [55] and Gabbard et al. [56] disagree on the usefulness of curriculum learning. Whereas George et al. find a noticeable improvement by using curriculum learning, Gabbard et al. find no difference in the final performance of the network.

We aim to determine the impact curriculum learning has on the final performance and speed of convergence of these networks. By doing so we optimize the sensitivity of the networks tested here and hope that our findings generalize also to state-of-the-art machine learning search algorithms [57, 61, 287, 312].

Our study contains 10 curriculum learning and 5 fixed interval training strategies. An overview can be found in Table 6.3. The minimum SNR allowed in any of these strategies is ≥ 5. We choose SNR 5 as a lower bound as this is roughly the lowest single detector SNR at which signals seen in multiple detectors can be confidently distinguished from noise [206, 356].

We test 5 different conditions for the optimal SNR contained in the training data for both types of strategies. For curriculum strategies, these conditions prescribe when the SNR of the training data is lowered. For fixed interval strategies the conditions are the interval from which the SNR for each sample is drawn.

Curriculum strategies use either the validation loss, the validation accuracy, or the number of epochs since the last step as conditions. For validation loss and validation accuracy we choose either a threshold or wait until the values stabilize and do not improve anymore. The latter are labeled by a prefix "plateau" throughout this paper. We choose a threshold of 0.95 for the validation accuracy and 0.2 for the validation loss. These values are arbitrary but proved to work well. When using the plateau conditions we lower the training range when the validation loss or validation accuracy, respectively, do not improve by more than 0.01% for 6 consecutive epochs. Finally, we also test lowering the training SNR irrespective of any of the metrics, by waiting 5 epochs between steps. We choose to wait 5 epochs to allow the network enough time at each signal strength while ensuring we reach the minimum SNR. No extensive studies testing different values were made.

We test two different approaches to lowering the training SNR. All curriculum strategies start with SNRs which are uniformly drawn from the interval $[90, 100]$. Strategies that are given the postfix "relative" lower the bounds of this interval by 10% at each step. The ranges are not lowered further when the lower bound of the interval reaches SNR 5. Strategies without the postfix "relative" lower the bounds of the interval by a fixed value of 5 at each step. This procedure is also continued down to a minimum bound of

SNR 5.

For fixed interval training strategies we test training on a single SNR as well as a fixed size interval of SNRs. We choose to train on fixed single SNRs 8, 15 and 30 to cover the low, mid and high SNRs respectively. Training on a single SNR allows us to test how well the network generalizes to lower and higher SNRs than it has seen during training. By drawing the SNR from an interval we aim to reduce the dependence on a specific signal strength. We choose two strategies that draw SNRs from a fixed range. One covers only the lowest range used by any of the non-relative curriculum strategies, i.e. it draws the signal SNRs from the interval SNR $\in [5, 15]$. The other draws the SNRs from the entire range of SNRs seen by the curriculum strategies, SNR $\in [5, 100]$.

6.2.4 Matched-filter baseline

In order to assess how sensitive the trained networks are in relation to conventional searches, we perform a matched-filter analysis of the test set described in subsection 6.2.1. To do so, we utilize the PyCBC analysis toolkit [370].

The template bank covers component masses from $10\,\mathrm{M}_\odot$ to $50\,\mathrm{M}_\odot$ and is constructed to lose no more than 3% of the SNR of any signal due to its discreteness. The templates are placed stochastically. In total, the bank contains 598 templates.

The search is implemented by `pycbc_inspiral`. We configured it to output a set of times where any template of the bank convolved with the data exceeds a matched-filter SNR of 5. Unlike the optimal SNR, the matched-filter SNR is the match of a detector data segment with a template, and so it varies based on the noise realization, while the optimal SNR assumes a noise realization that is constant zero. Combining the times where the threshold is exceeded with the corresponding matched-filter SNR and by using this SNR as ranking statistic, we obtain a set of triggers. We then process these triggers as described in subsection 6.2.2 to find events and calculate FARs and sensitive distances.

The configuration files are included in the data release [366].

6.3 Results

6.3.1 Sensitivities

We are able to reproduce or in some cases even improve on the results given in [56]. The top panel of Figure 6.2 shows the efficiency of one network as

Table 6.3: An overview of the different training strategies tested in this work. The "Curriculum" type strategies lower the SNR of the training samples whenever the condition in the last column is fulfilled. All of them start with SNR $\in [90, 100]$. Curriculum strategies with the postfix "relative" in their name lower the boundaries of the interval by 10% at each step, until the lower limit falls below SNR 5. The other curriculum strategies lower the bounds by a fixed value of 5, until the lower limit reaches SNR 5. A metric fulfills the plateau condition when it has not improved by more than 0.01% for 6 consecutive epochs. The "Fixed interval" type strategies use a single SNR range for the entire training. Their interval is given in the last column.

Type	Name	Condition
Curriculum	accuracy	when validation accuracy ≥ 0.95
	accuracy relative	
	epochs	every 5 epochs
	epochs relative	
	loss	when validation loss ≤ 0.2
	loss relative	
	plateau accuracy	6 epochs validation accuracy plateau
	plateau accuracy relative	
	plateau loss	6 epochs validation loss plateau
	plateau loss relative	
Fixed interval	SNR 30	SNR $= 30$
	SNR 15	SNR $= 15$
	SNR 8	SNR $= 8$
	low	SNR $\in [5, 15]$
	full	SNR $\in [5, 100]$

a function of the SNR at fixed FAPs calculated on the efficiency set. We compare our findings to theirs and find excellent agreement with the results shown in Figure 3 of [56], which closely reproduced efficiencies of matched filtering. The efficiencies at FAPs down to 10^{-3} for most other training strategies also closely follow the findings of Gabbard et al. We are, therefore, able to robustly reproduce the findings of [56].

In Figure 6.3 we show the evolution of the efficiency of the 50 networks trained on the fixed interval SNR $\in [5, 15]$ as the number of training epochs is increased at a FAP of 10^{-4}. Each panel of the plot shows the efficiency for a chosen SNR which allows us to observe how well the networks perform during different stages of the training at different signal strengths. This is especially interesting for curriculum strategies, where the SNR in the training set is adjusted as the network trains. The grey lines show the evolution of the efficiency for the different network initializations. The black, dashed line is the average of the grey lines. We highlight the evolution of a single network in dark grey. The red, dashed, vertical line signifies the epoch of maximum efficiency over all 50 networks and 200 epochs.

All networks in Figure 6.3 converge to similar efficiencies during the first $\sim$ 100 epochs. However, as training continues sudden drops to zero efficiency occur which become more frequent at later epochs. As a result the average efficiency drops continuously after some time. All networks show this behavior and thus the influence of an unlucky initialization can be ruled out. Furthermore, the drops are observed at all SNRs simultaneously and, therefore, do not depend on the signal strength.

The same effect can be seen in the top panel of Figure 6.2. For FAPs $\geq 10^{-3}$ the curves behave as expected. As one lowers the FAP the efficiency at any given SNR is expected to drop. Visually this manifests in a shift of the efficiency curves toward higher SNRs. Ideally, this behavior would be true for any FAP. However, at a FAP of 10^{-4} the efficiency collapses and becomes a constant 0.

The drops to zero efficiency are caused by noise samples which are attributed a p-score of 1. Since the Softmax activation on the last layer restricts outputs to the interval $[0, 1]$, no signal samples can achieve a p-score larger than the threshold and thus they cannot be distinguished from noise.

Many of the noise samples attributed a p-score of 1 are caused by numerical rounding errors in the Softmax activation

$$\text{Softmax}\left(\boldsymbol{x}\right)_i = \frac{\exp\left(x_i\right)}{\sum_{j=0}^{N} \exp\left(x_j\right)} \ , \tag{6.7}$$

where $\boldsymbol{x} = (x_0, x_1, \ldots, x_N)$ is the vector of outputs of the previous layer in the network, and $N + 1$ is the number of neurons in the layer.

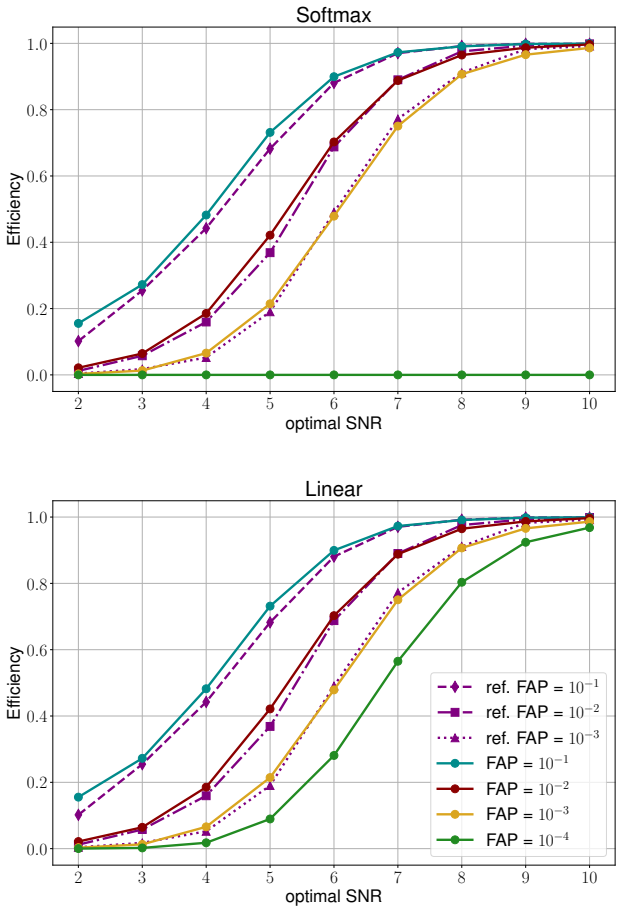


Figure 6.2: The efficiency as a function of optimal SNR at different FAPs. The network was trained on SNRs drawn from the fixed interval $[5, 15]$. We used epoch 186 of the network with the lowest efficiency at that epoch to produce this figure. The top panel shows the efficiency when the last layer uses a Softmax activation, the bottom panel shows the same network with the USR modification. We determine the threshold on the network output using a set of 400 000 pure noise samples. Any of the 10 000 signals at each SNR exceeding this threshold are counted as detected. We compare our findings to Figure 3 of the reference [56] which closely reproduces efficiencies of matched filtering.

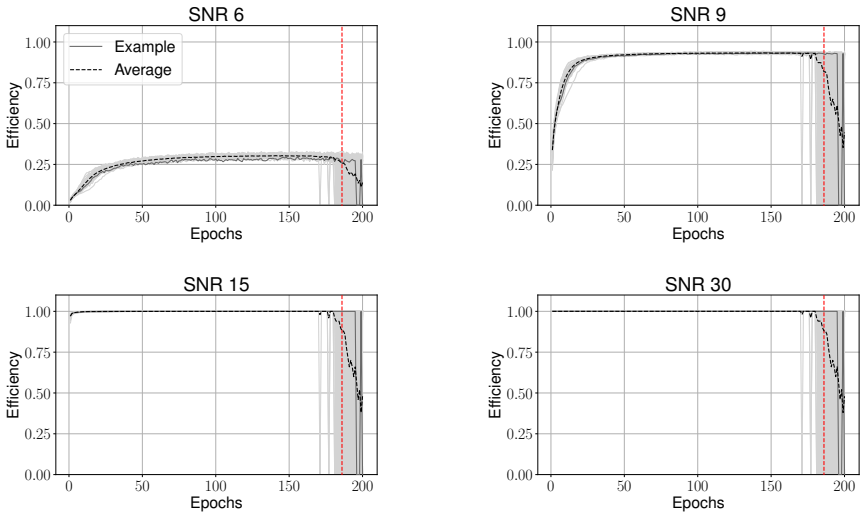


Figure 6.3: The evolution of the efficiency as a function of the epochs at different optimal SNRs. Training used the fixed SNR interval $[5, 15]$. The individual evolutions of all 50 runs are included as grey curves that form overlapping grey bands when plotted together. The dashed black line is the average of those. In dark grey we highlight the evolution of the efficiency for a single network. At the epoch marked by the red, dashed, vertical line we select the network with the highest, lowest and closest to average efficiency for further testing. The curves are computed at a FAP of 10^{-4}.

The networks operate with single precision (32-bit) floating point numbers. Therefore, small changes in the values of $\boldsymbol{x}$ may cause a roundoff error due to the rapid change in scale of the exponential functions. When this occurs, the fraction may evaluate to 1 even when mathematically (6.7) may never be 1.

We removed the final activation of the pre-trained network in an attempt to avoid the rounding errors. To do so, we recast (6.7) for $N = 1$ into

$$\frac{\exp(x_0)}{\exp(x_0) + \exp(x_1)} = \frac{1}{1 + \exp(x_1 - x_0)}, \tag{6.8}$$

and impose thresholds for the efficiency calculation on the difference $x_0 - x_1$ directly rather than $\text{Softmax}(\boldsymbol{x})_0$. Since (6.8) is bijective, there exists a direct relation between thresholds in $x_0 - x_1$ and the thresholds on $\text{Softmax}(\boldsymbol{x})_0$. We use $x_0 - x_1$ rather than $x_1 - x_0$ as our ranking statistic since $x_0 - x_1 > \hat{x}_0 - \hat{x}_1 \Leftrightarrow \text{Softmax}(\boldsymbol{x})_0 > \text{Softmax}(\hat{\boldsymbol{x}})_0$. We call this modification unbounded Softmax replacement.

The resulting efficiency is depicted in the bottom panel of Figure 6.2. Figure 6.4 shows the efficiency evolution at different optimal SNRs. We find that the drops to zero efficiency vanish when we apply USR. This is the case for all training strategies we explored and more examples are shown in the appendix (see Figure 6.9 to Figure 6.12).

One could also try to resolve the rounding issue by using double precision (64-bit) floating point numbers instead of single precision when applying the Softmax layer. We have tested a numerically safe implementation of the Softmax and found that its first output is rounded up to one even for quadruple (128-bit) precision when the difference $x_0 - x_1 > 45$. This is a relatively low value that indeed occurs for some noise realizations in our experiments. Although using higher precision for the Softmax layer increases the range of values it can operate on, the USR still solves roundoff issues more robustly.

The efficiency is a metric that is easy to calculate and physically more relevant than the accuracy of the network. However, it does not deal with samples where waveforms are misaligned in the data or take into account longer stretches of time. It is, therefore, only an approximation to the true statistic we want to calculate: the sensitive volume.

To assess if the efficiency is a good approximation to this statistic, we calculate the sensitive volume of three chosen networks for every training strategy as described in subsection 6.2.2. The networks are chosen from the 50 different initializations based on their efficiency at a selected epoch. We pick the networks with the highest, lowest and closest to average efficiency and denote them with "High", "Low" and "Mean", respectively, from here

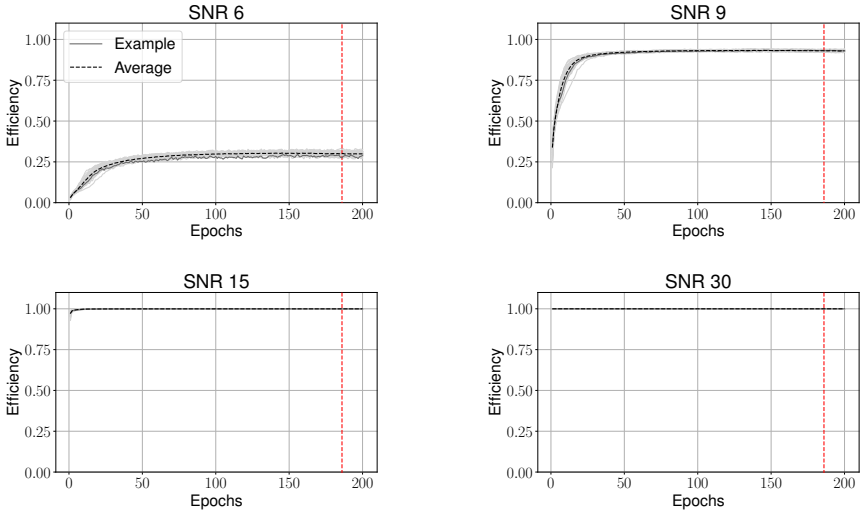


Figure 6.4: The evolution of the efficiency as a function of the epochs at different optimal SNRs. Training used the fixed SNR interval $[5, 15]$. The individual evolutions of all 50 runs are included as grey curves that form an overlapping grey band when plotted together. The dashed black line is the average of those. In dark grey we highlight the evolution of the efficiency for a single network. At the epoch marked by the red, dashed, vertical line we select the network with the highest, lowest and closest to average efficiency for further testing. The curves are computed at a FAP of 10^{-4}. This figure shows the same networks as Figure 6.3 after applying USR. This prevents the efficiency to drop to 0.

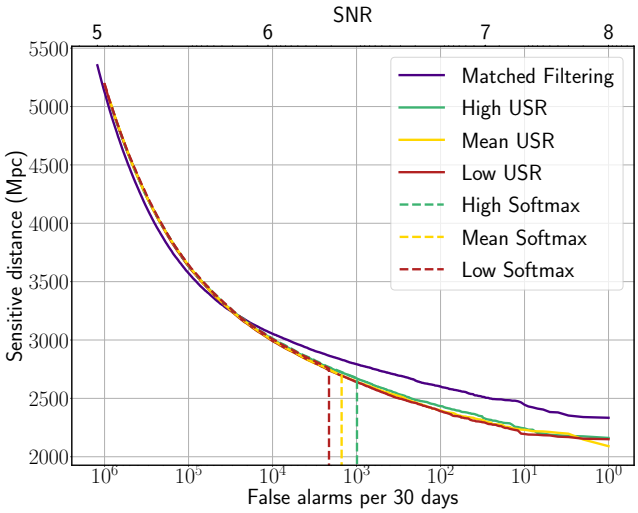


Figure 6.5: The sensitive distance as a function of the FAR (bottom horizontal axis) for different search algorithms. We compare differently initialized networks trained on data containing signals with SNR $\in [5, 15]$ to an equivalent matched-filter search. The dashed lines show the original networks, the filled lines show the corresponding network when USR is applied. The labels "High" (green), "Mean" (yellow) and "Low" (red) correspond to the networks with the highest, closest to average and lowest efficiency at epoch 186, respectively. In purple we show the equivalent matched-filter search that operates with a template bank containing 598 templates. The top horizontal axis shows the SNR threshold for the matched-filter search corresponding to the FAR on the bottom axis.

on out. If the efficiency at a fixed FAP is a good indicator of the networks sensitivity we expect the sensitive volume to scale with the efficiency.

Figure 6.5 shows the sensitive distance as a function of the FAR computed for the three networks trained on the fixed, low SNR interval. It compares the networks with (dashed) and without (continuous) the final Softmax activation and shows an equivalent matched-filter search in purple as reference. We find that the network is sensitive to sources up to a distance of 2150 Mpc with 1 false alarm per month.

The sensitive radii of all converged deep learning searches lie within 3.4% of each other for FARs where all of them are non-zero. However, the sensitivity of the networks with the final Softmax activation drops to zero for FARs $\leq \mathcal{O}(10^3)$ per month. This drop is caused by $\mathcal{O}(10^3)$ false alarms with a p-score of 1. This saturation of the final activation can be alleviated

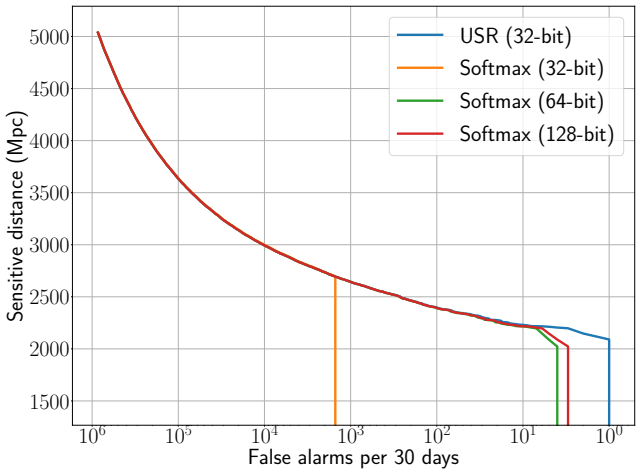


Figure 6.6: Sensitivity of the "mean" run of the "fixed low" strategy using the Softmax layer of various floating point precisions as well as the USR. A similar behavior of USR performing at least as well as the Softmax with all precisions was observed in all 45 evaluated runs.

by applying the USR modification and using the new output as a ranking statistic.

All tested networks have also been re-evaluated using higher precision floating point data types for the final activation function evaluation (example shown in Fig. 6.6). This resulted in the networks remaining sensitive at FARs down to 3 per month. However, applying the USR modification allowed us to test the network down to a FAR of 1 per month. Additionally, casting to a higher precision considerably increases computation time in the network due to hardware optimizations of GPUs for single precision floating point operations. In our view, the effectiveness of the USR outweights the benefits of using higher precision, hence we only report results obtained with the USR modification.

We had expected to find networks with higher efficiencies to be more sensitive, even within different initializations of the same training strategy. As such in Figure 6.5 we expected to find the sensitivity curve labeled "High" to be above the one labeled "Mean" above the one labeled "Low". While this is true in the example shown in Figure 6.5 in some regions, for other training strategies the order is arbitrary. All initializations converge to basically the same sensitivity. Sensitivity plots for all training strategies are provided in the data release [366].

The machine learning algorithms are compared to an equivalent matched-filter search, shown in purple in Figure 6.5. All searches perform equally well for FARs $\geq 10^5$ per month. For smaller FARs the matched-filter search is sensitive to sources which are up to 200 Mpc farther away. The deep learning search retains at least 91.5% of the sensitivity compared to the matched-filter search at all FARs.

This result shows that with minimal modification to the architecture the original network from [56] achieves a sensitivity comparable to matched filtering for short BBH signals in simulated Gaussian noise even at FARs previously untested for this particular network architecture.

All results above were obtained on data generated and whitened by the exact same PSD used during training. For realistic searches, this assumption does not hold as the PSD in the detectors drifts over time [14]. To assert that the network does not depend strongly on the exact PSD used during training, we also evaluated the sensitivity using a version of the training-PSD scaled by a constant factor of 1.05 in all frequency bins. This reduced the sensitive distance at all FARs by roughly $1/\sqrt{1.05}$, in agreement with the theoretical expectation.

We also tested the effects of using a realistic variation of the PSD. To determine the variation, we used 20 PSDs derived on real data from the O3a observing run [149], chose one as reference, and divided it by all the others. We then determined the PSD ratio that had the largest mean deviation from unity and multiplied it with the training PSD to obtain a realistically varied PSD. Generating and whitening the data by this varied PSD reduces the sensitivity at FARs below 100 per month to around the same level as is observed for the scaled PSD.

Finally, we tested whitening by a different PSD than the one used for generating the data. For this purpose we used a second PSD variation to whiten the data generated by the first PSD variation described above. To obtain the second PSD, we used the PSD ratio that had the smallest, instead of the largest, mean deviation from unity. This simulates a worst-case scenario for realistic PSD variations. We find that the sensitivity drops by as much as 20% compared to using the correct PSD for whitening. This analysis shows that the network is robust against differences between the training PSD and the PSD of the analyzed data, as long as the correct PSD is used for whitening.

6.3.2 Training strategies

We trained 50 networks for every training strategy discussed in subsection 6.2.3. Figure 6.7 shows the evolution of the efficiency at SNR 9 for

every training strategy. The networks use a Softmax activation on the final layer. While we also monitor different SNRs, it is this region we are most interested in for three reasons. The first is practical in nature. Above an SNR of 9 networks trained with almost all training strategies recover close to 100% of the signals. It is, therefore, impossible to separate them by efficiency. Secondly, most GWs are expected to be detected at low SNRs [179]. Hence, efficiency at low SNRs is most important. Lastly, SNR 8 is often used as a threshold above which matched-filter searches can comfortably detect most signals (compare Figure 6.5). By probing the efficiency close to this threshold we can get a sense of how well the search is doing overall.

Most training strategies do not have a major impact on the maximum efficiency. With the exception of training with a fixed SNR 15 and 30 all converged networks reach efficiencies of 90% ("Fixed full range") to 94% ("Fixed 8"). At SNR 6 the efficiency consistently drops to 24% ("Fixed full range") to 33% ("Loss relative") for all converged networks other than the above mentioned exceptions. Above an SNR of 12 the efficiencies reach 100% for all networks except "Fixed 15" and "Fixed 30". Those only achieve 100% efficiency at SNR 15 and 21 respectively.

The relative plateau strategies did not manage to converge within the first 200 epochs. For these, we have extended the training length to 400 epochs, which has allowed these runs to converge. They have reached comparable efficiencies to those mentioned above.

The main difference between all runs is the number of epochs required to reach a converged state. One can see that strategies which supply low SNR signals earlier reach their maximal efficiency earlier. This is especially emphasized with the runs "Fixed 8", "Fixed full range" and "Fixed low range". The curriculum strategies that use the accuracy or loss as their condition also converge quickly. They too supply low SNR samples very early on, as the respective condition is fulfilled at each of the first few epochs. Waiting for a set number of epochs to pass hinders the ability of the network to see low SNR signals early on and, therefore, takes more time to converge. The slowest converging strategies wait for the loss or accuracy to stop improving. They effectively have to wait at least 6 epochs before lowering the training range. Using a relative approach to lowering the SNR range further decreases the speed at which low SNRs are explored. In the most extreme cases the networks do not converge within the given number of epochs.

Finally, some training strategies become unstable toward the end. All of these unstable strategies converge relatively fast. This suggests that the longer one trains a converged network the more likely the efficiency is to collapse. We, therefore, expect that strategies where the efficiency did not collapse during the first 200 epochs would see a similar problem during later

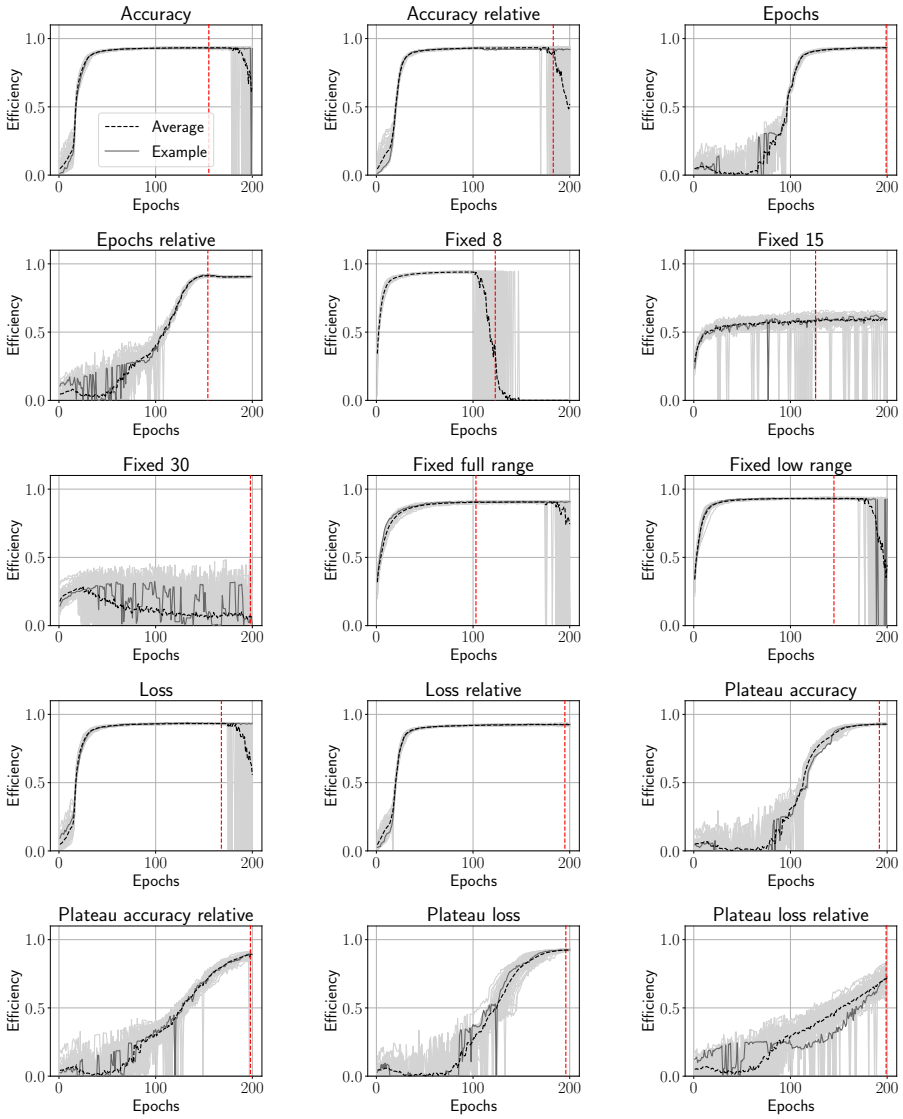


Figure 6.7: The efficiency for all 15 tested training strategies as a function of the training epochs at SNR 9 and a FAP of 10^{-4}. The light grey curves show the efficiency for the 50 independent initialized training runs. The black dashed line shows the average over these individual runs. We highlight the evolution of a single run in dark grey. The vertical, red, dashed line signifies the epoch with the largest efficiency. We choose 3 networks at this epoch for which to calculate the sensitive volume.

epochs.

The breakdown of the efficiency was resolved by the USR modification in subsection 6.3.1. Figure 6.8 shows the evolution of the efficiency at SNR 9 when this fix is applied. We find that the drops to zero efficiency are removed but the qualitative features of the efficiency curves stay the same.

All efficiency plots were generated from the TensorFlow version of the networks. When training with PyTorch we found the results virtually indistinguishable from the TensorFlow version.

We repeated our tests on networks with different capacities, although this was not the main focus of the present work, to ensure that our findings are robust against a few specific architecture changes. We found no significant differences in the final efficiency, although the speed of training convergence varied. Such studies are left to future work.

6.4 Conclusions

In this paper, we revisited the first deep learning GW search algorithms and compared them directly to a matched-filter search. We showed that for the considered parameter space and for a single detector the networks retain performance closely following matched filtering even on long duration continuous data sets and when considering FAR thresholds down to once per month. While there are now more sophisticated deep learning algorithms available that enhance the capabilities of the first proofs of concept, we think that there is still a lot to be learned from these first steps.

Our initial focus was the optimization of the data presentation to these networks. Two kinds of training strategies were previously explored; curriculum learning, where training samples become more difficult to classify as training continues, and fixed interval training, where the complexity of the training set stays constant.

We found that the particular strategy is of little importance to the eventual performance of the network. It depends a lot more on the presence of sufficiently complex samples in the training set. In particular, we found that the networks are able to generalize low SNR signals to high SNR ones but not vice versa.

On the other hand, the training strategy does have an impact on the time it takes the network to converge. Since high SNR examples are not as important to the performance of the network, strategies that provide low SNR samples earlier converge faster. In conclusion, we recommend training deep learning search algorithms on a fixed range of low SNR signals.

We use efficiency as our metric of performance during training. As this

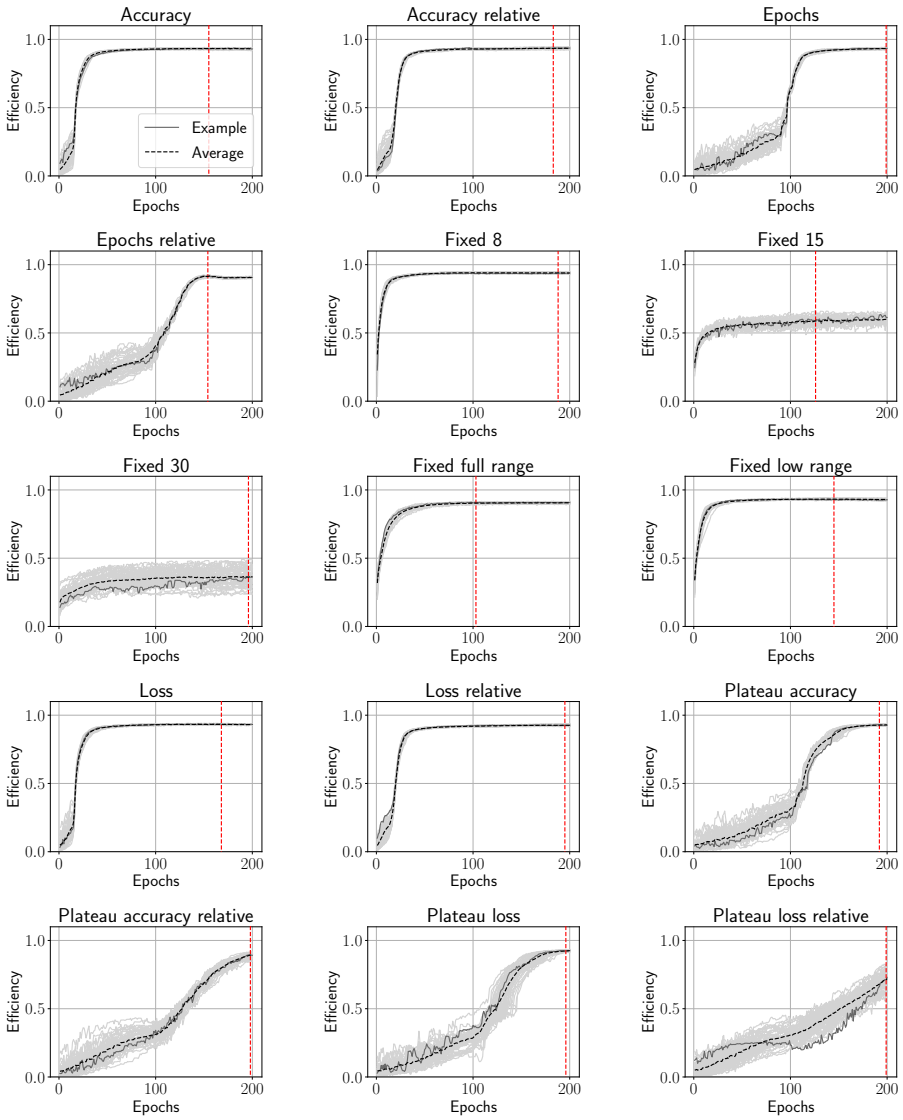


Figure 6.8: The efficiency for all 15 tested training strategies as a function of the training epochs at SNR 9 and a FAP of 10^{-4}. The light grey curves show the efficiency for the 50 independent initialized training runs. The black dashed line shows the average over these individual runs. We highlight the evolution of a single run in dark grey. The vertical, red, dashed line signifies the epoch with the largest efficiency. We choose 3 networks at this epoch for which to calculate the sensitive volume. This figure shows the same networks as Figure 6.7 with the USR modification applied. With this modification the efficiency stays > 0 at all times.

statistic has been used in previous publications, it allows us to verify that we have converged to the expected performance.

The efficiency at FAPs $\leq 10^{-4}$ dropped to zero when networks were trained for extended periods of time. This was unexpected and limited our ability to test the search.

We found the drops in efficiency to be caused by numerical instabilities in the final activation function of the networks. By removing the Softmax activation on the final layer and imposing thresholds directly on the linear output of the network, we were able to lift the limitations on the testable FAPs. This USR modification has proven to be simple and effective, as no re-training of the networks is required and virtually unlimited low FAPs can be tested.

To compare the deep learning based searches to an equivalent matched-filter search we calculated the sensitive volumes as functions of the FARs on a month of simulated data. We found that the machine learning algorithm is able to closely follow the performance of the traditional algorithm even down to FARs of 1 per month, when USR is used.

The results given here are limited to a single detector, Gaussian noise and signals from BBHs, which are relatively short in duration and comparatively simple to detect with existing methods. Parts of the parameter space, like the inclusion of higher-order modes [41], eccentricity [358], or precession [40], where current searches are computationally limited, are not yet included. However, it is expected that neural networks may generalize efficiently to these more difficult signals. Deep learning detection algorithms for spinning black holes with precession were recently explored for the first time by [57]. There is also ongoing work to construct neural network searches targeting long duration signals [61, 287, 291, 303]. Considering real noise may enable deep learning algorithms to outperform matched filtering, which is only known to be optimal for stationary Gaussian noise. Multiple studies have shown that neural networks adapt well to non-stationary noise contaminated with glitches [54, 57, 291, 359].

6.5 Appendix: Efficiency curve examples

This appendix provides examples of the usefulness of USR for various training strategies explored in this paper. The plots show the efficiency as a function of training epochs at 4 distinct SNRs with and without the application of the USR. For both shown examples the USR manages to remove the efficiency breakdown entirely.

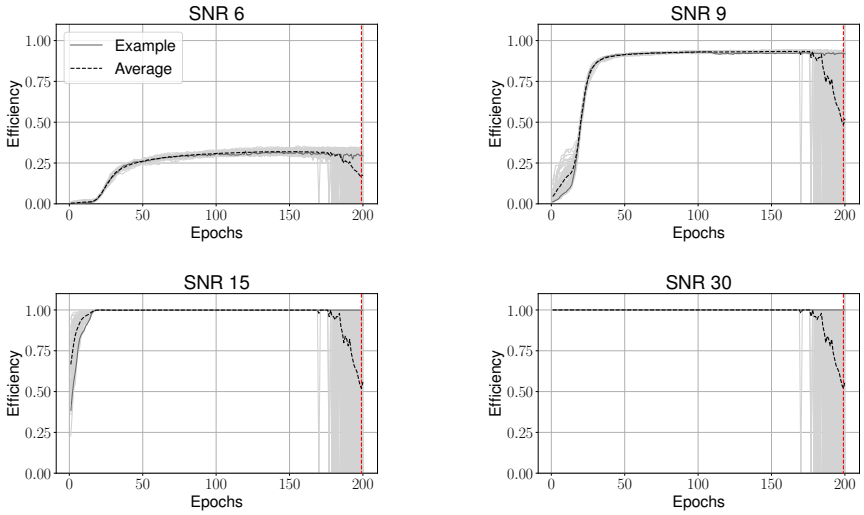


Figure 6.9: Efficiency evolution of the "Accuracy relative" strategy using the Softmax output as a ranking statistic.

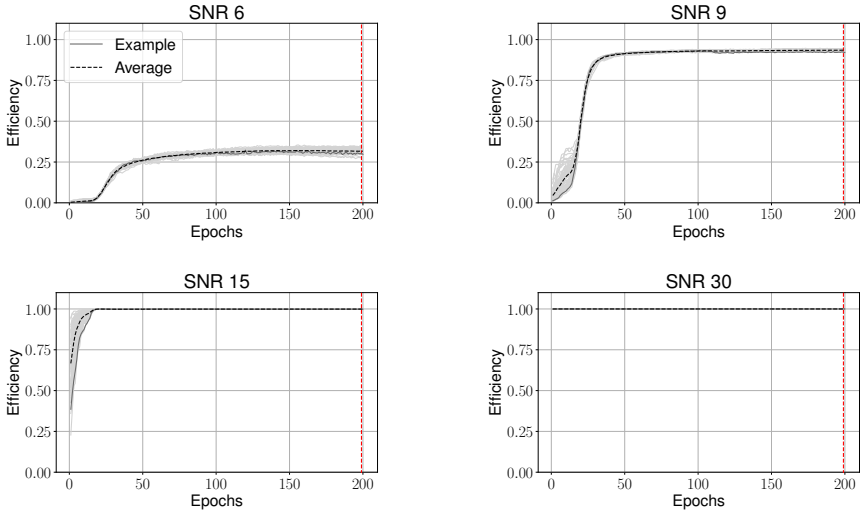


Figure 6.10: Efficiency evolution of the "Accuracy relative" strategy using the USR modification.

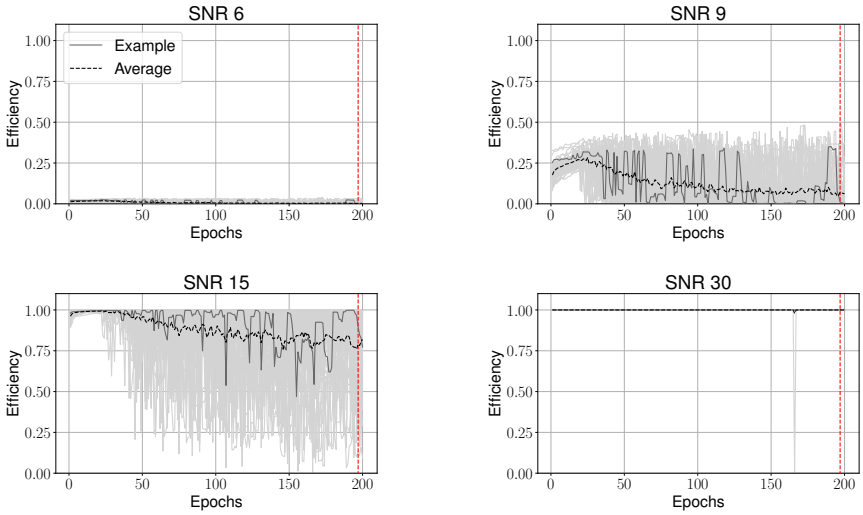


Figure 6.11: Efficiency evolution of the "Fixed 30" strategy using the Softmax output as a ranking statistic.

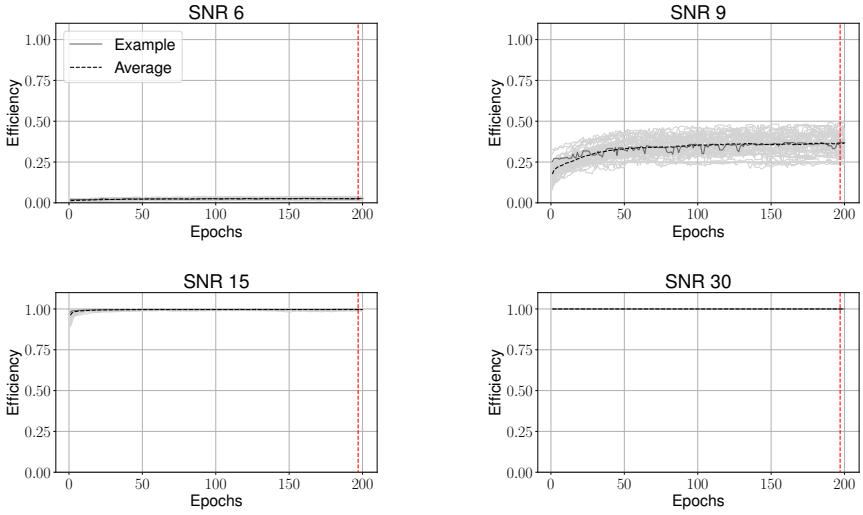


Figure 6.12: Efficiency evolution of the "Fixed 30" strategy using the USR modification.

Chapter 7

From One to Many: A Deep Learning Coincident Gravitational-Wave Search

Contents

7.0 Abstract 132

7.1 Introduction 132

7.2 Coincident Search from Independent Single-Detector Networks 135

7.2.1 Architecture 135

7.2.2 Data Sets and Training 136

7.2.3 Single Detector Events 138

7.2.4 Coincident Events 138

7.2.5 Background Estimation 139

7.2.6 Sensitivity 140

7.2.7 Matched Filtering 141

7.2.8 Evaluation and Comparison to Matched Filtering . 142

7.3 Two Detector Network 145

7.3.1 Architecture 146

7.3.2 Data Sets and Training 146

7.3.3 Coincident Events 148

7.3.4 Background Estimation 148

7.3.5 Evaluation and Comparison to Matched Filtering . 149

7.4 Conclusions 149

This chapter is essentially a full reprint of [64] with some minor edits for formatting. It explores the capability of deep learning GW search algorithms trained on a single detector to be applied in a coincidence search.

7.0 Abstract

Gravitational waves from the coalescence of compact-binary sources are now routinely observed by Earth bound detectors. The most sensitive search algorithms convolve many different pre-calculated gravitational waveforms with the detector data and look for coincident matches between different detectors. Machine learning is being explored as an alternative approach to building a search algorithm that has the prospect to reduce computational costs and target more complex signals. In this work we construct a two-detector search for gravitational waves from binary black hole mergers using neural networks trained on non-spinning binary black hole data from a single detector. The network is applied to the data from both observatories independently and we check for events coincident in time between the two. This enables the efficient analysis of large quantities of background data by time-shifting the independent detector data. We find that while for a single detector the network retains 91.5% of the sensitivity matched filtering can achieve, this number drops to 83.9% for two observatories. To enable the network to check for signal consistency in the detectors, we then construct a set of simple networks that operate directly on data from both detectors. We find that none of these simple two-detector networks are capable of improving the sensitivity over applying networks individually to the data from the detectors and searching for time coincidences.

7.1 Introduction

Gravitational waves (GWs) are now routinely observed by the two Advanced LIGO detectors [6] and the Advanced Virgo detector [7]. At the end of the last observing period, the KAGRA detector [8] joined the network and is expected to aid observations in the future. During three observing runs ≈ 90 GWs from compact binary sources have been identified, almost all of which are consistent with the merger of Binary Black Hole (BBH) systems [14, 15, 28, 206, 346, 356].

Many searches for GWs from compact-binary coalescence use matched filtering to separate potential signals from the background detector noise

[59, 161, 343, 356]. Matched filtering is a technique that convolves a set of pre-calculated template waveforms, each representing a possible source with different component masses, spins, etc., with the detector's data and is known to be optimal for Gaussian noise [30]. A Signal-to-Noise Ratio (SNR) time series is calculated for each template waveform; candidates are identified by a peak in the SNR time series that also passes data quality [125, 371, 372] checks. In a second step the candidate detections from one detector are cross-validated with the candidate detections from other detectors to further increase the significance of the reported events and rule out false positives [60, 158, 356]. For sources where the gravitational-wave signal is unknown or poorly modeled other search algorithms detect coincident excess power in different detectors and do not require a model [38].

Deep learning has started to be explored as an alternative approach to building an algorithm to detect GWs [48, 55–57, 61, 288, 291, 303, 359, 360]. It may potentially target signals which are currently challenging for matched filter search algorithms due to computational limitations [57, 373, 374]. The computational cost of these modeled searches scales with the number of templates required by the parameter space. Certain effects like higher-order modes [41], precession [40], eccentricity [168, 358], or the inclusion of sub-solar mass systems [171, 172] potentially require millions of templates and are thus computationally prohibitive to analyze. Deep learning may also be more sensitive when the noise is non-Gaussian [190, 282, 291].

In our previous work [63] we explored the sensitivity of a simple neural network to non-spinning BBH sources in Gaussian noise for a single detector. We tested how different training strategies influence the training procedure and the final efficiency of the network. Our results showed that under the given conditions the network can closely reproduce the sensitivity of matched filtering and that most efficient convergence is reached when a range of low SNR signals is provided throughout training.

Here we extend our previous work to two detectors. To do so, we use the same single detector network explored in [63] and apply it individually to the data from both observatories. This procedure produces a list of candidate events for each detector. We then search for coincident events between the two, where two events are assumed to be coincident if they are within the maximum time-of-flight difference between both detectors. We assume this difference to be 0.1 s since the networks are trained to be insensitive to variations on such scale.

The network uses the Unbounded Softmax Replacement (USR) modification we introduced in [63]. It outputs a single detector ranking statistic. Here we use it to construct a network ranking statistic. This network ranking statistic turns out to be the sum of the individual ranking statistics minus a

correction factor.

The main advantage of this approach is the trivial computation of the search background which enables robust detection claims at comparable statistical significance (<1 per 100 years) to existing production methodology. By applying time shifts larger than the time-of-flight difference between the detectors to the data from only one observatory, we can create large amounts of data which by construction cannot contain any astrophysical coincident candidates. By applying the time shifts to the single detector events rather than the input data directly, we can skip re-evaluating the entire test set and efficiently look for coincident events. This is a well established method that has already been successfully applied [60, 158, 356]. By this approach we can probe the search down to a false-alarm rate (FAR) of 1 false-alarm per $\mathcal{O}(10^3)$ months. The FAR estimates how often a candidate is produced by the search under the null hypothesis of no astrophysical candidates. Our FAR-estimate is limited by the assigned hardware resources rather than the available data.

We compare this search to an equivalent matched filter search [375]. We find that the deep learning search still retains 92.4% of the sensitivity of a two-detector matched filter search when the latter is restricted to using the timing difference between the detectors as the only means for determining coincident events. However, the matched filter search also extracts some information on the parameters of the signal. When we also require matching templates and the phase and amplitude of the triggered templates to be consistent between detectors [163], the machine learning search only retains 83.9% of the sensitivity.

We then construct a single network that operates on the data from both detectors. The idea is that the network may then be able to learn, summarize, and cross-correlate signal characteristics between detectors. To do so, we remove the last layer of the original networks applied to the individual detectors and concatenate their output. Thereby the input data are compressed to a 128 dimensional latent space. Dense layers are used to correlate the concatenated outputs and condense it into a single ranking statistic.

Using a single network complicates the background estimation, as time shifts between the detectors can in principle not be applied after evaluating the individual data streams. However, the two-detector network architecture is constructed such that the data from different detectors is analyzed by individual sub-networks, concatenated and processed by a third sub-network. This enables us to process the bulk of the data only once and apply time shifts to the individual detector sub-network outputs. To obtain the ranking statistic we are then only required to run the time-shifted data through the final, small sub-network.

We find that networks constructed this way are not able to improve the sensitivity over a time coincidence analysis of the single detector machine learning events. We test three different approaches to training these networks but none show any improvement.

7.2 Coincident Search from Independent Single-Detector Networks

The algorithm explored in this section uses a network trained on data from a single detector and uses it to find coincidences in multiple detectors. It is one of the most simple extensions and has two advantages. First, networks trained on data from a single detector can be re-used which reduces requirements to computational resources. Second, the search background can be estimated using well established and efficient algorithms allowing for much higher confidence in candidate detections.

7.2.1 Architecture

We use the same network as in [63], which is an adaptation of the network presented in [56]. It consists of 6 stacked convolutional layers followed by 3 dense layers. An overview of the architecture is given in Table 7.1.

The last layer contains a Softmax activation function, which we remove during testing. In [63] we showed that this modification, which we called Unbounded Softmax Replacement (USR), allows the network to be tested at lower FARs than otherwise possible.

The Softmax activation for the first output neuron is given by

$$p \coloneqq \text{Softmax}\left(\boldsymbol{x}\right)_0 = \frac{1}{1+\exp\left(-\Delta x\right)}, \tag{7.1}$$

where $\boldsymbol{x} = (x_0, x_1)$ is the network output before the activation function and $\Delta x = x_0 - x_1$. When Δx is strongly positive, the denominator in (7.1) and thus the fraction numerically evaluates to 1. This leads to problems when setting the threshold value to use to determine true positive detections [63].

However, equation (7.1) is bijective and can be inverted

$$-\Delta x = \log\left[\frac{1}{p} - 1\right]. \tag{7.2}$$

This quantity is monotonic and we can thus do statistics on Δx directly, avoiding numerical instabilities while still using the Softmax activation during training.

Table 7.1: A detailed overview of the architecture for the single detector neural network. Rows are grouped by their influence on the shape of the data. The layers are to be read from left to right and top to bottom to construct the network.

layer type	kernel size	output shape
Input + BatchNorm1d		2048×1
Conv1D + ELU	64	1985×8
Conv1D	32	1954×8
MaxPool1D + ELU	4	488×8
Conv1D + ELU	32	457×16
Conv1D	16	442×16
MaxPool1D + ELU	3	147×16
Conv1D + ELU	16	132×32
Conv1D	16	117×32
MaxPool1D + ELU	2	58×32
Flatten		1856
Dense + Dropout + ELU		64
Dense + Dropout + ELU		64
Dense + Softmax		2

7.2.2 Data Sets and Training

The input to the network is a time series of 1 s duration sampled at 2048 Hz. This allows for signals up to a frequency of 1024 Hz to be resolved which is sufficient for the considered parameter space.

The network is trained on signals from non-spinning BBHs with component masses m_1, m_2 uniformly distributed from $10\,\mathrm{M}_\odot$ to $50\,\mathrm{M}_\odot$. We enforce $m_1 \geq m_2$ and for each pair of masses uniformly draw 5 coalescence phases $\phi_0 \in [0, 2\pi]$. The signals are generated with the waveform model `SEOBNRv4_opt` [369] (optimized version of `SEOBNRv4` [118]) and scaled to varying optimal SNRs in the range $[5, 15]$ during training. The time of merger is varied from 0.6 s to 0.8 s from the start of the input window to decrease the dependency of the network on the exact signal position. Each signal is whitened by the analytic model for the detector power spectral density (PSD) `aLIGOZeroDetHighPower` [319]. For further details on the training set please refer to [63].

Notably, we do not vary the sky position, inclination or polarization during training. For a single detector, variations in these parameters can be fully expressed by changes in the distance, which is fixed by choosing a specific

SNR, and the phase ϕ_0. For a two detector setup this degeneracy is broken as a time-of-flight difference is introduced and the amplitudes and phases are correlated in the two detectors. However, our search algorithm is largely parameter agnostic. This means that its output does not depend on the amplitude or phase. Thus, we do not have information on whether or not the search responds to consistent signals. Finally, the time-of-flight difference is on the order of the variation of the merger time within the training set and can, therefore, not be resolved. In section 7.3 the network has access to data from both observatories and the data is adjusted accordingly.

All noise is Gaussian and simulated from the `aLIGOZeroDetHighPower` PSD [319]. We explicitly generate colored noise and whiten it afterwards. This in principle allows to extend our training to real noise.

The training set contains 200 000 noise samples, 100 000 of which are combined with 100 000 unique signals. The validation set[1] contains 400 000 noise samples and 10 000 unique signal samples, which we subsequently scale to SNRs $3, 6, 9, 12, 15, 18, 21, 24, 27$ and 30. This set is used to calculate the *efficiency* of the network at a fixed False-Alarm Probability (FAP) of 10^{-4}. The FAP is the fraction of discrete noise samples misclassified as signals. The efficiency is the fraction of discrete signal samples correctly classified as signals at a given FAP.

The test set contains a month of continuous simulated noise for each of the two detectors in Hanford and Livingston. We inject signals with parameters drawn from the distributions shown in Table 7.2 into both data streams. Injections are separated by a random time between 16 s to 22 s. To enable the networks to process this data, the continuous stream is sliced into ≈ 26 million overlapping, correlated samples. Each sample is whitened individually by the analytic PSD.

We construct a second test set for background estimation. This set contains the same time domain noise as the first test set but no injections are performed. We pre-process this second data set in the same way we pre-process the first data set for the network to be able to process it.

The network is trained for 200 epochs and we use the network with the highest average efficiency over all SNRs for the analysis carried out here. We use the Adam optimizer with a learning rate of 10^{-5}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ [253]. We use a variant of the binary cross-entropy which was designed to stay finite as loss function

$$L(\boldsymbol{y}_\mathrm{t}, \boldsymbol{y}_\mathrm{p}) = -\frac{1}{N_\mathrm{b}} \sum_{i=1}^{N_\mathrm{b}} \boldsymbol{y}_{\mathrm{t},i} \cdot \log\left(\epsilon + (1-2\epsilon)\boldsymbol{y}_{\mathrm{p},i}\right), \tag{7.3}$$

[1] In out previous work [63] what we call validation set here was named efficiency set.

where $\boldsymbol{y}_\mathrm{t}$ is $(1,0)^T$ for a signal-class sample and $(0,1)^T$ for a noise-class sample, $\boldsymbol{y}_\mathrm{p}$ is the prediction of the network, $N_b = 32$ is the mini-batch size, and $\epsilon = 10^{-6}$.

We implemented the network using the high-level API Keras [376] of TensorFlow version 2.3.0 [364].

7.2.3 Single Detector Events

To apply the network to data of duration longer than the 1 s input of the network, we use a sliding window with step size 0.1 s. The contents of each window are whitened individually by the PSD model. At each step the network outputs a set of two numbers, the difference of which we use as our ranking statistic.

We apply the same network to the data from both detectors individually. We, thus, receive two output time series of ranking statistics. To determine notable events in the individual detectors we apply a threshold to both time series and cluster the resulting points above the threshold into events. A point exceeding the threshold is counted toward a cluster if it is within 0.2 s of the cluster boundaries. We choose a threshold on the USR output of -2.2, which corresponds to a Softmax output of 0.1.

The search algorithm produces a list of events, where an event is a tuple $(t, \Delta x)$. Each event is a time t at which the network predicts a signal to be present with a ranking statistic Δx. The ranking statistic can be used to assign a significance to the event.

7.2.4 Coincident Events

A signal will be present in the data of all detectors if it is of astrophyiscal origin. Its SNR in each detector depends on the location and orientation of the source. The number of false alarms can, thus, be reduced by requiring that the event is picked up by multiple detectors at similar times.

To quantify the significance of an event detected by more than one observatory, a combined ranking statistic is required. For simplicity we restrict our current analysis to two detectors. However, this approach is extendable to any number of detectors.

If the network was using the final Softmax activation during evaluation a combined ranking statistic would come straightforwardly from the interpretation of the output as a probability.

$$p_{H+L} = 1 - (1 - p_H)\,(1 - p_L) \tag{7.4}$$

The 1-to-1 relation between p and Δx given in equation (7.1) can be inserted into (7.4) to get

$$\begin{aligned} -\Delta x_{H+L} = &- \Delta x_H - \Delta x_L \\ &- \log\left[1 + e^{-\Delta x_H} + e^{-\Delta x_L}\right]. \end{aligned} \tag{7.5}$$

The combined ranking statistic is the sum of the single detector ranking statistics minus a correction term.

We consider an event in one detector to be coincident with another event in the other detector if the event times t_i are within 0.1 s of each other. This time difference is chosen to be the maximum time resolution the networks can achieve due to the time variation in the training set.

We construct a list of coincidence events from the single detector list by the above condition. Each coincident event is assigned the combined ranking statistic (7.5) and the time in the Hanford detector.

7.2.5 Background Estimation

To estimate the FAR at different ranking statistic values we evaluate the same noise used to search for signals but omit injecting the GWs. This ensures that all events found in this data set are noise artifacts and are not influenced by close by injections.

We apply the network to the data and determine events as described in subsection 7.2.3. We obtain two lists of events and search for coincidences as detailed in subsection 7.2.4.

The lowest FAR that can be probed is limited by the duration of the analyzed data. Our test set covers one month. The duration can be increased by shifting the data in one of the detectors by a time larger than the maximum time-of-flight duration between the detectors. Rather than shifting the data itself one may instead alter the event times returned by the search. This allows us to skip reanalyzing the full data for each time step and only requires us to look for coincidences between the events from one detector and the time shifted events from the second detector. Increasing the amount of background by applying time shifts is a well established method that has already been successfully applied in production searches [60, 158, 356].

We choose a time shift of 1024 s and apply any possible integer multiple of this step size. We then search for coincidences in these events as detailed in subsection 7.2.4. This procedure increases our background to ≈ 2400 months $= 200$ years.

A list of FARs at different network ranking statistics is obtained by counting the number of events in the way described above with a larger ranking statistic.

Table 7.2: Distributions of the parameters used for the injections in the test set.

Parameter	Uniform distribution
Component masses	$m_1, m_2 \in (10, 50)\,\mathrm{M}_\odot$
Spins	0
Coalescence phase	$\Phi_0 \in (0, 2\pi)$
Polarization	$\Psi \in (0, 2\pi)$
Inclination	$\cos\iota \in (-1, 1)$
Declination	$\sin\theta \in (-1, 1)$
Right ascension	$\varphi \in (-\pi, \pi)$
Distance	$\mathrm{d}^2 \in (500^2, 7000^2)\,\mathrm{Mpc}^2$

7.2.6 Sensitivity

The sensitive volume of a search can be estimated by

$$V(\mathcal{F}) \approx V(d_{\mathrm{max}}) \frac{N_s(\mathcal{F})}{N_{\mathrm{inj}}}, \tag{7.6}$$

when it is derived on data containing injections which are distributed uniformly in volume [60]. Here $\mathcal{F}$ is the FAR at which the volume is being calculated, d_{max} is the maximum distance of any injection, $V(d_{\mathrm{max}})$ is the volume of a sphere with radius d_{max}, $N_s(\mathcal{F})$ is the number of signals detected with a FAR $\leq \mathcal{F}$ and N_{inj} is the total number of injected signals. We report the radius of a sphere with volume $V(\mathcal{F})$ instead of the sensitive volume.

We analyze a month of simulated data from the two detectors Hanford and Livingston, assuming the PSD `aLIGOZeroDetHighPower` [319]. The data contains injections drawn from the distribution shown in Table 7.2. We apply the network to the data from both detectors individually as described in subsection 7.2.3. The resulting single detector events are correlated and a list of coincident events is produced as detailed in subsection 7.2.4. We then pick out any events that are within 0.3 s of an injection. These events are called foreground events from here on out.

To determine the search background, we evaluate the same month of noise used to find the foreground events. However, this data does not contain any injections. The networks return a list of single detector events, which are correlated and shifted in time to increase the effective duration of the analyzed data as detailed in subsection 7.2.5. The resulting coincident events are called background events from here on out.

We can then assign a FAR to any foreground event. To do so we count the number of background events with a ranking statistic larger than the

ranking statistic of the considered foreground event. This number is divided by the effective duration of the analyzed background to obtain a FAR. The sensitive volume is then obtained from equation (7.6) and converted to a distance. The sensitive distance as a function of the FAR is obtained by evaluating the sensitive volume at the FARs of all foreground events.

7.2.7 Matched Filtering

The template bank contains 598 unique waveforms and is constructed such that no more than 3% of the SNR of any signal is lost due to the discreteness of the bank. It covers the same mass range of $10\,\mathrm{M}_\odot$ to $50\,\mathrm{M}_\odot$ as the training set of the networks and spins are set to 0. The individual templates are generated using the waveform model `IMRPhenomD` [129, 130] and placed stochastically.

To run the matched filter search we use the program `pycbc_inspiral` [375]. It is setup to use a SNR threshold of 5 in both detectors to create two sets of single detector triggers. These two sets are then checked for coincidence by two different approaches.

One approach handles the matched filter triggers analogous to the network single detector triggers, i.e. they are clustered and turned into single detector events as described in subsection 7.2.3. In this case the ranking statistic is the SNR returned by the best matching template. We then look for coincidences as described in subsection 7.2.4 by requiring two events in different detectors to be separated by no more than 0.1 s. The combined ranking statistic in this case is given by

$$\rho_{H+L} = \sqrt{\rho_H^2 + \rho_L^2}. \tag{7.7}$$

This disregards the information about the possible parameters obtained from the best matching template and only looks for time coincidence, i.e. no signal consistency is required.

The other approach leverages the signal information and checks for phase and amplitude correlation as well as requiring that the templates matching the data are consistent between detectors. In particular we utilize the combined ranking statistic given in equation (2) of [163] and find coincidences as described therein.

7.2.8 Evaluation and Comparison to Matched Filtering

In Figure 7.1 we show the injections that were found and missed by the network coincident search at a FAR of 1 false alarm per month. The x-axis shows the optimal SNR of the injections in the Hanford detector and the y-axis shows the optimal SNR in the Livingston detector. The color indicates the network ranking statistic as calculated by equation (7.5). Missed injections are marked with a red cross. A network SNR of 8 as calculated by equation (7.7) is highlighted by the black line.

Figure 7.1 shows that the combined ranking statistic (7.5) is correlated with the network SNR. As the network SNR increases so does the combined ranking statistic. The loudest missed injection has a network SNR of 22.7. However, the signal is most dominantly seen in the Hanford detector with a single detector SNR of 22.6, whereas Livingston has an optimal SNR < 2 due to the location of the source. Therefore, it is not surprising that the signal does not show up in both detectors and is missed by the coincidence search. When considering only the detector in which the signal is observable with lower SNR, the loudest missed signal has a optimal SNR of 9.2 in that detector.

In Figure 7.2 we show the sensitive distance of different algorithms as a function of the FAR. The orange lines show the sensitivity curves of the machine learning based algorithms whereas the purple lines show the sensitivities of a comparable matched filter search. The dashed lines show the sensitivity of the searches when only a single detector is considered. We compare those to a two-detector search where we require coincident detections in both detectors. The filled orange line and the dash-dotted purple line show the comparison between the machine learning and matched filter algorithms, respectively, when both impose the same coincidence condition. The filled purple line shows a more realistic application of matched filtering where the consistency of the time of arrival, the phase, the amplitude, as well as the parameters of the best matching template are required.

We find a significant improvement of up to 20% at a given FAR when the machine learning algorithm has access to data from both detectors compared to using only data from a single detector. Furthermore, we can probe FARs down to $\approx 4 \times 10^{-4}$ false alarms per month without needing to increase the amount of evaluated data by applying time shifts between detectors as described in subsection 7.2.5. In principle this limit may be decreased even further and time shifts are only limited by the time-of-flight difference between the detectors. The large increase in the available background potentially greatly increases the statistical significance of any event.

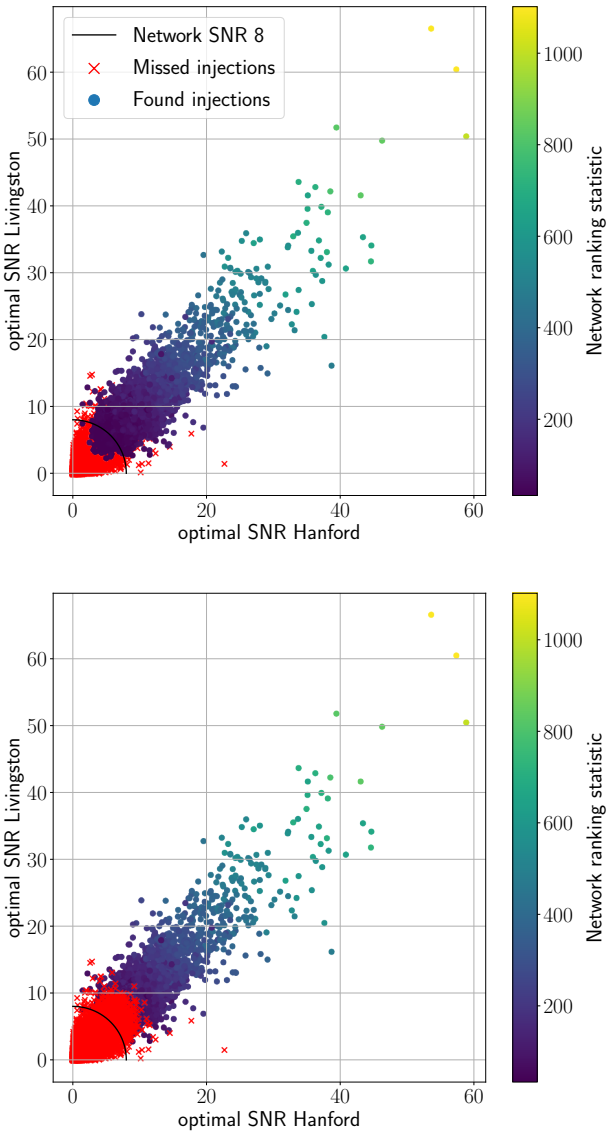


Figure 7.1: Found and missed injections from the test set as returned by the procedure discussed in section 7.2. The top panel overlays the missed injections by the found injections and the bottom panel reverses the order. The x- and y-axis show the optimal SNRs of the injections in the Hanford and Livingston detector, respectively. The color of found injections represents the combined ranking statistic as defined by equation (7.5). Missed injections are marked by a red cross. The black line indicates an optimal network SNR of 8. The plot is generated at a FAR of 1 false alarm per month.

The sensitivities of the machine learning search algorithms are compared to an equivalent matched filter search. For the single detector searches given by the dashed lines in Figure 7.2 we find that the machine learning algorithm retains at least 91.5% of the sensitivity at a fixed FAR of the matched filter analogue. This corresponds to a maximum absolute separation of 200 Mpc. This difference in sensitivity is basically unchanged when data from two detectors is considered and both the machine learning as well as the matched filter search calculate coincidences only based on the timing in the different detectors. The corresponding curves in Figure 7.2 are the filled orange and the dash-dotted purple line, respectively. In this case, the machine learning algorithm retains at least 92.4% of the sensitivity of the time coincidence matched filter search which corresponds to an absolute separation of 180 Mpc.

However, matched filtering also carries information about the intrinsic parameters of the source, the relative phase, and the relative amplitudes in the two detectors. This information can be used to further constrain coincidences and improve the ranking statistic [163] by testing for signal consistency. We compare the time coincidence machine learning search (filled, orange line in Figure 7.2) to this matched filter coincidence search utilizing signal consistency checks (filled, purple line in Figure 7.2). The machine learning search now only retains at least 83.9% of the sensitivity in FAR regions where both are defined. This corresponds to an absolute separation of 430 Mpc.

We truncate the sensitivity curve of any search that has access to data from both detectors in Figure 7.2 at a FAR of 10^3 false alarms per month. This is done due to a large number of true positives at high FARs originating from random noise coincidences. This means that the search returns a coincident event that is caused by a particular noise realization which happens to coincide with an injection with an optimal SNR below the trigger threshold. Many of these injections should thus not be recoverable but are detected at high FAR due to these noise fluctuations. At a FAR of 10^3 per month we expect less then $\mathcal{O}(10)$ of these false associations. Another reason to only compare the sensitivity at low FARs of the machine learning and the matched filtering based searches are the thresholds used to find triggers. The matched filter search uses a threshold of SNR 5 whereas the machine learning search uses a threshold on the USR ranking statistic of -2.2. Because there is no direct relation between these two statistics, we cannot guarantee that both thresholds correspond to similar signal strengths. It may be possible that one search excludes weak signals which are found by the other based on this difference in the threshold.

The sensitivity difference between machine learning and matched filtering stays constant between using data from a single detector and using data from two detectors when matched filtering may only check for time consistency

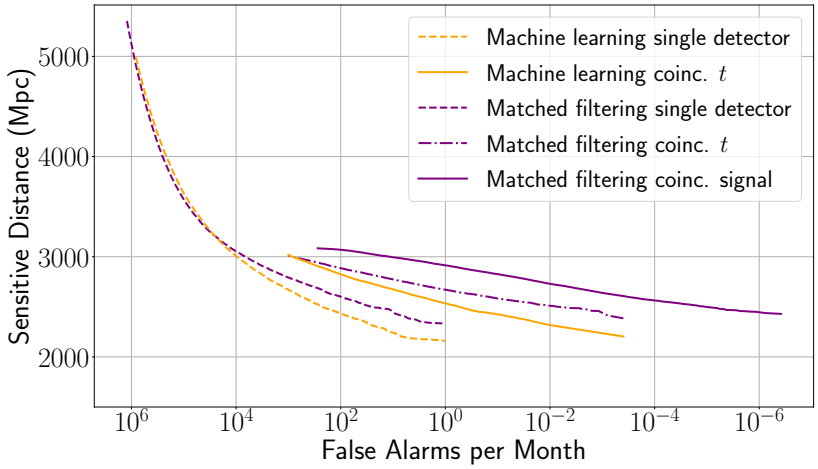


Figure 7.2: Shown are the sensitive distances of different search algorithms as a function of the FAR. In orange we show the sensitivity curves of the machine learning based searches presented in [63] and this work. In purple we show sensitivity curves of an equivalent matched filter search. The dashed lines are derived on data only from a single detector. A label "coinc. t" refers to events being tested for coincidence based solely on the time difference of the events in the two detectors. The label "coinc. signal" means that the matched filter search also checked for signal consistency based on the time-, phase-, amplitude-difference, and intrinsic parameters in the two detectors. Sensitivities derived on data from more than one detector are truncated at a FAR of 10^3 per month due to an increasing number of true detections caused by random coincident events in the noise.

between detection candidates from the two observatories. The performance difference increases when matched filtering also checks for signal consistency. It is, therefore, reasonable to believe that a multi detector machine learning search may be more sensitive when it too can check for signal consistency. This would either require the single detector network to output parameter estimates of the detected signal alongside a ranking statistic or a single network that uses the data from both detectors as input. In the following section 7.3 we explore the second hypothesis.

7.3 Two Detector Network

The deep learning algorithm presented in section 7.2 is significantly less sensitive than the full matched filter analysis that takes signal consistency into account. On the other hand, when the deep learning algorithm is compared

to the matched filter search where signal consistency is ignored, the difference in sensitivity is comparable to the difference in sensitivity for a single detector. This gives reason to believe that the difference in sensitivity compared to the full matched filter search could be reduced when the network may operate on the data from both detectors and consider coincidences itself.

7.3.1 Architecture

We construct a network that uses data from both detectors while still retaining the ability to efficiently estimate a large background. The network from section 7.2 is still applied to the data from the two detectors individually. However, the final layer is removed and the 64 output-neurons from both networks are concatenated. We then add 3 more fully connected layers to look for coincidences between the detectors. An overview of the network is shown in Figure 7.3.

The last layer from the single detector network is removed to create a large latent space. A matched filter search compresses the input data into the ranking statistic, the time of the merger, and the parameters of the best matching template. The intention is that 64 neurons may be sufficient for a comparable compression and that the additional layers that operate on the concatenated outputs could perform a signal consistency analysis.

The sub-networks A and B in Figure 7.3 are intended to act as encoders that reduce the 2048 dimensional input into a latent space of dimension 64. It may be interesting in the future to train these sub-networks initially as autoencoders [377] from which only the encoder is used for detection purposes afterwards. Autoencoders are neural networks which in the most simple form consist of an encoder network and a decoder network. The encoder network compresses the input to some lower dimensional latent representation whereas the decoder uses that lower dimensional representation to reconstruct the input. Other studies have already found that autoencoders have potential applications in GW data analysis [312, 378].

7.3.2 Data Sets and Training

The network is trained on data similar to that presented in subsection 7.2.2. However, the data is extended to two detectors and sources are uniformly distributed in the sky. The latter change is required due to the amplitude and phase correlations in the two detectors. We use the same number of noise and signal samples as in subsection 7.2.2.

We utilize the pre-trained single detector network used in section 7.2 in two different ways. In both cases the single detector parts of the two detector

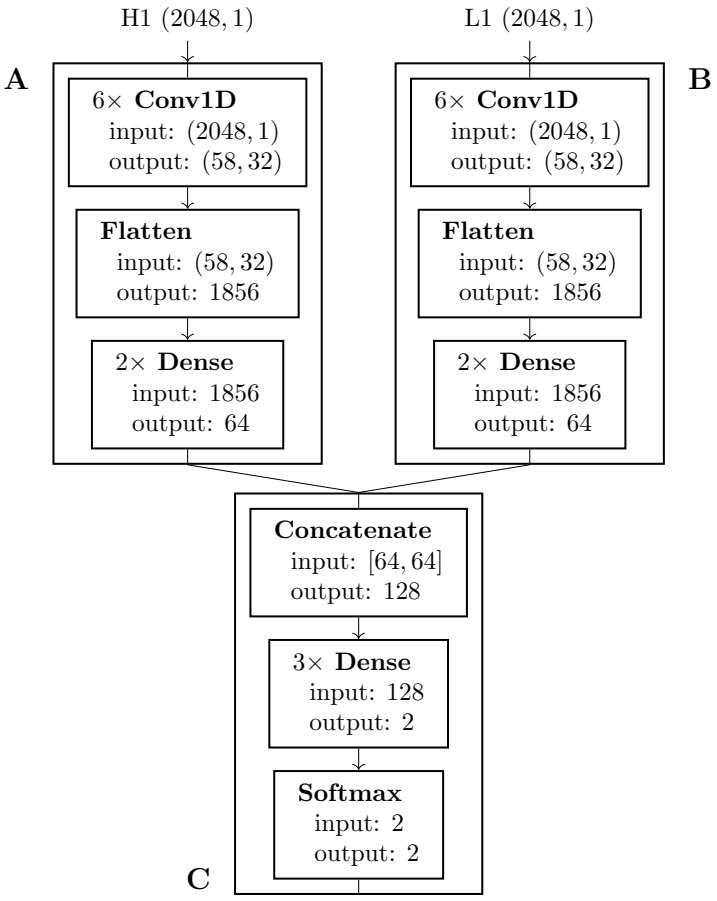


Figure 7.3: A high level overview of the two-detector architecture. The network consists of three sub-networks A, B, and C. A detailed description of the sub-networks A and B can be found in Table 7.1 by removing the final row. The fully connected Dense layers contain 128, 64, and 2 neurons in that order. All but the final Dense layer are equipped with an exponential linear unit (ELU) activation.

network (A and B in Figure 7.3) are initialized with the weights of the pre-trained model from section 7.2. However, for one of the two networks, these weights are then not optimized during training, leaving only the weights of the final fully connected layers (C in Figure 7.3) to be adjusted. This approach is known as transfer learning [379] and has been successfully applied for different problems [380–382]. The second network optimizes the weights of the entire network. We also train a third network of the same architecture, where all parameters are initialized randomly and optimized during training.

The same optimizer settings and loss function described in subsection 7.2.2 are used to train all three networks for 300 epochs. They are trained with a Softmax activation on the final layer, which is removed during evaluation. Each network is only trained once and the epoch with the highest efficiency on the validation set is chosen for further analysis.

7.3.3 Coincident Events

Because the networks output a single value when given the data from two detectors, we interpret that output as a coincidence ranking statistic at the corresponding time. We then perform the same clustering and thresholding described in subsection 7.2.3 to obtain a list of coincident events.

7.3.4 Background Estimation

Determining the background of the two detector network is more challenging than for the single detector network from subsection 7.2.5, as there is no direct way of performing time shift in a computationally efficient way. One would, therefore, naively be limited by the duration of the analyzed data or would have to re-evaluate the entire month of test data multiple times. However, the network is designed in such a way that the data from both detectors are still analyzed individually and combined only at later stages. We evaluate the single detector data individually with the sub-networks A and B from Figure 7.3 and store those outputs. We then permute the order of the outputs from sub-network B such that it corresponds to a time shift with respect to the output from sub-network A. Finally, sub-network C is applied to the concatenated data from sub-network A and B for many different time shifts. Since sub-network C is very simple and time shifts can be generated trivially this process generates $\mathcal{O}(1000)$ months of background within $< 12\,\mathrm{h}$ on a NVIDIA RTX 2070 Super.

7.3.5 Evaluation and Comparison to Matched Filtering

Figure 7.4 shows the sensitive distance of the various networks as a function of the FAR and compares them to the results presented in subsection 7.2.8. All curves are truncated at a FAR of 10^3 per month due to the large number of false associations described in subsection 7.2.8. The three networks utilizing the data from both detectors described in this section are labeled as "Machine learning network coinc.". The matched filter results are shown in purple, where the dash-dotted line considers only time coincidence and the filled line also takes the consistency of intrinsic source parameters, phase, and amplitude into account. The orange line corresponds to the network from section 7.2.

The networks described in this section were designed to be able to take signal consistency into account by reducing the input data to a large latent space. As such we were expecting sensitivities at low FARs to be larger than those obtained from time coincidence between single detector events produced by the single detector network.

However, we find that at low FARs all of the two detector networks are roughly as sensitive as the network tested in subsection 7.2.8. Therefore, they are still less sensitive than the matched filter equivalent and do not seem to take signal consistency into account. For high FARs, on the other hand, they are more sensitive. We suspect that the large time variation of the peak amplitude of $\pm 0.1\,\mathrm{s}$ may be responsible for this behavior. The networks are, thereby, trained to be insensitive to variations in timing of less then $0.1\,\mathrm{s}$, which may produce phase and amplitude variations in a broad range.

7.4 Conclusions

In this paper we have extended the single detector deep learning GW search algorithm from [56, 63] to two detectors and compared it to an equivalent matched filter algorithm. We found that the most simple extension, applying the one detector network to the data from two detectors individually and searching for coincident events, retains $\approx 92\%$ of the sensitivity of matched filtering, when only the time consistency between detectors is required. This fraction drops to $\approx 84\%$ when signal consistency between detectors is also considered.

To operate on data from two observatories, we constructed a two detector ranking statistic for the machine learning search based on the single detector USR ranking statistic proposed in [63]. This ranking statistic proved to be

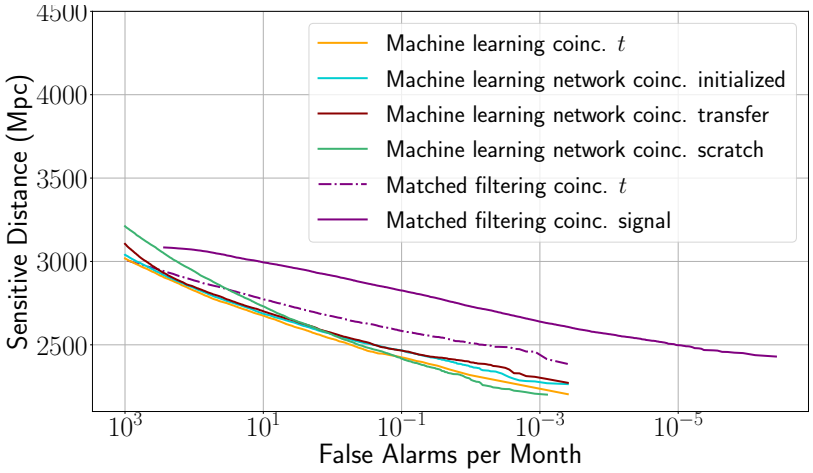


Figure 7.4: The sensitivity of different search algorithms as a function of the FAR. All shown algorithms operate on the data from two detectors. The curves labeled "Machine learning coinc." are neural network search algorithms that consider data from both detectors and an overview can be found in Figure 7.3. The network labeled "initialized" initializes the sub-networks A and B as shown in Figure 7.3 from the single detector network used in subsection 7.2.8 but optimizes them during the subsequent training. The network labeled "transfer" also initializes both sub-networks as the "initialized" network but freezes their weights. The network labeled "scratch" initializes all parameters of the network randomly. All other searches operate on the data from the individual detectors first and then search for coincident events. A label "coinc. t" refers to events being tested for coincidence based solely on the time difference of the events in the two detectors. The label "coinc. signal" means that the matched filter search also checked for signal consistency based on intrinsic parameters and the time-, phase-, and amplitude-difference in the two detectors. The curve labeled "Machine learning coinc. t" refers to the two-detector machine learning search analyzed in subsection 7.2.8. All sensitivities are truncated at a FAR of 10^3 per month due to a growing number of true positive detections caused by the coincidence of noise events.

correlated with the network SNR.

We also highlighted the advantages of using a single detector network to construct a two detector search. First, the single detector network does not need to be re-trained to be applied to the second detector, if both have similar noise characteristics. Second, this approach enables an efficient background estimation by applying relative time shifts to the recovered single detector events. This allows to test the two detector search to almost arbitrarily low FARs at low computational expenses. This method has already proven to be effective and reliable in state-of-the-art classical search algorithms [60, 158, 356].

Because using a single detector network restricts one to check for coincidences based solely on the timing difference, we tested a simple network that operates on data from both detectors directly. This allows the network in principle to construct internal signal representations which can be correlated between observatories. The network was constructed by removing the final layer of the single detector network, concatenating the outputs and adding a few fully connected layers to check for coincident events. The final fully connected layers, thus, receive 64 latent variables for each detector that can be checked for coincidence.

This design of the two detector network allowed us to do efficient background estimation. By applying relative time shifts to the outputs of the individual detector sub-networks, only the final few fully connected layers need to be evaluated for all shifts. The bulk of the computation, namely evaluating the input data of the detectors, only needs to be done once.

The network architecture was trained in three different ways; randomly initialized parameters for the entire network, parameters of the sub-networks initialized from the single detector network, and parameters of the individual detector sub-networks fixed to the single detector parameters and optimizing only the final fully connected layers.

We found that all of these networks have very similar performance at low FARs. Neither of them performed substantially better than the initial network that looked for time coincident events between the single detector network outputs. It, therefore, seems as if the network architecture explored here is unable to learn any additional information about the signal. This may be caused by the allowed time-variance of $\pm 0.1\,\mathrm{s}$ for signals in the training set, which may limit the time resolution of the network and thus overshadow correlations in any other parameters. More sophisticated network architectures with higher time resolution may improve our findings. First promising steps have already been taken by [57, 374]. Using an autoencoder to find a more meaningful latent representation of the input data may also be of use.

While the sensitivity was not improved by using a single network to pro-

cess the data of two detectors, we still want to highlight that the method of determining the background may be of use for future networks.

Here we limited our research to GWs from non-spinning binary black holes with signal duration $< 1\,\mathrm{s}$ and Gaussian noise. Any of these simplifications are desirable to be lifted. Especially considering real noise may increase the gap in sensitivity between the single detector and multi detector search algorithm, by vetoing glitches. While we considered only two detectors an extension to a larger network should be trivial and may follow studies such as [164].

Chapter 8

MLGWSC-1: The first Machine Learning Gravitational-Wave Search Mock Data Challenge

Contents

8.0 Abstract 154
8.1 Introduction 155
8.2 Methods 158
8.2.1 Challenge Resources 158
8.2.2 Test Data 158
8.2.3 Evaluation 164
8.2.4 Submission Requirements 167
8.3 Submissions 169
8.3.1 MFCNN 170
8.3.2 PyCBC 171
8.3.3 CNN-Coinc 172
8.3.4 TPI FSU Jena 173
8.3.5 Virgo-AUTh 174
8.3.6 cWB 175
8.4 Data release 176
8.5 Results and discussion 177
8.5.1 Sensitivities 178
8.5.2 Found and missed injections 185

8.5.3 Runtimes 186

8.6 Conclusions 188

This chapter is essentially a full reprint of [65] with some minor edits for formatting and adjustments to address comments from two referee reports. It discusses the results of a mock data challenge we organized that was targeted at placing machine learning GW searches into the context of existing, state-of-the-art algorithms.

8.0 Abstract

We present the results of the first Machine Learning Gravitational-Wave Search Mock Data Challenge (MLGWSC-1). For this challenge, participating groups had to identify gravitational-wave signals from binary black hole mergers of increasing complexity and duration embedded in progressively more realistic noise. The final of the 4 provided datasets contained real noise from the O3a observing run and signals up to a duration of 20 seconds with the inclusion of precession effects and higher order modes. We present the average sensitivity distance and runtime for the 6 entered algorithms derived from 1 month of test data unknown to the participants prior to submission. Of these, 4 are machine learning algorithms. We find that the best machine learning based algorithms are able to achieve up to 95% of the sensitive distance of matched-filtering based production analyses for simulated Gaussian noise at a false-alarm rate (FAR) of one per month. In contrast, for real noise, the leading machine learning search achieved 70%. For higher FARs the differences in sensitive distance shrink to the point where select machine learning submissions outperform traditional search algorithms at FARs $\geq$ 200 per month on some datasets. Our results show that current machine learning search algorithms may already be sensitive enough in limited parameter regions to be useful for some production settings. To improve the state-of-the-art, machine learning algorithms need to reduce the false-alarm rates at which they are capable of detecting signals and extend their validity to regions of parameter space where modeled searches are computationally expensive to run. Based on our findings we compile a list of research areas that we believe are the most important to elevate machine learning searches to an invaluable tool in gravitational-wave signal detection.

8.1 Introduction

The first gravitational-wave (GW) observation on September 14, 2015 [13] achieved by the LIGO and Virgo Collaboration [6, 7] started the era of GW astronomy. During the first observing run (O1) two more GWs from coalescing binary black holes (BBHs) were detected. The second observing run (O2) saw $\mathcal{O}(10)$ additional confident BBH detections as well as the first detection of a binary neutron star (BNS) merger [24, 150, 151, 212, 346, 383]. The third observing run (O3) was split into two parts, O3a and O3b. During O3a a further $\mathcal{O}(40)$ BBHs as well as a second BNS merger were reported [206, 356, 384]. O3b added another $\mathcal{O}(40)$ BBH events as well as finding the first two confident detections where the component masses are consistent with the merger of a neutron star black hole system (NSBH) [14, 15]. The fourth observing run (O4) is scheduled to begin in early 2023 and is expected to significantly increase the volume from which sources can be detected [194, 275].

GW signals are commonly identified in the background noise of the detectors using matched filtering [14, 59, 161, 343]. Matched filtering compares pre-computed models of expected signals, known as templates, with the data from the detectors [30]. When a model matches the data to a pre-defined degree and data-quality requirements are met, a candidate detection is reported. Loosely modelled searches [37–39], which look for coherent excess power in multiple detectors, are also employed by the LIGO-Virgo-KAGRA collaboration (LVK) to find potential signals.

The rate of detections has drastically increased from O1 to O3. This increase was enabled by continued detector upgrades at the two advanced LIGO observatories in Hanford and Livingston [6], as well as sensitivity improvements for the advanced Virgo detector [7]. With the entry into service of Kagra [8] a fourth observatory joined the network of ground based GW-detectors towards the end of O3. The rate of detections is expected to further increase during O4 as the sensitivity of the detectors improves and the volume from which sources can be detected grows.

With an increasing rate of detections, it is likely that systems with unexpected physical properties will be observed more frequently in the future. Optimally searching for these is a challenge for matched filtering based searches, where the computational cost scales linearly with the number of templates used. The inclusion of effects such as precession, eccentricity, or higher order modes requires millions of templates to not miss potential signals [40, 41, 44] and thus are computationally prohibitive, especially when real-time alerts should be issued. Loosely modeled searches are inherently capable of

detecting arbitrary sources at a fixed computational cost but are prone to miss more signals due to their lower sensitivity in parameter regions where accurate models exist.

In recent years, machine learning has been applied in many scientific fields to enable or improve research into computationally expensive topics [231]. Some examples include the prediction of protein structure used in pharmaceutical studies [232], improvements to material composition and synthesis [234], or event reconstruction at the Large Hadron Collider [235]. There is also ongoing research into using neural networks to discover closed form expressions from raw data [385] or optimizing machine learning algorithms to take advantage of physical symmetries of the underlying problem [386–388].

More relevant to this work, machine learning algorithms have also started to be explored as alternative algorithms for many GW data-analysis tasks. These include detector glitch classification [190, 281, 389], parameter estimation [311, 312, 315, 390, 391], continuous GW detection [302–305, 392–394], enhancements for existing pipelines [167, 296, 298, 300, 309, 317, 395, 396], surrogate waveform models [397–399], as well as various signal detection algorithms [55–57, 61, 63, 64, 287–289, 291, 307, 308, 359, 373, 374, 400–405]. For a summary of many methods we refer the reader to [48, 360]. In this work we focus solely on detection algorithms for BBH GW signals, which have been the most commonly observed type of sources to date [206, 356, 384]. These signals are the easiest to detect for machine learning algorithms due to their short duration.

Many of the works considering the usage of machine learning for GW signal detection are difficult to cross-compare. Most algorithms target different datasets and derived metrics are often motivated more by machine learning practices than by state-of-the-art GW searches. It is, therefore, hard to pinpoint exactly how capable machine learning search algorithms currently are and where the main difficulties arise. To achieve the goal of an objective characterization of machine learning GW search capabilities, a common ground for comparison is required.

Here we present the results of the first Machine Learning Gravitational-Wave Search Mock Data Challenge (MLGWSC-1). In an attempt to provide a common ground of comparison for different algorithms and in preparation of O4, we have calculated sensitive distances from 6 different submissions calculated on datasets of one month duration to collect and compare a suite of searches. We want to motivate the utilization of machine learning based searches in a production setting by providing a definitive resource to allow for easy comparison between different algorithms, be it machine learning based, matched filtering based, or completely unmodeled. This challenge is the first

of its kind[1] and hopefully more will be held in the future, expanding to more difficult scenarios.

The mock data used in this challenge consists of 4 datasets containing noise of increasing realism and signals with increasing complexity for the two detectors LIGO Hanford and LIGO Livingston [6]. The final dataset challenges participants to identify GWs from spinning BBHs with a duration of up to 20 s added to real detector noise from O3a. The signals also take precession effects and higher order modes into account.

Submissions are evaluated on mock data of one month duration for each of the four datasets. We calculate sensitive distances for each algorithm and estimate the computational efficiency based on the runtime. The final dataset should provide an accurate picture of the possible real-world performance these algorithms can achieve. However, we note that direct comparison of the runtime performance of the different algorithms is complicated by differing hardware usage and optimization.

We find that machine learning algorithms are already competitive with state-of-the-art searches on simulated data containing injections drawn from the limited parameter space covered by this challenge. The most sensitive machine learning algorithm manages to retain $\geq 93\%$ of the sensitive distance measured for the PyCBC pipeline [15] on Gaussian background data down to a false-alarm rate (FAR) of 1 per month. For higher FARs the separation between the approaches generally shrinks.

Most machine learning searches, as tested here, are less sensitive on real noise than on simulated data. The traditional algorithms handle this transition better. As a consequence, the most sensitive machine learning algorithm retains $\geq 70\%$ of the sensitive distance of the PyCBC search down to a FAR of 1 per month. However, the sensitivity achieved of machine learning algorithms on real data is still substantial and shows that they are capable of rejecting non-Gaussian noise artifacts without any hand-tuned glitch classification.

From the evaluation of the different datasets we conclude that the main difficulties for current machine learning algorithms are the ability to analyze the consistency of detected signals between detectors and the maximum duration of signals that can be detected. Solving these issues would allow for better performance at FARs < 1 per month and enable a fast detection of potentially electromagnetic bright sources such as BNS or NSBH mergers.

All code used in this challenge is open source and available at [407]. Therein we also collect the individual submissions by groups that have given

[1] There has previously been a public Kaggle challenge [406]. First in the sense of this paper refers to our setup of providing continuous data.

their consent, provide the analysis results, and make available all plots used in this paper for all submissions.

This paper is structured as follows. In section 8.2 we provide the details on the challenge, the datasets, as well as the evaluation process. All submissions are briefly introduced in section 8.3. The results of the challenge and a brief discussion can be found in section 8.5. We conclude and give an outlook into possible future work in section 8.6.

8.2 Methods

All submissions described in section 8.3 are evaluated on the same datasets, and all machine learning submissions are evaluated under the same conditions. Below we describe the provided material from the challenge, the requirements for the submitted algorithms, as well as the evaluation process.

8.2.1 Challenge Resources

In this challenge participants are asked to identify GW signals submerged in detector noise. To provide grounds of comparison, all submissions are evaluated on the same datasets. To allow for optimization of the submitted algorithms for the task at hand, participants had access to code that allowed them to generate arbitrary amounts of data equivalent to that used during the final evaluation of this challenge. All code used for data generation and algorithm evaluation is open source and can be found at [407].

In particular, participants had access to the code that was used to generate the final challenge sets, but not the specific seed that was used. The specifics of the datasets are described in subsection 8.2.2. They were also provided with the code that was used to generate the metrics we provide in this paper. Details on the metrics can be found in subsection 8.2.3.

8.2.2 Test Data

The challenge provides a script to generate semi-continuous raw test data for any of the four datasets described below. It allows the user to choose a specific seed and a total duration of the output data. The code subsequently generates up to three files; the first containing pure noise, the second containing the same noise with injected GW signals, and the third containing the parameters of the injected signals.

The files containing the pure noise and the noise with additive signals are of the same structure. They are HDF5 [408] files with two groups named “H1”

and "L1" containing data from the two detectors LIGO Hanford and LIGO Livingston, respectively. Each group consists of N HDF5-datasets, each holding the detector data of a single segment, as well as information on the GPS starting time of the segment, and its sampling rate. Each segment has a minimum duration of 2 h, is sampled at 2048 Hz, and contains continuous data. The files also contain information on the meta-data used to create the file. This meta-data is removed in the final challenge sets.

We chose to split data into smaller segments of uncorrelated noise for two reasons. First, real detectors are not equally sensitive for months at a time and data quality differs to an extent where certain data cannot be used for analyses. As such, any algorithm should be able to handle gaps in the data. Second, the noise characteristic varies over time. Segmenting simulated data allows us to easily incorporate different models for the power spectrum over the duration of the data. Subsequently, the noise model can be increased in complexity for the four datasets.

Minimal pre-processing is done on the data that is handed to the submitted algorithms. We only apply a low-frequency cutoff of 15 Hz which is used to enable a reduction in file size for real-detector data that has to be downloaded. The low-frequency cutoff reduces the dynamic range of the data, which allows us to scale the data and cast it to lower numerical precision. Any other pre-processing is left to the algorithms and is factored into the performance evaluation. The scaling is inverted during data loading.

A larger index of the dataset signifies a greater complexity and realism of the dataset. Participants may choose to optimize for any of the 4 datasets but are only allowed to submit a single algorithm, which is subsequently tested with all 4 datasets. We do this to test the ability of the search to generalize to slightly varying conditions.

Many parameters of the injected signals are drawn from the same distributions irrespective of the dataset. A summary of these distributions can be found in Table 8.1. All signals are generated using the waveform model `IMRPhenomXPHM` [133] with a lower frequency cutoff of 20 Hz. The waveform model was chosen for its ability to simulate both precession and higher-order modes. This setup assures that at least 33% of injected signals have an optimal network SNR < 4 and can thus not be detected. The merger times of two subsequent signals are seperated by a random time between 24 s to 30 s to avoid any overlap. We apply a taper to the start of each waveform.

In Figure 8.1 we show an overview of the intrinsic parameters used in this challenge and compare it to the parameter space searched by state of the art searches [14, 15].

Parameter	Uniform distribution
Coalescence phase	$\Phi_0 \in (0, 2\pi)$
Polarization	$\Psi \in (0, 2\pi)$
Inclination	$\cos\iota \in (-1, 1)$
Declination	$\sin\theta \in (-1, 1)$
Right ascension	$\varphi \in (-\pi, \pi)$
Chirp-Distance	$d_c^2 \in (130^2, 350^2)\,\mathrm{Mpc}^2$

Table 8.1: A summary of the distributions shared between all datasets from which parameters are drawn.

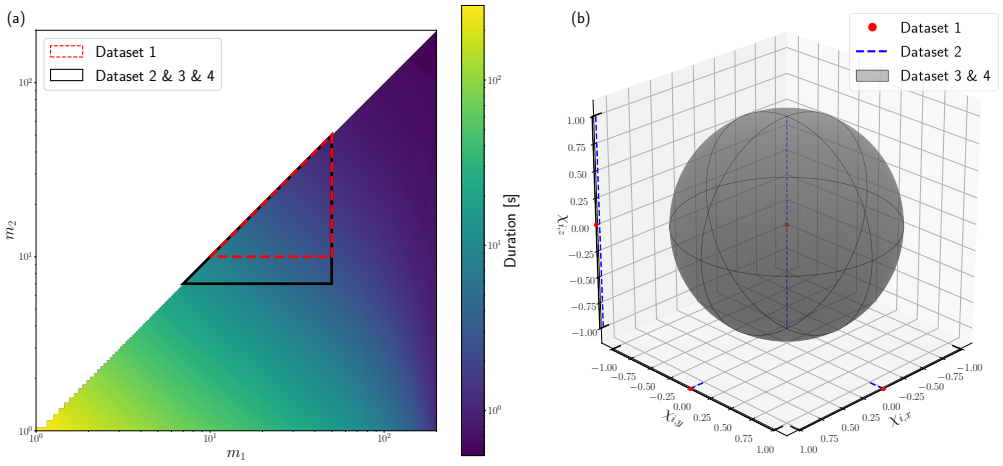


Figure 8.1: An illustration of the range for the intrinsic parameters covered by this challenge. The left panel (a) shows a typical range for the component masses used by state of the art searches [15]. The color indicates the duration of the waveform from 20 Hz. The triangles show the parameter regions covered by this challenge. The right panel (b) shows the component-spin χ_i distribution of the different datasets in this challenge.

Dataset 1

The noise from the first dataset is purely Gaussian and simulated from the PSD model `aLIGOZeroDetHighPower` [319] for both detectors. This means that the PSD used to generate the data contains no sharp peaks originating from factors such as the power grid, is the same for all segments, and is known to the participants.

Injected signals are non-spinning and no higher-order modes are simulated. The component masses are uniformly drawn from $10\,\mathrm{M}_\odot$ to $50\,\mathrm{M}_\odot$. We enforce the condition that the primary mass has to be equal or larger than the secondary mass. With this mass range, at a lower frequency cutoff of 20 Hz, and for non-spinning systems the signal duration is on the order of 1 s.

The first dataset represents a solved problem, as it has already been excessively studied in the past [55, 56, 64]. It is meant as a starting point where people new to the field can refer to existing literature to get off the ground initially. We expected many of the algorithms to perform equally well on this set.

The final challenge set for dataset 1 was generated with the seed 1 068 209 514 and a start time of 0.

Dataset 2

The noise for the second dataset is also purely Gaussian and simulated. However, in contrast to the first dataset the PSDs were derived from real data from O3a and as such contain power peaks at certain frequencies and are noisy. We generated a total of 20 PSDs for each detector. The PSDs used to generate the noise are randomly chosen from these lists and as such are unknown to the search algorithm. The lists themselves are known to the participants. The PSDs in both detectors are independent of each other but do not change over time.

Signals are now allowed to have a spin aligned with the orbital angular momentum with a magnitude between -0.99 and 0.99. Additionally, the mass range is adjusted to draw component masses from the range $7\,\mathrm{M}_\odot$ to $50\,\mathrm{M}_\odot$. This change increases the maximum duration of the signals at a lower frequency cutoff of 20 Hz to ≈ 20 s. No higher-order modes are simulated for this dataset and due to the aligned spin requirement no precession effects are present in the waveform.

The second dataset was intended to pose a considerable increase in difficulty to the first dataset. Using an unknown PSD which was derived from real data requires participants to estimate it during the analysis, if the algo-

rithm requires it. However, we expected that increasing the signal duration to up to 20 s would be the more prominent reason for an increase in difficulty as many previous machine learning algorithms have had trouble when dealing with large inputs [61, 213, 303, 312]. Finally, we did not expect a large increase in the difficulty of the dataset due to the inclusion of aligned spins.

The final challenge set for dataset 2 was generated with the seed 2 743 406 703 and a start time of 0.

Dataset 3

The noise for the third dataset is also simulated and purely Gaussian. The increase in difficulty of the noise comes from varying the PSDs over time. Instead of choosing a single random PSD from the list of 20 PSDs per detector described in Figure 8.2.2 and generating all noise with that one PSD, the PSD for dataset 3 is randomly chosen for each segment.

The mass range from $7\,\mathrm{M}_\odot$ to $50\,\mathrm{M}_\odot$ and subsequently the maximum signal duration of 20 s is unchanged compared to Figure 8.2.2. However, instead of requiring the spins to be aligned with the orbital angular momentum, their orientation is isotropically distributed with a magnitude between 0 and 0.99. As a consequence, precession effects are now present in the waveforms. Additionally, we also model all higher-order (l, m)-modes available in `IMRPhenomXPHM`, which are: $(2, 2)$, $(2, -2)$, $(2, 1)$, $(2, -1)$, $(3, 3)$, $(3, -3)$, $(3, 2)$, $(3, -2)$, $(4, 4)$, $(4, -4)$ [133].

The main challenge of this dataset was intended to be the inclusion of precession effects. While these are not as impactful for short duration, high mass systems, they can substantially alter the signal morphology for lower mass systems. Adding higher-order modes can also substantially increase signal complexity. Both of these effects are currently not modeled in any production search relying on accurate signal models, as their inclusion requires an increase in size of the filter bank to include millions of templates [40, 41]. As such, we expected many if not all of the submitted algorithms to struggle with this dataset. On the other hand, any machine learning based algorithm that operates successfully on this dataset may motivate the utilization of machine learning in production searches in the future by extending the searchable parameter space.

The final challenge set for dataset 3 was generated with the seed 470 182 217 and a start time of 0.

Dataset 4

Dataset 4 is the only dataset that contains real detector noise obtained from the Gravitational Wave Open Science Center (GWOSC) [149]. All noise was sampled from parts of O3a that had the "data" quality flag and none of the flags "CBC_CAT1", "CBC_CAT2", "CBC_HW_INJ", or "BURST_HW_INJ" were active. We consider only segments where the data from both LIGO Hanford and LIGO Livingston clear the above conditions and excluded 10 s around any detection listed in GWTC-2 [356]. Afterwards we discarded any segments shorter in duration than 2 h. To allow for different noise realizations, we shift the data from LIGO Livingston by a random time from 0 s to 240 s while keeping the data from LIGO Hanford fixed. The time shifts are independent for each segment and to avoid any possible overlap between neighbouring segments, we consider each segment on its own.

To reduce the amount of data that has to be downloaded by participants we pre-selected the suitable parts of the O3a data. We then applied a low frequency cutoff of 15 Hz to reduce the dynamic range of the data and multiplied the numerical values by a factor of $\approx 2^{69}$ to allow a lossless conversion to single precision. Finally, the data was converted to single precision and stored in a compressed format. This allowed us to provide a download link to a single file of 94 GB size containing enough data to generate up to 7 024 699 s≈81 d of coincident real noise for both detectors. The data was scaled by the constant factor to avoid the loss of dynamic range due to the conversion from double precision to single precision. When generating test data, the data is converted back to double precision and the scaling is inverted. The code used to downsample the data is also open source and available at [407].

The signals are generated equivalently to the signals in dataset 3, i.e. masses are uniformly drawn from $7\,M_\odot$ to $50\,M_\odot$, spins are isotropically distributed with a magnitude from 0 to 0.99, and all higher-order modes available in `IMRPhenomXPHM` are generated. Consequently, precession effects are simulated.

This dataset is intended to be indicative of a real-world application of the search in parameter regions which are currently sparsely searched. Given that many machine learning searches have proven to generalize well from Gaussian noise to real detector noise at higher FARs in the past [57, 288, 289, 359] we expected that machine learning algorithms that do well on dataset 3 will also be competitive for dataset 4. However, it was expected that handling short glitches may prove difficult for certain searches, especially those focusing most on the merger and ringdown.

The final challenge set for dataset 4 was generated with the seed 2 514 409 456 and a start time of 0.

8.2.3 Evaluation

All submissions are evaluated on the challenge sets, which are generated with a seed unknown to the participants at the time of submission. The evaluation is run on the Atlas computing cluster at the Albert-Einstein-Institut (AEI), Hannover. Groups that submitted an algorithm had no direct access to the evaluation stage[2] and final results presented in this work were only communicated back to the groups after the submission deadline had passed.

We compute two metrics for every submission and dataset. These are the wall-clock time required by the algorithm at hand to analyze one month of data as well as the sensitive distance of the search as a function of the false-alarm rate. In essence, the sensitivity as a function of the false-alarm rate is a receiver operating characteristic (ROC) curve that factors in the varying signal strengths of the injected GWs. It is a common measure of search sensitivity for production GW-searches [60] and thus allows for easy comparisons. We do not compute the ROC curve directly, for two reasons. First, it requires the number of a negative samples in the data. Since our data is continuous and the evaluation is left to the groups, defining a negative sample is not possible. Second, the ROC curve can be changed by choosing a different signal population. For instance, the ROC curve can be driven to zero by choosing a population of signals that are excessively far from the detectors. The sensitive distance normalizes the data by the injected population.

For the calculation of the sensitive distances we use two challenge sets for each of the 4 datasets. The first contains pure noise and we will call it the background set from here on out. The second contains the same noise as the background set but adds GW signals into it. This second set will be called the foreground set from here on out. As described in subsection 8.2.4 any search algorithm is expected to process these files and return lists of events, where an event is a combination of a GPS time, a ranking statistic-like quantity, and a value for the timing accuracy. We will call these events background or foreground events when they have been derived from the background or foreground set, respectively. For the remainder of this section we will refer to the ranking statistic-like quantity simply as ranking statistic, to simplify our statements.

To calculate the sensitivity as a function of the false-alarm rate, we need to determine the false-alarm rate as a function of the ranking statistic. Next

[2]This excludes submissions by the organization group. However, no member of the organization group accessed the challenge-data before the submission deadline or altered their algorithm after the submission deadline.

we can also determine the sensitivity as a function of the ranking statistic. Finally, we can combine the two, by evaluating both at the same values of the ranking statistic.

We use the ranking statistic of all background events as points where both the FAR as well as the sensitivity is evaluated. Each of these is certain to be a false positive and thus ensures that the FAR is unique at each threshold, as long as the search does not return identical ranking statistics for multiple background events.

To calculate the FAR at a given ranking statistic we count the number of background events with a ranking statistic greater than this threshold. We, subsequently, turn that into a rate by dividing the number of false-positives by the duration of the background data, i.e. 2 592 000 s. With $N_{\mathrm{FP},\mathcal{R}}$ the number of false-positives at a given ranking statistic $\mathcal{R}$ and T the time spanned by the background set, the FAR $\mathcal{F}$ can be calculated by

$$\mathcal{F} = \frac{N_{\mathrm{FP},\mathcal{R}}}{T}. \tag{8.1}$$

The sensitive volume of a search at FAR $\mathcal{F}$ can be calculated by [60]

$$V(\mathcal{F}) = \int d\boldsymbol{x} d\boldsymbol{\Lambda}\, \epsilon(\mathcal{F}; \boldsymbol{x}, \boldsymbol{\Lambda})\, \phi(\boldsymbol{x}, \boldsymbol{\Lambda}), \tag{8.2}$$

where $\boldsymbol{x}$ are the spatial coordinates of the injection, $\boldsymbol{\Lambda}$ are the injection parameters, $\epsilon(\mathcal{F}; \boldsymbol{x}, \boldsymbol{\Lambda})$ is the efficiency of the search at FAR $\mathcal{F}$, and $\phi(\boldsymbol{x}, \boldsymbol{\Lambda})$ is the distribution of the injection parameters $\boldsymbol{x}$ and $\boldsymbol{\Lambda}$.

When injections are performed uniformly in volume up to a maximum distance d_{max}, Equation 8.2 can be approximated by [60]

$$V(\mathcal{F}) \approx V(d_{\mathrm{max}}) \frac{N_{I,\mathcal{F}}}{N_I}, \tag{8.3}$$

where $V(d_{\mathrm{max}})$ is the volume of a sphere with radius d_{max}, $N_{I,\mathcal{F}}$ is the number of found injections at a FAR of $\mathcal{F}$, and N_I is the total number of injections performed. An injection is found if there is at least one foreground event that is within $\pm\Delta t$ of the injection, where Δt is the time variance assigned to the event by the search algorithm. The number of found injections at a given FAR considers only those foreground events where the ranking statistic assigned to the specified event is greater than the ranking statistic corresponding to the FAR. In machine learning terms Equation 8.3 is the recall at a given threshold on the network output multiplied by the volume of a sphere with radius d_{max}, assuming that each injection corresponds to exactly one true positive.

However, the injections in the datasets are not performed uniformly in volume, as we sample over the chirp-distance instead of the luminosity distance. The chirp-distance is given by [409]

$$d_c = d\left(\frac{\mathcal{M}_{c,0}}{\mathcal{M}_c}\right)^{5/6}, \tag{8.4}$$

where d is the luminosity distance, $\mathcal{M}_c = (m_1 m_2)^{3/5}/(m_1+m_2)^{1/5}$ is the chirp mass, and $\mathcal{M}_{c,0}$ =1.4/$2^{1/5}$ $\mathrm{M}_\odot$ is a fiducial chirp mass used as a basis for calculation. Note that in contrast to [409] we use the luminosity distance instead of the effective distance as our basis.

When sampling the injections from the distributions defined in Table 8.1 using the chirp-distance, effectively the maximum luminosity distance d is selected based on the chirp mass; the smaller the chirp mass, the smaller the maximum luminosity distance at which injections are placed. This allows us to increase the number of detectable low mass systems and, subsequently, make statistically meaningful statements about the sensitivity for these systems without requiring a large increase in the amount of data that needs to be analyzed. However, when considering a fixed chirp mass, injections are still placed uniformly within that sphere of the adjusted maximum luminosity distance. In Equation 8.3 we assumed that each injection was placed uniformly within the volume spanned by the sphere with volume $V(d_{\max})$. To adjust it for sampling over luminosity distance we have to factor in that the probed distance depends on the selected chirp mass. We, therefore, find

$$V(\mathcal{F}) \approx \frac{V(d_{\max})}{N_I} \sum_{i=1}^{N_{I,\mathcal{F}}} \frac{V\left(d_{c,\max}\left(\frac{\mathcal{M}_{c,i}}{\mathcal{M}_{c,0}}\right)^{5/6}\right)}{V\left(d_{c,\max}\left(\frac{\mathcal{M}_{c,\max}}{\mathcal{M}_{c,0}}\right)^{5/6}\right)}, \tag{8.5}$$

where $\mathcal{M}_{c,i}$ is the chirp mass of the i-th found injection, $d_{c,\max}$ is the upper limit on the injected chirp distances, and $\mathcal{M}_{c,\max}$ is the upper limit on the injected chirp masses. This expression can be simplified to yield

$$V(\mathcal{F}) \approx \frac{V(d_{\max})}{N_I} \sum_{i=1}^{N_{I,\mathcal{F}}} \left(\frac{\mathcal{M}_{c,i}}{\mathcal{M}_{c,\max}}\right)^{5/2}, \tag{8.6}$$

which is the formula we use to estimate the sensitive volume of a search algorithm. Instead of quoting the volume directly we convert it to the radius of a sphere with the corresponding volume and quote that instead.

We also measure the time the algorithm requires to evaluate an entire month of test data. Since all machine learning search algorithms are running

Hardware type	Specification
CPU	2× Intel Xeon Silver 4215, 8(16) cores(threads) at 2.5 GHz
GPU	8× NVIDIA RTX 2070 Super (8 GB VRAM)
RAM	192 GB

Table 8.2: Main hardware specifications available to each search algorithm during final testing.

on the same hardware these values can be used to compare the speed of the different analyses on the given hardware. For a summary of the available hardware resources please refer to Table 8.2. However, we expect the computational time to be dominated by pre-processing steps, which can in theory be heavily optimized. For this challenge, though, we did not expect many submissions to invest resources into optimizing their pre-processing and thus advise the reader to not overemphasize the provided numbers.

All runtimes are measured twice; once for the foreground set and once for the background set. In both cases the wall-time that has passed between calling the executable and it returning is measured.

8.2.4 Submission Requirements

All submissions are provided with the path to a single file containing the input data they have to process. In particular they have to be able to read HDF5 files, the structure of which is detailed in subsection 8.2.2. Importantly, no pre-processing other than the introduction of a low frequency cutoff of 15 Hz has been applied to the data. All other pre-processing has to be performed by the algorithms themselves. In addition to the path to the input data, each algorithm is provided with a second path at which it is expected to store a single HDF5 file. This file has to contain three one-dimensional datasets of equal size named "time", "stat", and "var".

The "time" dataset is expected to contain the GPS times at which the algorithm predicts a GW signal to be present. These are compared to the injection times to determine which injections were found, which were missed, and how many false positives the analysis produced.

The "stat" dataset is expected to contain a ranking-statistic like quantity for every GPS time in the "time" dataset. Here, ranking-statistic like quantity means a value where larger numbers indicate a higher degree of believe for the search to have found a GW signal. Having a ranking-statistic

Python package	Version
bilby	1.1.3
pycbc	efeaeb6
tensorflow-gpu	2.6.0
tensorflow-probability	0.14.0
torch	1.9.1+cu11

Table 8.3: A selective list of the core Python packages available to algorithms during evaluation. A complete list is given at [407].

like quantity associated to all candidate detections enables us to assign a statistical significance to any event.

The "var" dataset is expected to contain the estimated timing accuracy of the search algorithm for all GPS times in the "time" dataset. This value determines the window around the GPS time returned by the search within which an injection has had to be made in order to consider the detection a true positive and the injection to be found. This value may be constant for all times at which the search expects to have seen a signal. We allowed searches to specify this value themselves, as we felt it to be unsuitable for a signal detection challenge to require a fixed timing accuracy. In principle, this freedom can be abused by choosing an accessively high value of Δt and claiming all events as true positives. However, all groups have chosen values on similar scales and more importantly far shorter than the average separation of two injections.

Throughout the paper, we will refer to events returned by the search. By that we mean a single tuple $(t, \mathcal{R}, \Delta t)$ contained in the "time", "stat", and "var" datasets, respectively.

To be able to execute all algorithms without major problems, we ask participants to either provide a single executable that can be run on the Linux command-line utilizing only the provided software stack or to provide a singularity image that we can execute. In both cases the algorithms have to accept two positional command line arguments; the path to the input data file and the path at which the output file should be stored. The main Python packages available to submitted executables are listed in Table 8.3, for a full list refer to [407].

Each algorithm is executed by hand and closely monitored by the organization team of the challenge. Participants are not allowed to directly tune or influence the final evaluation.

To ensure that participants have submitted the correct version of their

algorithm and to make sure that their algorithm behaves as expected on the evaluation hardware and software, all algorithms are first evaluated on a validation set which is generated equivalently to the final test set. The results on this validation set are then communicated back to the submitting group. Once the group has approved that their algorithm performs within the expected margin of error, the algorithm is applied to the real challenge sets. These challenge sets are the same for all participants and were kept secret until the deadline for final submissions had passed.

Since multiple members of the organization team have submitted algorithms to this challenge, the challenge datasets were only generated after the submission deadline had passed. The script to generate test data provides an option to use a random seed. This option was used to generate the final challenge datasets and ensures that no submission had knowledge of the challenge set prior to the submission deadline.

We allowed all participants to retract their submissions at any point prior to the final publication of our results. This means that participants were allowed to retract their submissions even after they were informed about the performance of their algorithm on the final challenge sets and after they have seen the performance of other entries. No group made use of this freedom and retracted their submission after results were internally published.

8.3 Submissions

In this section we briefly introduce the different algorithms. For more details on the individual submissions we refer the reader to the original works cited within each subsection. The subsections are titled by the group name and are given in order of registration to the challenge.

All algorithm preparation was performed by the individual groups using their own available hardware resources. This crucially includes training of machine learning algorithms, for which no resources were provided by the organizers of this challenge. There were no strict requirements to submit algorithms that are based on machine learning techniques. We even encouraged the submission of a few traditional algorithms to quote a point of reference. However, the available resources detailed in subsection 8.2.3 for evaluation of the test sets are tailored to suit the needs of machine learning algorithms.

8.3.1 MFCNN

[3] The submission of the MFCNN group is based on the works from He et al. [290]. The authors of [290] refer to the model as matched-filtering convolutional neural network (MFCNN). MFCNN is a semi-coherent search model. The basic idea of the model is to use waveform templates as learnable weights in neural network layers. Analogously to the standard coincident matched-filtering searches the output of each matched-filtering layer is maximized and normalized in the unit of matched-filtering SNRs for each GW detector. However, triggers are not generated on a single detector. The remaining part of the neural network is a usual convolutional neural network that is employed afterwards to jointly analyze the output from all detectors. Finally, a SoftMax function is applied to evaluate the confidence score of a GW signal being present in the GW detector network. The architecture was designed to take the advantages of both matched-filtering and convolutional neural networks and combine them to search for real GW events in GWTC-1 [346]. To adapt to this challenge, the source code [410] of the submission was translated from the MXNet framework [411] used in the original work to a PyTorch [365] implementation.

The training data for the model is generated by the code that generates dataset 4. The training data are input into the model directly with none of the usual pre-processing such as band-pass or whitening, which is consistent with the original work [290]. In fact, the model is equipped with a whitening layer to estimate the power spectrum for each input data. The main modification used in this challenge is to randomly sample 25 templates in the first matched-filtering layer from the same parameter space used in dataset 4 of this challenge. It performs significantly better than the original gridded and fixed template configuration. The subsequent convolution network of the model is constructed using the current excellent lightweight models MobileNetV3 [412] which give state-of-the-art results in major computer vision problems. The submission uses curriculum learning, during which the model is trained with decreasing multiples of signal amplitude. The multiplicative factor is lowered from 50 to 1 until convergence. Multiple models were randomly initialized and trained on a NVIDIA Tesla V100 GPU, from which the best was chosen for this submission.

To search for triggers and evaluate the performance of the model, a sliding window approach is implemented. The evaluation data is divided into overlapping segments corresponding to the input size of the model. Subsequently, all segments are passed through the model resulting in a sequence

[3]The corresponding authors for the MFCNN submission are He Wang, Shichao Wu, Zong-Kuan Guo, Zhoujian Cao, and Zhixiang Ren.

of predictions and a table of SNR peaks from the 25 sorted matched filters. The step size is 1 second and a threshold of 0.5 is set on the network output as in [290]. The "time"-, "var"- and "stat"- dataset of the output file described in subsection 8.2.4 are derived from the table of SNR peaks associated with directly filtering the templates with the data. The GPS time and time variance of each trigger are designated as the median value and the interquartile range of SNR peaks from the nearby segments, respectively. We count the coincident SNR peaks between two detectors to quantify the ranking-statistic. Other experiments are still in progress and are supposed to be published alongside further details in a standalone paper.

The final version of the algorithm submitted by the MFCNN group was provided after the submission deadline had past. A vital flaw in their original contribution was discovered and was allowed to be fixed.

8.3.2 PyCBC

[4] The *PyCBC* submission is based on a standard configuration of the PyCBC-based archival search for compact-binary mergers [15]. The search infrastructure was used, in addition to cWB, for the first detection of gravitational waves, GW159014 [13], in production analyses by multiple groups to produce gravitational-wave catalogs [14, 15] and targeted analyses [358]. A similar low-latency PyCBC-Live analysis is also based around the same toolkit [58, 343]. The analysis uses matched filtering to identify candidate observations in combination with a bank of predetermined waveform templates that correspond to the expected gravitational-wave signals [30]. Matched filtering is known to be the optimal linear filter for stationary, Gaussian noise. To account for the potential non-Gaussian noise transients [371, 413, 414], each candidate and the surrounding data are checked for consistency with the expected signal [189, 209]. In addition, the properties of candidates, such as their time of arrival, amplitude, and phases in each detector are checked for consistency with an astrophysical population [163].

The empirically measured noise distribution and the consistency with the expected gravitational-wave signal are combined to calculate a ranking statistic for each potential candidate [163, 164]; this ranking statistic is used as the "stat" value of dataset output, along with its associate trigger time in "time". The "var" dataset is set to a constant of 0.25 s. Two template banks are used for the submitted results. For dataset 1, a template bank of non-spinning waveform templates, using the IMRPhenomD [129] model, is created using stochastic placement. Datasets 2, 3, and 4 were evaluated

[4] The corresponding author for the PyCBC submission is Alexander H. Nitz.

with a common template bank that includes templates that account for spin which is aligned with the orbital angular momentum. Furthermore, only the dominant mode of the gravitational-wave signal was used and effects such as precession were not accounted for. In both cases, the mass boundaries of the template bank conform to the challenge set parameters.

The final version of the algorithm submitted by the PyCBC group was provided after the submission deadline had past. A vital flaw in their original contribution was discovered and was allowed to be fixed. Furthermore, the PyCBC submission strictly speaking uses a different algorithm for dataset 1 than for all other datasets, as the template banks are not the same. The change in template banks was accepted, as this work does not focus on a runtime analysis.

8.3.3 CNN-Coinc

[5] This submission is based on the works from Gabbard et al. [56] and Schäfer et al. [64]. It utilizes the network architecture presented in [56] with a prepended batch-normalization layer [271]. As such the network processes 8 192 input samples, which corresponds to 4 s at a sampling rate of 2 kHz. The network is trained only once and applied to the data from both detectors individually. Afterwards the outputs are correlated to find coincident events as detailed in [64]. The source code for training the network and applying it to test data of the format used in this challenge is open source and can be found at [415]. The algorithm was designed to enable an easy and efficient estimation of the search background by applying time shifts between the individual detectors data. While this feature cannot be utilized in this challenge, the original paper [64] highlights the advantages of this approach.

The network is trained on parts of the real O3a noise from the Hanford detector as provided in this challenge. Signals are generated using the waveform approximant `IMRPhenomXPHM` [133] from the same parameter distribution used in datasets 3 and 4 in this challenge. Merger times of the signals are varied between 2.9 s to 3.1 s from the start of the input window of the network. The signals are pre-whitened by one of the provided Hanford PSDs used in datasets 2 and 3. Noise samples are non-overlapping parts taken from the real noise data provided by this challenge, where each segment is whitenened by an estimate of the PSD on that segment. The network was trained for 100 epochs using the loss and optimizer settings provided in [64] on a single NVIDIA RTX 2070. The epoch with the greatest binary accuracy on a single training run was chosen for this challenge.

[5] The corresponding author for the CNN-Coinc submission is Marlin B. Schäfer.

During evaluation the network is applied to the challenge-data using a sliding window approach. Each data segment is whitened by an estimate of the PSD of that segment obtained by Welch's method [30, 187]. All data is whitened before the network is applied for computational efficiency. Subsequently, the network is applied to the data via a sliding window with a step size of 204 samples $\approx 0.1\,\mathrm{s}$. Afterwards a threshold of 3.4 is applied on the unbounded Softmax replacement (USR) output, which was introduced in [63]. Coincident events are calculated using the same procedure and parameters as outlined in [64]. The "time"- and "stat"-dataset of the output file described in subsection 8.2.4 list the coincident event times and ranking statistic values, respectively. The time variance of the "var"-dataset is set to a constant value of $0.3\,\mathrm{s}$.

8.3.4 TPI FSU Jena

[6] This submission closely followed the method of [63], which is itself based on [56], with several modifications to adapt to the specifics of the challenge. The core of the algorithm is a convolutional neural network that accepts a 2×2048 input tensor corresponding to 1 second of data from 2 detectors sampled at 2048 Hz. Its architecture is derived from that of [63] and deviates from the original network by a larger size of the individual layers and a doubled number of convolutional layers. These modifications are the result of a hyperparameter variation experiment which found these settings to be optimal. A standalone publication on this submission giving further details on the methodology is in preparation. The final layer of the network is a Softmax layer over two inputs which is used for training and removed using the USR [63] during evaluation.

The network is trained on a dataset constructed by whitening a randomly chosen part of the real noise file and slicing it to produce 1-second noise samples and injecting whitened `IMRPhenomXPHM`-generated BBH waveforms into half the noise samples at SNRs uniformly drawn between 7 and 20. The waveform parameters are drawn from the same distributions as are used in dataset 4 of this challenge. The training dataset consists of 10^6 samples and the validation set of $2 \cdot 10^5$ samples.

During evaluation, each segment in the input file is whitened separately using the estimated PSD and sliced into 1-second segments at 0.1-second spacing. These are fed to the network with the USR applied. First-level triggers are selected by applying a threshold of -8, which are then clustered

[6] The corresponding authors for the TPI FSU Jena submission are Ondřej Zelenka, Bernd Brügmann, and Frank Ohme.

into events. For each event, the "time" and "stat" in the output file are the values of the highest ranking statistic first-level trigger of each cluster, and "var" is set to 0.2 seconds. The algorithm is implemented using the PyTorch framework [365] and spawns child processes to whiten individual segments. The network evaluation is performed by the parent process.

8.3.5 Virgo-AUTh

[7] This submission is based on a simple per-dataset binary classification scheme. Interestingly, it was found that training a model only on dataset 2 or only on dataset 4 can yield impressive results on the other datasets as well. Specifically, training samples from dataset 2 can generalize well to dataset 3 and 1 and not so well on dataset 4, whereas training samples from dataset 4 can generalize well on datasets 1, 2 and 3. Thus, training samples were only generated from dataset 4. An adaptive normalization mechanism [416] was used instead of batch normalization as the first layer, to handle non-stationary timeseries. For the neural network architecture a deep, ResNet-like model [240] with a depth of 54 layers was used.

One week of training data per dataset was generated and the generated injection parameters were used to construct all corresponding waveforms. This amounted to about 600k background segments of duration 1.25 s with a stride of 2 s between, i.e. the next sample starts 0.75 s after the end of the previous one, and about 580k waveforms, of which 300k were used for the injections. For validation, one day of data was used, resulting in about 86k noise segments and 3.2k waveforms. The noise segments and waveform segments are combined online during training, in a static manner, both for the training and for the validation sets. The input samples are whitened before feeding them to the classifier. The PSD is computed online per batch of 4.25 s with a stride of 3.1 s, and each 1.25 s segment inside this duration is whitened with the same PSD. To increase speed, the Welch method for computing the PSD was implemented in PyTorch [365] and whitening is implemented as the first layer of the final detection module. Notably, this approach of computing the PSD for every 4.25 s and whitening each 1.25 s segment in a sliding window manner was found to be faster than using a precomputed PSD for every 1.25 s (about 40% faster for one month of data). After whitening, the first and last 0.125 s (0.25 s total) are removed from each sample.

[7] The corresponding authors for the Virgo-AUTh submission are Paraskevi Nousi, Nikolaos Stergioulas, Panagiotis Iosif, Alexandra E. Koloniari, Anastasios Tefas, and Nikolaos Passalis.

The best results were obtained with a ResNet-52 type network. A Deep Adaptive Input Normalization (DAIN) layer [416] was used as the first layer after whitening, to handle distribution shifts that may be present. The final output is binary, i.e., noise plus waveforms or noise only, and the objective function used was a regularized binary cross entropy. The "var" parameter is set to 0.3 s, as the network predictions are high even when the time of coalescence is slightly outside the preset range. The "stat" parameter is set to the network confidence, i.e., a value in the $[0, 1]$ interval corresponding to the probability that a waveform is present. Finally, 0.125 s are added to the expected time of coalescence to account for the time lost in the whitening process.

A standalone publication on the methods used in this submission is in preparation.

8.3.6 cWB

[8] Coherent WaveBurst (cWB) is a waveform model-agnostic search pipeline for GW signals based on the constrained likelihood method [417–419]. The cWB pipeline has been used for the analysis of scientific data collected by the LIGO-Virgo detectors, targeting detection of signals from generic GW sources, including the compact binary mergers [14].

The cWB algorithm identifies the excess-power events in the time-frequency domain representation of strain data from multiple detectors [38, 165]. For each event, the cWB pipeline reconstructs the GW waveforms and estimates summary statistics which describe generic properties of the events like the coherence across the detector network, signal strength, and the time-frequency structure.

Recently, a boosted decision-tree algorithm, eXtreme-Gradient Boost (XGBoost) [301], was adopted and implemented within the cWB framework to automate the signal-noise classification of the cWB events [300]. Two types of input data are used for the supervised training: signal events (from simulations) and noise events (from background estimations). For each of those, a subset of cWB summary statistics is fed to XGBoost as input features to train a signal-noise model. As in [300], the detection statistic for the machine learning-enhanced cWB algorithm is defined by:

$$\eta_{\mathrm{r}} = \eta_0 \cdot W_{\mathrm{XGB}}, \tag{8.7}$$

where, η_0 is cWBs ranking statistic, and W_{XGB} is the penalty factor calculated by XGBoost ranging between 0 (noise) and 1 (signal).

[8] The corresponding authors for the cWB submission are Francesco Salemi, Gabriele Vedovato, Sergey Klimenko, and Tanmaya Mishra.

This methodology has been recently used in the full reanalysis of publicly available strain data from Advanced LIGO's Hanford and Livingston third observational run [167]: the machine learning-enhanced cWB outperforms the standard human-tuned signal-noise classification used for detection of the compact binary coalescences in the O3 run.

For this study, we chose to use machine learning-enhanced cWB; however, cWB typically rejects weak candidate triggers (i.e., with FAR $\gg 1$ per year) at early production stages. Moreover, the whole workflow is optimized for a trigger production which saturates at FAR ≈ 30 to 50 per month. Therefore, we modified cWB to increase the event production rate by almost 2 orders of magnitude: the result is a cWB with sub-threshold capabilities, able to speed up computation and reduce memory allocations.

While trying to provide the most "generic" result for this study, it was decided to re-use the XGBoost model which was developed for [167]: it should be noted that the model was trained on noise and signal events sets that differ substantially from those adopted for the data sets prepared for MLGWSC-1. The noise backgrounds for dataset 3 and dataset 4 appear to be significantly quieter than O3. Also, the signals were drawn from a spin-aligned stellar-mass BBHs population model with different component mass ranges [420] and with SEOBNRv4 waveforms [118]. The above-mentioned detection statistic, η_{r}, is used as the "stat" value of dataset output, along with its associated trigger peak-time in "time". The "var" dataset is set to a constant of 0.25 s.

The results from the cWB group were provided after the submission deadline had passed. The group assured that no tuning to the challenge set was performed.

8.4 Data release

We provide all source code as well as the evaluation results for all submissions at [407]. The repository contains all code accessible to the participants of the challenge, which most importantly includes a script to generate data and one to produce the sensitivity statistics we provide in section 8.5. The repository also contains code for basic visualization as part of the "contributions" folder. Adaption of these scripts were used to create the graphics in this paper. The challenge used the code of release 1.3 of the repository.

Alongside the code provided by the challenge organizers we publish the source code that was used to run the contributions for the groups PyCBC, CNN-Coinc, TPI FSU Jena, and Virgo-AUTh in the "submissions" folder of [407]. The submission code for the MFCNN group can be found at [410].

All analysis output files for all submissions created by our analysis are also publicly available and are stored in the "results" folder in [407]. For each group we make available the raw output on the foreground and the background for all 4 datasets. Additionally, all timing information is available. The exception is the cWB group, for which only results on datasets 3 and 4 are available.

The repository [407] also contains plots used in this paper for all groups, including versions we have not shown here. They can be found in the "plots" folder.

8.5 Results and discussion

In this section we provide the results of our evaluation process described in section 8.2 for all 6 submissions. We calculate and discuss sensitive distances, found-missed plots, and runtimes to provide a quantitative comparison between the different submissions. We specifically focus on the difference between machine learning and traditional algorithms and reason where the core differences in performance arise.

The four datasets we use in this study were chosen to answer different questions and serve different purposes. Dataset 1 was meant as an entry point to the challenge that represents a largely solved case [55, 56, 64]. We expected most submissions to perform very similarly on this dataset. The second dataset was intended as the first major step in difficulty. We expected its main challenge to be the longer duration of the injected signals, as many machine learning algorithms target shorter durations and struggle with large analysis segments [61, 213]. Dataset 3 includes precession and higher order mode effects in the injected signals that traditional, modeled searches are not optimized for[9] [40, 41, 44]. We wanted to test if machine learning algorithms could get closer in performance, or even outperform, the traditional searches in these regions. The intention of dataset 4 was to provide a challenge that is representative of a realistic search on real detector data and a limited parameter space. The data contains non-Gaussian noise artifacts, that can mimic GW signals [125, 421–423], which are strongly suppressed by sophisticated algorithms in traditional searches [59, 60, 125]. Most machine learning algorithms that target real noise do not make use of such noise-mitigation strategies and instead rely solely on the ability of the machine learning algorithm to identify noise artifacts. This approach was reported to be effective for higher FARs in the past [57, 288, 289, 359] and we were,

[9] A full search of the entire O3 data that includes higher order modes has been performed in [43].

therefore, expecting relatively minor difference between dataset 3 and dataset 4. Furthermore, most traditional algorithms use matched filtering, which is only proven to be optimal for signal recovery when the noise is stationary and Gaussian. Since neither of the two assumptions are true for real detector data, we were also interested to test if machine learning algorithms can perform better than these searches by learning a better noise representation.

8.5.1 Sensitivities

In this subsection we discuss the sensitive distances of the different submissions, which are a measure for how many sources can be detected at any given level of certainty, i.e. at a particular FAR. They are the core metric to determine the quality of any search. We focus on the low FAR region and truncate the plot at a FAR of 10^3 per month. We chose this cutoff for two reasons. First, to function as a standalone search, algorithms may only report events with low FARs. State of the art pipelines send out alerts only when the FAR is smaller than $\mathcal{O}(1)$ per month [58]. Second, for high FARs a non-negligible number of detections originate from false associations. This means that a large number of triggers that originate from random noise coincidences are close enough to an injection to be counted as true positives.

Since all machine learning submissions chose to optimize for dataset 4, results on all prior sets also test the capability of generalizing to different signal (sub)populations. Dataset 3 is a special case, as it uses the same distribution to draw the parameters of the injected signals as dataset 4. It, therefore, differs only in the noise contents and is a good test of the performance difference of different algorithms between simulated and real noise.

The results of this challenge are summarized in Figure 8.2 and Table 8.4. The four individual panels of Figure 8.2 show the sensitive distances as a function of the FAR for all submissions. The panels contain the results for dataset 1 to 4 from left to right and top to bottom. The errors on the sensitive distances estimated from the variance of the Monte-Carlo integration are smaller than 80 Mpc for all curves. In Table 8.4 we give the numeric values for the sensitive distances at three selected FAR values of 1, 10, and 100 per month for all submissions and datasets. We also provide information on the wall-clock time used to evaluate the different sets. Due to time constraints, we only show sensitivity curves for dataset 3 and 4 for the submission from the cWB group. We also note that PyCBC used a different template bank to analyze dataset 1 than for the remaining three datasets.

We find that the machine learning algorithms from the TPI FSU Jena group presented in subsection 8.3.4 and the Virgo-AUTh group presented in

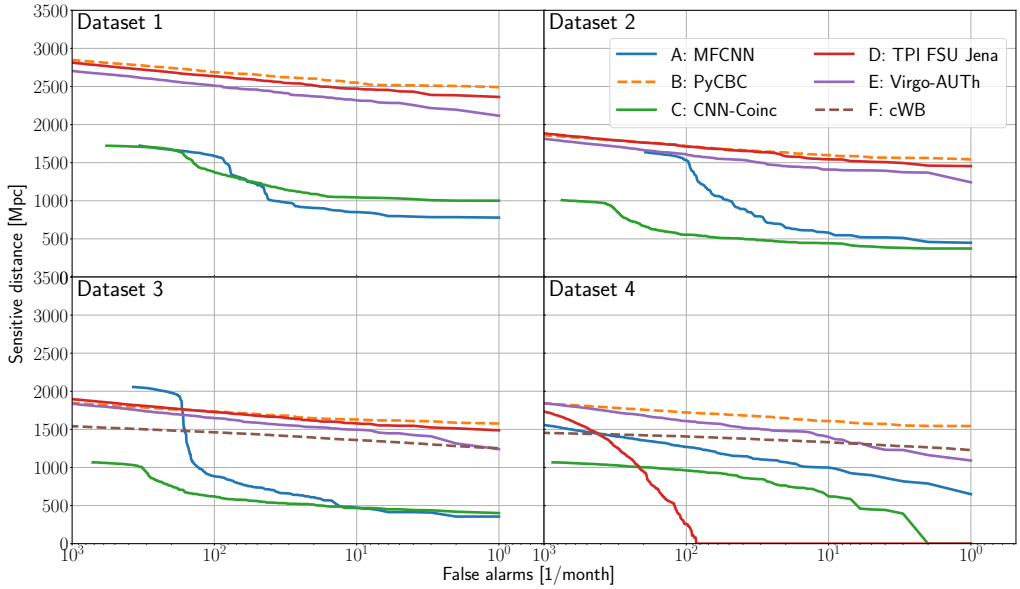


Figure 8.2: The sensitive distances of all submissions and all four datasets as functions of the FAR. Submissions that made use of a machine learning algorithm at their core are shown with solid lines, others with dashed lines. The FAR was calculated on a background set that does not contain any injections.

Dataset	Group	Sensitivity [Mpc] at FAR = x per month			Runtime [s]		
		$x = 100$	$x = 10$	$x = 1$	foreground	background	average
1	A: MFCNN	1586.90	852.18	779.21	42842	43820	43331
	B: PyCBC	2686.55	2550.57	2491.53	5406*	5092*	5249*
	C: CNN-Coinc	1372.30	1045.34	1001.55	14003	12996	13500
	D: TPI FSU Jena	2634.80	2472.31	2362.51	3758	3530	3644
	E: Virgo-AUTh	2511.95	2317.53	2116.38	5490	5520	5505
	F: cWB	N/A	N/A	N/A	N/A	N/A	N/A
2	A: MFCNN	1531.55	581.93	448.59	43431	40634	42033
	B: PyCBC	1719.98	1599.79	1543.79	157865**	161662**	159763**
	C: CNN-Coinc	554.32	443.58	373.64	14731	14976	14853
	D: TPI FSU Jena	1712.13	1544.28	1455.09	3920	3805	3862
	E: Virgo-AUTh	1608.97	1409.95	1242.37	5596	5748	5672
	F: cWB	N/A	N/A	N/A	N/A	N/A	N/A
3	A: MFCNN	885.46	472.96	355.83	37822	41251	39536
	B: PyCBC	1734.43	1630.35	1577.18	149025**	146683**	147854**
	C: CNN-Coinc	619.94	467.81	401.58	13628	14345	13986
	D: TPI FSU Jena	1727.73	1577.77	1487.67	3862	3621	3742
	E: Virgo-AUTh	1646.53	1494.98	1240.68	5450	5453	5451
	F: cWB	1461.56	1359.78	1252.09	5247***	N/A	N/A
4	A: MFCNN	1269.03	999.29	649.81	41942	46702	44322
	B: PyCBC	1722.43	1609.62	1544.33	162699**	163504**	163102**
	C: CNN-Coinc	960.70	620.85	0.00	12489	12431	12460
	D: TPI FSU Jena	257.87	0.00	0.00	3540	3487	3514
	E: Virgo-AUTh	1608.71	1400.30	1091.77	5462	5571	5516
	F: cWB	1406.88	1331.90	1229.14	4996***	N/A	N/A

Table 8.4: A summary of the analysis results for all submissions and all datasets. The columns labeled "Sensitivity" give the values for the sensitive distance at the three FARs 10^2 per month, 10^1 per month, and 10^0 per month rounded to the second decimal place. The values lie on the lines in Figure 8.2. The columns labeled "Runtime" list the time for evaluation of the foreground and background set in seconds, respectively. The runtime column labeled "average" lists the mean time obtained from evaluating the foreground and background data. Entries labeled "N/A" are not available, because they were not measured. The PyCBC times labeled with * are only approximations. The analysis did not run on the challenge hardware but made use of a compute cluster. Shown times are the result of scaling the computational costs to 16 CPU cores. The PyCBC times labeled with ** are approximations obtained in the same manner as the approximations labeled with *, but make use of a larger filter bank. The times of the cWB group marked with *** are approximations derived from dividing the CPU core-seconds reported by the search by 16 to normalize it to the challenge hardware.

subsection 8.3.5 are very close in sensitivity for datasets 1, 2, and 3. The submission from the TPI FSU Jena group reaches a slightly higher sensitive distance at all FARs for all of these three datasets. However, the Virgo-AUTh submission retains $\geq 90\%$ of the sensitive distance achieved by the TPI FSU Jena submissions for FARs ≥ 2 per month. At lower FARs the gap widens but the individual sensitivities carry large uncertainties due to low number statistics. For higher FARs this gap narrows to a separation of roughly 4% at a FAR of 1000 per month. We suspect that the difference between the two approaches is on the order that could be explained by different initializations of the training procedure.

On dataset 4 the submission from the Virgo-AUTh group manages to maintain a stable sensitivity for the full range of tested FARs. The submission from TPI FSU Jena, on the other hand, is dominated by background triggers and seemingly struggles to adjust to the non-Gaussian noise characteristics. For high FARs the sensitivity is on a similar scale as the submission from the Virgo-AUTh group and as was observed on previous datasets, backing up the hypothesis that rejecting background triggers is the main problem. This is surprising, as both algorithms were optimized on dataset 4 but performed similarly only on datasets 1 to 3. One reason for this result may be the neural network architectures used by the different groups. The Virgo-AUTh group uses a very deep ResNet that may be better suited to represent non-Gaussian noise artifacts. The architecture from the TPI FSU Jena group is a more straightforward convolutional architecture that may be limited in its ability to learn appropriate parameters.

The algorithms from the MFCNN group presented in subsection 8.3.1 and the CNN-Coinc group presented in subsection 8.3.3 also show similarities in sensitivity. Both are significantly less sensitive than the leading machine learning submission on all datasets. For datasets 1, 2, and 3, the MFCNN contribution achieves 32.5%, 30.8%, and 23.5% of the sensitive distances of the leading machine learning contribution, respectively. The CNN-Coinc submission reaches 42%, 25.5%, and 27% of the sensitivity of the leading machine learning contribution at the point of farthest separation. For dataset 4 the submission from the MFCNN and CNN-Coinc groups do comparatively better. They retain $\geq 68\%$ and $\geq 50\%$ of the sensitive distance of the leading machine learning submission down to a FAR of 10 per month, respectively. At a FAR of 1 per month the CNN-Coinc submission does not detect any signals, whereas the MFCNN still retains 60% of the sensitivity of the leading machine learning contribution.

On the first three datasets one can observe a steep gradient of the sensitivity curves at varying FARs for the MFCNN and CNN-Coinc submissions. At even higher FARs the curves level off again and return to a similar slope

observed at low FARs. The sudden increase leads to the MFCNN submission being more sensitive than the modeled PyCBC search by up to 15% on dataset 3 for FARs > 200 per month. This behavior is not present in any of the other submissions and we were not able to find a clear explanation. However, we observe that both algorithms have different trigger rates on the foreground and background set. If the background is estimated from the foreground data only, the sensitivity of both algorithms drops sharply. All other algorithms are robust to this change. We show these sensitivity curves in Figure 8.3. However, it was communicated to the groups before submission that sensitivities would be calculated using both the foreground and background data. For this reason, we do not discuss Figure 8.3 any further but would like to encourage possible future mock data challenges to drop the background set.

For all datasets we compare the leading machine learning submission to the submission from PyCBC presented in subsection 8.3.2. We also compare it to the submission from cWB presented in subsection 8.3.6 for datasets 3 and 4. These two are traditional, state-of-the-art search algorithms that have already been used successfully in past observation runs [13, 14, 424].

For dataset 1 we find that the machine learning search is able to achieve between 94% and 99% of the sensitivity obtained with PyCBC. These results are remarkably close and improve significantly on the findings from [64], which targeted a very similar dataset. However, the gap between the machine learning detection algorithm and the PyCBC search widens for lower FARs. Therefore, we expect that the PyCBC contribution will be able to attribute a substantially higher significance to many events. This is amplified by the ability of PyCBC to trivially increase the amount of data that can be used for background estimation by introducing time-slides between detectors [60, 64].

For dataset 2 the leading machine learning contribution gets even closer to the traditional algorithm from the PyCBC group. At low FARs ≤ 20 per month it retains $\geq 93.5\%$ of the sensitivity achieved by the PyCBC submission. For high FARs ≥ 200 per month it even manages to outperform the PyCBC submission and is up to 1.5% more sensitive.

From dataset 2 to dataset 3 all submissions experience a slight increase of the measured sensitive distance. This may be surprising at first but can be explained by the distribution of the effective spin. For dataset 3 the spin orientations are distributed isotropically, which causes the average effective spin to be smaller than in dataset 2. This leads to few systems with large effective spin. The PyCBC search gains up to 3% in sensitivity at low FARs, although it loses about 1% in sensitivity at high FARs. A similar change can be observed in the submission from TPI FSU Jena. Since both the

leading machine learning contribution and the PyCBC search gain similar amounts of sensitivity from dataset 2 to dataset 3 the comparison between the two does not change substantially. The submission from the TPI FSU Jena group is now up to 2.5% more sensitive at high FARs and still about 6% less sensitive at low FARs. The Virgo-AUTh, the MFCNN, and the CNN-Coinc submissions increase their sensitive distance by a larger fraction, suggesting that they benefit more from the signal population being closer to the distribution of signals in their training set. Dataset 3 is also the first dataset for which results from the cWB search are available. We find that cWB retains $\geq 80\%$ of the sensitive distance obtained by PyCBC over all tested FARs. Subsequently the leading machine learning submission achieves a sensitive distance greater by 15% to 23% over the range of tested FARs.

For dataset 4 the leading machine learning contribution now comes from the Virgo-AUTh group. Compared to PyCBC their algorithm retains $\geq 87\%$ of the sensitivity down to a FAR of 10 per month. For smaller FARs the sensitivity gap widens quickly. At a FAR of 1 per month the machine learning search achieves 70% of the sensitivity of PyCBC. The cWB submission evolves similarly to PyCBC and retains $\geq 79\%$ of the sensitive distance. At high FARs the leading machine learning search manages a sensitive distance up to 27% larger than that of cWB. For low FARs the sensitive distance falls off quicker than that of cWB. At a FAR of 1 per month the cWB search is 12.5% more sensitive than the Virgo-AUTh submission. For lower FARs we expect this difference to become larger, as the production level search algorithms are tuned for lower FARs than tested in this work. In comparison to the sensitivity difference on dataset 3 the machine learning submission from Virgo-AUTh does not retain as much sensitivity on real noise as the PyCBC or cWB submissions.

The results on dataset 1 demonstrate that machine learning detection algorithms are already capable of rivaling traditional search algorithms for simulated data at FARs ≥ 1 per month. A previous study [64] had identified the capability of machine learning searches to build an internal representation of the signal morphology as the main problem to achieve comparable sensitivities to traditional algorithms. Such a signal representation would allow the algorithms to compare detections in multiple detectors and require them to be consistent. The two leading machine learning algorithms in this challenge seem to have overcome this limitation, at least for high FAR detections.

For dataset 2 we expected machine learning searches to decline in sensitivity more strongly than traditional searches. This expectation was provoked by the short duration of data that is processed by most machine learning searches at each step. As the signals injected into dataset 2 are of longer duration than those used in dataset 1, the machine learning algorithms in-

herently lose some amount of sensitivity due to considering only small parts of the signal. We estimate this loss to account for at most a 1% difference in sensitivity. However, we observe the opposite effect for the two leading machine learning algorithms, which get even closer in sensitivity to the PyCBC submission compared to dataset 1. This may be caused by the distribution of signals in the training data used for the machine learning algorithms. Since both algorithms optimized for dataset 4, most signals in the training data will have non-zero spin. Therefore, the challenge set for dataset 2 is closer in nature to the training data, which may have introduced a bias that leads to higher sensitivities for spinning systems or in other words a slightly reduced sensitivity to non-spinning systems.

Dataset 3 was intended to test if machine learning searches are capable of outperforming traditional algorithms for precessing systems and signals carrying higher order mode information. We do not find substantial evidence in support of this hypothesis from the sensitivity curves. However, the challenge set 3 contains only very few signals with strong evidence for precession and higher order modes, as most signals are still relatively short. The impact on the overall sensitivity from these signals is, therefore, minor. Surprisingly, the leading machine learning search is still on par with PyCBC and manages to be significantly more sensitive even at the lowest tested FARs than the unmodeled cWB search. It must be noted that the cWB submission was not optimized for the parameter space used in this challenge. We, thus, expect this gap to narrow if more effort were to be used to tune the cWB pipeline.

The change in the relative difference in sensitivity between the PyCBC submission and the leading machine learning contribution, as well as the change in difference to the cWB submission, from dataset 3 to dataset 4 suggests that many machine learning algorithms currently used by the community are not yet capable of treating real noise as well as sophisticated traditional algorithms. We suspect that one major factor may be non-Gaussian noise artifacts that are misclassified as signals by machine learning algorithms, while the traditional searches excise them from the data or reject them on other bases. Another reason may be the non-stationary character of the noise that may lead to different sensitivities at different times. However, this would have also been a factor in dataset 3, where the PSDs used to simulate the noise change over the duration of the challenge set. However, since the leading machine learning search does retain sensitivity at all FARs it must have learned to reject most non-Gaussian noise artifacts, which is in line with expectations from studies carried out at higher FARs [288, 289, 291, 359].

8.5.2 Found and missed injections

We generate found-missed plots for all submissions and show a few selected ones. The ones not included in this paper can be found in the associated data release [407]. These plots highlight specific areas in parameter space where the machine learning searches are already competitive and those where more work is required. Specifically, we provide plots for chirp mass M_c versus decisive effective chirp-distance $D_{c,\text{eff}}$, τ_0 versus mass ratio q, and the effective precession spin χ_p [180] versus inclination with respect to the line of sight θ_{jn}. To first order τ_0 is the time to merger from the lower frequency cutoff of the waveform [68, 425]. The decisive effective chirp-distance is a measure for how strong the signal can be observed in the detector that has the worse sensitivity due to source location and orientation. The effective chirp-distance is the chirp-distance at which a source with the same intrinsic parameters and sky location but an optimal orientation would have been observed from at the same amplitude as the injected signal. The decisive effective chirp-distance is then the larger of the two effective chirp-distances from the two detectors. Therefore, the $M_c/D_{c,\text{eff}}$ plot informs about the ability to detect signals as a function of the SNR in the detector that is less sensitive to the signal. We also include information on the ranking statistic like quantity returned for each detected event, to highlight how strongly it is correlated to the SNR. The τ_0 versus q plot highlights how well long and short duration signals are recovered. It also gives information on the mass ratio, which is an important parameter for the strength of precession effects. The main plot used to determine the impact precession effects and higher order modes have on the detectability of signals is the χ_p versus θ_{jn} plot.

In Figure 8.4 we show the found injections from dataset 1 in the M_c-$D_{c,\text{eff}}$-plane for the PyCBC and TPI FSU Jena submissions, respectively. Both plots clearly show that closer injections are generally attributed a higher confidence to be a real signal. This indicates that the ranking statistic like quantities for both algorithms are actually correlated with the signal strength. Similar correlations can be observed for all submissions. Furthermore, all signals with large $D_{c,\text{eff}}$ are missed by both searches, showing that sensitivity estimates are not limited by the injected population. From Figure 8.4 we find that the chirp mass distribution from the TPI FSU Jena submission favors chirp masses in the region $M_c \in [20, 35]$, which is not true for the PyCBC submission. We attribute this bias to the training set, which contained signals drawn from the distributions used for dataset 3 and 4. The probability distribution of the chirp mass for these sets is shaped such that about 51% of signals are being drawn from the mass range $20\,\mathrm{M}_\odot$ to $35\,\mathrm{M}_\odot$. A similar bias is not so evident for the other machine learning submissions

but may be masked by other effects. The PyCBC submission uses a uniform prior on the chirp mass and thus avoids this bias.

In Figure 8.5 we compare the found injections from dataset 2 in the τ_0-q-plane for the PyCBC and TPI FSU Jena submissions. The plots show that the two searches are competitive in the comparable mass region and identify similar signals. The main difference between the two searches can be observed in the τ_0 distribution of found signals. Most of the signals with large values for τ_0, i.e. long duration signals, are missed by the TPI FSU Jena submission. These crucially include many signals that the PyCBC submission identifies with relatively high confidence. Therefore, the short duration of the input windows used by the TPI FSU Jena submission still seem to be a limiting factor for the sensitivity. This limitation will likely be more severe if longer duration signals from sources like BNS or NSBH systems were considered.

In Figure 8.6 we compare the θ_{jn} and χ_p values of the injections from dataset 3 that are found by one algorithm but missed by the other. The two algorithms come from the PyCBC group and the TPI FSU Jena group. If either algorithm adapted better to signals with strong precession or higher order modes content, we would expect to see a clustering from that search in the scatter plot. However, we do not observe this clustering, which backs up our observation from the sensitivity curves that the machine learning algorithm from the TPI FSU Jena group has not learned a better representation of precessing systems or signals with higher order mode content than the modeled PyCBC search, which only includes non-precessing signals in its template bank. However, the amount of impact precession or higher order modes have on the detectability of short duration signals used in this study are small. A real test of this hypothesis would require the analysis of long duration signals.

8.5.3 Runtimes

All runtimes in this section are given in terms of wall-clock times obtained on equivalent hardware, which is listed in Table 8.2. The runtimes are largely independent of the dataset for all submissions. We, therefore, discuss them only in summary. An overview of the times can be found in Table 8.4. They were measured by applying each algorithm to the foreground and background of each challenge set. We report the time between the algorithm call and it returning. To avoid bottlenecks, all files were transferred to the local storage of the individual compute nodes before calling the algorithm. The output was also written to said local storage and transferred back only after the algorithm returned. It should be noted that the runtimes are heavily dependent on the amount of optimization of the algorithms. The main objective for this

challenge was the sensitivity and not the runtime.

The PyCBC and cWB submissions are exceptions as their runtimes were not measured on the same hardware. Instead they were run on compute clusters making heavy use of parallelized work over multiple CPUs. The times reported here are approximations by normalizing the compute time to 16 CPU cores available in the compute nodes used for this challenge. Furthermore, for the evaluation of dataset 1 PyCBC used a different template bank than those for dataset 2 to 4 was used. This bank was substantially smaller, resulting in faster evaluation. cWB times were reported to us only on the foreground data in CPU core seconds.

We find that of the machine learning algorithms the submission from the TPI FSU Jena group is the fastest, evaluating an entire month of archival data in about 1 h. It utilizes a single GPU when evaluating the network. The second fastest algorithm is the submission from the Virgo-AUTh group. It evaluates a month of data in 1.5 h on a single GPU and is thus about 50% slower than the fastest algorithm. Notably, the two fastest algorithms are also the two most sensitive machine learning searches. The algorithm from the CNN-Coinc group requires almost 4 h on a single GPU to evaluate the same amount of data but is significantly less sensitive. However, none of these algorithms are limited by the GPU performance. The differences in execution time can be mainly attributed to the difference in optimization of the pre-processing steps. The submission from the MFCNN group on the other hand does not apply any pre-processing directly. They instead use a neural network to carry out this computation. They operate on all 8 available GPUs and manage to evaluate the month of data in ≈ 11.5 h.

For dataset 1 the PyCBC submission has a runtime comparable to that of the submission from the Virgo-AUTh group. On all other datasets it requires roughly 43 h to evaluate the month of data. The large difference in runtime between the datasets is caused by the smaller template bank that is used only for dataset 1. Contrary to the machine learning algorithms, the PyCBC submission did not utilize GPUs and ran on CPUs only. However, PyCBC is a production level search pipeline and as such has been optimized to run on CPUs. It is not limited by the pre-processing but rather by the matched filter operation. It should be noted that PyCBC is still the most sensitive search presented here and gains in computational efficiency could be obtained by reducing the number of templates. This would effectively trade off search sensitivity for lower computational cost.

The PyCBC submission is implemented on the CPU as a GPU implementation is inherently more difficult to optimize. GPUs, on the other hand, are usually far more efficient from a cost to performance and energy to performance standpoint [426]. One advantage of machine learning algorithms

is that they make use of well optimized libraries such as PyTorch [365] or TensorFlow [364] that utilize GPUs for their computations. This makes the implementation of search algorithms on GPUs relatively straightforward and allows researchers to focus on optimizing the sensitivity of their algorithm rather than having to spend time on optimizing the algorithmic implementation.

The runtimes in this challenge are measured under the assumption that the lowest required FAR is 1 per month. In a real search lower FARs are beneficial especially for rare signals. Therefore, most traditional searches are tuned to be most sensitive at FARs well below the level tested in this challenge. PyCBC for instance can extend its background by introducing time-slides [60], thereby potentially lowering the FARs of detected events. This process is a trivial operation that requires a fraction of the computational cost of the actual filtering stage. If machine learning algorithms are not specifically designed to allow for a similar approach, lowering the FARs of detections requires multiple complete re-evaluations of the time-shifted data. This would in turn lead to a linear increase in the computational cost, i.e. lowering the potential FAR of an event by an order of magnitude would lead to an order of magnitude increase in the computational cost.

8.6 Conclusions

In this paper we have presented the results of the first Machine Learning Gravitational Wave Search Mock Data Challenge (MLGWSC-1). The study compiled curves showing the sensitive distances from 4 different machine learning submissions and compared them to 2 state-of-the-art traditional search algorithms; the modeled PyCBC [60] pipeline and the unmodeled coherent wave burst search [38, 165]. We established a common dataset and means for evaluation. We hope that other researchers will continue to make use of the resources presented in this work to allow for quantitative comparisons between different machine learning approaches and to traditional filtering techniques. As research continues and machine learning search algorithms become more sensitive, we want to motivate other groups to host new challenges, focusing on other parts of parameter space or targeting different observing strategies.

The key observations of this challenge are:

1. Machine learning search algorithms are competitive in sensitivity compared to state-of-the-art searches on simulated data and the limited parameter space explored in this challenge.

2. Most of the tested machine learning algorithms struggle to effectively handle real noise, which is contaminated with non-Gaussian noise artifacts.

3. Traditional search algorithms are capable of detecting signals at lower FARs, thus making detections more confident.

4. The tested machine learning searches struggle to identify long duration signals.

Therefore, the main challenges for current machine learning searches are the operation on real noise, the confidence in detections due to comparatively high FARs, and the detection of long duration signals. The last of those three is a major hurdle to confidently detect signals from BNS and NSBH systems. Improvements in any of these fields would be beneficial. Specifically, we identify the following key research areas:

1. Improve the ability to compare signal parameters, or representations thereof, between detectors to check for consistency and reject noise artifacts.

2. Improve the ability to calculate large amounts of background, for instance by designing algorithms that can trivially evaluate time-slides of the input data.

3. Increase the duration of data that is processed by machine learning algorithms to enable the detection of long duration signals.

This challenge shows the potential of machine learning algorithms to act as GW detection pipelines. We have shown that these algorithms are competitive in a realistic scenario to state-of-the-art searches today. They operate at low computational cost and allow for a trivial implementation of the algorithms on highly efficient GPUs, rather than relying on CPUs. We believe that this work justifies more research on this topic, especially in areas where machine learning may have a tangible impact on the rapid identification of GWs.

However, we do acknowledge that the research carried out here operates on a limited parameter space. Moreover, the targeted parameter space is not the computationally expensive part of the search space of traditional searches. About 1% of the total size of the template bank used in [15] is dedicated to the area this study searches. To have the greatest impact on real searches machine learning algorithms need to be extended to target either the low mass region, where signals are long and the computational cost of

matched filtering rises rapidly, or the high mass region where signals and noise artifacts are difficult to distinguish.

We also want to mention that we did not receive a submission utilizing one of the most promising neural network architectures for GW detection of the recent past. A WaveNet based architecture, that uses dilated convolutions, has been reported to do well for this kind of task [57, 289, 427]. We also did not receive submission based on many other neural network architectures that have been used in the past, such as autoencoders [307, 308, 378, 405], inception networks [61, 303], or two dimensional convolutions that analyze time-frequency decompositions [291]. We hope that some of these approaches will be adopted to the requirements of this challenge and evaluated on the datasets presented here, to allow for a quantitative comparison.

Future mock data challenges could target longer duration signals, concentrating on BNS and NSBH systems. These are potentially EM-bright and would, therefore, be of particular interest. Furthermore, these signals stem from regions of parameter space where traditional searches are computationally expensive to run. For even longer signals, sub-solar mass black holes could be targeted. Existing modeled searches in those regions make use of several million templates and are computationally limited [173]. Another avenue may be very massive systems, which can be difficult to distinguish from noise artifacts. Finally, we recommend that future mock data challenges drop the notion of a foreground and background set and only provide data files containing injections. This would eliminate further sources of error and be more true to a realistic application, where no true GW-free background exists.

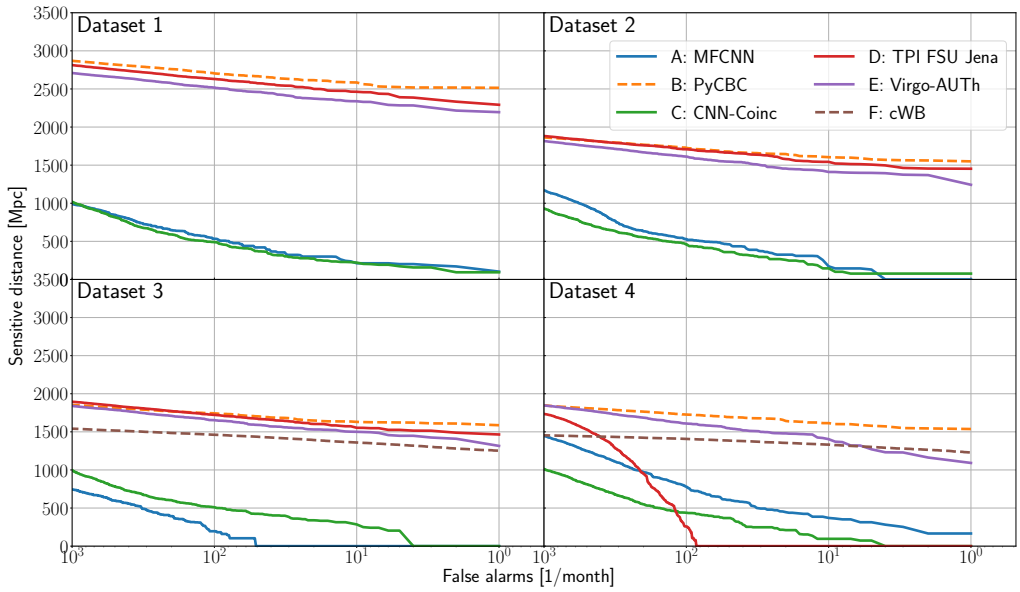


Figure 8.3: The sensitive distances of all submissions and all four datasets as functions of the FAR. The sensitive distances are calculated using only the data from the foreground file. The FAR is determined from the false positives on that data. Submissions that made use of a machine learning algorithm at their core are shown with solid lines, others with dashed lines. This figure differs from Figure 8.2 as the algorithms from MFCNN and CNN-Coinc behave differently on the foreground and the background.

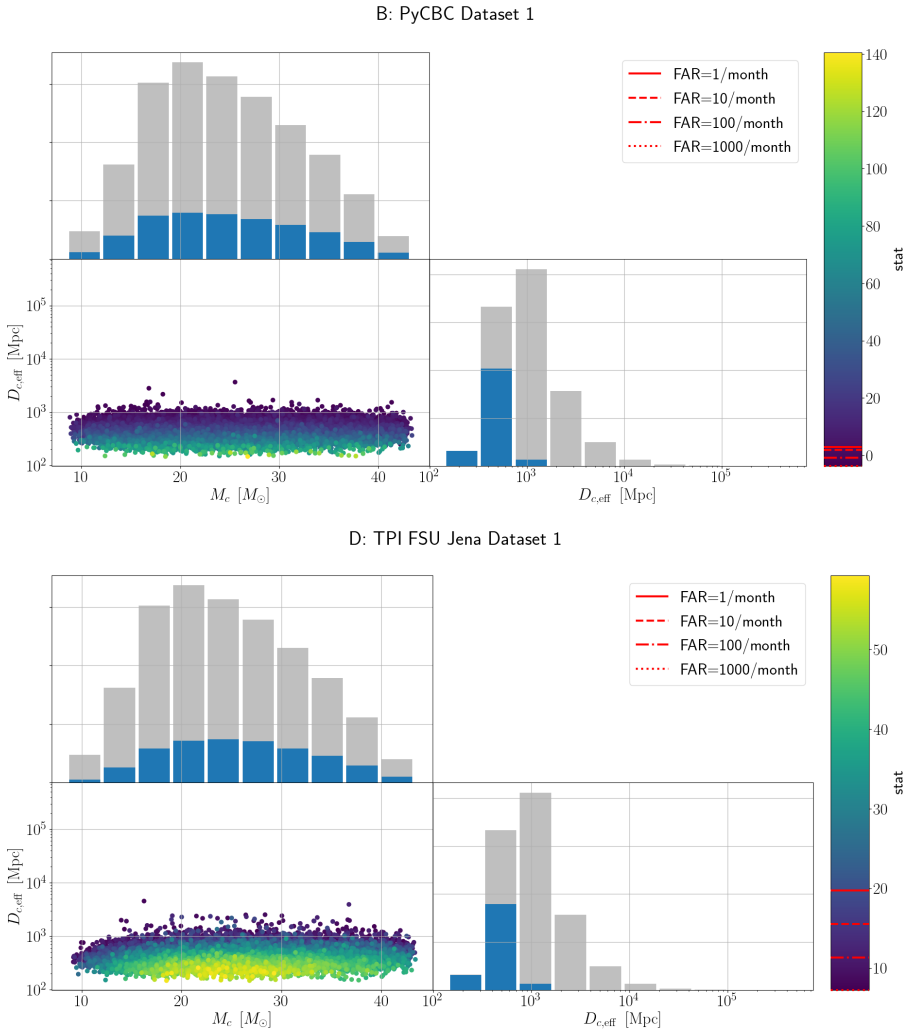


Figure 8.4: The injections from dataset 1 identified by the PyCBC and TPI FSU Jena submissions with a FAR $\leq 10^3$ per month in the chirp mass M_c versus decisive effective chirp-distance $D_{c,\mathrm{eff}}$ plane. The blue bars in the histograms show the one dimensional marginal distributions of the found injections. The gray bars show the distribution of injected signals, including those missed by the search. The color shows the "stat" value attributed to the injection by the algorithm. The red lines in the colorbar highlight the thresholds on the "stat" to achieve different FARs.

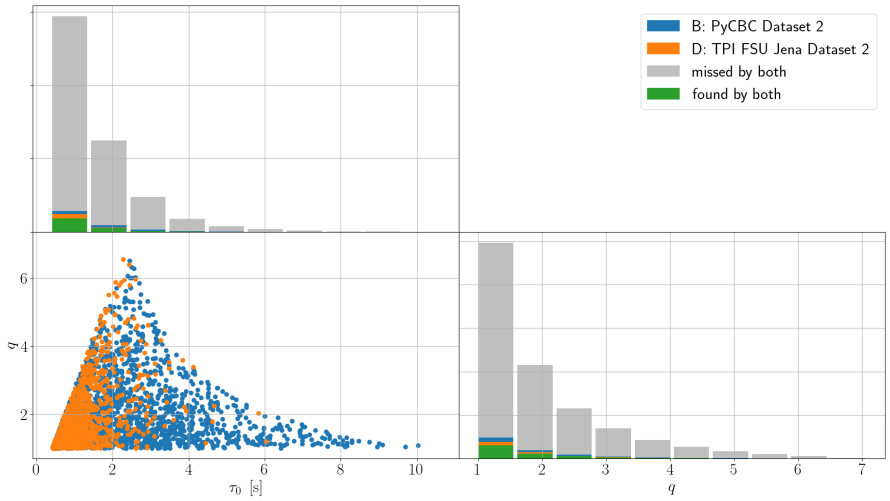


Figure 8.5: The injections from dataset 2 identified by the PyCBC and TPI FSU Jena submissions with a FAR $\leq 10^3$ per month in the signal duration τ_0 versus mass ratio q plane. The scatter plot shows injections that are found only by one of the two algorithms. Injections that are missed or found by both are only shown in the 1D marginal distributions.

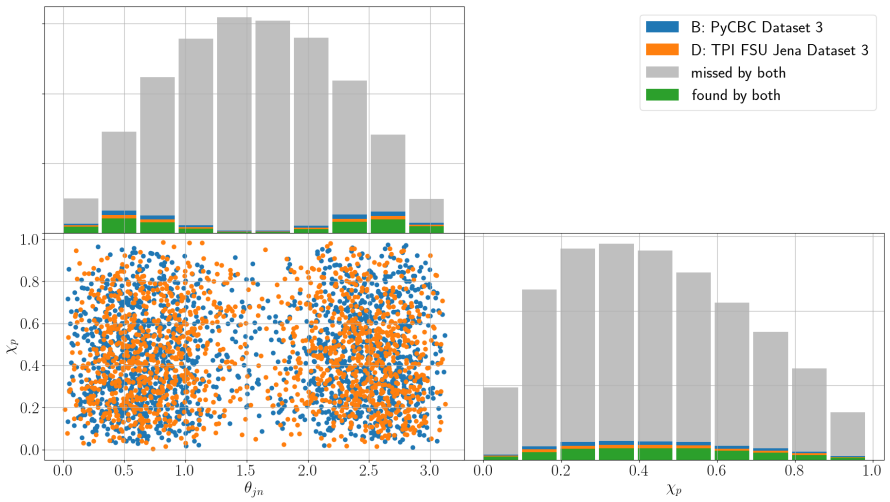


Figure 8.6: The injections from dataset 3 identified by the PyCBC and TPI FSU Jena submissions with a FAR $\leq 10^3$ per month in the inclination to spin axis θ_{jn} to χ_p plane. The scatter plot shows injections that are found only by one of the two algorithms. Injections that are missed or found by both are only shown in the 1D marginal distributions.

Chapter 9

Internship at Bosch

Contents

9.1 Basics of Object Detection 196

9.2 Self-Supervised Pre-Training 200

9.3 Research at Bosch 203

9.3.1 Introduction 204

9.3.2 Methods 206

9.3.3 Experiments 209

9.3.4 Conclusions 213

This chapter is somewhat independent from the rest of this thesis and summarizes a voluntary internship I did at Bosch Hildesheim, Germany. However, many advancements of ML algorithms are developed in the industry and gaining an insight into their development environment is of great benefit to academic research. At Bosch I was part of the corporate research department focusing on unsupervised pre-training for computer vision (CV) algorithms. Specifically, I was tasked with developing and testing self-supervised learning algorithms for object detection. Due to the independence of this chapter from the others, I will give a brief introduction to deep learning object detection algorithms and self-supervised pre-training algorithms before describing my research at Bosch.

During my 4 month internship I managed to test a framework originally presented in [428] on data used internally at Bosch. Additionally, I experimented with multiple new and original ideas targeted at improving the original framework for the application at Bosch. My time at Bosch resulted in a total of 4 invention reports, one of which will be discussed below. The others are either too far from the core topics of this thesis or are not published yet.

9.1 Basics of Object Detection

Traditionally, CV challenges like the ILSVRC have mainly focused on image classification. The task in image classification is to provide a single label to an image. The famous ImageNet dataset contains more than 21 000 classes and 14 million images [429]. In object detection the task is to locate and annotate objects in an image. So rather than providing one global label to the entire image, the algorithm needs to specify one or multiple regions and apply a label to every one of them.

Once a good image classifier is available, getting an accurate object detector is in principal not too difficult. Theoretically, the classifier can be applied to each possible sub-region of the image to get a label. When the image classifier is setup such that it has a garbage class that is used for all images that do not contain any object, it would be able to detect and correctly annotate objects in an image. However, this direct approach is prohibitively computationally expensive, due to the factorial growth of sub-regions with the dimensions of the input image. To reduce that cost at the price of accuracy, one can instead use pre-determined regions in the image and classify those. An alternative is to use a fast algorithm to filter out the majority of uninteresting regions with low accuracy and only classify the remaining ones.

Before an overview of the evolution of deep learning object detection algorithms is given, a few core concepts have to be defined. First, the output of an object detector in the context discussed here is a rectangular bounding box of an object, whose sides are parallel to the edges of the image. Each bounding box has to be classified into one of $N + 1$ categories, where N is the number of object classes. The last class is the background class, that represents non-detections. To quantify how well a predicted bounding box aligns with a labeled box, a quantity known as the *intersection over union* (IoU) is commonly used. It is defined as the area of the intersection of two rectangles divided by the area of their union. To quantify the performance of an object detection algorithm, the most common metric is the *mean average precision* (mAP). The *precision* is the number of true positive detections divided by the number of total detections. It measures how likely a detection is to be a true positive and depends on the requirement for how well a bounding box has to be aligned with the ground truth to be considered a true positive, i.e. it depends on IoU requirements. The IoU also influences how likely a ground truth is to be recovered by the algorithm. This is measured by the *recall*, which is the number of true positives divided by the number of ground truths. Since both the precision as well as the recall are a function of the IoU, they can be plotted against one another. The average precision,

confusingly, is defined as the area under the precision-recall curve [430]. The mAP averages the average precision of all classes.

It is common to divide the architecture of an object detection NN into two parts. Usually, a network pre-trained on image classification data is used to produce a feature map. This part of the object detection network is known as the *backbone*. To adjust the output of the backbone to object detection and to produce the desired bounding boxes and classifications additional networks are attached to the backbone. These are known as *heads* and often fine tuned on comparatively smaller amounts of object detection data sets.

The first major deep learning based object detection algorithm is known as R-CNN [263] and selects regions of interest using the traditional algorithm "selective search" [431]. Afterwards, it crops the proposed regions from the input image, projects them to a fixed size, and applies the then state-of-the-art AlexNet [49] to create fixed-size feature maps. In a third step, each feature map is evaluated by SVMs to find the class for the proposed region, where one class is a background class. The algorithm is trained, by first training the AlexNet on an image classification task. In a next step, the network is fine tuned on an object detection dataset. For this stage of training, the last layer of the AlexNet is replaced by a $N + 1$ dimensional dense layer, where N is the number of object classes and the additional output is used for background detections. For each image of the training set a selective search is used to propose regions of interest, which are then compared to the ground truth boxes. Proposed regions with an IoU ≥ 0.5 with a ground truth box are defined as positive boxes and the rest are defined as negative boxes, i.e. background. From the proposed regions 32 positive boxes and 96 negative boxes are sampled, warped, and used as training data for fine tuning. In a final step, the last layer of the fine tuned network is discarded and one SVM is trained for each class. The resulting method outperformed previous existing methods by almost 20% in mAP [263].

The follow-up of R-CNN is Fast R-CNN [268]. It still uses the selective search algorithm to generate region proposals. However, the network was upgraded to a VGG16 network [260] that was pre-trained on image classification data. Instead of applying it to different warped sub-regions of the image, the image is now processed only once as a whole. The proposed regions are then used to select parts of the resulting feature map, which are pooled to a fixed size using a process called RoIPooling. It also replaces the SVMs by a fully connected NN, which has two outputs. One is the previous classification into $N + 1$ classes. The other outputs 4 values, which are used to adjust the bounding box proposed by the selective search. Fine tuning is a lot easier than for R-CNN, as the entire network can be trained as a whole using standard deep learning optimizers. It improves over the original

R-CNN by more than 15% in mAP [268].

The final version of R-CNN I want to briefly discuss here is Faster R-CNN [50]. It replaces the selective search region proposal algorithm by a NN, thus making every part of the object detection pipeline a deep learning model. This NN is called the region proposal network (RPN) and introduces the concept of anchors. The RPN processes its input to create a feature map, where each pixel is associated with a region in the input image. The size and shape of the region is the same for every pixel, but the location shifts such that the full image is covered. The regions are called *anchors*. To have anchors of different sizes and aspect ratios, the output of the RPN has multiple channels, where each can be associated with a different anchor size. The association is only learned and does not necessarily have to resemble the receptive fields of the pixels. For each anchor, two output values are produced; one is a binary classification into background/foreground, and the other are four values that adjust the location and size of the anchor. The remaining architecture is equivalent to that of Fast R-CNN [268]. During inference, input images are processed by a CNN that produces a feature map, which is then fed to the RPN. All regions that are classified as foreground by the RPN are cropped from the feature map of the initial CNN using the RoIAlign layer from Fast R-CNN [268] and fed to a classifier. Training is a more complicated five step process. First, the backbone is pre-trained on image classification data. Next, the RPN is trained to find correct region proposals. For this IoU thresholds are chosen to select positive and negative anchor examples. The entire network, i.e. backbone and RPN, is trained using a form of SGD. The resulting RPN is then used to generate region proposals to train a fresh Fast R-CNN, which uses the original pre-trained backbone weights. Once the Fast R-CNN is trained, the backbone is extracted, its weights are frozen, and a new RPN is attached. This RPN is trained again, where only the RPN weights are optimized. Finally, the weights of the RPN are also frozen, and the detection heads of the Fast R-CNN trained before are fine tuned using the new RPN weights. It improves the mAP over Fast R-CNN by about 2% but more importantly increases evaluation speed. Where previous versions of R-CNN used multiple seconds to evaluate a single image, mainly limited by the selective search, Faster R-CNN allows the evaluation of ≈ 5 images per second.

The different R-CNN variants are known as two stage detectors, as they propose regions of interest in a first step and classify each region in a second step. While this is a very accurate method, it is computationally expensive as each region of interest has to be processed on its own. To speed up object detection, single stage algorithms have been developed. Instead of only classifying regions of interest, they classify all of a set of pre-defined

regions simultaneously.

One of the first major single stage object detectors that allowed to do object detection in real time and with accuracy comparable to Fast R-CNN was named You Only Look Once (YOLO) [264]. The network divides the input images into $S \times S$ cells. For each cell the network predicts B bounding boxes and for all bounding boxes a confidence score is predicted. The confidence score gives an estimate of how likely it is that the box contains an object and how accurate the box is. Each cell is also associated with a probability score for all object classes, conditional on the predicted probability that the cell contains an object. The network architecture is also built up of a backbone with additional layers attached to handle object detection. The backbone is pre-trained on image classification data. To fine tune the architecture to object detection, ground truths are assigned to the grid cells which contain the center of the ground truth. The model can then be trained end-to-end, meaning that it is a single training loop. Since YOLO was introduced multiple improvements have been made and at the time of writing seven versions were released, of which only the first three are associated with the original authors [292, 410, 432–435].

Another single stage object detector, which many modern object detectors are based on, is the single shot multibox detector (SSD) presented in [265]. A variant of its architecture known as RetinaNet [272] was used in the experiments presented below. The core idea of SSD is to adjust the architecture of the network to simultaneously process different object scales. The backbone of an object detector usually reduces the size of the feature map and increases the number of channels as the data passes through the network. The SSD architecture attaches heads to intermediate layers of the backbone. The pixels from the output of the heads are then interpreted as belonging to anchor boxes in the original image in the same way Faster R-CNN did. The area covered by the anchor boxes is determined by the layer of the backbone at which the head was attached. Different channels in the output correspond to different aspect ratios of the anchor boxes. The training targets are created from the labeled data by IoU requirements of the ground truth boxes with the anchor boxes. Since the model is a one stage detector, it can be trained end-to-end. When it was released, it improved the mAP over Faster R-CNN by almost 3% while simultaneously being substantially faster to evaluate.

The SSD architecture was improved over the years. The Retinanet [272] used a Resnet50 [240] as backbone and attached a feature pyramid network (FPN) [273]. The FPN takes the different feature map scales used by SSD and introduces additional connections that allow smaller scale features to be informed by larger scale features. For FPNs a naming convention is used

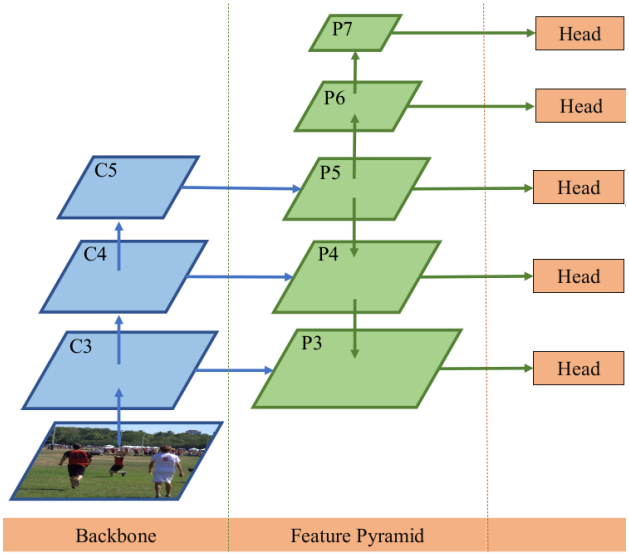


Figure 9.1: High-level overview of the structure of a FPN. It shows a typical FPN with levels P3 to P7, where P6 and P7 are built on top of P5. Arrows indicate connections, that can be of varying kind. Connections from P5 to P4 and P4 to P5 involve upsampling. This figure was taken from figure 2 in [438].

to specify how it is built. The different levels of the backbone from which features are extracted are called C1, C2, and so on. Once they have been processed by the FPN, the outputs are called P1, P2, and so on, depending on which C-level they represent. Additionally, some FPNs have a larger number of P-levels than C-levels. If that is the case, one can either build them on the output of the C-levels or the P-levels. See Figure 9.1 for a visualization. Another popular detector is called EfficientDet [436], which uses an EfficientNet [437] backbone and introduces a bi-directional FPN allowing for information to flow from small scales to large scales.

9.2 Self-Supervised Pre-Training

In this section I will introduce the concept of self-supervised learning (SSL), focusing on a framework known as *contrastive* learning. The goal of SSL is to extract general information from raw data without labels. In the context of object detection, this data are images without any labeled objects. The hope is that the network will learn a general representation of images, which can then be fine tuned on comparatively few labeled training examples. This has the advantage of being computationally less expensive than full supervised

learning, as plenty of raw data exists but labeling them is costly. Contrastive learning uses positive and negative pairs, where in the context of CV positive pairs are usually different augmentations of the same image and negative pairs are augmentations of different images. The loss in this framework then encourages positive pairs to be similar and negative pairs to be dissimilar. Below I will give a brief overview of a few recent works in this field.

A framework known as SimCLR (simple framework for contrastive learnong of visual representations) was introduced in [439]. It consists of two networks f and g that operate on images x. The network f is the core network that after fine tuning is supposed to be an image classifier, network g translates the output of f into a latent representation z. For each input two different augmentations $\tilde{x}$ and $\tilde{x}'$ are constructed and the latent representations z and z' are generated. For a batch of size N we define $\hat{z}_{2i} = z_i$ and $\hat{z}_{2i+1} = z'$. The networks are then trained as one using the loss function

$$L = \frac{1}{2N} \sum_{k=1}^{N} [l(2k, 2k+1) + l(2k+1, 2k)] \tag{9.1}$$

where,

$$l(i,j) = -\log \left[\frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} (1-\delta_{ki}) \exp(s_{i,k}/\tau)} \right] \tag{9.2}$$

and

$$s_{i,j} = \frac{\hat{z}_i \cdot \hat{z}_j}{||z_i|| ||z_j||}. \tag{9.3}$$

The numerator in (9.2) represents the positive pairs, whereas the denominator represents the negative pairs. The loss is minimized, when the cosine similarity (9.3) for the positive pairs is minimized and for the negative pairs is maximized. The authors of [439] find that strong augmentations of the images improve their framework substantially and that large batch sizes (≥ 256) are useful. After fine tuning the network, they achieve performance almost equal to a full supervised training on ImageNet data. In a follow-up paper the authors improve on the results by utilizing very large networks [440] and noticing that the larger the pre-trained model the less impactful the number of labeled images for fine tuning.

The authors of [441] improve on the performance of SimCLR by introducing two branches of the same architecture. They call their approach Bootstrap Your Own Latent (BYOL) and its main advantage is that the loss does not contain any negative examples. This greatly reduces the computational cost during training, as fewer computations are required per batch

sample. Instead of feeding two augmentations of the image to the same network, they introduce a second network, which is identical in architecture the first, to process the second augmentation. This setup can be understood as a teacher-student setup. Only the student is optimized directly through gradient descent, the teacher is updated by an exponential moving average of the student parameters. This procedure is also known as applying a stop-gradient to the teacher. Finally, a third network ϕ is introduced that is hypothesized to project the output of the student to the output of the teacher. The teacher and student produce two outputs z and z' for every input image and its different augmentations x and x', i.e. $z = g_{\mathrm{T}}(f_{\mathrm{T}}(x))$, $z' = \phi_{\mathrm{S}}(g_{\mathrm{S}}(f_{\mathrm{S}}(x')))$, where the subscripts T and S refer to the teacher and student, respectively. The networks f and g serve the same roles as for SimCLR. The loss is given by a MSE

$$L = ||g_{\mathrm{T}}(f_{\mathrm{T}}(x)) - \phi_{\mathrm{S}}(g_{\mathrm{S}}(f_{\mathrm{S}}(x')))||^2 + ||g_{\mathrm{T}}(f_{\mathrm{T}}(x')) - \phi_{\mathrm{S}}(g_{\mathrm{S}}(f_{\mathrm{S}}(x)))||^2. \quad (9.4)$$

In [442] the authors present the SimSiam framework, which combines the advantages of SimCLR with those of BYOL. Their approach trains on positive pairs only and uses a stop gradient as BYOL, but does not require an exponential moving average for the teacher. They also remove the network g from both approaches and are left with the network f and the projector ϕ. This greatly simplifies the overall setup for SSL training and as a consequence greatly reduces the required batch sizes. While their final performance after fine tuning is weaker than that of BYOL, they manage to use less resources, which makes SimSiam viable even on modest hardware. As loss they use a negative cosine similarity.

The final SSL framework I want to introduce is Selective object Contrastive learning (SoCo) proposed in [428]. Previous SSL frameworks were used primarily for image classification tasks and only adapted to object detection by utilizing the pre-trained backbones. SoCo is a SSL framework specifically designed for object detection, allowing to pre-train the backbone and heads of a Mask-RCNN [269]. Mask-RCNN is a special Faster-RCNN that creates a pixel mask for different objects, instead of just generating bounding boxes. The proposed method is guided by the BYOL paper in the sense that it uses the three networks f_{S}, g_{S}, and ϕ_{S} for the student branch and the two networks f_{T} and g_{T} for the teacher, as well as using a stop gradient for the teacher and an exponential moving average to update the teacher parameters. What is called "student" here is named "online network" and what is called "teacher" is named "target network" in [428]. The network f is a Resnet50 [240] with a FPN attached. The FPN is defined as in [272] with levels P2 to P5. To adapt the BYOL framework to object detection,

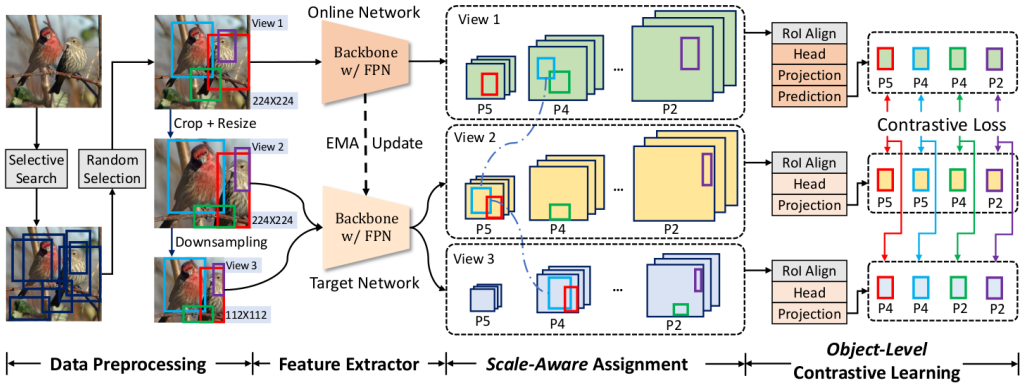


Figure 9.2: An overview of the SoCo framework. What is called "Online Network" in the figure was described as student network in this chapter and what is called "Target Network" in the figure is the teacher network of this chapter. Different augmentations are used for view 1, 2, and 3. Different sized bounding boxes from the selective search are assigned different levels in the FPN. This figure was taken from [428] where it is figure 1.

SoCo produces regions of interest using the selective search algorithm [431]. Based on the size and location of the region proposal a corresponding area in one of the feature maps is selected and turned into a fixed size output by a layer named RoIAlign, which was introduced in [269]. It is a special form of RoIPooling [268] that interpolates its values to better align the input and the output. The output of the RoIAlign is then processed by the head, before being fed into the networks g and ϕ. To induce scale invariance of the object detector, the augmentation of one image is required to create a crop of the input image and resize it to the original size. Also, a third augmentation is processed, which is a downsized version of the augmented image that was cropped and resized. The loss is the sum of the cosine similarities of the output from the first augmentation with the cropped and resized augmentation and the first augmentation with the downsized version. See Figure 9.2 for a visualization of the framework.

9.3 Research at Bosch

At Bosch I was tasked with evaluating the usefulness of the approach called Selective object Contrastive learning (SoCo) presented in [428] to their existing object detection networks. The main challenges lay in two distinct points. First, the paper was mainly targeting two stage detectors, where the first stage samples possible regions of interest and the second stage classifies

these regions. For the application that my work was targeting, anchor based single stage detectors are the prevalent approach, resulting in only portions of the architecture being able to be pre-trained using [428]. Second, the code that was published [443] alongside the paper [428] needed to be validated and adapted for the networks, as well as the data used in the project.

Below I also present an adaption of the framework given in [428] that I proposed to pre-train most of the single stage object detector architecture and eliminate the need to run a computationally expensive selective search [431] before training. At the core it tries to encourage the network to create similar representations for two different augmentations of the same image. Only regions in the resulting feature maps that are present in both augmentations are compared. I evaluate the performance of a network pre-trained with this adaption, which I call SSDoCo (Single Shot Detector object Contrastive learning), against the same network pre-trained with the SoCo framework.

9.3.1 Introduction

Training deep learning object detection algorithms is a difficult task. Creating classified bounding boxes as labels for the training data is far more time consuming, and therefore expensive, than creating labels that classify the entire image into a predetermined number of categories. For this reason, the interest in making un-labeled data usable for training purposes has picked up a lot of interest in the recent past [439, 441, 442]. One of the most successful approaches so far has been a concept known as contrastive learning, which was introduced in 9.2. Most of these algorithms focus on the task of image classification, where they have shown to improve performance even over full supervised training [442]. Although these algorithms are focused on image classification, they can still have a beneficial effect on object detection, as most object detection networks consist of a general backbone that aims to create a feature representation and an object detection specific part, like a feature pyramid network (FPN), that tries to extract information at different scales. Afterward, classification and bounding box regression heads are attached to get the final output.

The backbone has traditionally been taken from state-of-the-art image classification algorithms [439, 441, 442] and was always trained on image classification tasks. It is, therefore, reasonable to believe that SSL pre-training algorithms devised for image classification may improve object detection algorithms, simply by improving the classification performance of the backbone. On the other hand, object detection datasets are often different from image classification datasets. In image classification one commonly has images showing only a single object. For object detection training sets, these kinds

of images are rare, and most images show multiple objects. Furthermore, additional parts of the object detection architecture, such as FPNs or the detection heads, cannot be initialized from pure backbone pre-training and have to be trained from scratch on limited labeled data. To this end, it may be beneficial to develop pre-training algorithms that are tailored toward object detection, for instance by using contrastive loss functions that encourage similarity only in regions where the same objects are shown.

One of the first works presenting a SSL framework that focuses on object detection is the SoCo approach given in [428]. Their goal was to pre-train the full architecture of a Mask-RCNN [269] that learned object level representations rather than global representations of the image. This means that rather than learning to produce similar feature maps of two different augmentations for the entire image, it focusses on producing similar representations only in regions where objects are likely to be present.

To select regions in which objects are likely to be present, SoCo preprocesses the entire training set using a selective search [431], which is a generic algorithm that proposes regions of interest. This stage is computationally expensive but has to be executed only once per dataset. Whenever an image is used for training, a subset of the proposed regions is randomly selected.

In a next step, the two different augmentations of the input image are given to the two branches of their network. Both branches have the same architecture and mainly use a ResNet50 backbone [240] with a FPN [273]. For each selected and proposed region, the feature map that best matches the scale of the bounding box is selected and the part that corresponds to the feature map is cropped out. It is then passed to a RoIAlign layer which was introduced in[269] to create a fixed size feature map. This can subsequently be processed by a detection head. The output of this detection head is further processed by additional layers aimed to project the output of the two branches into the same space. Finally, the output of one branch is processed by a final prediction module, that tries to translate the output of one branch to the output of the other branch. Afterward, the outputs of the two branches are compared by a cosine-similarity loss [428].

The SoCo framework allows to pre-train most of the Mask-RCNN architecture; backbone, FPN, and classification heads. In fact, it can be easily adopted to pre-train most two stage detectors. However, adopting it to one stage detectors is not trivial. One stage detectors do a classification and bounding box regression for a fixed number of pre-determined locations in an image. Since all positions must be classified, selecting only a few of them for comparison has the potential to severely diminish the final performance.

To adapt the SoCo framework to single stage detectors we propose SSDoCo (Single Shot Detector object Contrastive learning). Instead of com-

paring only select regions, SSDoCo compares compatible regions. One of the core principles of SoCo is that one branch receives a cropped part of the original image as input to induce positional and scale invariance. SSDoCo uses the same concept but changes how the loss is constructed. It sums the cosine similarity loss of all compatible feature maps by cropping and interpolating the feature maps at different levels of the feature pyramid. This change allows us to effectively pre-train the entire single shot object detector, aside from the final layers.

9.3.2 Methods

We use a Retinanet [272] with a ResNet50 [240] as backbone for our single stage object detector. The feature pyramid contains levels P3 to P7, where P6 is built from C5. For pre-training we use a single head consisting of 4 stacked convolutional layers with a kernel size of 3, 256 channels, and ReLU activations. This architecture outputs 5 distinct feature maps of different scales. For fine-tuning both the classification head and the regression head are initialized with the weights from the single head during pre-training.

The SSDoCo framework follows the teacher-student model, where only the student is optimized directly using a gradient based optimizer and the teacher is updated through an exponential moving average of the student network parameters. Note that what we call the student is the "online" network in [428] and what we call teacher is there known as the "target" network.

Both branches have multiple components. The student is composed of the model that should be trained, an alignment stage described below, a projector network, and a predictor network. The teacher is built the same way but does not contain a predictor network. Both the model as well as the projector of the teacher are updated by an exponential moving average of the student model and neither receive gradient updates. See Figure 9.3 for a visualization of the framework.

The two branches receive different augmentations of the same image. We follow [428] for the choice of augmentations, but do not construct a "View 3". The exact transformations are listed in Table 9.1. The teacher receives a random sub-view of the input to the student, the area of which is randomly selected between 50% and 100%. The sub-view is subsequently resized to the original image size using bilinear interpolation. To enforce symmetry, we swap the inputs of the student and teacher after the forward pass and compute the loss as a sum of both results.

For the loss we use a variant of cosine similarity, which we shift and scale

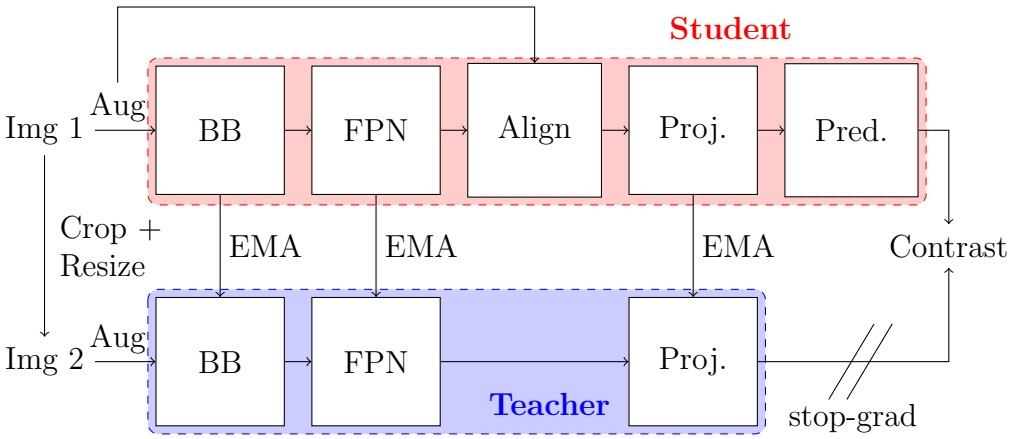


Figure 9.3: An overview of the SSDoCo framework. "Img 1" is an input image and "Img 2" is constructed from it by cropping and resizing to the original resolution. Each image subsequently gets its own augmentation "Aug", before being passed through the backbone "BB" and the FPN. The "Align" step selects and interpolates the parts of the feature maps that are visible in "Img 2". The networks "Proj." and "Pred." are the projection and prediction networks. Finally, the contrastive loss is computed for both branches and the gradient is propagated back through the student. The parameters of the teacher are updated as an exponential moving average "EMA" of the student parameters.

#	Name	Description	Student		Teacher	
			Params	Prob	Params	Prob
1	Resize	Resize smaller side to 224 pixels	–	1	–	1
2	Crop	Random crop to 224x224 pixels	–	1	–	1
3	Random Crop + Resize	Random square crop and resize to 224x224 pixels	–	0	scale from 0.5 to 1	1
4	Horizonal Flip	Flip image horizontally	–	0.5	–	0.5
5	Color Jitter	Apply various color transformations	brightness 0.4 contrast 0.4 saturation 0.2 hue 0.1	0.8	brightness 0.4 contrast 0.4 saturation 0.2 hue 0.1	0.8
6	Grayscale	Convert image to Grayscale	–	0.2	–	0.2
7	Gaussian Blur	Blur the image	sigma from 0.1 to 2 kernel size 15×15	1	sigma from 0.1 to 2 kernel size 15×15	0.2
8	Solarize	Solarize image	Threshold 128/255	0.2	–	0
9	Norm	Normalize the input data	Mean: 0.5 Std: 0.225	1	Mean: 0.5 Std: 0.225	1

Table 9.1: The image augmentations used for the SSDoCo training. The "Prob" columns give the probability of applying the given transformation.

to return values between 0 and 2. The loss is given by

$$L(y_1, y_2) = 2 + \frac{1}{n}\sum_{i=1}^{n} \mathrm{L2}\big(y_1^i, y_2^i\big), \tag{9.5}$$

where y_x is the output of model x, and y_x^i is output i of model x. The function L2 is given by

$$\mathrm{L2}(y_1, y_2) = -2\frac{y_1 \cdot y_2}{||y_1|| \cdot ||y_2||}. \tag{9.6}$$

However, naively comparing the five different feature maps would yield inconsistent results, as the input images are at different scales and show different regions of the same image. For this reason, we compare only those feature maps that are on similar scale and project the feature maps that were derived on the full image onto the sub-region given to the other branch of the Siamese network.

The feature pyramid in the network is built such that the scale of the features double at each level. We, therefore, enumerate the five output feature maps from smallest scale (i.e. largest feature map) to largest scale (i.e. smallest feature map). We then compare feature map $j + i$ of the unscaled image with feature map j of the scaled image, where i is given by

$$i = \lfloor \log_2(s) \rfloor, \tag{9.7}$$

with $1/s$ being the scale chosen in the Random Crop + Resize transformation. The index j ranges from 0 to $n - i$, where n is the number of feature maps, i.e. 5. Our settings are chosen such that $1 < s < 2$ and, therefore, $i = 0$. Consequently, we always compare the same levels of feature maps and only have to take care of the projection.

For the projection we interpret each pixel of the feature map as if it is associated to an anchor box in the input image. We then transform the anchor box-coordinates in the scaled branch to their (sub-pixel) position in the un-scaled image. This process takes re-scaling, cropping boundaries, and horizontal flips into account. We then interpolate the feature map of the unscaled branch at those transformed coordinates, to obtain a comparable grid. We use bilinear interpolation. Afterward the feature maps are flattened, and the cosine similarity loss described above is applied.

9.3.3 Experiments

All pre-training uses a set of 1.6 million diverse unlabeled images taken from Flickr [444]. The data set was created internally to test the influence of large

data sets which show widely differing scenes that are not specific to the final domain the network should operate on.

Once the pre-training is finished, all networks are fine tuned on the same set of labeled data. This set is also sampled from Flickr and contains 5 500 images. For each epoch, 5 000 batches are randomly sampled from the set and the network is trained for 40 epochs with a batch size of 8 or 80 epochs with a batch size of 4, depending on memory limitations of the hardware.

To compare the performance, we report the log average miss rate (LAMR) averaged over all six classes. We use the Caltech protocol given in [445] to determine true and false positives. The miss rate is averaged over five different false positive rates, where the false positive rate is given in terms of false positives per image. The miss rate is the number of false negatives divided by the number of ground truth boxes. Averaging over multiple false positive rates gives a more stable estimate of the detector performance. Lower values of the log average miss rate indicate better performance.

As baseline we fine tuned the network from random initialization. We found that the minimum LAMR after 40 epochs was $\approx 53\%$. To find any improvement, this baseline has to be beaten. Furthermore, we used ImageNet pre-trained weights as initialization for the backbone only. The FPN and heads were initialized randomly. In this setup we found a minimal LAMR of $\approx 33\%$.

SoCo

We adjusted the code published alongside the SoCo paper [443] to use the Retinanet discussed above. However, we removed the heads from the Retinanet and only replaced the backbone and FPN, as both components differ to the original implementation only in minor details. The greatest difference is the inclusion of the levels P6 and P7, which were dropped from the original work due to the size of the resulting feature maps. Our network uses the outputs $\{\mathrm{P3}, \mathrm{P4}, \mathrm{P5}, \mathrm{P6}, \mathrm{P7}\}$. Consequently, we also had to adjust the awareness scales. We define object proposals of area within the range of $\{0 - 48^2, 49^2 - 96^2, 97^2 - 192^2, 193^2 - 208^2, 209^2 - 224^2\}$ pixels to $\{\mathrm{P3}, \mathrm{P4}, \mathrm{P5}, \mathrm{P6}, \mathrm{P7}\}$, respectively. Due to hardware limitations we also had to lower the batch size to 64 samples.

Before training, we applied the selective search to the entire training set. To reduce runtime, we resized all images such that the smaller side has a size of 224 pixels, before applying selective search. Afterward, the boxes were scaled to match the original image size.

We pre-trained the architecture for 10 epochs with a warmup of 1 epoch using the same LARS optimizer [446] and settings as given in the SoCo

repository [443]. We found that the outputs of the network grow during pre-training. This forced us to not use mixed precision training during fine tuning and, therefore, reduce the batch size to 4. This fine tuning seemed to be unstable and only converging occasionally. When the fine tuning converged, performance was on the order of a randomly initialized network.

To make sure that the architecture was not a problem, we changed the backbone and FPN to match those cited in the paper [428]. We also changed the association of the area of proposals with FPN levels back to the original implementation. However, pre-training and fine tuning showed the same problem as before, with the network converging only sometimes. Converged networks could not beat a random initialization during fine tuning.

When we fine tune the network, we only initialize those parts of the network that have had pre-training. The remaining weights and biases are initialized randomly. This means that the P6 and P7 levels are initialized randomly in our second experiment during fine tuning. In both cases, we tested initializing both the backbone and the FPN as well as only the backbone. Neither option yielded any improvements over the other.

SSDoCo

To evaluate the SSDoCo approach, we tested multiple different ideas aimed at improving performance. All tests use the same backbone and FPN of the Retinanet, but the projector and predictor were varied. We also experimented with different optimizers and regularization techniques.

Initially, following the architecture of SimSiam [442], the projectors were removed and the predictors for the different feature maps were 1×1 convolutions that only altered the channel dimension. We trained the network with the Adam [253] optimizer. The transformation aligning the outputs was placed after the output of the predictor. These experiments showed that the output was continually growing, the longer we trained. This caused fine tuning to be unstable and diverge. Consequently, we introduced L2 regularization [447], to reduce the numerical values of the weights and, thereby, the output. We also switched to the LARS [446] optimizer, following the recommendations of the SoCo paper.

After these alterations, the network output stayed on average below 1. However, the loss quickly fell during the first few batches and continued to grow afterward. This problem was rectified by lowering the initial learning rate by an order of magnitude. The resulting algorithm trained smoothly but checking the outputs for different inputs revealed that the output was constant for different inputs.

We initially tried to force the network to avoid collapsing to constant

outputs by introducing a new regularization term to the loss. This term penalizes low weight variance and is given by

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(s\mathrm{Var}\left[\theta_i\right] + \epsilon \right)^{-1}, \tag{9.8}$$

where s is a scale factor, ϵ is a small constant, and θ_i are the parameters of layer i. The function Var calculates the variance of the parameters. After some experiments we found that this regularization seems to solve the problem. However, fine tuning resulted in a network that is on par with a randomly initialized network.

To align the SSDoCo training more strongly with both the SoCo training and previous SSL pre-training experience, we introduced a projector using global average pooling, followed by two dense layers. The predictor was altered to consist of two dense layers, where the first is a bottleneck, i.e. having fewer neurons than its input and the subsequent output. We also moved the transformation aligning the outputs of the teacher and the student to take place between the Retinanet and the projector. So, the student is composed of a Retinanet, followed by the output alignment, the projector, and finally the predictor. The teacher network has the same structure but drops the predictor. Both the Retinanet and the projector of the teacher are updated through an exponential moving average of the student.

Aligning the setup with previous experience, the output dropped to expected small scales. However, fine tuning still did not produce a network that could beat a random initialization.

We also tried fine tuning all architectures by initializing only the backbone and FPN of the network with the pre-trained weights. This had proven helpful in other scenarios. However, this could not improve fine tuning results either.

We had planned to test the influence of the batch size, the size of the bottleneck layer, and the influence of different data set sizes on the final performance of the algorithm. Due to a missing baseline, we could not perform these experiments.

Pre-training was carried out on four NVIDIA A6000 GPUs for a total batch size of 128. Training for 10 epochs on 1.6 million images required about 4 days. We used a learning rate of 0.1 and a momentum of 0.9 in the LARS optimizer. We used the same weight decay implementation and settings as given in [443].

9.3.4 Conclusions

We have tested the SoCo framework presented in [428] and a custom extension of it – named SSDoCo – on a diverse dataset. Our goal was to pre-train a Retinanet for complex object detection tasks on a large unlabeled dataset and fine tune it on application specific data. The frameworks allowed to pre-train both the backbone as well as the FPN, while the SSDoCo approach was envisioned to also allow pre-training of the detection- and classification-heads.

We found that neither of the two contrastive pre-training methods managed to produce network parameters that could beat a random initialization during a fixed fine tuning. Especially, the findings of SoCo could not be reproduced on our data. We hypothesize that this could be due to two reasons. First, our datasets contain many objects, compared to MS Coco [448], which was used in SoCo. This could be challenging for a framework that tries to align object views. Second, limited hardware resources restricted the batch size we could use during pre-training to 64 samples. The lowest batch size results are reported on in [428] is 512; a 8 times increase over our setup.

The SSDoCo setup allowed us to train with a batch size of 128. The training produced much more stable results, that managed to converge every time during fine tuning. However, the performance after fine tuning could not outperform random initialization. Given that the SoCo framework, which the SSDoCo approach is loosely based on, did not produce positive results on our datasets, we believe it to be unlikely that SSDoCo will yield tangible performance benefits. However, other SSL methods have already proven to be successful, which justifies further research in this area.

Chapter 10

Conclusions and Outlook

This thesis has analyzed the applicability of ML methods to the problem of detecting GW signals from CBC sources in comparatively strong noise. It studied both BNS and BBH signals and introduced novel methods to expand the capabilities of existing ML algorithms. A core contribution of this thesis is an objective comparison of sensitivities between ML methods and existing search pipelines. Several studies highlighted the importance of calculating sensitivities normalized by the population of GW sources derived from long-duration continuous data to make representative claims about GW detection capability.

Each chapter of this thesis has solved select problems of ML based searches for CBC signals to advance the field from a proof of principle level to actual applications. Chapter 4 introduced a novel deep learning based search for BNS signals that outperforms previous ML methods at low FARs. While it is not yet competitive with matched filter searches, it provides a method to reduce the number of data samples that need to be processed. This reduction in data size for BNS signals is crucial for NNs, as they struggle with large inputs. Chapter 6 introduced a simple modification to the NNs used in early deep learning BBH search algorithms. This extension enabled the algorithm to operate at FARs $\mathcal{O}(1)$ per month and be competitive to a matched filter baseline. Chapter 7 applied a coincidence strategy to deep learning searches optimized on a single detector and showed that the background can be trivially extended to test the algorithm down to FARs $\mathcal{O}(10^{-2})$ per year. Chapter 8 presents the results of a global mock data challenge organized by the author of this thesis. The mock data challenge makes the tools and experiences gained from chapters 4 to 7 publicly available by publishing open source software and reference data sets that can be used to evaluate any search on the provided parameter space. The evaluation results can be compared to existing literature, as the study includes reference sensitivities from

current production search pipelines.

The summarized takeaways from the various analyses presented in this thesis are as follows. NNs are already competitive in sensitivity to matched filtering when BBH signals are considered. This is highlighted by the results found in chapter 8. There, the best ML search retains $\geq 70\%$ of the sensitivity achieved by the state-of-the-art matched filter based PyCBC search pipeline and becomes comparable at FARs > 100 per month, even in real noise. However, while the most sensitive algorithm requires less time to process the data than PyCBC, the probed parameter region is still efficiently searched by existing methods. In regions of parameter space, where matched filtering becomes computationally more expensive, deep learning is also limited in its capability. Especially long duration BNS and NSBH signals are still challenging to NN searches. This is in line with chapter 4 where we observed that, while we improve the state-of-the-art for deep learning BNS searches, our algorithm is handily outperformed by matched filtering. We also found evidence of the same problem in chapter 8, where even high SNR long duration signals were missed by all deep learning searches. Furthermore, the results presented in chapter 7 suggest that a major problem for ML algorithms in the future may be the unavailability of signal consistency tests to reject many noise artifacts. Typically deep learning searches are used as binary classifiers that decide only between the presence and absence of a signal in the parameter region they were trained on. The lack of crude source parameter estimates in this setting complicates a coincidence analysis, which greatly decreases the FARs that can be trivially tested.

Challenges for ML based GW search algorithms are plentiful, but should not be discouraging. Great progress has been made in the last few years in terms of capability and clarity of results. Their enormous potential, the rapid development of deep learning methods that can be imported from other research areas, and the easy utilization of graphics cards are all good reasons to continue trying to overcome existing problems. Furthermore, some ML algorithms are already invaluable tools in some areas of GW astronomy. These include glitch classification [190] and improvements to ranking statistics to reject non-Gaussian noise artifacts [300].

The identified weaknesses of current approaches also provide a clear path for future research. It is necessary to develop ML algorithms capable of reliably and rapidly detecting weak, long duration GWs from sources such as BNS or NSBH mergers. These sources are of special interest as they potentially emit an EM counterpart, which can be used to extract further information from the system and constrain stellar models. Their rapid detection can increase the EM observation time, hopefully to a point where the prompt emission can be observed. It is also desirable to develop ML GW

searches that operate at low FARs on the order of one per year. A promising route studied in this thesis is the adoption of a coincidence analysis scheme as it is used by state-of-the-art production searches. To achieve this goal and reliably reject non-Gaussian noise artifacts, developing signal consistency checks for deep learning based algorithms is important. Organizing future mock data challenges would be beneficial to benchmark the progress of the field accurately. The mock data challenge discussed in chapter 8 has provided a baseline that can be easily extended to more difficult regions of parameter space.

Beyond the clear path painted by the challenges of existing methods identified in this thesis, there are further interesting avenues to apply ML in GW astronomy. One approach to utilize ML in GW detection that has often been proposed but never actually been explored beyond an initial study [449] is to do a hierarchical search. Fast ML algorithms can be used to flag candidate detections at high FARs, thereby rejecting most of the noise. The candidate detections can then be checked using matched filtering to reduce computational costs while preserving the sensitivity of state-of-the-art analyses. The work in this thesis has already proven that deep learning searches can be more sensitive than matched filtering at very high FARs and may, therefore, be great first-stage filters. Another application not explored in this thesis that has gathered much interest in the recent past is the rapid production of posteriors to determine the parameters of GW sources [312, 315]. Such development could reduce the computational costs by orders of magnitude due to the long runtimes of existing parameter estimation codes and the trivial evaluation of their deep learning counterparts.

Future GW observation runs and detectors are expected to provide an increased rate of detections. The increased number of observations will allow us to make more reliable statements about the population of astrophysical objects and hopefully lead to new and exciting insights into the Universe. At the same time, the large number of expected detections provides a challenge for data analysts that need to process them in a timely manner. This is where ML algorithms may be able to provide solutions previous algorithms cannot. How effective ML algorithms will be applied in broad GW astronomy tasks and how strongly they will be utilized remains to be seen. However, this thesis shows that ML has already passed many of the hurdles needed for an application as a GW search pipeline.

Bibliography

[1] William Herschel. "'The Scientific Papers of Sir William Herschel' at 100". In: *Astronomy & Geophysics* 53.2 (Apr. 2012), pp. 2.13–2.13. ISSN: 1366-8781. DOI: 10.1111/j.1468-4004.2012.53213.x. eprint: https://academic.oup.com/astrogeo/article-pdf/53/2/2.13/19333277/53-2-2.13.pdf. URL: https://doi.org/10.1111/j.1468-4004.2012.53213.x (cit. on p. 1).

[2] A. A. Penzias and R. W. Wilson. "A Measurement of Excess Antenna Temperature at 4080 Mc/s." In: *The Astrophysical Journal* 142 (July 1965), pp. 419–421. DOI: 10.1086/148307 (cit. on p. 1).

[3] Jonathan P. Gardner et al. "The James Webb Space Telescope". In: *Space Sci. Rev.* 123 (2006), p. 485. DOI: 10.1007/s11214-006-8315-7. arXiv: astro-ph/0606175 (cit. on p. 1).

[4] John P. Hughes et al. "Nucleosynthesis and Mixing in Cassiopeia A". In: *The Astrophysical Journal* 528.2 (Jan. 2000), pp. L109–L113. DOI: 10.1086/312438. URL: https://doi.org/10.1086/312438 (cit. on p. 1).

[5] Ray W. Klebesadel, Ian B. Strong, and Roy A. Olson. "Observations of Gamma-Ray Bursts of Cosmic Origin". In: *The Astrophysical Journal Letters* 182 (June 1973), p. L85. DOI: 10.1086/181225 (cit. on p. 1).

[6] J. Aasi et al. "Advanced LIGO". In: *Class. Quant. Grav.* 32 (2015), p. 074001. DOI: 10.1088/0264-9381/32/7/074001. arXiv: 1411.4547 [gr-qc] (cit. on pp. 1, 2, 27, 28, 30, 32, 36, 94, 106, 132, 155, 157).

[7] F. Acernese et al. "Advanced Virgo: a second-generation interferometric gravitational wave detector". In: *Class. Quant. Grav.* 32.2 (2015), p. 024001. DOI: 10.1088/0264-9381/32/2/024001. arXiv: 1408.3978 [gr-qc] (cit. on pp. 1, 27, 28, 32, 94, 132, 155).

[8] T. Akutsu et al. “KAGRA: 2.5 Generation Interferometric Gravitational Wave Detector”. In: *Nature Astron.* 3.1 (2019), pp. 35–40. DOI: 10.1038/s41550-018-0658-y. arXiv: 1811.08079 [gr-qc] (cit. on pp. 1, 2, 27, 28, 29, 35, 94, 132, 155).

[9] Harald Luck et al. “The upgrade of GEO600”. In: *J. Phys. Conf. Ser.* 228 (2010). Ed. by Zsuzsa Marka and Szabolcs Marka, p. 012012. DOI: 10.1088/1742-6596/228/1/012012. arXiv: 1004.0339 [gr-qc] (cit. on pp. 1, 28).

[10] A. Einstein. “Näherungsweise Integration der Feldgleichungen der Gravitation”. In: *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften (Berlin)* (Jan. 1916), pp. 688–696 (cit. on pp. 1, 10).

[11] J. Weber. “Detection and Generation of Gravitational Waves”. In: *Phys. Rev.* 117 (1960), pp. 306–313. DOI: 10.1103/PhysRev.117.306 (cit. on pp. 1, 27).

[12] J. H. Taylor and J. M. Weisberg. “A new test of general relativity: Gravitational radiation and the binary pulsar PS R 1913+16”. In: *Astrophys. J.* 253 (1982), pp. 908–920. DOI: 10.1086/159690 (cit. on pp. 1, 26, 27).

[13] B. P. Abbott et al. “Observation of Gravitational Waves from a Binary Black Hole Merger”. In: *Phys. Rev. Lett.* 116.6 (2016), p. 061102. DOI: 10.1103/PhysRevLett.116.061102. arXiv: 1602.03837 [gr-qc] (cit. on pp. 1, 2, 25, 27, 43, 102, 155, 171, 182).

[14] R. Abbott et al. “GWTC-3: Compact Binary Coalescences Observed by LIGO and Virgo During the Second Part of the Third Observing Run”. In: (Nov. 2021). arXiv: 2111.03606 [gr-qc] (cit. on pp. 1, 10, 26, 27, 28, 31, 36, 41, 42, 76, 123, 132, 155, 159, 171, 175, 182).

[15] Alexander H. Nitz et al. “4-OGC: Catalog of gravitational waves from compact-binary mergers”. In: (Dec. 2021). arXiv: 2112.06878 [astro-ph.HE] (cit. on pp. 1, 10, 27, 28, 42, 75, 132, 155, 157, 159, 160, 171, 189, 271).

[16] B. P. Abbott et al. “Tests of general relativity with GW150914”. In: *Phys. Rev. Lett.* 116.22 (2016). [Erratum: Phys.Rev.Lett. 121, 129902 (2018)], p. 221101. DOI: 10.1103/PhysRevLett.116.221101. arXiv: 1602.03841 [gr-qc] (cit. on pp. 1, 11, 27).

[17] R. Abbott et al. “Tests of General Relativity with GWTC-3”. In: (Dec. 2021). arXiv: 2112.06861 [gr-qc] (cit. on pp. 1, 11, 27).

[18] B. P. Abbott et al. "GW170817: Measurements of neutron star radii and equation of state". In: *Phys. Rev. Lett.* 121.16 (2018), p. 161101. DOI: 10.1103/PhysRevLett.121.161101. arXiv: 1805.11581 [gr-qc] (cit. on pp. 1, 27, 93).

[19] Collin D. Capano et al. "Stringent constraints on neutron-star radii from multimessenger observations and nuclear theory". In: *Nature Astron.* 4.6 (2020), pp. 625–632. DOI: 10.1038/s41550-020-1014-6. arXiv: 1908.10352 [astro-ph.HE] (cit. on pp. 1, 27, 93).

[20] M. Fishbach et al. "A Standard Siren Measurement of the Hubble Constant from GW170817 without the Electromagnetic Counterpart". In: *Astrophys. J. Lett.* 871.1 (2019), p. L13. DOI: 10.3847/2041-8213/aaf96e. arXiv: 1807.05667 [astro-ph.CO] (cit. on pp. 1, 27, 93).

[21] M. Soares-Santos et al. "First Measurement of the Hubble Constant from a Dark Standard Siren using the Dark Energy Survey Galaxies and the LIGO/Virgo Binary–Black-hole Merger GW170814". In: *Astrophys. J. Lett.* 876.1 (2019), p. L7. DOI: 10.3847/2041-8213/ab14f1. arXiv: 1901.01540 [astro-ph.CO] (cit. on pp. 1, 27).

[22] R. Abbott et al. "The population of merging compact binaries inferred using gravitational waves through GWTC-3". In: (Nov. 2021). arXiv: 2111.03634 [astro-ph.HE] (cit. on pp. 1, 11, 27).

[23] R. Abbott et al. "Search for Subsolar-Mass Binaries in the First Half of Advanced LIGO's and Advanced Virgo's Third Observing Run". In: *Phys. Rev. Lett.* 129.6 (2022), p. 061104. DOI: 10.1103/PhysRevLett.129.061104. arXiv: 2109.12197 [astro-ph.CO] (cit. on pp. 1, 11, 27).

[24] B. P. Abbott et al. "GW170817: Observation of Gravitational Waves from a Binary Neutron Star Inspiral". In: *Phys. Rev. Lett.* 119.16 (2017), p. 161101. DOI: 10.1103/PhysRevLett.119.161101. arXiv: 1710.05832 [gr-qc] (cit. on pp. 1, 93, 102, 155).

[25] D. A. Coulter et al. "Swope Supernova Survey 2017a (SSS17a), the Optical Counterpart to a Gravitational Wave Source". In: *Science* 358 (2017), p. 1556. DOI: 10.1126/science.aap9811. arXiv: 1710.05452 [astro-ph.HE] (cit. on pp. 1, 93).

[26] M. Soares-Santos et al. "The Electromagnetic Counterpart of the Binary Neutron Star Merger LIGO/Virgo GW170817. I. Discovery of the Optical Counterpart Using the Dark Energy Camera". In: *Astro-*

phys. J. Lett. 848.2 (2017), p. L16. DOI: 10.3847/2041-8213/aa9059. arXiv: 1710.05459 [astro-ph.HE] (cit. on pp. 1, 102).

[27] B. P. Abbott et al. "Multi-messenger Observations of a Binary Neutron Star Merger". In: *Astrophys. J. Lett.* 848.2 (2017), p. L12. DOI: 10.3847/2041-8213/aa91c9. arXiv: 1710.05833 [astro-ph.HE] (cit. on pp. 1, 93, 102).

[28] R. Abbott et al. "Observation of Gravitational Waves from Two Neutron Star–Black Hole Coalescences". In: *Astrophys. J. Lett.* 915.1 (2021), p. L5. DOI: 10.3847/2041-8213/ac082e. arXiv: 2106.15163 [astro-ph.HE] (cit. on pp. 2, 132).

[29] LIGO Scientific. *LIGO Technology*. 2022. URL: https://www.ligo.caltech.edu/page/ligo-technology (visited on 10/12/2022) (cit. on p. 2).

[30] Bruce Allen et al. "FINDCHIRP: An Algorithm for detection of gravitational waves from inspiraling compact binaries". In: *Phys. Rev. D* 85 (2012), p. 122006. DOI: 10.1103/PhysRevD.85.122006. arXiv: gr-qc/0509116 (cit. on pp. 2, 28, 40, 41, 42, 103, 133, 155, 171, 173).

[31] Alessandra Buonanno and Thibault Damour. "Transition from inspiral to plunge in binary black hole coalescences". In: *Phys. Rev. D* 62 (2000), p. 064015. DOI: 10.1103/PhysRevD.62.064015. arXiv: gr-qc/0001013 (cit. on pp. 2, 24).

[32] Parameswaran Ajith et al. "Phenomenological template family for black-hole coalescence waveforms". In: *Class. Quant. Grav.* 24 (2007). Ed. by B. Krishnan, M. A. Papa, and Bernard F. Schutz, S689–S700. DOI: 10.1088/0264-9381/24/19/S31. arXiv: 0704.3764 [gr-qc] (cit. on pp. 2, 25).

[33] LIGO Scientific. *Introduction to LIGO and Gravitational Waves: Ripples in space-time*. 2022. URL: https://www.ligo.org/science/GW-GW2.php (visited on 10/12/2022) (cit. on p. 2).

[34] M. Heurs. "Gravitational wave detection using laser interferometry beyond the standard quantum limit". In: *Phil. Trans. Roy. Soc. Lond. A* 376.2120 (2018), p. 20170289. DOI: 10.1098/rsta.2017.0289 (cit. on pp. 2, 35).

[35] Norbert Wiener et al. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*. Vol. 113. 21. MIT press Cambridge, MA, 1949 (cit. on p. 2).

[36] LIGO Scientific Collaboration and VIRGO Collaboration. *Online Pipelines*. 2018. URL: https://emfollow.docs.ligo.org/userguide/analysis/searches.html (visited on 09/10/2019) (cit. on pp. 2, 102).

[37] S. Klimenko et al. "Constraint likelihood analysis for a network of gravitational wave detectors". In: *Phys. Rev. D* 72 (2005), p. 122002. DOI: 10.1103/PhysRevD.72.122002. arXiv: gr-qc/0508068 (cit. on pp. 2, 79, 103, 155).

[38] S. Klimenko et al. "Method for detection and reconstruction of gravitational wave transients with networks of advanced detectors". In: *Phys. Rev. D* 93.4 (2016), p. 042004. DOI: 10.1103/PhysRevD.93.042004. arXiv: 1511.05999 [gr-qc] (cit. on pp. 2, 4, 28, 79, 103, 133, 155, 175, 188).

[39] Ryan Lynch et al. "Information-theoretic approach to the gravitational-wave burst detection problem". In: *Phys. Rev. D* 95.10 (2017), p. 104046. DOI: 10.1103/PhysRevD.95.104046. arXiv: 1511.05955 [gr-qc] (cit. on pp. 2, 103, 155).

[40] Ian Harry et al. "Searching for Gravitational Waves from Compact Binaries with Precessing Spins". In: *Phys. Rev. D* 94.2 (2016), p. 024012. DOI: 10.1103/PhysRevD.94.024012. arXiv: 1603.02444 [gr-qc] (cit. on pp. 2, 103, 128, 133, 155, 162, 177).

[41] Ian Harry, Juan Calderón Bustillo, and Alex Nitz. "Searching for the full symphony of black hole binary mergers". In: *Phys. Rev. D* 97.2 (2018), p. 023004. DOI: 10.1103/PhysRevD.97.023004. arXiv: 1709.09181 [gr-qc] (cit. on pp. 2, 103, 128, 133, 155, 162, 177).

[42] Juan Calderón Bustillo, Pablo Laguna, and Deirdre Shoemaker. "Detectability of gravitational waves from binary black holes: Impact of precession and higher modes". In: *Phys. Rev. D* 95.10 (2017), p. 104038. DOI: 10.1103/PhysRevD.95.104038. arXiv: 1612.02340 [gr-qc] (cit. on p. 2).

[43] Koustav Chandra et al. "First gravitational-wave search for intermediate-mass black hole mergers with higher order harmonics". In: (July 2022). arXiv: 2207.01654 [gr-qc] (cit. on pp. 2, 28, 177).

[44] Rahul Dhurkunde and Alexander H. Nitz. "Sensitivity of spin-aligned searches for neutron star-black hole systems using future detectors". In: (July 2022). arXiv: 2207.14645 [astro-ph.IM] (cit. on pp. 2, 24, 155, 177).

[45] LIGO Scientific Collaboration. *Instrument Science White Paper*. 2017. URL: https://dcc.ligo.org/public/0142/T1700231/003/T1700231-v3.pdf (visited on 08/27/2022) (cit. on pp. 2, 37, 94).

[46] Michele Maggiore et al. "Science Case for the Einstein Telescope". In: *JCAP* 03 (2020), p. 050. DOI: 10.1088/1475-7516/2020/03/050. arXiv: 1912.02622 [astro-ph.CO] (cit. on pp. 2, 37).

[47] David Reitze et al. "Cosmic Explorer: The U.S. Contribution to Gravitational-Wave Astronomy beyond LIGO". In: *Bull. Am. Astron. Soc.* 51.7 (2019), p. 035. arXiv: 1907.04833 [astro-ph.IM] (cit. on pp. 2, 37).

[48] Elena Cuoco et al. "Enhancing Gravitational-Wave Science with Machine Learning". In: *Mach. Learn. Sci. Tech.* 2.1 (2021), p. 011002. DOI: 10.1088/2632-2153/abb93a. arXiv: 2005.03745 [astro-ph.HE] (cit. on pp. 3, 28, 75, 76, 103, 133, 156).

[49] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105 (cit. on pp. 3, 48, 70, 75, 197).

[50] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2015. DOI: 10.48550/ARXIV.1506.01497. URL: https://arxiv.org/abs/1506.01497 (cit. on pp. 3, 70, 198).

[51] OpenAI et al. *Dota 2 with Large Scale Deep Reinforcement Learning*. 2019. arXiv: 1912.06680 [cs.LG] (cit. on pp. 3, 48).

[52] Reinhard Prix and Badri Krishnan. "Targeted search for continuous gravitational waves: Bayesian versus maximum-likelihood statistics". In: *Class. Quant. Grav.* 26 (2009). Ed. by Mario Diaz, Fredrick Jenet, and Soumya Mohanty, p. 204013. DOI: 10.1088/0264-9381/26/20/204013. arXiv: 0907.2569 [gr-qc] (cit. on p. 3).

[53] Paul T. Baker et al. "Multivariate Classification with Random Forests for Gravitational Wave Searches of Black Hole Binary Coalescence". In: *Phys. Rev. D* 91.6 (2015), p. 062004. DOI: 10.1103/PhysRevD.91.062004. arXiv: 1412.6479 [gr-qc] (cit. on pp. 3, 76).

[54] Jingkai Yan et al. "Generalized approach to matched filtering using neural networks". In: *Phys. Rev. D* 105.4 (2022), p. 043006. DOI: 10.1103/PhysRevD.105.043006. arXiv: 2104.03961 [astro-ph.IM] (cit. on pp. 3, 78, 128).

[55] Daniel George and E. A. Huerta. "Deep Neural Networks to Enable Real-time Multimessenger Astrophysics". In: *Phys. Rev. D* 97.4 (2018), p. 044039. DOI: 10.1103/PhysRevD.97.044039. arXiv: 1701.00008 [astro-ph.IM] (cit. on pp. 3, 28, 76, 77, 79, 103, 104, 106, 110, 113, 133, 156, 161, 177).

[56] Hunter Gabbard et al. "Matching matched filtering with deep networks for gravitational-wave astronomy". In: *Phys. Rev. Lett.* 120.14 (2018), p. 141103. DOI: 10.1103/PhysRevLett.120.141103. arXiv: 1712.06041 [astro-ph.IM] (cit. on pp. 3, 7, 77, 79, 80, 103, 104, 105, 106, 110, 113, 114, 116, 117, 123, 133, 135, 149, 156, 161, 172, 173, 177).

[57] Wei Wei et al. "Deep Learning Ensemble for Real-time Gravitational Wave Detection of Spinning Binary Black Hole Mergers". In: *Phys. Lett. B* 812 (2021), p. 136029. DOI: 10.1016/j.physletb.2020.136029. arXiv: 2010.15845 [gr-qc] (cit. on pp. 3, 28, 78, 113, 128, 133, 151, 156, 163, 177, 190).

[58] Alexander H. Nitz et al. "Rapid detection of gravitational waves from compact binary mergers with PyCBC Live". In: *Phys. Rev. D* 98.2 (2018), p. 024050. DOI: 10.1103/PhysRevD.98.024050. arXiv: 1805.11174 [gr-qc] (cit. on pp. 4, 28, 83, 89, 94, 102, 105, 109, 171, 178).

[59] Cody Messick et al. "Analysis Framework for the Prompt Discovery of Compact Binary Mergers in Gravitational-wave Data". In: *Phys. Rev. D* 95.4 (2017), p. 042001. DOI: 10.1103/PhysRevD.95.042001. arXiv: 1604.04324 [astro-ph.IM] (cit. on pp. 4, 28, 43, 44, 133, 155, 177).

[60] Samantha A. Usman et al. "The PyCBC search for gravitational waves from compact binary coalescence". In: *Class. Quant. Grav.* 33.21 (2016), p. 215004. DOI: 10.1088/0264-9381/33/21/215004. arXiv: 1508.02357 [gr-qc] (cit. on pp. 4, 28, 40, 42, 43, 44, 45, 46, 79, 80, 133, 134, 139, 140, 151, 164, 165, 177, 182, 188).

[61] Marlin B. Schäfer, Frank Ohme, and Alexander H. Nitz. "Detection of gravitational-wave signals from binary neutron star mergers using machine learning". In: *Phys. Rev. D* 102.6 (2020), p. 063015. DOI: 10.1103/PhysRevD.102.063015. arXiv: 2006.01509 [astro-ph.HE] (cit. on pp. 5, 6, 28, 77, 83, 84, 85, 86, 87, 88, 90, 103, 104, 110, 111, 113, 128, 133, 156, 162, 177, 190, 271).

[62] Alexander H. Nitz, Marlin Schäfer, and Tito Dal Canton. "Gravitational-wave Merger Forecasting: Scenarios for the Early Detection and Localization of Compact-binary Mergers with Ground-based Observatories". In: *Astrophys. J. Lett.* 902 (2020), p. L29. DOI: 10.3847/2041-8213/abbc10. arXiv: 2009.04439 [astro-ph.HE] (cit. on pp. 5, 6, 75, 93, 94, 96, 97, 98, 271).

[63] Marlin B. Schäfer et al. "Training strategies for deep learning gravitational-wave searches". In: *Phys. Rev. D* 105.4 (2022), p. 043002. DOI: 10.1103/PhysRevD.105.043002. arXiv: 2106.03741 [astro-ph.IM] (cit. on pp. 5, 7, 101, 133, 135, 136, 137, 145, 149, 156, 173, 271).

[64] Marlin B. Schäfer and Alexander H. Nitz. "From one to many: A deep learning coincident gravitational-wave search". In: *Phys. Rev. D* 105.4 (2022), p. 043003. DOI: 10.1103/PhysRevD.105.043003. arXiv: 2108.10715 [astro-ph.IM] (cit. on pp. 5, 7, 132, 156, 161, 172, 173, 177, 182, 183, 271).

[65] Marlin B. Schäfer et al. "MLGWSC-1: The first Machine Learning Gravitational-Wave Search Mock Data Challenge". In: (Sept. 2022). arXiv: 2209.11146 [astro-ph.IM] (cit. on pp. 5, 7, 28, 154, 271).

[66] Marlin Benedikt Schäfer. *Analysis of Gravitational-Wave Signals from Binary Neutron Star Mergers Using Machine Learning*. Sept. 2019. DOI: 10.15488/7467 (cit. on pp. 6, 11, 66, 83, 84, 86, 87, 88).

[67] A. Einstein. "Die Grundlage der allgemeinen Relativitätstheorie". In: *Annalen der Physik* 354.7 (1916), pp. 769–822. DOI: https://doi.org/10.1002/andp.19163540702. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/andp.19163540702. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.19163540702 (cit. on p. 10).

[68] Michelle Maggiore. *Gravitational Waves Volume 1: Theory and Experiments*. Oxford University Press, 2016. ISBN: 978-0-19-857074-5 (cit. on pp. 10, 11, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 27, 29, 33, 35, 39, 40, 185).

[69] Michelle Maggiore. *Gravitational Waves Volume 2: Astrophysics and Cosmology*. Oxford University Press, 2019. ISBN: 978-0-19-857089-9 (cit. on pp. 10, 11, 24).

[70] Frans Pretorius. "Binary Black Hole Coalescence". In: (Aug. 2007). arXiv: 0710.1338 [gr-qc] (cit. on p. 10).

[71] Mark Hannam. "Status of black-hole-binary simulations for gravitational-wave detection". In: *Class. Quant. Grav.* 26 (2009). Ed. by Patrick Sutton and Deirdre Shoemaker, p. 114001. DOI: 10.1088/0264-9381/26/11/114001. arXiv: 0901.2931 [gr-qc] (cit. on pp. 10, 23).

[72] Ian Hinder. "The Current Status of Binary Black Hole Simulations in Numerical Relativity". In: *Class. Quant. Grav.* 27 (2010). Ed. by Sascha Husa and Badri Krishnan, p. 114004. DOI: 10.1088/0264-9381/27/11/114004. arXiv: 1001.5161 [gr-qc] (cit. on p. 10).

[73] Joan M. Centrella et al. "The Final Merger of Black-Hole Binaries". In: *Ann. Rev. Nucl. Part. Sci.* 60 (2010), pp. 75–100. DOI: 10.1146/annurev.nucl.010909.083246. arXiv: 1010.2165 [gr-qc] (cit. on p. 10).

[74] Hans-Peter Nollert. "TOPICAL REVIEW: Quasinormal modes: the characteristic 'sound' of black holes and neutron stars". In: *Class. Quant. Grav.* 16 (1999), R159–R216. DOI: 10.1088/0264-9381/16/12/201 (cit. on pp. 10, 11).

[75] A. Bauswein et al. "Equation-of-state dependence of the gravitational-wave signal from the ring-down phase of neutron-star mergers". In: *Phys. Rev. D* 86 (2012), p. 063001. DOI: 10.1103/PhysRevD.86.063001. arXiv: 1204.1888 [astro-ph.SR] (cit. on p. 11).

[76] Kostas D. Kokkotas and Bernd G. Schmidt. "Quasinormal modes of stars and black holes". In: *Living Rev. Rel.* 2 (1999), p. 2. DOI: 10.12942/lrr-1999-2. arXiv: gr-qc/9909058 (cit. on p. 11).

[77] Vladimir Dergachev and Maria Alessandra Papa. "Results from the First All-Sky Search for Continuous Gravitational Waves from Small-Ellipticity Sources". In: *Phys. Rev. Lett.* 125.17 (2020), p. 171101. DOI: 10.1103/PhysRevLett.125.171101. arXiv: 2004.08334 [gr-qc] (cit. on p. 11).

[78] B. Steltner et al. "Einstein@Home All-sky Search for Continuous Gravitational Waves in LIGO O2 Public Data". In: *Astrophys. J.* 909.1 (2021), p. 79. DOI: 10.3847/1538-4357/abc7c9. arXiv: 2009.12260 [astro-ph.HE] (cit. on p. 11).

[79] R. Abbott et al. "All-sky search for continuous gravitational waves from isolated neutron stars using Advanced LIGO and Advanced Virgo O3 data". In: (Jan. 2022). arXiv: 2201.00697 [gr-qc] (cit. on p. 11).

[80] B. P. Abbott et al. “A First Targeted Search for Gravitational-Wave Bursts from Core-Collapse Supernovae in Data of First-Generation Laser Interferometer Detectors”. In: *Phys. Rev. D* 94.10 (2016), p. 102001. DOI: 10.1103/PhysRevD.94.102001. arXiv: 1605.01785 [gr-qc] (cit. on p. 11).

[81] Pau Amaro-Seoane et al. “Laser Interferometer Space Antenna”. In: (Feb. 2017). arXiv: 1702.00786 [astro-ph.IM] (cit. on pp. 11, 37).

[82] D. N. Matsakis, J. H. Taylor, and T. Marshall Eubanks. “A statistic for describing pulsar and clock stabilities.” In: *Astronomy and Astrophysics* 326 (Oct. 1997), pp. 924–928 (cit. on pp. 11, 27).

[83] J. P. W. Verbiest et al. “Timing stability of millisecond pulsars and prospects for gravitational-wave detection”. In: *Monthly Notices of the Royal Astronomical Society* 400.2 (Dec. 2009), pp. 951–968. DOI: 10.1111/j.1365-2966.2009.15508.x. URL: https://doi.org/10.1111%2Fj.1365-2966.2009.15508.x (cit. on pp. 11, 27).

[84] Paul D. Lasky. “Gravitational Waves from Neutron Stars: A Review”. In: *Publ. Astron. Soc. Austral.* 32 (2015), e034. DOI: 10.1017/pasa.2015.35. arXiv: 1508.06643 [astro-ph.HE] (cit. on p. 11).

[85] Bharat Ratra and Michael S. Vogeley. “The Beginning and Evolution of the Universe”. In: *Publ. Astron. Soc. Pac.* 120 (2008), pp. 235–265. DOI: 10.1086/529495. arXiv: 0706.1565 [astro-ph] (cit. on p. 11).

[86] Luc Blanchet. “Gravitational Radiation from Post-Newtonian Sources and Inspiralling Compact Binaries”. In: *Living Reviews in Relativity* 9.1 (2006), p. 4. DOI: 10.12942/lrr-2006-4. URL: https://doi.org/10.12942/lrr-2006-4 (cit. on pp. 11, 20, 21, 22, 23, 87).

[87] Robert M. Wald. *General Relativity*. The University of Chicago Press, 1984. ISBN: 0-226-87033-2 (cit. on p. 13).

[88] Norbert Dragon. *Stichworte und Ergänzungen zu Mathematische Methoden der Physik*. 2016. URL: https://www.itp.uni-hannover.de/fileadmin/itp/emeritus/dragon/rech.pdf (cit. on p. 13).

[89] Charles W. Misner, Kip S. Thorne, and John Archibald Wheeler. *Gravitation*. Princeton University Press, 2017. ISBN: 978-0-691-17779-3 (cit. on pp. 13, 14, 17, 19).

[90] Luc Blanchet et al. “Gravitational wave forms from inspiralling compact binaries to second postNewtonian order”. In: *Class. Quant. Grav.* 13 (1996), pp. 575–584. DOI: 10.1088/0264-9381/13/4/002. arXiv: gr-qc/9602024 (cit. on p. 23).

[91] K. G. Arun et al. "The 2.5PN gravitational wave polarisations from inspiralling compact binaries in circular orbits". In: *Class. Quant. Grav.* 21 (2004). [Erratum: Class.Quant.Grav. 22, 3115 (2005)], pp. 3771–3802. DOI: 10.1088/0264-9381/21/15/010. arXiv: gr-qc/0404085 (cit. on p. 23).

[92] Thibault Damour. "Coalescence of two spinning black holes: an effective one-body approach". In: *Phys. Rev. D* 64 (2001), p. 124013. DOI: 10.1103/PhysRevD.64.124013. arXiv: gr-qc/0103018 (cit. on pp. 23, 24).

[93] Luc Blanchet et al. "Gravitational radiation from inspiralling compact binaries completed at the third post-Newtonian order". In: *Phys. Rev. Lett.* 93 (2004), p. 091101. DOI: 10.1103/PhysRevLett.93.091101. arXiv: gr-qc/0406012 (cit. on p. 23).

[94] Guillaume Faye, Luc Blanchet, and Alessandra Buonanno. "Higher-order spin effects in the dynamics of compact binaries. I. Equations of motion". In: *Phys. Rev. D* 74 (2006), p. 104033. DOI: 10.1103/PhysRevD.74.104033. arXiv: gr-qc/0605139 (cit. on p. 23).

[95] Luc Blanchet, Alessandra Buonanno, and Guillaume Faye. "Higher-order spin effects in the dynamics of compact binaries. II. Radiation field". In: *Phys. Rev. D* 74 (2006). [Erratum: Phys.Rev.D 75, 049903 (2007), Erratum: Phys.Rev.D 81, 089901 (2010)], p. 104034. DOI: 10.1103/PhysRevD.81.089901. arXiv: gr-qc/0605140 (cit. on p. 23).

[96] Thibault Damour, Piotr Jaranowski, and Gerhard Schaefer. "Hamiltonian of two spinning compact bodies with next-to-leading order gravitational spin-orbit coupling". In: *Phys. Rev. D* 77 (2008), p. 064032. DOI: 10.1103/PhysRevD.77.064032. arXiv: 0711.1048 [gr-qc] (cit. on p. 23).

[97] L. Santamaria et al. "Matching post-Newtonian and numerical relativity waveforms: systematic errors and a new phenomenological model for non-precessing black hole binaries". In: *Phys. Rev. D* 82 (2010), p. 064016. DOI: 10.1103/PhysRevD.82.064016. arXiv: 1005.3306 [gr-qc] (cit. on pp. 23, 25, 26).

[98] K. S. Thorne. "Multipole Expansions of Gravitational Radiation". In: *Rev. Mod. Phys.* 52 (1980), pp. 299–339. DOI: 10.1103/RevModPhys.52.299 (cit. on p. 24).

[99] A. Buonanno and T. Damour. "Effective one-body approach to general relativistic two-body dynamics". In: *Phys. Rev. D* 59 (1999), p. 084006. DOI: 10.1103/PhysRevD.59.084006. arXiv: gr-qc/9811091 (cit. on p. 24).

[100] Thibault Damour et al. "Accurate Effective-One-Body waveforms of inspiralling and coalescing black-hole binaries". In: *Phys. Rev. D* 78 (2008), p. 044039. DOI: 10.1103/PhysRevD.78.044039. arXiv: 0803.3162 [gr-qc] (cit. on p. 24).

[101] Thibault Damour et al. "Gravitational waves from black hole binary inspiral and merger: The Span of third postNewtonian effective one-body templates". In: *Phys. Rev. D* 67 (2003), p. 064028. DOI: 10.1103/PhysRevD.67.064028. arXiv: gr-qc/0211041 (cit. on p. 24).

[102] Thibault Damour and Alessandro Nagar. "Faithful effective-one-body waveforms of small-mass-ratio coalescing black-hole binaries". In: *Phys. Rev. D* 76 (2007), p. 064028. DOI: 10.1103/PhysRevD.76.064028. arXiv: 0705.2519 [gr-qc] (cit. on p. 24).

[103] Alessandra Buonanno, Yanbei Chen, and Thibault Damour. "Transition from inspiral to plunge in precessing binaries of spinning black holes". In: *Phys. Rev. D* 74 (2006), p. 104005. DOI: 10.1103/PhysRevD.74.104005. arXiv: gr-qc/0508067 (cit. on p. 24).

[104] Enrico Barausse, Etienne Racine, and Alessandra Buonanno. "Hamiltonian of a spinning test-particle in curved spacetime". In: *Phys. Rev. D* 80 (2009). [Erratum: Phys.Rev.D 85, 069904 (2012)], p. 104025. DOI: 10.1103/PhysRevD.85.069904. arXiv: 0907.4745 [gr-qc] (cit. on p. 24).

[105] Frans Pretorius. "Evolution of binary black hole spacetimes". In: *Phys. Rev. Lett.* 95 (2005), p. 121101. DOI: 10.1103/PhysRevLett.95.121101. arXiv: gr-qc/0507014 (cit. on p. 24).

[106] Manuela Campanelli et al. "Accurate evolutions of orbiting black-hole binaries without excision". In: *Phys. Rev. Lett.* 96 (2006), p. 111101. DOI: 10.1103/PhysRevLett.96.111101. arXiv: gr-qc/0511048 (cit. on p. 24).

[107] John G. Baker et al. "Gravitational wave extraction from an inspiraling configuration of merging black holes". In: *Phys. Rev. Lett.* 96 (2006), p. 111102. DOI: 10.1103/PhysRevLett.96.111102. arXiv: gr-qc/0511103 (cit. on p. 24).

[108] Alessandra Buonanno et al. "Toward faithful templates for non-spinning binary black holes using the effective-one-body approach". In: *Phys. Rev. D* 76 (2007), p. 104049. DOI: 10.1103/PhysRevD.76.104049. arXiv: 0706.3732 [gr-qc] (cit. on p. 24).

[109] Thibault Damour and Alessandro Nagar. "An Improved analytical description of inspiralling and coalescing black-hole binaries". In: *Phys. Rev. D* 79 (2009), p. 081503. DOI: 10.1103/PhysRevD.79.081503. arXiv: 0902.0136 [gr-qc] (cit. on p. 25).

[110] Alessandra Buonanno et al. "Effective-one-body waveforms calibrated to numerical relativity simulations: Coalescence of non-spinning, equal-mass black holes". In: *Phys. Rev. D* 79 (2009), p. 124028. DOI: 10.1103/PhysRevD.79.124028. arXiv: 0902.0790 [gr-qc] (cit. on p. 25).

[111] Enrico Barausse and Alessandra Buonanno. "An Improved effective-one-body Hamiltonian for spinning black-hole binaries". In: *Phys. Rev. D* 81 (2010), p. 084024. DOI: 10.1103/PhysRevD.81.084024. arXiv: 0912.3517 [gr-qc] (cit. on p. 25).

[112] Yi Pan et al. "Effective-one-body waveforms calibrated to numerical relativity simulations: coalescence of non-precessing, spinning, equal-mass black holes". In: *Phys. Rev. D* 81 (2010), p. 084041. DOI: 10.1103/PhysRevD.81.084041. arXiv: 0912.3466 [gr-qc] (cit. on p. 25).

[113] Yi Pan et al. "Inspiral-merger-ringdown multipolar waveforms of non-spinning black-hole binaries using the effective-one-body formalism". In: *Phys. Rev. D* 84 (2011), p. 124052. DOI: 10.1103/PhysRevD.84.124052. arXiv: 1106.1021 [gr-qc] (cit. on p. 25).

[114] Yi Pan et al. "Inspiral-merger-ringdown waveforms of spinning, precessing black-hole binaries in the effective-one-body formalism". In: *Phys. Rev. D* 89.8 (2014), p. 084006. DOI: 10.1103/PhysRevD.89.084006. arXiv: 1307.6232 [gr-qc] (cit. on p. 25).

[115] Andrea Taracchini et al. "Prototype effective-one-body model for non-precessing spinning inspiral-merger-ringdown waveforms". In: *Phys. Rev. D* 86 (2012), p. 024011. DOI: 10.1103/PhysRevD.86.024011. arXiv: 1202.0790 [gr-qc] (cit. on p. 25).

[116] Andrea Taracchini et al. "Effective-one-body model for black-hole binaries with generic mass ratios and spins". In: *Phys. Rev. D* 89.6 (2014), p. 061502. DOI: 10.1103/PhysRevD.89.061502. arXiv: 1311.2544 [gr-qc] (cit. on p. 25).

[117] B. P. Abbott et al. “Properties of the Binary Black Hole Merger GW150914”. In: *Phys. Rev. Lett.* 116.24 (2016), p. 241102. DOI: 10.1103/PhysRevLett.116.241102. arXiv: 1602.03840 [gr-qc] (cit. on pp. 25, 43).

[118] Alejandro Bohé et al. “Improved effective-one-body model of spinning, nonprecessing binary black holes for the era of gravitational-wave astrophysics with advanced detectors”. In: *Phys. Rev. D* 95.4 (2017), p. 044028. DOI: 10.1103/PhysRevD.95.044028. arXiv: 1611.03703 [gr-qc] (cit. on pp. 25, 107, 136, 176).

[119] Serguei Ossokine et al. “Multipolar Effective-One-Body Waveforms for Precessing Binary Black Holes: Construction and Validation”. In: *Phys. Rev. D* 102.4 (2020), p. 044055. DOI: 10.1103/PhysRevD.102.044055. arXiv: 2004.09442 [gr-qc] (cit. on p. 25).

[120] Michael Pürrer. “Frequency domain reduced order models for gravitational waves from aligned-spin compact binaries”. In: *Class. Quant. Grav.* 31.19 (2014), p. 195010. DOI: 10.1088/0264-9381/31/19/195010. arXiv: 1402.4146 [gr-qc] (cit. on p. 25).

[121] Michael Pürrer. “Frequency domain reduced order model of aligned-spin effective-one-body waveforms with generic mass-ratios and spins”. In: *Phys. Rev. D* 93.6 (2016), p. 064041. DOI: 10.1103/PhysRevD.93.064041. arXiv: 1512.02248 [gr-qc] (cit. on p. 25).

[122] P. Ajith et al. “Inspiral-merger-ringdown waveforms for black-hole binaries with non-precessing spins”. In: *Phys. Rev. Lett.* 106 (2011), p. 241101. DOI: 10.1103/PhysRevLett.106.241101. arXiv: 0909.2867 [gr-qc] (cit. on pp. 25, 26).

[123] Patricia Schmidt, Mark Hannam, and Sascha Husa. “Towards models of gravitational waveforms from generic binaries: A simple approximate mapping between precessing and non-precessing inspiral signals”. In: *Phys. Rev. D* 86 (2012), p. 104063. DOI: 10.1103/PhysRevD.86.104063. arXiv: 1207.3088 [gr-qc] (cit. on pp. 25, 31).

[124] Mark Hannam et al. “Simple Model of Complete Precessing Black-Hole-Binary Gravitational Waveforms”. In: *Phys. Rev. Lett.* 113.15 (2014), p. 151101. DOI: 10.1103/PhysRevLett.113.151101. arXiv: 1308.3271 [gr-qc] (cit. on p. 25).

[125] Benjamin P Abbott et al. “A guide to LIGO–Virgo detector noise and extraction of transient gravitational-wave signals”. In: *Class. Quant. Grav.* 37.5 (2020), p. 055002. DOI: 10.1088/1361-6382/ab685e. arXiv: 1908.11170 [gr-qc] (cit. on pp. 25, 36, 43, 133, 177).

[126] Lionel London et al. "First higher-multipole model of gravitational waves from spinning and coalescing black-hole binaries". In: *Phys. Rev. Lett.* 120.16 (2018), p. 161102. DOI: 10.1103/PhysRevLett.120.161102. arXiv: 1708.00404 [gr-qc] (cit. on p. 25).

[127] Sebastian Khan et al. "Phenomenological model for the gravitational-wave signal from precessing binary black holes with two-spin effects". In: *Phys. Rev. D* 100.2 (2019), p. 024059. DOI: 10.1103/PhysRevD.100.024059. arXiv: 1809.10113 [gr-qc] (cit. on p. 25).

[128] Sebastian Khan et al. "Including higher order multipoles in gravitational-wave models for precessing binary black holes". In: *Phys. Rev. D* 101.2 (2020), p. 024056. DOI: 10.1103/PhysRevD.101.024056. arXiv: 1911.06050 [gr-qc] (cit. on p. 25).

[129] Sascha Husa et al. "Frequency-domain gravitational waves from nonprecessing black-hole binaries. I. New numerical waveforms and anatomy of the signal". In: *Phys. Rev. D* 93.4 (2016), p. 044006. DOI: 10.1103/PhysRevD.93.044006. arXiv: 1508.07250 [gr-qc] (cit. on pp. 26, 141, 171).

[130] Sebastian Khan et al. "Frequency-domain gravitational waves from nonprecessing black-hole binaries. II. A phenomenological model for the advanced detector era". In: *Phys. Rev. D* 93.4 (2016), p. 044007. DOI: 10.1103/PhysRevD.93.044007. arXiv: 1508.07253 [gr-qc] (cit. on pp. 26, 141).

[131] Geraint Pratten et al. "Setting the cornerstone for a family of models for gravitational waves from compact binaries: The dominant harmonic for nonprecessing quasicircular black holes". In: *Phys. Rev. D* 102.6 (2020), p. 064001. DOI: 10.1103/PhysRevD.102.064001. arXiv: 2001.11412 [gr-qc] (cit. on p. 26).

[132] Cecilio García-Quirós et al. "Multimode frequency-domain model for the gravitational wave signal from nonprecessing black-hole binaries". In: *Phys. Rev. D* 102.6 (2020), p. 064002. DOI: 10.1103/PhysRevD.102.064002. arXiv: 2001.10914 [gr-qc] (cit. on p. 26).

[133] Geraint Pratten et al. "Computationally efficient models for the dominant and subdominant harmonic modes of precessing binary black holes". In: *Phys. Rev. D* 103.10 (2021), p. 104056. DOI: 10.1103/PhysRevD.103.104056. arXiv: 2004.06503 [gr-qc] (cit. on pp. 26, 159, 162, 172).

[134] Michael Boyle et al. "The SXS Collaboration catalog of binary black hole simulations". In: *Class. Quant. Grav.* 36.19 (2019), p. 195006. DOI: 10.1088/1361-6382/ab34e2. arXiv: 1904.04831 [gr-qc] (cit. on p. 26).

[135] James Healy and Carlos O. Lousto. "Fourth RIT binary black hole simulations catalog: Extension to eccentric orbits". In: *Phys. Rev. D* 105.12 (2022), p. 124010. DOI: 10.1103/PhysRevD.105.124010. arXiv: 2202.00018 [gr-qc] (cit. on p. 26).

[136] Yoshinta Setyawati, Michael Pürrer, and Frank Ohme. "Regression methods in waveform modeling: a comparative study". In: *Class. Quant. Grav.* 37.7 (2020), p. 075012. DOI: 10.1088/1361-6382/ab693b. arXiv: 1909.10986 [astro-ph.IM] (cit. on p. 26).

[137] Jonathan Blackman et al. "Fast and Accurate Prediction of Numerical Relativity Waveforms from Binary Black Hole Coalescences Using Surrogate Models". In: *Phys. Rev. Lett.* 115.12 (2015), p. 121102. DOI: 10.1103/PhysRevLett.115.121102. arXiv: 1502.07758 [gr-qc] (cit. on p. 26).

[138] Mark A. Scheel et al. "Improved methods for simulating nearly extremal binary black holes". In: *Class. Quant. Grav.* 32.10 (2015), p. 105009. DOI: 10.1088/0264-9381/32/10/105009. arXiv: 1412.1803 [gr-qc] (cit. on p. 26).

[139] Béla Szilágyi. "Key Elements of Robustness in Binary Black Hole Evolutions using Spectral Methods". In: *Int. J. Mod. Phys. D* 23.7 (2014), p. 1430014. DOI: 10.1142/S0218271814300146. arXiv: 1405.3693 [gr-qc] (cit. on p. 26).

[140] Jonathan Blackman et al. "A Surrogate Model of Gravitational Waveforms from Numerical Relativity Simulations of Precessing Binary Black Hole Mergers". In: *Phys. Rev. D* 95.10 (2017), p. 104023. DOI: 10.1103/PhysRevD.95.104023. arXiv: 1701.00550 [gr-qc] (cit. on p. 26).

[141] Jonathan Blackman et al. "Numerical relativity waveform surrogate model for generically precessing binary black hole mergers". In: *Phys. Rev. D* 96.2 (2017), p. 024058. DOI: 10.1103/PhysRevD.96.024058. arXiv: 1705.07089 [gr-qc] (cit. on p. 26).

[142] Vijay Varma et al. "High-accuracy mass, spin, and recoil predictions of generic black-hole merger remnants". In: *Phys. Rev. Lett.* 122.1 (2019), p. 011101. DOI: 10.1103/PhysRevLett.122.011101. arXiv: 1809.09125 [gr-qc] (cit. on p. 26).

[143] Vijay Varma et al. "Surrogate models for precessing binary black hole simulations with unequal masses". In: *Phys. Rev. Research.* 1 (2019), p. 033015. DOI: 10.1103/PhysRevResearch.1.033015. arXiv: 1905.09300 [gr-qc] (cit. on p. 26).

[144] R. A. Hulse and J. H. Taylor. "Discovery of a pulsar in a binary system". In: *Astrophys. J. Lett.* 195 (1975), pp. L51–L53. DOI: 10.1086/181708 (cit. on p. 26).

[145] Joel M. Weisberg and Joseph H. Taylor. "Relativistic binary pulsar B1913+16: Thirty years of observations and analysis". In: *ASP Conf. Ser.* 328 (2005), p. 25. arXiv: astro-ph/0407149 (cit. on p. 27).

[146] J. Weber. "Gravitational-Wave-Detector Events". In: *Phys. Rev. Lett.* 20.23 (1968), p. 1307. DOI: 10.1103/PhysRevLett.20.1307 (cit. on p. 27).

[147] J. Weber. "Evidence for discovery of gravitational radiation". In: *Phys. Rev. Lett.* 22 (1969), pp. 1320–1324. DOI: 10.1103/PhysRevLett.22.1320 (cit. on p. 27).

[148] Matthew Pitkin et al. "Gravitational Wave Detection by Interferometry (Ground and Space)". In: *Living Rev. Rel.* 14 (2011), p. 5. DOI: 10.12942/lrr-2011-5. arXiv: 1102.3355 [astro-ph.IM] (cit. on pp. 27, 32).

[149] Rich Abbott et al. "Open data from the first and second observing runs of Advanced LIGO and Advanced Virgo". In: *SoftwareX* 13 (2021), p. 100658. DOI: 10.1016/j.softx.2021.100658. arXiv: 1912.11716 [gr-qc] (cit. on pp. 27, 39, 123, 163).

[150] Tejaswi Venumadhav et al. "New search pipeline for compact binary mergers: Results for binary black holes in the first observing run of Advanced LIGO". In: *Phys. Rev. D* 100.2 (2019), p. 023011. DOI: 10.1103/PhysRevD.100.023011. arXiv: 1902.10341 [astro-ph.IM] (cit. on pp. 27, 44, 155).

[151] Tejaswi Venumadhav et al. "New binary black hole mergers in the second observing run of Advanced LIGO and Advanced Virgo". In: *Phys. Rev. D* 101.8 (2020), p. 083030. DOI: 10.1103/PhysRevD.101.083030. arXiv: 1904.07214 [astro-ph.HE] (cit. on pp. 27, 28, 155).

[152] Seth Olsen et al. "New binary black hole mergers in the LIGO-Virgo O3a data". In: *Phys. Rev. D* 106.4 (2022), p. 043009. DOI: 10.1103/PhysRevD.106.043009. arXiv: 2201.02252 [astro-ph.HE] (cit. on pp. 27, 28).

[153] Andrey A. Shoom et al. “Testing GR with the Gravitational Wave Inspiral Signal GW170817”. In: (May 2021). arXiv: 2105.02191 [gr-qc] (cit. on p. 27).

[154] N. V. Krishnendu and Frank Ohme. “Testing General Relativity with Gravitational Waves: An Overview”. In: *Universe* 7.12 (2021), p. 497. DOI: 10.3390/universe7120497. arXiv: 2201.05418 [gr-qc] (cit. on p. 27).

[155] Misao Sasaki et al. “Primordial black holes—perspectives in gravitational wave astronomy”. In: *Class. Quant. Grav.* 35.6 (2018), p. 063001. DOI: 10.1088/1361-6382/aaa7b4. arXiv: 1801.05235 [astro-ph.CO] (cit. on p. 27).

[156] S. Basak et al. “Constraints on Compact Dark Matter from Gravitational Wave Microlensing”. In: *Astrophys. J.* 926.2 (2022), p. L28. DOI: 10.3847/2041-8213/ac4dfa. arXiv: 2109.06456 [gr-qc] (cit. on p. 27).

[157] Floor S. Broekgaarden and Edo Berger. “Formation of the First Two Black Hole–Neutron Star Mergers (GW200115 and GW200105) from Isolated Binary Evolution”. In: *Astrophys. J. Lett.* 920.1 (2021), p. L13. DOI: 10.3847/2041-8213/ac2832. arXiv: 2108.05763 [astro-ph.HE] (cit. on p. 27).

[158] Surabhi Sachdev et al. “The GstLAL Search Analysis Methods for Compact Binary Mergers in Advanced LIGO’s Second and Advanced Virgo’s First Observing Runs”. In: (Jan. 2019). arXiv: 1901.08580 [gr-qc] (cit. on pp. 28, 102, 133, 134, 139, 151).

[159] Chad Hanna et al. “Fast evaluation of multidetector consistency for real-time gravitational wave searches”. In: *Phys. Rev. D* 101.2 (2020), p. 022003. DOI: 10.1103/PhysRevD.101.022003. arXiv: 1901.02227 [gr-qc] (cit. on p. 28).

[160] Kipp Cannon et al. “GstLAL: A software framework for gravitational wave discovery”. In: (Oct. 2020). arXiv: 2010.05082 [astro-ph.IM] (cit. on p. 28).

[161] T. Adams et al. “Low-latency analysis pipeline for compact binary coalescences in the advanced gravitational wave detector era”. In: *Class. Quant. Grav.* 33.17 (2016), p. 175012. DOI: 10.1088/0264-9381/33/17/175012. arXiv: 1512.02864 [gr-qc] (cit. on pp. 28, 43, 44, 102, 133, 155).

[162] F. Aubin et al. "The MBTA pipeline for detecting compact binary coalescences in the third LIGO–Virgo observing run". In: *Class. Quant. Grav.* 38.9 (2021), p. 095004. DOI: 10.1088/1361-6382/abe913. arXiv: 2012.11512 [gr-qc] (cit. on p. 28).

[163] Alexander H. Nitz et al. "Detecting binary compact-object mergers with gravitational waves: Understanding and Improving the sensitivity of the PyCBC search". In: *Astrophys. J.* 849.2 (2017), p. 118. DOI: 10.3847/1538-4357/aa8f50. arXiv: 1705.01513 [gr-qc] (cit. on pp. 28, 44, 88, 89, 134, 141, 144, 171).

[164] Gareth S. Davies et al. "Extending the PyCBC search for gravitational waves from compact binary mergers to a global network". In: *Phys. Rev. D* 102.2 (2020), p. 022004. DOI: 10.1103/PhysRevD.102.022004. arXiv: 2002.08291 [astro-ph.HE] (cit. on pp. 28, 152, 171).

[165] S. Klimenko and Guenakh Mitselmakher. "A wavelet method for detection of gravitational wave bursts". In: *Class. Quant. Grav.* 21 (2004), S1819–S1830. DOI: 10.1088/0264-9381/21/20/025 (cit. on pp. 28, 175, 188).

[166] S. Klimenko et al. "Localization of gravitational wave sources with networks of advanced detectors". In: *Phys. Rev. D* 83 (2011), p. 102001. DOI: 10.1103/PhysRevD.83.102001. arXiv: 1101.5408 [astro-ph.IM] (cit. on p. 28).

[167] T. Mishra et al. "Search for binary black hole mergers in the third observing run of Advanced LIGO-Virgo using coherent WaveBurst enhanced with machine learning". In: *Phys. Rev. D* 105.8 (2022), p. 083018. DOI: 10.1103/PhysRevD.105.083018. arXiv: 2201.01495 [gr-qc] (cit. on pp. 28, 156, 176).

[168] Amber K. Lenon, Duncan A. Brown, and Alexander H. Nitz. "Eccentric binary neutron star search prospects for Cosmic Explorer". In: *Phys. Rev. D* 104.6 (2021), p. 063011. DOI: 10.1103/PhysRevD.104.063011. arXiv: 2103.14088 [astro-ph.HE] (cit. on pp. 28, 133).

[169] B. P. Abbott et al. "Search for Subsolar Mass Ultracompact Binaries in Advanced LIGO's Second Observing Run". In: *Phys. Rev. Lett.* 123.16 (2019), p. 161102. DOI: 10.1103/PhysRevLett.123.161102. arXiv: 1904.08976 [astro-ph.CO] (cit. on p. 28).

[170] Alexander Harvey Nitz and Yi-Fan Wang. "Search for Gravitational Waves from High-Mass-Ratio Compact-Binary Mergers of Stellar Mass and Subsolar Mass Black Holes". In: *Phys. Rev. Lett.* 126.2 (2021),

p. 021103. DOI: 10.1103/PhysRevLett.126.021103. arXiv: 2007.03583 [astro-ph.HE] (cit. on p. 28).

[171] Alexander H. Nitz and Yi-Fan Wang. "Search for gravitational waves from the coalescence of sub-solar mass and eccentric compact binaries". In: (Feb. 2021). DOI: 10.3847/1538-4357/ac01d9. arXiv: 2102.00868 [astro-ph.HE] (cit. on pp. 28, 133).

[172] Alexander H. Nitz and Yi-Fan Wang. "Search for Gravitational Waves from the Coalescence of Subsolar-Mass Binaries in the First Half of Advanced LIGO and Virgo's Third Observing Run". In: *Phys. Rev. Lett.* 127.15 (2021), p. 151101. DOI: 10.1103/PhysRevLett.127.151101. arXiv: 2106.08979 [astro-ph.HE] (cit. on pp. 28, 133).

[173] Alexander H. Nitz and Yi-Fan Wang. "Broad search for gravitational waves from subsolar-mass binaries through LIGO and Virgo's third observing run". In: *Phys. Rev. D* 106.2 (2022), p. 023024. DOI: 10.1103/PhysRevD.106.023024. arXiv: 2202.11024 [astro-ph.HE] (cit. on pp. 28, 190).

[174] C. Affeldt et al. "Advanced techniques in GEO 600". In: *Class. Quant. Grav.* 31.22 (2014), p. 224002. DOI: 10.1088/0264-9381/31/22/224002 (cit. on p. 28).

[175] K. L. Dooley et al. "GEO 600 and the GEO-HF upgrade program: successes and challenges". In: *Class. Quant. Grav.* 33 (2016), p. 075009. DOI: 10.1088/0264-9381/33/7/075009. arXiv: 1510.00317 [physics.ins-det] (cit. on p. 28).

[176] R. Abbott et al. "First joint observation by the underground gravitational-wave detector KAGRA with GEO 600". In: *PTEP* 2022.6 (2022), 063F01. DOI: 10.1093/ptep/ptac073. arXiv: 2203.01270 [gr-qc] (cit. on p. 28).

[177] GEO600. *GEO600 Advanced Technologies*. 2022. URL: https://www.geo600.org/23441/advanced-technologies (visited on 09/02/2022) (cit. on p. 29).

[178] Wolfgang Demtröder. *Experimentalphysik 2: Elektirzität und Optik*. Springer Spektrum, 2013. ISBN: 978-3-642-29943-8. DOI: 10.1007/978-3-642-29944-5 (cit. on p. 29).

[179] Bernard F. Schutz. "Networks of gravitational wave detectors and three figures of merit". In: *Class. Quant. Grav.* 28 (2011), p. 125023. DOI: 10.1088/0264-9381/28/12/125023. arXiv: 1102.5421 [astro-ph.IM] (cit. on pp. 31, 33, 124).

[180] Patricia Schmidt, Frank Ohme, and Mark Hannam. "Towards models of gravitational waveforms from generic binaries II: Modelling precession effects with a single effective precession parameter". In: *Phys. Rev. D* 91.2 (2015), p. 024043. DOI: 10.1103/PhysRevD.91.024043. arXiv: 1408.1810 [gr-qc] (cit. on pp. 31, 185).

[181] Stephen W Hawking and Werner Israel. *Three hundred years of gravitation*. Cambridge University Press, 1989. ISBN: 0-521-34312-7 (cit. on p. 34).

[182] Katherine L Dooley et al. "Angular control of optical cavities in a radiation pressure dominated regime: the Enhanced LIGO case". In: *J. Opt. Soc. Am. A* 30 (2013), p. 2618. DOI: 10.1364/JOSAA.30.002618. arXiv: 1310.3662 [physics.ins-det] (cit. on p. 32).

[183] Aaron Buikema et al. "Sensitivity and performance of the Advanced LIGO detectors in the third observing run". In: *Phys. Rev. D* 102.6 (2020), p. 062003. DOI: 10.1103/PhysRevD.102.062003. arXiv: 2008.01301 [astro-ph.IM] (cit. on p. 32).

[184] C. M. Caves. "Quantum Mechanical Noise in an Interferometer". In: *Phys. Rev. D* 23 (1981), pp. 1693–1708. DOI: 10.1103/PhysRevD.23.1693 (cit. on p. 35).

[185] J. Aasi et al. "Enhancing the sensitivity of the LIGO gravitational wave detector by using squeezed states of light". In: *Nature Photon.* 7 (2013), pp. 613–619. DOI: 10.1038/nphoton.2013.177. arXiv: 1310.0383 [quant-ph] (cit. on p. 35).

[186] James Lough et al. "First Demonstration of 6 dB Quantum Noise Reduction in a Kilometer Scale Gravitational Wave Observatory". In: *Phys. Rev. Lett.* 126.4 (2021), p. 041102. DOI: 10.1103/PhysRevLett.126.041102. arXiv: 2005.10292 [physics.ins-det] (cit. on p. 35).

[187] P. Welch. "The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms". In: *IEEE Transactions on Audio and Electroacoustics* 15.2 (1967), pp. 70–73. DOI: 10.1109/TAU.1967.1161901 (cit. on pp. 36, 173).

[188] Bruce Allen et al. "Observational limit on gravitational waves from binary neutron stars in the galaxy". In: *Phys. Rev. Lett.* 83 (1999), p. 1498. DOI: 10.1103/PhysRevLett.83.1498. arXiv: gr-qc/9903108 (cit. on p. 36).

[189] Bruce Allen. "χ^2 time-frequency discriminator for gravitational wave detection". In: *Phys. Rev. D* 71 (2005), p. 062001. DOI: 10.1103/PhysRevD.71.062001. arXiv: gr-qc/0405045 (cit. on pp. 36, 44, 171).

[190] Michael Zevin et al. "Gravity Spy: Integrating Advanced LIGO Detector Characterization, Machine Learning, and Citizen Science". In: *Class. Quant. Grav.* 34.6 (2017), p. 064003. DOI: 10.1088/1361-6382/aa5cea. arXiv: 1611.04596 [gr-qc] (cit. on pp. 36, 76, 79, 133, 156, 216).

[191] Florent Robinet et al. "Omicron: a tool to characterize transient noise in gravitational-wave detectors". In: *SoftwareX* 12 (2020), p. 100620. DOI: 10.1016/j.softx.2020.100620. arXiv: 2007.11374 [astro-ph.IM] (cit. on pp. 36, 76).

[192] Bala Iyer et al. *LIGO-India, Proposal of the Consortium for Indian Initiative in Gravitational-wave Observations (IndIGO)*. 2011. URL: https://dcc.ligo.org/LIGO-M1100296/public (visited on 08/27/2022) (cit. on pp. 37, 94).

[193] M. Saleem et al. "The science case for LIGO-India". In: *Class. Quant. Grav.* 39.2 (2022), p. 025004. DOI: 10.1088/1361-6382/ac3b99. arXiv: 2105.01716 [gr-qc] (cit. on p. 37).

[194] B. P. Abbott et al. "Prospects for observing and localizing gravitational-wave transients with Advanced LIGO, Advanced Virgo and KAGRA". In: *Living Rev. Rel.* 21.1 (2018), p. 3. DOI: 10.1007/s41114-020-00026-9. arXiv: 1304.0670 [gr-qc] (cit. on pp. 37, 75, 94, 155).

[195] European Space Agency. *LISA*. 2022. URL: https://sci.esa.int/web/lisa (visited on 08/28/2022) (cit. on p. 37).

[196] Steve Sabia, Pat Tyler, and James Ira Thorpe. *LISA*. 2022. URL: https://lisa.nasa.gov/ (visited on 08/28/2022) (cit. on p. 37).

[197] P. McNamara, S. Vitale, and K. Danzmann. "LISA Pathfinder". In: *Class. Quant. Grav.* 25 (2008). Ed. by Susan M. Scott and David E. McClelland, p. 114034. DOI: 10.1088/0264-9381/25/11/114034 (cit. on p. 37).

[198] M. Armano et al. "Sub-Femto- g Free Fall for Space-Based Gravitational Wave Observatories: LISA Pathfinder Results". In: *Phys. Rev. Lett.* 116.23 (2016), p. 231101. DOI: 10.1103/PhysRevLett.116.231101 (cit. on p. 37).

[199] M. Armano et al. "Beyond the Required LISA Free-Fall Performance: New LISA Pathfinder Results down to 20 μHz". In: *Phys. Rev. Lett.* 120.6 (2018), p. 061101. DOI: 10.1103/PhysRevLett.120.061101 (cit. on p. 37).

[200] Jun Luo et al. "TianQin: a space-borne gravitational wave detector". In: *Class. Quant. Grav.* 33.3 (2016), p. 035010. DOI: 10.1088/0264-9381/33/3/035010. arXiv: 1512.02076 [astro-ph.IM] (cit. on p. 37).

[201] G. B. Hobbs et al. "Gravitational wave detection using pulsars: status of the Parkes Pulsar Timing Array project". In: *Publ. Astron. Soc. Austral.* 26 (2009), p. 103. DOI: 10.1071/AS08023. arXiv: 0812.2721 [astro-ph] (cit. on p. 37).

[202] P. Demorest, J. Lazio, and A. Lommen. "Gravitational Wave Astronomy Using Pulsars: Massive Black Hole Mergers & the Early Universe". In: (Feb. 2009). arXiv: 0902.2968 [astro-ph.CO] (cit. on p. 37).

[203] Md F. Alam et al. "The NANOGrav 12.5 yr Data Set: Observations and Narrowband Timing of 47 Millisecond Pulsars". In: *Astrophys. J. Suppl.* 252.1 (2021), p. 4. DOI: 10.3847/1538-4365/abc6a0. arXiv: 2005.06490 [astro-ph.HE] (cit. on p. 37).

[204] Zaven Arzoumanian et al. "The NANOGrav 12.5 yr Data Set: Search for an Isotropic Stochastic Gravitational-wave Background". In: *Astrophys. J. Lett.* 905.2 (2020), p. L34. DOI: 10.3847/2041-8213/abd401. arXiv: 2009.04496 [astro-ph.HE] (cit. on p. 37).

[205] Sumit Kumar, Alexander H. Nitz, and Xisco Jiménez Forteza. "Parameter estimation with non stationary noise in gravitational waves data". In: (Feb. 2022). arXiv: 2202.12762 [astro-ph.IM] (cit. on p. 38).

[206] Alexander H. Nitz et al. "3-OGC: Catalog of Gravitational Waves from Compact-binary Mergers". In: *Astrophys. J.* 922.1 (2021), p. 76. DOI: 10.3847/1538-4357/ac1c03. arXiv: 2105.09151 [astro-ph.HE] (cit. on pp. 41, 42, 102, 109, 113, 132, 155, 156, 271).

[207] Frank Ohme. "Bridging the Gap between Post-Newtonian Theory and Numerical Relativity in Gravitational-Wave Data Analysis". PhD thesis. Potsdam U., 2012 (cit. on p. 42).

[208] Ian W. Harry, Bruce Allen, and B. S. Sathyaprakash. "A Stochastic template placement algorithm for gravitational wave data analysis". In: *Phys. Rev. D* 80 (2009), p. 104014. DOI: 10.1103/PhysRevD.80.104014. arXiv: 0908.2090 [gr-qc] (cit. on p. 42).

[209] Alexander Harvey Nitz. "Distinguishing short duration noise transients in LIGO data to improve the PyCBC search for gravitational waves from high mass binary black hole mergers". In: *Class. Quant. Grav.* 35.3 (2018), p. 035016. DOI: 10.1088/1361-6382/aaa13d. arXiv: 1709.08974 [gr-qc] (cit. on pp. 42, 171).

[210] C. M. Biwer et al. "PyCBC Inference: A Python-based parameter estimation toolkit for compact binary coalescence signals". In: *Publ. Astron. Soc. Pac.* 131.996 (2019), p. 024503. DOI: 10.1088/1538-3873/aaef0b. arXiv: 1807.10312 [astro-ph.IM] (cit. on p. 43).

[211] Derek Davis et al. "LIGO detector characterization in the second and third observing runs". In: *Class. Quant. Grav.* 38.13 (2021), p. 135014. DOI: 10.1088/1361-6382/abfd85. arXiv: 2101.11673 [astro-ph.IM] (cit. on pp. 43, 44).

[212] Alexander H. Nitz et al. "2-OGC: Open Gravitational-wave Catalog of binary mergers from analysis of public Advanced LIGO and Virgo data". In: *Astrophys. J.* 891 (Mar. 2020), p. 123. DOI: 10.3847/1538-4357/ab733f. arXiv: 1910.05331 [astro-ph.HE] (cit. on pp. 44, 95, 155).

[213] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org (cit. on pp. 46, 47, 48, 50, 52, 53, 54, 55, 56, 57, 58, 59, 61, 62, 63, 64, 65, 66, 67, 68, 70, 72, 73, 75, 84, 162, 177).

[214] Nils J. Nilsson. *The Quest for Artificial Intelligence*. Cambridge University Press, Oct. 2009. ISBN: 9780521122931. URL: http://www.cambridge.org/us/0521122937 (cit. on p. 46).

[215] D. B. Lenat and R. V. Guha. *Building large knowledge-based systems; representation and inference in the Cyc project*. Addison-Wesley Longman Publishing Co., Inc., 1989 (cit. on p. 47).

[216] Tin Kam Ho. "Random decision forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1. 1995, 278–282 vol.1. DOI: 10.1109/ICDAR.1995.598994 (cit. on p. 47).

[217] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 1992, pp. 144–152. ISBN: 089791497X. DOI: 10.1145/130385.130401. URL: https://doi.org/10.1145/130385.130401 (cit. on p. 47).

[218] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media, Inc., 2019. ISBN: 978-1-492-03264-9 (cit. on pp. 47, 48, 49, 50, 52, 55, 56, 59, 62, 64, 65, 70, 75).

[219] Zoubin Ghahramani. "Unsupervised Learning". In: *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*. Ed. by Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 72–112. ISBN: 978-3-540-28650-9. DOI: 10.1007/978-3-540-28650-9_5. URL: https://doi.org/10.1007/978-3-540-28650-9_5 (cit. on p. 47).

[220] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. ISBN: 9780262039246. URL: https://mitpress.mit.edu/books/reinforcement-learning-second-edition (cit. on p. 47).

[221] Chelsea Finn et al. "Deep Spatial Autoencoders for Visuomotor Learning". In: (Sept. 2015). arXiv: 1509.06113 (cit. on p. 48).

[222] Ahmad EL Sallab et al. "Deep Reinforcement Learning framework for Autonomous Driving". In: *Electronic Imaging* 2017.19 (Jan. 2017), pp. 70–76. ISSN: 2470-1173. DOI: doi:10.2352/ISSN.2470-1173.2017.19.AVM-023. URL: https://www.ingentaconnect.com/content/ist/ei/2017/00002017/00000019/art00012 (cit. on p. 48).

[223] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. ISSN: 1476-4687. DOI: 10.1038/nature14236. URL: https://doi.org/10.1038/nature14236 (cit. on p. 48).

[224] Harald Steck et al. "Deep Learning for Recommender Systems: A Netflix Case Study". In: *AI Magazine* 42.3 (Nov. 2021), pp. 7–18. DOI: 10.1609/aimag.v42i3.18140. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/18140 (cit. on p. 48).

[225] Andre Esteva et al. “A guide to deep learning in healthcare”. In: *Nature Medicine* 25.1 (Jan. 2019), pp. 24–29. ISSN: 1546-170X. DOI: 10.1038/s41591-018-0316-z. URL: https://doi.org/10.1038/s41591-018-0316-z (cit. on p. 48).

[226] Christian Szegedy et al. “Going Deeper With Convolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015 (cit. on p. 48).

[227] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y (cit. on pp. 48, 73).

[228] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587 (Jan. 2016), pp. 484–489. ISSN: 1476-4687. DOI: 10.1038/nature16961. URL: https://doi.org/10.1038/nature16961 (cit. on p. 48).

[229] Geoffrey Hinton et al. “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 82–97. DOI: 10.1109/MSP.2012.2205597 (cit. on p. 48).

[230] Aaron van den Oord et al. *WaveNet: A Generative Model for Raw Audio*. 2016. DOI: 10.48550/ARXIV.1609.03499. URL: https://arxiv.org/abs/1609.03499 (cit. on pp. 48, 70).

[231] Allison McCarn Deiana et al. “Applications and Techniques for Fast Machine Learning in Science”. In: (Oct. 2021). arXiv: 2110.13041 [cs.LG] (cit. on pp. 48, 70, 156).

[232] Mohammed AlQuraishi. “Machine learning in protein structure prediction”. In: *Current Opinion in Chemical Biology* 65 (2021). Mechanistic Biology * Machine Learning in Chemical Biology, pp. 1–8. ISSN: 1367-5931. DOI: https://doi.org/10.1016/j.cbpa.2021.04.005. URL: https://www.sciencedirect.com/science/article/pii/S1367593121000508 (cit. on pp. 48, 156).

[233] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596.7873 (Aug. 2021), pp. 583–589. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03819-2. URL: https://doi.org/10.1038/s41586-021-03819-2 (cit. on p. 48).

[234] Keith T. Butler et al. “Machine learning for molecular and materials science”. In: *Nature* 559.7715 (July 2018), pp. 547–555. ISSN: 1476-4687. DOI: 10.1038/s41586-018-0337-2. URL: https://doi.org/10.1038/s41586-018-0337-2 (cit. on pp. 48, 156).

[235] Lindsey Gray, Thomas Klijnsma, and Shamik Ghosh. “A Dynamic Reduction Network for Point Clouds”. In: (Mar. 2020). arXiv: 2003.08013 [cs.CV] (cit. on pp. 48, 156).

[236] Michael Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL: http://neuralnetworksanddeeplearning.com/index.html (cit. on p. 48).

[237] Warren S. McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259 (cit. on pp. 48, 50, 52).

[238] Gualtiero Piccinini. “The First Computational Theory of Mind and Brain: A Close Look at Mcculloch and Pitts’s “Logical Calculus of Ideas Immanent in Nervous Activity””. In: *Synthese* 141.2 (Aug. 2004), pp. 175–215. ISSN: 1573-0964. DOI: 10.1023/B:SYNT.0000043018.52445.3e. URL: https://doi.org/10.1023/B:SYNT.0000043018.52445.3e (cit. on pp. 48, 52).

[239] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386. DOI: https://doi.org/10.1037/h0042519 (cit. on p. 49).

[240] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385 (cit. on pp. 50, 64, 73, 74, 75, 78, 174, 199, 202, 205, 206).

[241] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985 (cit. on pp. 50, 51, 59).

[242] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*. 2015. DOI: 10.48550/ARXIV.1511.07289. URL: https://arxiv.org/abs/1511.07289 (cit. on p. 51).

[243] Kunihiko Fukushima. "Cognitron: A self-organizing multilayered neural network". In: *Biological Cybernetics* 20.3 (Sept. 1975), pp. 121–136. ISSN: 1432-0770. DOI: 10.1007/BF00342633. URL: https://doi.org/10.1007/BF00342633 (cit. on p. 51).

[244] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *ICML*. 2010, pp. 807–814. URL: https://icml.cc/Conferences/2010/papers/432.pdf (cit. on p. 51).

[245] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0 (cit. on p. 52).

[246] Jeffrey L. Elman. "Finding Structure in Time". In: *Cognitive Science* 14.2 (1990), pp. 179–211. DOI: https://doi.org/10.1207/s15516709cog1402_1. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1207/s15516709cog1402_1. URL: https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog1402_1 (cit. on p. 52).

[247] Wenpeng Yin et al. *Comparative Study of CNN and RNN for Natural Language Processing*. 2017. DOI: 10.48550/ARXIV.1702.01923. URL: https://arxiv.org/abs/1702.01923 (cit. on pp. 52, 70).

[248] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (cit. on p. 52).

[249] Lewis Tunstall, Leandro von Werra, and Thomas Wolf. *Natural Language Processing with Transformers*. O'Reilly Media, Inc., 2022. URL: https://transformersbook.com/ (cit. on p. 52).

[250] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208 (cit. on p. 53).

[251] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks". In: *Neural Networks* 3.5 (1990), pp. 551–560. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(90)90005-6. URL: https://www.sciencedirect.com/science/article/pii/0893608090900056 (cit. on p. 53).

[252] David H. Wolpert. "The Lack of A Priori Distinctions Between Learning Algorithms". In: *Neural Computation* 8.7 (Oct. 1996), pp. 1341–1390. ISSN: 0899-7667. DOI: 10.1162/neco.1996.8.7.1341. eprint: https://direct.mit.edu/neco/article-pdf/8/7/1341/813495/neco.1996.8.7.1341.pdf. URL: https://doi.org/10.1162/neco.1996.8.7.1341 (cit. on p. 54).

[253] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980 (cit. on pp. 61, 64, 65, 108, 137, 211).

[254] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. URL: https://proceedings.mlr.press/v9/glorot10a.html (cit. on p. 64).

[255] T. Tieleman and G. Hinton. *COURSERA: Neural Networks for Machine Learning*. Slide 29 in Lecture 6.5. 2012 (cit. on p. 64).

[256] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12.61 (2011), pp. 2121–2159. URL: http://jmlr.org/papers/v12/duchi11a.html (cit. on p. 64).

[257] Yann LeCun. "Generalization and network design strategies". In: *Connectionism in perspective* 19.143-155 (1989), p. 18 (cit. on pp. 65, 70).

[258] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *Computer Vision – ECCV 2014*. Cham: Springer International Publishing, 2014, pp. 818–833. ISBN: 978-3-319-10590-1 (cit. on p. 67).

[259] Y. LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791 (cit. on p. 70).

[260] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. DOI: 10.48550/ARXIV.1409.1556. URL: https://arxiv.org/abs/1409.1556 (cit. on pp. 70, 197).

[261] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. DOI: 10.48550/ARXIV.1704.04861. URL: https://arxiv.org/abs/1704.04861 (cit. on p. 70).

[262] Omar Elharrouss et al. *Backbones-Review: Feature Extraction Networks for Deep Learning and Deep Reinforcement Learning Approaches*. 2022. DOI: 10.48550/ARXIV.2206.08016. URL: https://arxiv.org/abs/2206.08016 (cit. on pp. 70, 74).

[263] Ross Girshick et al. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2014. DOI: 10.48550/ARXIV.1311.2524. URL: https://arxiv.org/abs/1311.2524 (cit. on pp. 70, 197).

[264] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2015. DOI: 10.48550/ARXIV.1506.02640. URL: https://arxiv.org/abs/1506.02640 (cit. on pp. 70, 199).

[265] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, Sept. 2016, pp. 21–37. ISBN: 978-3-319-46448-0. DOI: https://doi.org/10.1007/978-3-319-46448-0_2 (cit. on pp. 70, 199).

[266] Marc'aurelio Ranzato, Y-lan Boureau, and Yann Cun. "Sparse Feature Learning for Deep Belief Networks". In: *Advances in Neural Information Processing Systems*. Ed. by J. Platt et al. Vol. 20. Curran Associates, Inc., 2007. URL: https://proceedings.neurips.cc/paper/2007/file/c60d060b946d6dd6145dcbad5c4ccf6f-Paper.pdf (cit. on p. 70).

[267] Hossein Gholamalinezhad and Hossein Khosravi. *Pooling Methods in Deep Neural Networks, a Review*. 2020. DOI: 10.48550/ARXIV.2009.07485. URL: https://arxiv.org/abs/2009.07485 (cit. on p. 70).

[268] Ross Girshick. "Fast R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015 (cit. on pp. 70, 197, 198, 203).

[269] Kaiming He et al. "Mask R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017 (cit. on pp. 70, 202, 203, 205).

[270] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html (cit. on p. 71).

[271] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *arXiv e-prints*, arXiv:1502.03167 (Feb. 2015), arXiv:1502.03167. arXiv: 1502.03167 [cs.LG] (cit. on pp. 72, 73, 105, 172).

[272] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2017. DOI: 10.48550/ARXIV.1708.02002. URL: https://arxiv.org/abs/1708.02002 (cit. on pp. 74, 199, 202, 206).

[273] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. "DetectoRS: Detecting Objects With Recursive Feature Pyramid and Switchable Atrous Convolution". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 10213–10224 (cit. on pp. 74, 199, 205).

[274] Christian Szegedy et al. *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. 2016. DOI: 10.48550/ARXIV.1602.07261. URL: https://arxiv.org/abs/1602.07261 (cit. on p. 74).

[275] Craig Cahillane and Georgia Mansell. "Review of the Advanced LIGO Gravitational Wave Observatories Leading to Observing Run Four". In: *Galaxies* 10.1 (2022), p. 36. DOI: 10.3390/galaxies10010036. arXiv: 2202.00847 [gr-qc] (cit. on pp. 75, 155).

[276] S. Mukherjee, R. Obaid, and B. Matkarimov. "Classification of glitch waveforms in gravitational wave detector characterization". In: *J. Phys. Conf. Ser.* 243 (2010). Ed. by Fulvio Ricci, p. 012006. DOI: 10.1088/1742-6596/243/1/012006 (cit. on p. 75).

[277] Rahul Biswas et al. "Application of machine learning algorithms to the study of noise artifacts in gravitational-wave data". In: *Phys. Rev. D* 88.6 (2013), p. 062003. DOI: 10.1103/PhysRevD.88.062003. arXiv: 1303.6984 [astro-ph.IM] (cit. on p. 75).

[278] Jade Powell et al. "Classification methods for noise transients in advanced gravitational-wave detectors". In: *Class. Quant. Grav.* 32.21 (2015), p. 215012. DOI: 10.1088/0264-9381/32/21/215012. arXiv: 1505.01299 [astro-ph.IM] (cit. on p. 75).

[279] Jade Powell et al. "Classification methods for noise transients in advanced gravitational-wave detectors II: performance tests on Advanced LIGO data". In: *Class. Quant. Grav.* 34.3 (2017), p. 034002. DOI: 10.1088/1361-6382/34/3/034002. arXiv: 1609.06262 [astro-ph.IM] (cit. on p. 75).

[280] A Effler et al. "Environmental influences on the LIGO gravitational wave detectors during the 6th science run". In: *Classical and Quantum Gravity* 32.3 (Jan. 2015), p. 035017. DOI: 10.1088/0264-9381/32/3/035017. URL: https://doi.org/10.1088/0264-9381/32/3/035017 (cit. on p. 76).

[281] S. Bahaadini et al. "Machine learning for Gravity Spy: Glitch classification and dataset". In: *Info. Sci.* 444 (2018), pp. 172–186. DOI: 10.1016/j.ins.2018.02.068 (cit. on pp. 76, 156).

[282] Reed Essick et al. "iDQ: Statistical Inference of Non-Gaussian Noise with Auxiliary Degrees of Freedom in Gravitational-Wave Detectors". In: (May 2020). arXiv: 2005.12761 [astro-ph.IM] (cit. on pp. 76, 133).

[283] Nikhil Mukund et al. "Transient Classification in LIGO data using Difference Boosting Neural Network". In: *Phys. Rev. D* 95.10 (2017), p. 104059. DOI: 10.1103/PhysRevD.95.104059. arXiv: 1609.07259 [astro-ph.IM] (cit. on p. 76).

[284] Marco Cavaglia, Kai Staats, and Teerth Gill. "Finding the origin of noise transients in LIGO data with machine learning". In: *Commun. Comput. Phys.* 25.4 (2019), pp. 963–987. DOI: 10.4208/cicp.OA-2018-0092. arXiv: 1812.05225 [physics.data-an] (cit. on p. 76).

[285] S. B. Coughlin et al. "Classifying the unknown: discovering novel gravitational-wave detector glitches using similarity learning". In: *Phys. Rev. D* 99.8 (2019), p. 082002. DOI: 10.1103/PhysRevD.99.082002. arXiv: 1903.04058 [astro-ph.IM] (cit. on p. 76).

[286] Shasvath J. Kapadia, Thomas Dent, and Tito Dal Canton. "Classifier for gravitational-wave inspiral signals in nonideal single-detector data". In: *Phys. Rev. D* 96.10 (2017), p. 104015. DOI: 10.1103/PhysRevD.96.104015. arXiv: 1709.02421 [astro-ph.IM] (cit. on p. 76).

[287] Plamen G. Krastev. "Real-Time Detection of Gravitational Waves from Binary Neutron Stars using Artificial Neural Networks". In: *Phys. Lett. B* 803 (2020), p. 135330. DOI: 10.1016/j.physletb.2020.135330. arXiv: 1908.03151 [astro-ph.IM] (cit. on pp. 77, 80, 89, 90, 103, 113, 128, 156).

[288] Plamen G. Krastev et al. "Detection and Parameter Estimation of Gravitational Waves from Binary Neutron-Star Mergers in Real LIGO Data using Deep Learning". In: *Phys. Lett. B* 815 (2021), p. 136161. DOI: 10.1016/j.physletb.2021.136161. arXiv: 2012.13101 [astro-ph.IM] (cit. on pp. 77, 133, 156, 163, 177, 184).

[289] Timothy D. Gebhard et al. "Convolutional neural networks: a magic bullet for gravitational-wave detection?" In: *Phys. Rev. D* 100.6 (2019), p. 063015. DOI: 10.1103/PhysRevD.100.063015. arXiv: 1904.08693 [astro-ph.IM] (cit. on pp. 77, 91, 103, 156, 163, 177, 184, 190).

[290] He Wang et al. "Gravitational-wave signal recognition of LIGO data by deep learning". In: *Phys. Rev. D* 101.10 (2020), p. 104003. DOI: 10.1103/PhysRevD.101.104003. arXiv: 1909.13442 [astro-ph.IM] (cit. on pp. 78, 170, 171).

[291] Wei Wei and E. A. Huerta. "Deep learning for gravitational wave forecasting of neutron star mergers". In: *Phys. Lett. B* 816 (2021), p. 136185. DOI: 10.1016/j.physletb.2021.136185. arXiv: 2010.09751 [gr-qc] (cit. on pp. 78, 107, 128, 133, 156, 184, 190).

[292] Glenn Jocher et al. *ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations*. Version v6.2. Aug. 2022. DOI: 10.5281/zenodo.7002879. URL: https://doi.org/10.5281/zenodo.7002879 (cit. on pp. 78, 199).

[293] Gregory Baltus et al. "Convolutional neural networks for the detection of the early inspiral of a gravitational-wave signal". In: *Phys. Rev. D* 103 (2021), p. 102003. DOI: 10.1103/PhysRevD.103.102003. arXiv: 2104.00594 [gr-qc] (cit. on p. 79).

[294] Grégory Baltus et al. "Detecting the early inspiral of a gravitational-wave signal with convolutional neural networks". In: May 2021. DOI: 10.1109/CBMI50038.2021.9461919. arXiv: 2105.13664 [gr-qc] (cit. on p. 79).

[295] Hang Yu et al. "Early warning of coalescing neutron-star and neutron-star-black-hole binaries from the nonstationary noise background using neural networks". In: *Phys. Rev. D* 104.6 (2021), p. 062004. DOI:

10.1103/PhysRevD.104.062004. arXiv: 2104.09438 [gr-qc] (cit. on p. 79).

[296] Shreejit Jadhav et al. "Improving significance of binary black hole mergers in Advanced LIGO data using deep learning: Confirmation of GW151216". In: *Phys. Rev. D* 104.6 (2021), p. 064051. DOI: 10.1103/PhysRevD.104.064051. arXiv: 2010.08584 [gr-qc] (cit. on pp. 79, 156).

[297] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2818–2826. DOI: 10.1109/CVPR.2016.308 (cit. on p. 79).

[298] Connor McIsaac and Ian Harry. "Using machine learning to autotune chi-squared tests for gravitational wave searches". In: *Phys. Rev. D* 105.10 (2022), p. 104056. DOI: 10.1103/PhysRevD.105.104056. arXiv: 2203.03449 [gr-qc] (cit. on pp. 79, 156).

[299] Sunil Choudhary et al. "SiGMa-Net: Deep learning network to distinguish binary black hole signals from short-duration noise transients". In: (Feb. 2022). arXiv: 2202.08671 [gr-qc] (cit. on p. 79).

[300] Tanmaya Mishra et al. "Optimization of model independent gravitational wave search for binary black hole mergers using machine learning". In: *Phys. Rev. D* 104.2 (2021), p. 023014. DOI: 10.1103/PhysRevD.104.023014. arXiv: 2105.04739 [gr-qc] (cit. on pp. 79, 80, 156, 175, 216).

[301] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785 (cit. on pp. 79, 175).

[302] Christoph Dreissigacker et al. "Deep-Learning Continuous Gravitational Waves". In: *Phys. Rev. D* 100.4 (2019), p. 044009. DOI: 10.1103/PhysRevD.100.044009. arXiv: 1904.13291 [gr-qc] (cit. on pp. 80, 103, 107, 156).

[303] Christoph Dreissigacker and Reinhard Prix. "Deep-Learning Continuous Gravitational Waves: Multiple detectors and realistic noise". In: *Phys. Rev. D* 102.2 (2020), p. 022005. DOI: 10.1103/PhysRevD.102.022005. arXiv: 2005.04140 [gr-qc] (cit. on pp. 80, 128, 133, 156, 162, 190).

[304] Banafsheh Beheshtipour and Maria Alessandra Papa. "Deep learning for clustering of continuous gravitational wave candidates". In: *Phys. Rev. D* 101.6 (2020), p. 064009. DOI: 10.1103/PhysRevD.101.064009. arXiv: 2001.03116 [gr-qc] (cit. on pp. 80, 156).

[305] B. Beheshtipour and M. A. Papa. "Deep learning for clustering of continuous gravitational wave candidates II: identification of low-SNR candidates". In: *Phys. Rev. D* 103.6 (2021), p. 064027. DOI: 10.1103/PhysRevD.103.064027. arXiv: 2012.04381 [gr-qc] (cit. on pp. 80, 156).

[306] Xue-Ting Zhang et al. "Detecting gravitational waves from extreme mass ratio inspirals using convolutional neural networks". In: *Phys. Rev. D* 105.12 (2022), p. 123027. DOI: 10.1103/PhysRevD.105.123027. arXiv: 2202.07158 [astro-ph.HE] (cit. on p. 80).

[307] Filip Morawski et al. "Anomaly detection in gravitational waves data using convolutional autoencoders". In: *Mach. Learn. Sci. Tech.* 2.4 (2021), p. 045014. DOI: 10.1088/2632-2153/abf3d0. arXiv: 2103.07688 [astro-ph.IM] (cit. on pp. 80, 156, 190).

[308] Eric A. Moreno et al. "Source-agnostic gravitational-wave detection with recurrent autoencoders". In: *Mach. Learn. Sci. Tech.* 3.2 (2022), p. 025001. DOI: 10.1088/2632-2153/ac5435. arXiv: 2107.12698 [gr-qc] (cit. on pp. 80, 156, 190).

[309] Miriam Cabero, Ashish Mahabal, and Jess McIver. "GWSkyNet: a real-time classifier for public gravitational-wave candidates". In: *Astrophys. J. Lett.* 904.1 (2020), p. L9. DOI: 10.3847/2041-8213/abc5b5. arXiv: 2010.11829 [gr-qc] (cit. on pp. 80, 156).

[310] Thomas C. Abbott et al. "GWSkyNet-Multi: A Machine-learning Multiclass Classifier for LIGO–Virgo Public Alerts". In: *Astrophys. J.* 927.2 (2022), p. 232. DOI: 10.3847/1538-4357/ac5019. arXiv: 2111.04015 [astro-ph.IM] (cit. on p. 80).

[311] Alvin J. K. Chua and Michele Vallisneri. "Learning Bayesian posteriors with neural networks for gravitational-wave inference". In: *Phys. Rev. Lett.* 124.4 (2020), p. 041102. DOI: 10.1103/PhysRevLett.124.041102. arXiv: 1909.05966 [gr-qc] (cit. on pp. 80, 104, 156).

[312] Hunter Gabbard et al. "Bayesian parameter estimation using conditional variational autoencoders for gravitational-wave astronomy". In: *Nature Phys.* 18.1 (2022), pp. 112–117. DOI: 10.1038/s41567-021-01425-7. arXiv: 1909.06296 [astro-ph.IM] (cit. on pp. 80, 104, 113, 146, 156, 162, 217).

[313] Stephen R. Green, Christine Simpson, and Jonathan Gair. "Gravitational-wave parameter estimation with autoregressive neural network flows". In: *Phys. Rev. D* 102.10 (2020), p. 104057. DOI: 10.1103/PhysRevD.102.104057. arXiv: 2002.07656 [astro-ph.IM] (cit. on pp. 81, 104).

[314] Stephen R. Green and Jonathan Gair. "Complete parameter inference for GW150914 using deep learning". In: *Mach. Learn. Sci. Tech.* 2.3 (2021), 03LT01. DOI: 10.1088/2632-2153/abfaed. arXiv: 2008.03312 [astro-ph.IM] (cit. on p. 81).

[315] Maximilian Dax et al. "Real-Time Gravitational Wave Science with Neural Posterior Estimation". In: *Phys. Rev. Lett.* 127.24 (2021), p. 241103. DOI: 10.1103/PhysRevLett.127.241103. arXiv: 2106.12594 [gr-qc] (cit. on pp. 81, 156, 217).

[316] Chayan Chatterjee et al. "Rapid localization of gravitational wave sources from compact binary coalescences using deep learning". In: (July 2022). arXiv: 2207.14522 [gr-qc] (cit. on p. 81).

[317] Michael J. Williams, John Veitch, and Chris Messenger. "Nested sampling with normalizing flows for gravitational-wave inference". In: *Phys. Rev. D* 103.10 (2021), p. 103006. DOI: 10.1103/PhysRevD.103.103006. arXiv: 2102.11056 [gr-qc] (cit. on pp. 81, 104, 156).

[318] Joshua S Speagle. "dynesty: a dynamic nested sampling package for estimating Bayesian posteriors and evidences". In: *Monthly Notices of the Royal Astronomical Society* 493.3 (Feb. 2020), pp. 3132–3158. ISSN: 0035-8711. DOI: 10.1093/mnras/staa278. eprint: https://academic.oup.com/mnras/article-pdf/493/3/3132/32890730/staa278.pdf. URL: https://doi.org/10.1093/mnras/staa278 (cit. on p. 81).

[319] LIGO Scientific Collaboration. *LIGO Algorithm Library - LALSuite*. free software (GPL). 2018. DOI: 10.7935/GT1W-FZ16 (cit. on pp. 87, 107, 136, 137, 140, 161).

[320] Serge Droz et al. "Gravitational waves from inspiraling compact binaries: Validity of the stationary phase approximation to the Fourier transform". In: *Phys. Rev. D* 59 (1999), p. 124016. DOI: 10.1103/PhysRevD.59.124016. arXiv: gr-qc/9901076 (cit. on p. 87).

[321] Guillaume Faye et al. "The third and a half post-Newtonian gravitational wave quadrupole mode for quasi-circular inspiralling compact binaries". In: *Class. Quant. Grav.* 29 (2012), p. 175004. DOI: 10.1088/0264-9381/29/17/175004. arXiv: 1204.1043 [gr-qc] (cit. on p. 87).

[322] Plamen G. Krastev. "Real-Time Detection of Gravitational Waves from Binary Neutron Stars using Artificial Neural Networks". In: (2019). arXiv: 1908.03151v1. URL: https://arxiv.org/abs/1908.03151v1 (cit. on pp. 89, 90).

[323] A. Goldstein et al. "An Ordinary Short Gamma-Ray Burst with Extraordinary Implications: Fermi-GBM Detection of GRB 170817A". In: *Astrophys. J. Lett.* 848.2 (2017), p. L14. DOI: 10.3847/2041-8213/aa8f41. arXiv: 1710.05446 [astro-ph.HE] (cit. on pp. 93, 102).

[324] V. Savchenko et al. "INTEGRAL Detection of the First Prompt Gamma-Ray Signal Coincident with the Gravitational-wave Event GW170817". In: *Astrophys. J. Lett.* 848.2 (2017), p. L15. DOI: 10.3847/2041-8213/aa8f94. arXiv: 1710.05449 [astro-ph.HE] (cit. on pp. 93, 102).

[325] B.P. Abbott et al. "Gravitational Waves and Gamma-rays from a Binary Neutron Star Merger: GW170817 and GRB 170817A". In: *Astrophys. J. Lett.* 848.2 (2017), p. L13. DOI: 10.3847/2041-8213/aa920c. arXiv: 1710.05834 [astro-ph.HE] (cit. on p. 93).

[326] B. P. Abbott et al. "Properties of the binary neutron star merger GW170817". In: *Phys. Rev. X* 9.1 (2019), p. 011001. DOI: 10.1103/PhysRevX.9.011001. arXiv: 1805.11579 [gr-qc] (cit. on p. 93).

[327] David Radice et al. "GW170817: Joint Constraint on the Neutron Star Equation of State from Multimessenger Observations". In: *Astrophys. J. Lett.* 852.2 (2018), p. L29. DOI: 10.3847/2041-8213/aaa402. arXiv: 1711.03647 [astro-ph.HE] (cit. on p. 93).

[328] Kenta Kiuchi et al. "Revisiting the lower bound on tidal deformability derived by AT 2017gfo". In: *Astrophys. J. Lett.* 876.2 (2019), p. L31. DOI: 10.3847/2041-8213/ab1e45. arXiv: 1903.01466 [astro-ph.HE] (cit. on p. 93).

[329] Benjamin P Abbott et al. "Model comparison from LIGO–Virgo data on GW170817's binary components and consequences for the merger remnant". In: *Class. Quant. Grav.* 37.4 (2020), p. 045006. DOI: 10.1088/1361-6382/ab5f7c. arXiv: 1908.01012 [gr-qc] (cit. on p. 93).

[330] C. Guidorzi et al. "Improved Constraints on H_0 from a Combined Analysis of Gravitational-wave and Electromagnetic Emission from GW170817". In: *Astrophys. J. Lett.* 851.2 (2017), p. L36. DOI: 10.3847/2041-8213/aaa009. arXiv: 1710.06426 [astro-ph.CO] (cit. on p. 93).

[331] Kenta Hotokezaka et al. "A Hubble constant measurement from superluminal motion of the jet in GW170817". In: *Nature Astron.* 3.10 (2019), pp. 940–944. DOI: 10.1038/s41550-019-0820-1. arXiv: 1806.10596 [astro-ph.CO] (cit. on p. 93).

[332] Brian D. Metzger. "Kilonovae". In: *Living Rev. Rel.* 23.1 (2020), p. 1. DOI: 10.1007/s41114-019-0024-0. arXiv: 1910.01617 [astro-ph.HE] (cit. on p. 93).

[333] Ariadna Murguia-Berthier et al. "The Fate of the Merger Remnant in GW170817 and its Imprint on the Jet Structure". In: *Astrophys. J.* 908.2 (2021), p. 152. DOI: 10.3847/1538-4357/abd08e. arXiv: 2007.12245 [astro-ph.HE] (cit. on p. 94).

[334] Yiyang Wu and Andrew MacFadyen. "GW170817 Afterglow Reveals that Short Gamma-Ray Bursts are Neutron Star Mergers". In: *Astrophys. J. Lett.* 880.2 (2019), p. L23. DOI: 10.3847/2041-8213/ab2fd4. arXiv: 1905.02665 [astro-ph.HE] (cit. on p. 94).

[335] Davide Lazzati, Riccardo Ciolfi, and Rosalba Perna. "Intrinsic properties of the engine and jet that powered the short gamma-ray burst associated with GW170817". In: *Astrophys. J.* 898.1 (2020), p. 59. DOI: 10.3847/1538-4357/ab9a44. arXiv: 2004.10210 [astro-ph.HE] (cit. on p. 94).

[336] Iair Arcavi. "The First Hours of the GW170817 Kilonova and the Importance of Early Optical and Ultraviolet Observations for Constraining Emission Models". In: *Astrophys. J. Lett.* 855.2 (2018), p. L23. DOI: 10.3847/2041-8213/aab267. arXiv: 1802.02164 [astro-ph.HE] (cit. on p. 94).

[337] Brad M. S. Hansen and Maxim Lyutikov. "Radio and x-ray signatures of merging neutron stars". In: *Mon. Not. Roy. Astron. Soc.* 322 (2001), p. 695. DOI: 10.1046/j.1365-8711.2001.04103.x. arXiv: astro-ph/0003218 (cit. on p. 94).

[338] Eleonora Troja, Stephan Rosswog, and Neil Gehrels. "Precursors of short gamma-ray bursts". In: *Astrophys. J.* 723 (2010), pp. 1711–1717. DOI: 10.1088/0004-637X/723/2/1711. arXiv: 1009.1385 [astro-ph.HE] (cit. on p. 94).

[339] David Tsang et al. "Resonant Shattering of Neutron Star Crusts". In: *Phys. Rev. Lett.* 108 (2012), p. 011102. DOI: 10.1103/PhysRevLett.108.011102. arXiv: 1110.0467 [astro-ph.HE] (cit. on p. 94).

[340] Brian D. Metzger and Charles Zivancev. "Pair Fireball Precursors of Neutron Star Mergers". In: *Mon. Not. Roy. Astron. Soc.* 461.4 (2016), pp. 4435–4440. DOI: 10.1093/mnras/stw1800. arXiv: 1605.01060 [astro-ph.HE] (cit. on p. 94).

[341] Jie-Shuang Wang et al. "Fast Radio Bursts from the Inspiral of Double Neutron Stars". In: *Astrophys. J. Lett.* 822.1 (2016), p. L7. DOI: 10.3847/2041-8205/822/1/L7. arXiv: 1603.02014 [astro-ph.HE] (cit. on p. 94).

[342] Tomoki Wada, Masaru Shibata, and Kunihito Ioka. "Analytic properties of the electromagnetic field of binary compact stars and electromagnetic precursors to gravitational waves". In: *PTEP* 2020.10 (2020), 103E01. DOI: 10.1093/ptep/ptaa126. arXiv: 2008.04661 [astro-ph.HE] (cit. on p. 94).

[343] Tito Dal Canton et al. "Real-time Search for Compact Binary Mergers in Advanced LIGO and Virgo's Third Observing Run Using PyCBC Live". In: *Astrophys. J.* 923.2 (2021), p. 254. DOI: 10.3847/1538-4357/ac2f9a. arXiv: 2008.07494 [astro-ph.HE] (cit. on pp. 94, 133, 155, 171).

[344] L. Barsotti et al. *The A+ design curve*. https://dcc.ligo.org/public/0149/T1800042/005/T1800042-v5.pdf. 2018 (cit. on p. 94).

[345] Alexander H. Nitz and Tito Dal Canton. "Pre-merger Localization of Compact-binary Mergers with Third-generation Observatories". In: *Astrophys. J. Lett.* 917.2 (2021), p. L27. DOI: 10.3847/2041-8213/ac1a75. arXiv: 2106.15259 [astro-ph.HE] (cit. on p. 94).

[346] B. P. Abbott et al. "GWTC-1: A Gravitational-Wave Transient Catalog of Compact Binary Mergers Observed by LIGO and Virgo during the First and Second Observing Runs". In: *Phys. Rev. X* 9.3 (2019), p. 031040. DOI: 10.1103/PhysRevX.9.031040. arXiv: 1811.12907 [astro-ph.HE] (cit. on pp. 95, 132, 155, 170).

[347] Leo P. Singer and Larry R. Price. "Rapid Bayesian position reconstruction for gravitational-wave transients". In: *Phys. Rev. D* 93.2 (2016), p. 024013. DOI: 10.1103/PhysRevD.93.024013. arXiv: 1508.03634 [gr-qc] (cit. on p. 95).

[348] Michael W. Coughlin et al. "Optimizing multitelescope observations of gravitational-wave counterparts". In: *Mon. Not. Roy. Astron. Soc.* 489.4 (2019), pp. 5775–5783. DOI: 10.1093/mnras/stz2485. arXiv: 1909.01244 [astro-ph.IM] (cit. on p. 99).

[349] Aaron Tohuvavohu et al. "Gamma-Ray Urgent Archiver for Novel Opportunities (GUANO): Swift/BAT Event Data Dumps on Demand to Enable Sensitive Subthreshold GRB Searches". In: *Astrophys. J.* 900.1 (2020), p. 35. DOI: 10.3847/1538-4357/aba94f. arXiv: 2005.01751 [astro-ph.HE] (cit. on p. 99).

[350] Eric C. Bellm et al. "The Zwicky Transient Facility: System Overview, Performance, and First Results". In: *Publications of the Astronomical Society of the Pacific* 131.995 (Dec. 2018), p. 018002. DOI: 10.1088/1538-3873/aaecbe. URL: https://doi.org/10.1088/1538-3873/aaecbe (cit. on p. 99).

[351] Michael W. Coughlin et al. "GROWTH on S190425z: Searching thousands of square degrees to identify an optical or infrared counterpart to a binary neutron star merger with the Zwicky Transient Facility and Palomar Gattini IR". In: *Astrophys. J. Lett.* 885.1 (2019), p. L19. DOI: 10.3847/2041-8213/ab4ad8. arXiv: 1907.12645 [astro-ph.HE] (cit. on p. 99).

[352] Victor Kornilov et al. "Robotic optical telescopes global network MASTER II. Equipment, structure, algorithms". In: *Exper. Astron.* 33 (2012), p. 173. DOI: 10.1007/s10686-011-9280-z. arXiv: 1111.6904 [astro-ph.IM] (cit. on p. 99).

[353] B. S. Acharya et al. "Introducing the CTA concept". In: *Astropart. Phys.* 43 (2013), pp. 3–18. DOI: 10.1016/j.astropartphys.2013.01.007 (cit. on p. 99).

[354] Clancy W. James et al. "Using negative-latency gravitational wave alerts to detect prompt radio bursts from binary neutron star mergers with the Murchison Widefield Array". In: *Mon. Not. Roy. Astron. Soc.* 489.1 (2019), pp. L75–L79. DOI: 10.1093/mnrasl/slz129. arXiv: 1908.08688 [astro-ph.HE] (cit. on p. 99).

[355] J. Camp and TAP Team. "Transient Astrophysics Probe". In: *The Space Astrophysics Landscape for the 2020s and Beyond*. Vol. 2135. Apr. 2019, p. 5027 (cit. on p. 99).

[356] R. Abbott et al. "GWTC-2: Compact Binary Coalescences Observed by LIGO and Virgo During the First Half of the Third Observing Run". In: *Phys. Rev. X* 11 (2021), p. 021053. DOI: 10.1103/PhysRevX.11.021053. arXiv: 2010.14527 [gr-qc] (cit. on pp. 102, 109, 113, 132, 133, 134, 139, 151, 155, 156, 163).

[357] Shaun Hooper et al. "Summed Parallel Infinite Impulse Response (SPIIR) Filters For Low-Latency Gravitational Wave Detection". In: *Phys. Rev. D* 86 (2012), p. 024012. DOI: 10.1103/PhysRevD.86.024012. arXiv: 1108.3186 [gr-qc] (cit. on p. 102).

[358] Alexander H. Nitz, Amber Lenon, and Duncan A. Brown. "Search for Eccentric Binary Neutron Star Mergers in the first and second observing runs of Advanced LIGO". In: *Astrophys. J.* 890 (Dec. 2019), p. 1. DOI: 10.3847/1538-4357/ab6611. arXiv: 1912.05464 [astro-ph.HE] (cit. on pp. 103, 128, 133, 171).

[359] Daniel George and E. A. Huerta. "Deep Learning for Real-time Gravitational Wave Detection and Parameter Estimation: Results with Advanced LIGO Data". In: *Phys. Lett. B* 778 (2018), pp. 64–70. DOI: 10.1016/j.physletb.2017.12.053. arXiv: 1711.03121 [gr-qc] (cit. on pp. 103, 128, 133, 156, 163, 177, 184).

[360] E. A. Huerta and Zhizhen Zhao. "Advances in Machine and Deep Learning for Modeling and Real-Time Detection of Multi-messenger Sources". In: 2021. DOI: 10.1007/978-981-15-4702-7_47-1. arXiv: 2105.06479 [astro-ph.IM] (cit. on pp. 103, 133, 156).

[361] Heming Xia et al. "Improved deep learning techniques in gravitational-wave data analysis". In: *Phys. Rev. D* 103.2 (2021), p. 024040. DOI: 10.1103/PhysRevD.103.024040. arXiv: 2011.04418 [astro-ph.HE] (cit. on p. 103).

[362] Stefano Schmidt et al. "Machine Learning Gravitational Waves from Binary Black Hole Mergers". In: *Phys. Rev. D* 103.4 (2021), p. 043020. DOI: 10.1103/PhysRevD.103.043020. arXiv: 2011.01958 [gr-qc] (cit. on p. 104).

[363] Yoshua Bengio et al. "Curriculum Learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: Association for Computing Machinery, 2009, pp. 41–48. ISBN: 9781605585161. DOI: 10.1145/1553374.1553380. URL: https://doi.org/10.1145/1553374.1553380 (cit. on p. 104).

[364] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/ (cit. on pp. 104, 138, 188).

[365] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (cit. on pp. 104, 170, 174, 188).

[366] Marlin B. Schäfer et al. *Training Strategies for Deep Learning Gravitational-Wave Searches*. https://github.com/gwastro/ml-training-strategies. 2021 (cit. on pp. 105, 114, 122).

[367] R. Abbott et al. "Search for Gravitational Waves Associated with Gamma-Ray Bursts Detected by Fermi and Swift During the LIGO-Virgo Run O3a". In: *Astrophys. J.* 915.2 (2021), p. 86. DOI: 10.3847/1538-4357/abee15. arXiv: 2010.14550 [astro-ph.HE] (cit. on p. 105).

[368] B. P. Abbott et al. "All-sky search for long-duration gravitational wave transients with initial LIGO". In: *Phys. Rev. D* 93.4 (2016), p. 042005. DOI: 10.1103/PhysRevD.93.042005. arXiv: 1511.04398 [gr-qc] (cit. on p. 105).

[369] Caleb Devine, Zachariah B. Etienne, and Sean T. McWilliams. "Optimizing spinning time-domain gravitational waveforms for Advanced LIGO data analysis". In: *Class. Quant. Grav.* 33.12 (2016), p. 125025. DOI: 10.1088/0264-9381/33/12/125025. arXiv: 1601.03393 [astro-ph.HE] (cit. on pp. 107, 136).

[370] Alexander H. Nitz et al. *PyCBC Software*. https://github.com/gwastro/pycbc. 2018 (cit. on p. 114).

[371] L. Nuttall et al. "Improving the Data Quality of Advanced LIGO Based on Early Engineering Run Results". In: *Class. Quant. Grav.* 32.24 (2015), p. 245005. DOI: 10.1088/0264-9381/32/24/245005. arXiv: 1508.07316 [gr-qc] (cit. on pp. 133, 171).

[372] B. P. Abbott et al. "GW150914: The Advanced LIGO Detectors in the Era of First Discoveries". In: *Phys. Rev. Lett.* 116.13 (2016), p. 131103. DOI: 10.1103/PhysRevLett.116.131103. arXiv: 1602.03838 [gr-qc] (cit. on p. 133).

[373] Wei Wei et al. "Deep Learning with Quantized Neural Networks for Gravitational-wave Forecasting of Eccentric Compact Binary Coalescence". In: *Astrophys. J.* 919.2 (2021), p. 82. DOI: 10.3847/1538-4357/ac1121. arXiv: 2012.03963 [gr-qc] (cit. on pp. 133, 156).

[374] E. A. Huerta et al. "Accelerated, scalable and reproducible AI-driven gravitational wave detection". In: *Nature Astron.* 5.10 (2021), pp. 1062–1068. DOI: 10.1038/s41550-021-01405-0. arXiv: 2012.08545 [gr-qc] (cit. on pp. 133, 151, 156).

[375] Alex Nitz et al. *gwastro/pycbc: 1.18.0 release of PyCBC*. Version v1.18.0. Feb. 2021. DOI: 10.5281/zenodo.4556907. URL: https://doi.org/10.5281/zenodo.4556907 (cit. on pp. 134, 141).

[376] François Chollet et al. *Keras*. https://keras.io. 2015 (cit. on p. 138).

[377] Mark A. Kramer. "Nonlinear principal component analysis using autoassociative neural networks". In: *AIChE Journal* 37.2 (1991), pp. 233–243. DOI: https://doi.org/10.1002/aic.690370209. eprint: https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.690370209. URL: https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209 (cit. on p. 146).

[378] Hongyu Shen et al. "Denoising Gravitational Waves with Enhanced Deep Recurrent Denoising Auto-Encoders". In: (Mar. 2019). DOI: 10.1109/ICASSP.2019.8683061. arXiv: 1903.03105 [astro-ph.CO] (cit. on pp. 146, 190).

[379] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. "A survey of transfer learning". In: *Journal of Big Data* 3.1 (May 2016), p. 9. ISSN: 2196-1115. DOI: 10.1186/s40537-016-0043-6. URL: https://doi.org/10.1186/s40537-016-0043-6 (cit. on p. 148).

[380] Chuanqi Tan et al. "A Survey on Deep Transfer Learning". In: *Artificial Neural Networks and Machine Learning – ICANN 2018*. Ed. by Věra Kůrková et al. Cham: Springer International Publishing, 2018, pp. 270–279. ISBN: 978-3-030-01424-7. DOI: 10.1007/978-3-030-01424-7_27 (cit. on p. 148).

[381] Daniel George, Hongyu Shen, and E. A. Huerta. "Classification and unsupervised clustering of LIGO data with Deep Transfer Learning". In: *Phys. Rev. D* 97.10 (2018), p. 101501. DOI: 10.1103/PhysRevD.97.101501 (cit. on p. 148).

[382] Reymond Mesuga and Brian James Bayanay. "A Deep Transfer Learning Approach on Identifying Glitch Wave-form in Gravitational Wave Data". In: (July 2021). arXiv: 2107.01863 [gr-qc] (cit. on p. 148).

[383] Alexander H. Nitz et al. "1-OGC: The first open gravitational-wave catalog of binary mergers from analysis of public Advanced LIGO data". In: *Astrophys. J.* 872.2 (2019), p. 195. DOI: 10.3847/1538-4357/ab0108. arXiv: 1811.01921 [gr-qc] (cit. on p. 155).

[384] R. Abbott et al. "GWTC-2.1: Deep Extended Catalog of Compact Binary Coalescences Observed by LIGO and Virgo During the First Half of the Third Observing Run". In: (Aug. 2021). arXiv: 2108.01045 [gr-qc] (cit. on pp. 155, 156).

[385] Miles Cranmer et al. "Discovering Symbolic Models from Deep Learning with Inductive Biases". In: (June 2020). arXiv: 2006.11287 [cs.LG] (cit. on p. 156).

[386] Tess E. Smidt, Mario Geiger, and Benjamin Kurt Miller. "Finding symmetry breaking order parameters with Euclidean neural networks". In: *Phys. Rev. Res.* 3.1 (2021), p. L012002. DOI: 10.1103/PhysRevResearch.3.L012002. arXiv: 2007.02005 [cs.LG] (cit. on p. 156).

[387] Mark P. Oxley et al. "Probing atomic-scale symmetry breaking by rotationally invariant machine learning of multidimensional electron scattering". In: *npj Computational Materials* 7.1 (May 2021), p. 65. ISSN: 2057-3960. DOI: 10.1038/s41524-021-00527-3. URL: https://doi.org/10.1038/s41524-021-00527-3 (cit. on p. 156).

[388] Maximilian Dax et al. "Group equivariant neural posterior estimation". In: (Nov. 2021). arXiv: 2111.13139 [cs.LG] (cit. on p. 156).

[389] Daniel George, Hongyu Shen, and E. A. Huerta. "Deep Transfer Learning: A new deep learning glitch classification method for advanced LIGO". In: (June 2017). arXiv: 1706.07446 [gr-qc] (cit. on p. 156).

[390] Deep Chatterjee et al. "A Machine Learning Based Source Property Inference for Compact Binary Mergers". In: *Astrophys. J.* 896.1 (2020), p. 54. DOI: 10.3847/1538-4357/ab8dbe. arXiv: 1911.00116 [astro-ph.IM] (cit. on p. 156).

[391] Alistair McLeod et al. "Rapid Mass Parameter Estimation of Binary Black Hole Coalescences Using Deep Learning". In: (Jan. 2022). arXiv: 2201.11126 [astro-ph.IM] (cit. on p. 156).

[392] Filip Morawski, Michał Bejger, and Paweł Cieciel\ag. "Convolutional neural network classifier for the output of the time-domain F-statistic all-sky search for continuous gravitational waves". In: *Mach. Learn. Sci. Tech.* 1.2 (2020), p. 025016. DOI: 10.1088/2632-2153/ab86c7. arXiv: 1907.06917 [astro-ph.IM] (cit. on p. 156).

[393] Andrew L. Miller et al. "How effective is machine learning to detect long transient gravitational waves from neutron stars in a real search?" In: *Phys. Rev. D* 100.6 (2019), p. 062005. DOI: 10.1103/PhysRevD.100.062005. arXiv: 1909.02262 [astro-ph.IM] (cit. on p. 156).

[394] Takahiro S. Yamamoto et al. "Assessing the impact of non-Gaussian noise on convolutional neural networks that search for continuous gravitational waves". In: *Phys. Rev. D* 106.2 (2022), p. 024025. DOI: 10.1103/PhysRevD.106.024025. arXiv: 2206.00882 [gr-qc] (cit. on p. 156).

[395] Chayan Chatterjee et al. "Using Deep Learning to Localize Gravitational Wave Sources". In: *Phys. Rev. D* 100.10 (2019), p. 103025. DOI: 10.1103/PhysRevD.100.103025. arXiv: 1909.06367 [astro-ph.IM] (cit. on p. 156).

[396] Dixeena Lopez et al. "Gaussian mixture modeling utilization in all-sky search for short-duration gravitational wave burst". In: (Dec. 2021). arXiv: 2112.06608 [gr-qc] (cit. on p. 156).

[397] Sebastian Khan and Rhys Green. "Gravitational-wave surrogate models powered by artificial neural networks". In: *Phys. Rev. D* 103.6 (2021), p. 064015. DOI: 10.1103/PhysRevD.103.064015. arXiv: 2008.12932 [gr-qc] (cit. on p. 156).

[398] Paraskevi Nousi et al. "Autoencoder-driven Spiral Representation Learning for Gravitational Wave Surrogate Modelling". In: (July 2021). arXiv: 2107.04312 [cs.LG] (cit. on p. 156).

[399] Styliani-Christina Fragkouli et al. "Deep Residual Error and Bag-of-Tricks Learning for Gravitational Wave Surrogate Modeling". In: (Mar. 2022). arXiv: 2203.08434 [astro-ph.IM] (cit. on p. 156).

[400] Vasileios Skliris, Michael R. K. Norman, and Patrick J. Sutton. "Real-Time Detection of Unmodelled Gravitational-Wave Transients Using Convolutional Neural Networks". In: (Sept. 2020). arXiv: 2009.14611 [astro-ph.IM] (cit. on p. 156).

[401] Asad Khan and E. A. Huerta. "AI and extreme scale computing to learn and infer the physics of higher order gravitational wave modes of quasi-circular, spinning, non-precessing binary black hole mergers". In: (Dec. 2021). arXiv: 2112.07669 [astro-ph.IM] (cit. on p. 156).

[402] Wen-Hong Ruan et al. "Rapid search for massive black hole binary coalescences using deep learning". In: (Nov. 2021). arXiv: 2111.14546 [astro-ph.IM] (cit. on p. 156).

[403] Pranshu Chaturvedi et al. “Inference-optimized AI and high performance computing for gravitational wave detection at scale”. In: (Jan. 2022). arXiv: 2201.11133 [gr-qc] (cit. on p. 156).

[404] Grégory Baltus et al. “Convolutional neural network for gravitational-wave early alert: Going down in frequency”. In: (May 2022). arXiv: 2205.04750 [gr-qc] (cit. on p. 156).

[405] P. Bacon, A. Trovato, and M. Bejger. “Denoising gravitational-wave signals from binary black holes with dilated convolutional autoencoder”. In: (May 2022). arXiv: 2205.13513 [gr-qc] (cit. on pp. 156, 190).

[406] European Gravitational Observatory - EGO. *G2Net Gravitational Wave Detection Kaggle Challenge*. 2021. URL: https://github.com/gwastro/ml-mock-data-challenge-1 (visited on 08/10/2022) (cit. on p. 157).

[407] Marlin Schäfer and Ondřej Zelenka. *MLGWSC-1 Github repository*. 2021. URL: https://github.com/gwastro/ml-mock-data-challenge-1 (visited on 02/23/2022) (cit. on pp. 157, 158, 163, 168, 176, 177, 185).

[408] The HDF Group. *Hierarchical Data Format, version 5*. 1997-2022. URL: https://www.hdfgroup.org/HDF5/ (cit. on p. 158).

[409] B. Abbott et al. “Search for gravitational waves from binary inspirals in S3 and S4 LIGO data”. In: *Phys. Rev. D* 77 (2008), p. 062002. DOI: 10.1103/PhysRevD.77.062002. arXiv: 0704.3368 [gr-qc] (cit. on p. 166).

[410] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. 2022. DOI: 10.48550/ARXIV.2207.02696. URL: https://arxiv.org/abs/2207.02696 (cit. on pp. 170, 176, 199).

[411] Tianqi Chen et al. *MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems*. 2015. DOI: 10.48550/ARXIV.1512.01274. URL: https://arxiv.org/abs/1512.01274 (cit. on p. 170).

[412] Andrew Howard et al. *Searching for MobileNetV3*. 2019. DOI: 10.48550/ARXIV.1905.02244. URL: https://arxiv.org/abs/1905.02244 (cit. on p. 170).

[413] Miriam Cabero et al. “Blip glitches in Advanced LIGO data”. In: *Class. Quant. Grav.* 36.15 (2019), p. 15. DOI: 10.1088/1361-6382/ab2e14. arXiv: 1901.05093 [physics.ins-det] (cit. on p. 171).

[414] Derek Davis and Marissa Walker. "Detector Characterization and Mitigation of Noise in Ground-Based Gravitational-Wave Interferometers". In: *Galaxies* 10.1 (2022), p. 12. DOI: 10.3390/galaxies10010012 (cit. on p. 171).

[415] Marlin Schäfer. *CNN-Coinc Github repository*. 2022. URL: https://github.com/MarlinSchaefer/cnn-coinc (visited on 06/15/2022) (cit. on p. 172).

[416] Nikolaos Passalis et al. "Deep adaptive input normalization for time series forecasting". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.9 (2019), pp. 3760–3765 (cit. on pp. 174, 175).

[417] Marco Drago et al. "coherent WaveBurst, a pipeline for unmodeled gravitational-wave data analysis". In: *SoftwareX* 14 (2021), p. 100678. ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2021.100678. URL: https://www.sciencedirect.com/science/article/pii/S2352711021000236 (cit. on p. 175).

[418] Sergey Klimenko et al. *cWB pipeline library: 6.4.1*. Version cWB-6.4.1. Dec. 2021. DOI: 10.5281/zenodo.5798976. URL: https://doi.org/10.5281/zenodo.5798976 (cit. on p. 175).

[419] cWB Development Team. *coherent WaveBurst Homepage*. 2021. URL: https://gwburst.gitlab.io/ (cit. on p. 175).

[420] Colm Talbot and Eric Thrane. "Measuring the Binary Black Hole Mass Spectrum with an Astrophysically Motivated Parameterization". In: *The Astrophysical Journal* 856.2 (Apr. 2018), p. 173. ISSN: 1538-4357. DOI: 10.3847/1538-4357/aab34c. URL: http://dx.doi.org/10.3847/1538-4357/aab34c (cit. on p. 176).

[421] J. Aasi et al. "Characterization of the LIGO detectors during their sixth science run". In: *Class. Quant. Grav.* 32.11 (2015), p. 115012. DOI: 10.1088/0264-9381/32/11/115012. arXiv: 1410.7764 [gr-qc] (cit. on p. 177).

[422] B. P. Abbott et al. "Characterization of transient noise in Advanced LIGO relevant to gravitational wave signal GW150914". In: *Class. Quant. Grav.* 33.13 (2016), p. 134001. DOI: 10.1088/0264-9381/33/13/134001. arXiv: 1602.03844 [gr-qc] (cit. on p. 177).

[423] Derek Davis et al. "Incorporating information from LIGO data quality streams into the PyCBC search for gravitational waves". In: (Apr. 2022). arXiv: 2204.03091 [gr-qc] (cit. on p. 177).

[424] B. P. Abbott et al. "Observing gravitational-wave transient GW150914 with minimal assumptions". In: *Phys. Rev. D* 93.12 (2016). [Addendum: Phys.Rev.D 94, 069903 (2016)], p. 122004. DOI: 10.1103/PhysRevD.93.122004. arXiv: 1602.03843 [gr-qc] (cit. on p. 182).

[425] Thomas Cokelaer. "Gravitational waves from inspiralling compact binaries: Hexagonal template placement and its efficiency in detecting physical signals". In: *Phys. Rev. D* 76 (2007), p. 102004. DOI: 10.1103/PhysRevD.76.102004. arXiv: 0706.4437 [gr-qc] (cit. on p. 185).

[426] Rahul Dhurkunde, Henning Fehrmann, and Alexander H. Nitz. "Hierarchical approach to matched filtering using a reduced basis". In: *Phys. Rev. D* 105.10 (2022), p. 103001. DOI: 10.1103/PhysRevD.105.103001. arXiv: 2110.13115 [astro-ph.IM] (cit. on p. 187).

[427] Alexander Schmitt et al. "Investigating Deep Neural Networks for Gravitational Wave Detection in Advanced LIGO Data". In: *Proceedings of the 2nd International Conference on Computer Science and Software Engineering*. CSSE 2019. Xi'an, China: Association for Computing Machinery, 2019, pp. 73–78. ISBN: 9781450371728. DOI: 10.1145/3339363.3339377. URL: https://doi.org/10.1145/3339363.3339377 (cit. on p. 190).

[428] Fangyun Wei et al. "Aligning Pretraining for Detection via Object-Level Contrastive Learning". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 22682–22694. arXiv: 2106.02637. URL: https://proceedings.neurips.cc/paper/2021/file/bf5cd8b2509011b9502a72296edc14a0-Paper.pdf (cit. on pp. 195, 202, 203, 204, 205, 206, 211, 213).

[429] Li Fei-Fei et al. *ImageNet Website*. 2022. URL: https://image-net.org/about.php (visited on 07/22/2022) (cit. on p. 196).

[430] Shivy Yohanandan. *mAP (mean Average Precision) might confuse you!* 2020. URL: https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2 (visited on 10/16/2022) (cit. on p. 197).

[431] J. R. R. Uijlings et al. "Selective Search for Object Recognition". In: *International Journal of Computer Vision* 104.2 (Sept. 2013), pp. 154–171. DOI: 10.1007/s11263-013-0620-5. URL: https://doi.org/10.1007/s11263-013-0620-5 (cit. on pp. 197, 203, 204, 205).

[432] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". In: *arXiv e-prints*, arXiv:1612.08242 (Dec. 2016), arXiv:1612.08242. arXiv: 1612.08242 [cs.CV] (cit. on p. 199).

[433] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. DOI: 10.48550/ARXIV.1804.02767. URL: https://arxiv.org/abs/1804.02767 (cit. on p. 199).

[434] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. DOI: 10.48550/ARXIV.2004.10934. URL: https://arxiv.org/abs/2004.10934 (cit. on p. 199).

[435] Chuyi Li et al. *YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications*. 2022. DOI: 10.48550/ARXIV.2209.02976. URL: https://arxiv.org/abs/2209.02976 (cit. on p. 199).

[436] Mingxing Tan, Ruoming Pang, and Quoc V. Le. "EfficientDet: Scalable and Efficient Object Detection". In: (2019). DOI: 10.48550/ARXIV.1911.09070. URL: https://arxiv.org/abs/1911.09070 (cit. on p. 200).

[437] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: (2019). DOI: 10.48550/ARXIV.1905.11946. URL: https://arxiv.org/abs/1905.11946 (cit. on p. 200).

[438] Zhi Tian et al. *FCOS: Fully Convolutional One-Stage Object Detection*. 2019. DOI: 10.48550/ARXIV.1904.01355. URL: https://arxiv.org/abs/1904.01355 (cit. on p. 200).

[439] Ting Chen et al. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. DOI: 10.48550/ARXIV.2002.05709. URL: https://arxiv.org/abs/2002.05709 (cit. on pp. 201, 204).

[440] Ting Chen et al. *Big Self-Supervised Models are Strong Semi-Supervised Learners*. 2020. DOI: 10.48550/ARXIV.2006.10029. URL: https://arxiv.org/abs/2006.10029 (cit. on p. 201).

[441] Jean-Bastien Grill et al. *Bootstrap your own latent: A new approach to self-supervised Learning*. 2020. DOI: 10.48550/ARXIV.2006.07733. URL: https://arxiv.org/abs/2006.07733 (cit. on pp. 201, 204).

[442] Xinlei Chen and Kaiming He. *Exploring Simple Siamese Representation Learning*. 2020. DOI: 10.48550/ARXIV.2011.10566. URL: https://arxiv.org/abs/2011.10566 (cit. on pp. 202, 204, 211).

[443] Fangyun Wei et al. *Github: SoCo*. 2022. URL: https://github.com/hologerry/SoCo (visited on 09/20/2022) (cit. on pp. 204, 210, 211, 212).

[444] Flickr. *Flickr*. 2022. URL: https://www.flickr.com/ (visited on 10/16/2022) (cit. on p. 209).

[445] Piotr Dollar et al. "Pedestrian Detection: An Evaluation of the State of the Art". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.4 (2012), pp. 743–761. DOI: 10.1109/TPAMI.2011.155 (cit. on p. 210).

[446] Yang You, Igor Gitman, and Boris Ginsburg. *Large Batch Training of Convolutional Networks*. 2017. DOI: 10.48550/ARXIV.1708.03888. URL: https://arxiv.org/abs/1708.03888 (cit. on pp. 210, 211).

[447] Andrew Y. Ng. "Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance". In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 78. ISBN: 1581138385. DOI: 10.1145/1015330.1015435. URL: https://doi.org/10.1145/1015330.1015435 (cit. on p. 211).

[448] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 740–755. ISBN: 978-3-319-10602-1. DOI: 10.48550/ARXIV.1405.0312. URL: https://arxiv.org/abs/1405.0312 (cit. on p. 213).

[449] Chetan Verma et al. "Employing Deep Learning for Detection of Gravitational Waves from Compact Binary Coalescences". In: (Oct. 2021). arXiv: 2110.01883 [gr-qc] (cit. on p. 217).

Appendix A

Acknowledgments

Throughout the three years that I have spent working on this thesis I had support from many people, who I would like to thank here.

First of, I would like to thank the entire Albert-Einstein-Institute for their support, the opportunities they have provided me with, and the friendly working environment. The discussions I could partake in were always very fruitful. I specifically want to thank Bruce Allen, Alexander Harvey Nitz, and Badri Krishnan for allowing me to join the institute and the compact binary merger group. I also want to thank the Atlas computing team at the Albert-Einstein-Institute for their support, without whom most of the work presented in this thesis would not have been possible.

I am especially grateful for the guidance I have received by Alexander Harvey Nitz, who in most aspects has been my mentor. He suggested many of the research topics covered in this thesis but was also always open to new ideas and allowed me to experiment. His insights and tips have been invaluable to the completion of this thesis.

Another person who has greatly influenced my scientific evolution is Frank Ohme. He was my first point of contact with the Albert-Einstein-Institute and has always tried to enable my development wherever possible. His scientific input has greatly improved the quality of the works presented in this thesis.

I would also like to thank all co-authors of the various publications that have culminated in this thesis. These people are Bernd Brügmann, Miriam Cabero, Zhoujian Cao, Collin D. Capano, Elena Cuoco, Tito Dal Canton, Rahul Dhurkunde, Zong-Kuan Guo, Eliu A. Huerta, Panagiotis Iosif, Shilpa Kastha, Sergey Klimenko, Alexandra E. Koloniari, Sumit Kumar, Chris Messenger, Tanmaya Mishra, Alexander H. Nitz, Paraskevi Nousi, Frank Ohme, Nikolaos Passalis, Zhixiang Ren, Francesco Salemi, Nikolaos Stergioulas, Anastasios Tefas, Gabriele Vedovato, He Wang, Yi-Fan Wang, Shichao Wu,

and Ondřej Zelenka.

A major contribution to the completion of this thesis have been my family and my close friends. Their moral and emotional support has been essential for my well being during this time. Most importantly I want to express my gratitude towards my mother Marie Therese Schäfer and my partner Nadine Speer, who were a pillar especially during difficult times. I also want to thank my father Thomas Eberhard Kurt Schäfer as well as Doris Maria Schulz, Helmger Hans Roderich Schotola, Felizitas Kaya, Horst Paul Hermann Schrödter, and Brigitta Beatrice Kuhr for support during all stages of my life.

I would further like to thank Bosch Hildesheim for allowing me to have a look into the non-academic work life and for interesting insights into computer vision applications of deep learning. I would like to especially thank Niklas Beuter, Hendrik Klose, and Christoph Dreißigacker, who made the internship possible. Furthermore, I would like to thank Lukas Enderich, who I closely worked with, as well as the entire closed loop learning project for the warm welcome and the open discussions.

Finally, I would like to thank the people who proofread the majority of this thesis. They significantly improved clarity, grammar, and spelling. Thank you Tobias Blenke, Christoph Dreißigacker, Lukas Enderich, Tom Junker, Alexander H. Nitz, Frank Ohme, Lina Schmitz, Nadine Speer, and Ondřej Zelenka.

Appendix B

List of Publications

The research for this thesis has resulted in the following publications:

[61] Marlin B. Schäfer, Frank Ohme, and Alexander H. Nitz, Detection of gravitational-wave signals from binary neutron star mergers using machine learning. Phys. Rev. D 102.6 (2020)

[62] Alexander H. Nitz, Marlin B. Schäfer, and Tito Dal Canton, Forecasting Gravitational-wave Merger Forecasting: Scenarios for the Early Detection and Localization of Compact-binary Mergers with Ground-based Observatories. Astrophys. J. Lett. 902 (2020)

[206] Alexander H. Nitz, Collin D. Capano, Sumit Kumar, Yi-Fan Wang, Shilpa Kastha, Marlin B. Schäfer, Rahul Dhurkunde and Miriam Cabero, 3-OGC: Catalog of Gravitational Waves from Compact-binary Mergers. Astrophys. J. 922.1 (2021)

[63] Marlin B. Schäfer and Ondřej Zelenka, Alexander H. Nitz, Frank Ohme and Bernd Brügmann, Training strategies for deep learning gravitational-wave searches. Phys. Rev. D 105.4 (2022)

[64] Marlin B. Schäfer and Alexander H. Nitz, From one to many: A deep learning coincident gravitational-wave search. Phys. Rev. D 105.4 (2022)

[15] Alexander H. Nitz, Sumit Kumar, Yi-Fan Wang, Shilpa Kastha, Shichao Wu, Marlin B. Schäfer, Rahul Dhurkunde and Collin D. Capano, 4-OGC: Catalog of gravitational waves from compact-binary mergers. pre-print arXiv: 2112.06878 (2021)

[65] Marlin B. Schäfer, Ondřej Zelenka, Alexander H. Nitz, He Wang, Shichao Wu, Zong-Kuan Guo, Zhoujian Cao, Zhixiang Ren, Paraskevi Nousi,

Nikolaos Stergioulas, Panagiotis Iosif, Alexandra E. Koloniari, Anastasios Tefas, Nikolaos Passalis, Francesco Salemi, Gabriele Vedovato, Sergey Klimenko, Tanmaya Mishra, Bernd Brügmann, Elena Cuoco, E.A. Huerta, Chris Messenger and Frank Ohme, MLGWSC-1: The first Machine Learning Gravitational-Wave Search Mock Data Challenge. pre-print arXiv: 2209.11146 (2022)

Curriculum Vitae

Marlin Benedikt Schäfer

Date of birth	28th December 1995
Place of birth	Hannover, Germany
Email	marlin.schaefer@aei.mpg.de

Education

2017 – 2019 MSc Physik – Leibniz Universität Hannover

Master thesis

Analysis of Gravitational-Wave Signals from Binary Neutron Star Mergers Using Machine Learning

Supervisor: Dr. Frank Ohme

2014 – 2018 BSc Physik – Leibniz Universität Hannover

Bachelor thesis

Massenbestimmung kompakter Binärsysteme durch Analyse von Gravitationswellen

Determination of Component Masses from Compact Binary System by Analyzing Gravitational Waves

Supervisor: Prof. Domenico Giulini

2006 – 2014 Abitur – Kaiser-Wilhelm- und Ratsgymnasium Hannover

Experience

Nov. 2019 – Dec. 2022	Doctoral researcher – Albert-Einstein-Institute Hannover
Jun. 2022 – Sep. 2022	Research intern – Robert Bosch GmbH Hildesheim
Apr. 2019 – Sep. 2019	Assistant researcher – Albert-Einstein-Institute Hannover
Oct. 2016 – Mar. 2019	Tutor for theoretical physics – Leibniz Universität Hannover
Aug. & Sep. 2016 – 2018	Tutor for mathematics – Leibniz Universität Hannover