First machine learning gravitational-wave search mock data challenge

Marlin B. Schäfer,[1,2] Ondřej Zelenka,[3,4] Alexander H. Nitz,[1,2] He Wang,[5] Shichao Wu,[1,2] Zong-Kuan Guo,[5] Zhoujian Cao,[6] Zhixiang Ren,[7] Paraskevi Nousi,[8] Nikolaos Stergioulas,[9] Panagiotis Iosif,[10,9] Alexandra E. Koloniari,[9] Anastasios Tefas,[8] Nikolaos Passalis,[8] Francesco Salemi,[11,12] Gabriele Vedovato,[13] Sergey Klimenko,[14] Tanmaya Mishra,[14] Bernd Brügmann,[3,4] Elena Cuoco,[15,16,17] E. A. Huerta,[18,19] Chris Messenger,[20] and Frank Ohme[1,2]

[1]*Max-Planck-Institut für Gravitationsphysik, Albert-Einstein-Institut, D-30167 Hannover, Germany*
[2]*Leibniz Universität Hannover, D-30167 Hannover, Germany*
[3]*Friedrich-Schiller-Universität Jena, D-07743 Jena, Germany*
[4]*Michael Stifel Center Jena, D-07743 Jena, Germany*
[5]*CAS Key Laboratory of Theoretical Physics, Institute of Theoretical Physics, Chinese Academy of Sciences, Beijing 100190, China*
[6]*Department of Astronomy, Beijing Normal University, Beijing 100875, China*
[7]*Peng Cheng Laboratory, Shenzhen, 518055, China*
[8]*Department of Informatics, Aristotle University of Thessaloniki, GR-54124 Thessaloniki, Greece*
[9]*Department of Physics, Aristotle University of Thessaloniki, GR-54124 Thessaloniki, Greece*
[10]*GSI Helmholtz Center for Heavy Ion Research, Planckstraße 1, 64291 Darmstadt, Germany*
[11]*Università di Trento, Dipartimento di Fisica, I-38123 Povo, Trento, Italy*
[12]*INFN, Trento Institute for Fundamental Physics and Applications, I-38123 Povo, Trento, Italy*
[13]*INFN, Sezione di Padova, I-35131 Padova, Italy*
[14]*Department of Physics, University of Florida, PO Box 118440, Gainesville, Florida 32611-8440, USA*
[15]*European Gravitational Observatory (EGO), I-56021 Cascina, Pisa, Italy*
[16]*Scuola Normale Superiore, Piazza dei Cavalieri 7, I-56126 Pisa, Italy*
[17]*INFN, Sezione di Pisa, Largo Bruno Pontecorvo, 3, I-56127 Pisa, Italy*
[18]*Data Science and Learning Division, Argonne National Laboratory, Lemont, Illinois 60439, USA*
[19]*Department of Computer Science, University of Chicago, Chicago, Illinois 60637, USA*
[20]*SUPA, School of Physics and Astronomy, University of Glasgow, Glasgow G12 8QQ, United Kingdom*

(Received 23 September 2022; accepted 28 November 2022; published 27 January 2023)

We present the results of the first Machine Learning Gravitational-Wave Search Mock Data Challenge. 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 s with the inclusion of precession effects and higher order modes. We present the average sensitivity distance and run-time 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 ≥ 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.

DOI: 10.1103/PhysRevD.107.023021

I. INTRODUCTION

The first gravitational-wave (GW) observation on September 14, 2015 [1] achieved by the LIGO and Virgo Collaboration [2,3] 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 [4–9]. 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 [10–12]. 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) [13,14]. 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 [15,16].

GW signals are commonly identified in the background noise of the detectors using matched filtering [13,17–19]. Matched filtering compares precomputed models of expected signals, known as templates, with the data from the detectors [20]. When a model matches the data to a predefined degree and data-quality requirements are met, a candidate detection is reported. Loosely modeled searches [21–23], 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 [2], as well as sensitivity improvements for the advanced Virgo detector [3]. With the entry into service of Kagra [24] 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 [25–27] 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 [28]. Some examples include the prediction of protein structure used in pharmaceutical studies [29], improvements to material composition and synthesis [30], or event reconstruction at the Large Hadron Collider [31]. There is also ongoing research into using neural networks to discover closed form expressions from raw data [32] or optimizing machine learning algorithms to take advantage of physical symmetries of the underlying problem [33–35].

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 [36–38], parameter estimation [39–43], continuous GW detection [44–50], enhancements for existing pipelines [51–58], surrogate waveform models [59–61], as well as various signal detection algorithms [62–82]. For a summary of many methods we refer the reader to [83,84]. 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 [10–12]. 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 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 [2]. The final dataset challenges participants to identify GWs from spinning BBHs with a duration of up to 20 s added to real detector

[1]There has previously been a public Kaggle challenge [85]. First in the sense of this paper refers to our setup of providing continuous data.

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 run-time. 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 run-time 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 [14] 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 [86]. Therein we also collect the individual submissions by groups that have given 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 Sec. II we provide the details on the challenge, the datasets, as well as the evaluation process. All submissions are briefly introduced in Sec. III. The results of the challenge and a brief discussion can be found in Sec. V. We conclude and give an outlook into possible future work in Sec. VI.

II. METHODS

All submissions described in Sec. III 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.

A. 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 [86].

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 Sec. II B. 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 Sec. II C.

B. Test data

The challenge provides a script to generate semicontinuous 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 [87] 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 metadata used to create the file. This metadata 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 preprocessing 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

TABLE I. A summary of the distributions shared between all datasets from which parameters are drawn.

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	$d_c^2 \in (130^2, 350^2)$ Mpc^2

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 preprocessing 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 I. All signals are generated using the waveform model `IMRPhenomXPHM` [88] 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 separated by a random time between 24 to 30 s to avoid any overlap. We apply a taper to the start of each waveform.

In Fig. 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 [13,14].

1. Dataset 1

The noise from the first dataset is purely Gaussian and simulated from the PSD model `aLIGOZeroDetHighPower` [89] 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 nonspinning and no higher-order modes are simulated. The component masses are uniformly drawn from 10 $M_\odot$ to 50 $M_\odot$. 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 nonspinning 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 [62,64,76]. 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.

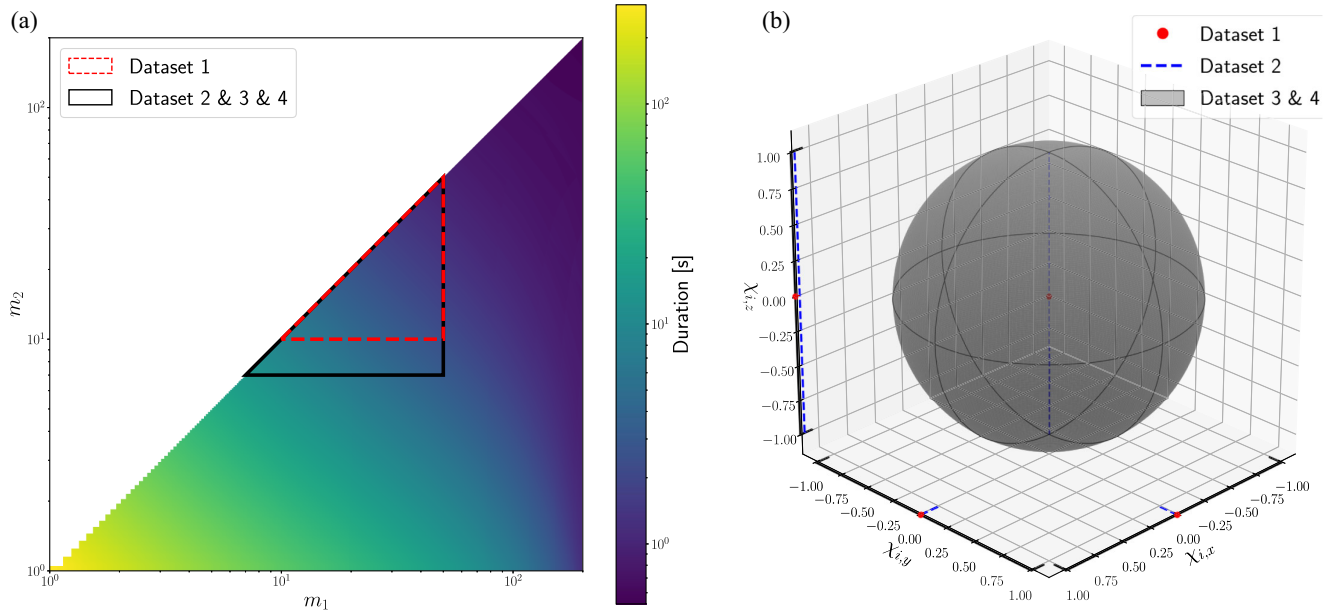

FIG. 1. An illustration of the range for the intrinsic parameters covered by this challenge. The left panel (a) shows a typical range for the component masses used by state-of-the-art searches [14]. The color indicates the duration of the waveform from 20 Hz. The triangles show the parameter regions covered by this challenge. The right panel (b) shows the component-spin χ_i distribution of the different datasets in this challenge.

The final challenge set for dataset 1 was generated with the seed 1068209514 and a start time of 0.

2. 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 $M_\odot$ to 50 $M_\odot$. 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 algorithm 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 [40,47,69,90]. 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 2743406703 and a start time of 0.

3. 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 Sec. II B 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 $M_\odot$ to 50 $M_\odot$ and subsequently the maximum signal duration of 20 s is unchanged compared to Sec. II B 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 $(2, 2)$, $(2, -2)$, $(2, 1)$, $(2, -1)$, $(3, 3)$, $(3, -3)$, $(3, 2)$, $(3, -2)$, $(4, 4)$, $(4, -4)$ [88].

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 [25,26]. 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 470182217 and a start time of 0.

4. Dataset 4

Dataset 4 is the only dataset that contains real detector noise obtained from the Gravitational Wave Open Science Center (GWOSC) [91]. 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 [10]. 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 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 preselected 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 7024699 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 down sample the data is also open source and available at [86].

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 [63,65,67,68] 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 2514409456 and a start time of 0.

C. 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 [92] 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 Sec. II D 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 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. 2592000 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

$$\mathcal{F} = \frac{N_{\mathrm{FP},\mathcal{R}}}{T}. \tag{1}$$

The sensitive volume of a search at FAR $\mathcal{F}$ can be calculated by [92]

$$V(\mathcal{F}) = \int d\boldsymbol{x} d\boldsymbol{\Lambda} \epsilon(\mathcal{F}; \boldsymbol{x}, \boldsymbol{\Lambda}) \phi(\boldsymbol{x}, \boldsymbol{\Lambda}), \tag{2}$$

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}, Eq. (2) can be approximated by [92]

$$V(\mathcal{F}) \approx V(d_{\mathrm{max}}) \frac{N_{I,\mathcal{F}}}{N_I}, \tag{3}$$

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

[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.

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 Eq. (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 [93]

$$d_c = d\left(\frac{\mathcal{M}_{c,0}}{\mathcal{M}_c}\right)^{5/6}, \tag{4}$$

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 [93] we use the luminosity distance instead of the effective distance as our basis.

When sampling the injections from the distributions defined in Table I 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 Eq. (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

$$V(\mathcal{F}) \approx \frac{V(d_{\max})}{N_I} \sum_{i=1}^{N_{I,\mathcal{F}}} \frac{V\left(d_{c,\max}\left(\frac{\mathcal{M}_{c,i}}{\mathcal{M}_{c,0}}\right)^{5/6}\right)}{V\left(d_{c,\max}\left(\frac{\mathcal{M}_{c,\max}}{\mathcal{M}_{c,0}}\right)^{5/6}\right)}, \tag{5}$$

where $\mathcal{M}_{c,i}$ is the chirp mass of the ith 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

$$V(\mathcal{F}) \approx \frac{V(d_{\max})}{N_I} \sum_{i=1}^{N_{I,\mathcal{F}}} \left(\frac{\mathcal{M}_{c,i}}{\mathcal{M}_{c,\max}}\right)^{5/2}, \tag{6}$$

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.

TABLE II. Main hardware specifications available to each search algorithm during final testing.

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

We also measure the time the algorithm requires to evaluate an entire month of test data. Since all machine learning search algorithms are running 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 II. However, we expect the computational time to be dominated by preprocessing steps, which can in theory be heavily optimized. For this challenge, though, we did not expect many submissions to invest resources into optimizing their preprocessing and thus advise the reader to not overemphasize the provided numbers.

All run-times 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.

D. 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 Sec. II B. Importantly, no preprocessing other than the introduction of a low frequency cutoff of 15 Hz has been applied to the data. All other preprocessing 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-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

TABLE III. A selective list of the core PYTHON packages available to algorithms during evaluation. A complete list is given at [86].

PYTHON package	Version
bilby	1.1.3
pycbc	efeaeb6
tensorflow-gpu	2.6.0
tensorflow-probability	0.14.0
torch	1.9.1 + cu11

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 excessively 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 III, for a full list refer to [86].

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.

III. 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 Sec. II C for evaluation of the test sets are tailored to suit the needs of machine learning algorithms.

A. MFCNN[3]

The submission of the MFCNN group is based on the works from He *et al.* [94]. The authors of [94] refer to the model as matched-filtering convolutional neural network (MFCNN). MFCNN is a semicoherent 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

[3]The corresponding authors for the MFCNN submission are He Wang, Shichao Wu, Zong-Kuan Guo, Zhoujian Cao, and Zhixiang Ren.

combine them to search for real GW events in GWTC-1 [5]. To adapt to this challenge, the source code [95] of the submission was translated from the MXNet framework [96] used in the original work to a PyTorch [97] 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 preprocessing such as band pass or whitening, which is consistent with the original work [94]. 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 [98] 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 of predictions and a table of SNR peaks from the 25 sorted matched filters. The step size is 1 s and a threshold of 0.5 is set on the network output as in [94]. The "time", "var", and "stat" dataset of the output file described in Sec. II D 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 stand alone 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.

B. PyCBC[4]

The *PyCBC* submission is based on a standard configuration of the PyCBC-based archival search for compact-binary mergers [14]. The search infrastructure was used, in addition to cWB, for the first detection of gravitational waves, GW159014 [1], in production analyses by multiple groups to produce gravitational-wave catalogs [13,14] and targeted analyses [99]. A similar low-latency PyCBC-Live analysis is also based around the same toolkit [18,100]. 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 [20]. Matched filtering is known to be the optimal linear filter for stationary, Gaussian noise. To account for the potential non-Gaussian noise transients [101–103], each candidate and the surrounding data are checked for consistency with the expected signal [104,105]. 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 [106].

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 [106,107]; 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 nonspinning waveform templates, using the IMRPhenomD [108] model, is created using stochastic placement. Datasets 2, 3, and 4 were evaluated 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.

C. CNN-Coinc[5]

This submission is based on the works from Gabbard *et al.* [64] and Schäfer *et al.* [76]. It utilizes the network architecture presented in [64] with a prepended batch-normalization layer [109]. As such the network processes 8192 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

[4]The corresponding author for the PyCBC submission is Alexander H. Nitz.

[5]The corresponding author for the CNN-Coinc submission is Marlin B. Schäfer.

outputs are correlated to find coincident events as detailed in [76]. 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 [110]. 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 [76] 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` [88] from the same parameter distribution used in datasets 3 and 4 in this challenge. Merger times of the signals are varied between 2.9 to 3.1 s from the start of the input window of the network. The signals are prewhitened by one of the provided Hanford PSDs used in datasets 2 and 3. Noise samples are nonoverlapping 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 [76] on a single NVIDIA RTX 2070. The epoch with the greatest binary accuracy on a single training run was chosen for this challenge.

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 [20,111]. All data are 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 ≈ 0.1 s. Afterwards a threshold of 3.4 is applied on the unbounded Softmax replacement (USR) output, which was introduced in [75]. Coincident events are calculated using the same procedure and parameters as outlined in [76]. The "time" and "stat" dataset of the output file described in Sec. II D 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 s.

D. TPI FSU Jena[6]

This submission closely followed the method of [75], which is itself based on [64], 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 s of data from 2 detectors sampled at 2048 Hz. Its architecture is derived from that of [75] 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 [75] 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 $1-$ s 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 2×10^5 samples.

During evaluation, each segment in the input file is whitened separately using the estimated PSD and sliced into 1-second segments at a step size of 0.1 s. 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 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 s. The algorithm is implemented using the PyTorch framework [97] and spawns child processes to whiten individual segments. The network evaluation is performed by the parent process.

E. 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 [112] was used instead of batch normalization as the first layer, to handle nonstationary timeseries. For the neural network architecture a deep, ResNet-like model [113] 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 600000 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 580000 waveforms, of which 300000 were used for the injections. For validation, one day of data was used, resulting in about 86000 noise segments and 3200 waveforms. The noise segments and waveform segments are combined online during

[6]The corresponding authors for the TPI FSU Jena submission are Ondřej Zelenka, Bernd Brügmann, and Frank Ohme.

[7]The corresponding authors for the Virgo-AUTh submission are Paraskevi Nousi, Nikolaos Stergioulas, Panagiotis Iosif, Alexandra E. Koloniari, Anastasios Tefas, and Nikolaos Passalis.

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 [97] and whitening is implemented as the first layer of the final detection module. Notably, this approach of computing the PSD for every 4.25s 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 0.125s (0.25s total) are removed from each sample.

The best results were obtained with a ResNet-52 type network. A Deep Adaptive Input Normalization (DAIN) layer [112] 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 stand alone publication containing details on the Virgo-AUTh submission can be found at [114].

F. cWB[8]

Coherent WaveBurst (cWB) is a waveform model-agnostic search pipeline for GW signals based on the constrained likelihood method [115–117]. 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 [13].

The cWB algorithm identifies the excess-power events in the time-frequency domain representation of strain data from multiple detectors [22,118]. 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) [119], was adopted and implemented within the cWB framework to automate the signal-noise classification of the cWB events [55]. 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 [55], the detection statistic for the machine learning-enhanced cWB algorithm is defined by

$$\eta_r = \eta_0 \cdot W_{\mathrm{XGB}}, \tag{7}$$

where, η_0 is cWB s ranking statistic, and W_{XGB} is the penalty factor calculated by XGBoost ranging between 0 (noise) and 1 (signal).

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 [57]: 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 subthreshold 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 reuse the XGBoost model which was developed for [57]: it should be noted that the model was trained on noise and signal events sets that differ substantially from those adopted for the datasets 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 [120] and with SEOBNRv4 waveforms [121]. 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.

IV. DATA RELEASE

We provide all source code as well as the evaluation results for all submissions at [86]. 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 Sec. V. The repository also contains code for basic visualization as part of the "contributions" folder. Adaptations of these scripts were used to create the

[8]The corresponding authors for the cWB submission are Francesco Salemi, Gabriele Vedovato, Sergey Klimenko, Tanmaya Mishra.

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 [86]. The submission code for the MFCNN group can be found at [95].

All analysis output files for all submissions created by our analysis are also publicly available and are stored in the "results" folder in [86]. 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 [86] 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.

V. RESULTS AND DISCUSSION

In this section we provide the results of our evaluation process described in Sec. II for all 6 submissions. We calculate and discuss sensitive distances, found-missed plots, and run-times 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 [62,64,76]. 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 [69,90]. Dataset 3 includes precession and higher order mode effects in the injected signals that traditional, modeled searches are not optimized for[9] [25–27]. 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 [123–126], which are strongly suppressed by sophisticated algorithms in traditional searches [17,92,125]. 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 [63,65,67,68] and we were, 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.

[9]A full search of the entire O3 data that includes higher order modes has been performed in [122].

A. 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 [100]. 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 Fig. 2 and Table IV. The four individual panels of Fig. 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 IV 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 Sec. III D and the Virgo-AUTh group presented in Sec. III E are very close in sensitivity for datasets 1, 2, and 3. The submission from the

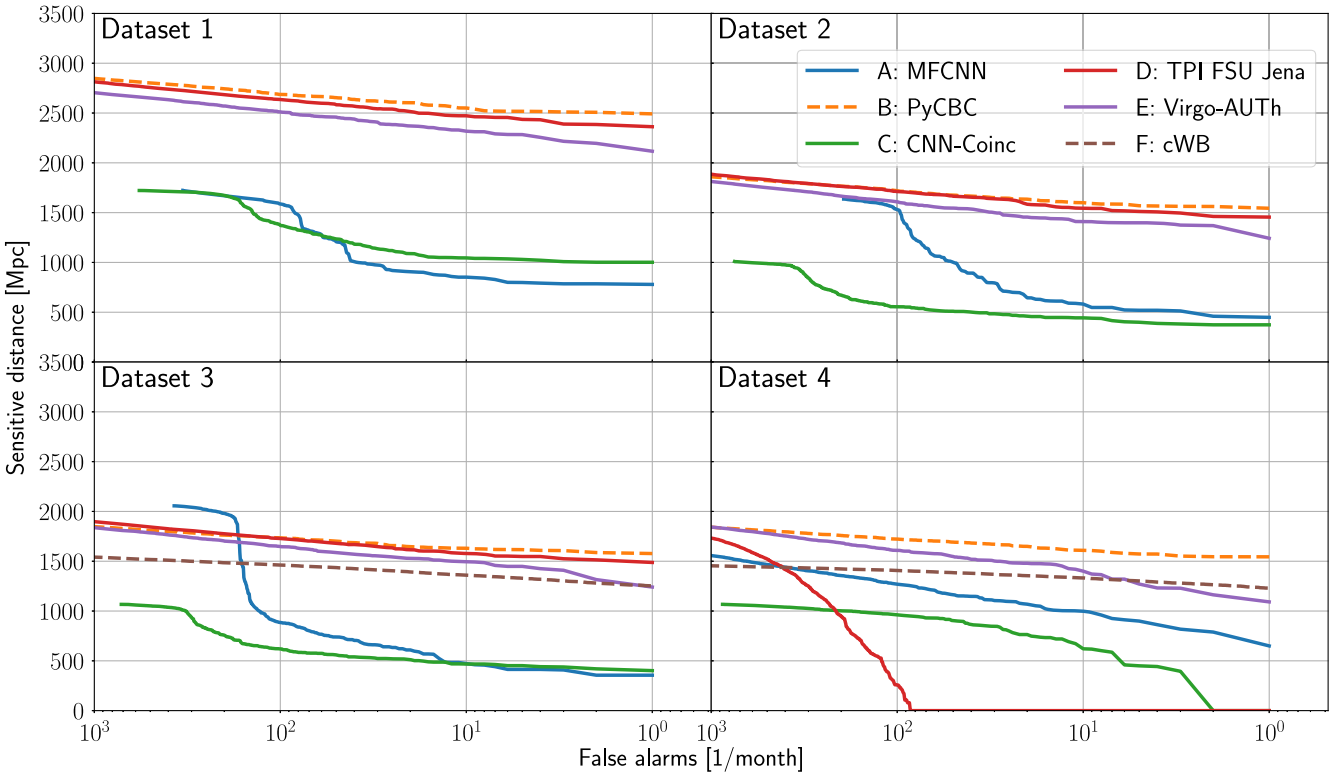

FIG. 2. The sensitive distances of all submissions and all four datasets as functions of the FAR. Submissions that made use of a machine learning algorithm at their core are shown with solid lines, others with dashed lines. The FAR was calculated on a background set that does not contain any injections.

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 Sec. III A and the CNN-Coinc group presented in Sec. III C 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

TABLE IV. 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 Fig. 2. The columns labeled "Run-time" list the time for evaluation of the foreground and background set in seconds, respectively. The run-time column labeled "average" lists the mean time obtained from evaluating the foreground and background data. Entries labeled "Not available" 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.

		Sensitivity [Mpc] at FAR $= x$ per month			Run-time [s]		
Dataset	Group	$x = 100$	$x = 10$	$x = 1$	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	...	...	...	...	...	...
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	...	...	...	...	...	...
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***	Not available	Not available
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***	Not available	Not available

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 Fig. 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 Fig. 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 Sec. III B. We also compare it to the submission from cWB presented in Sec. III F 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 [1,13,127].

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 [76], 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 [76,92].

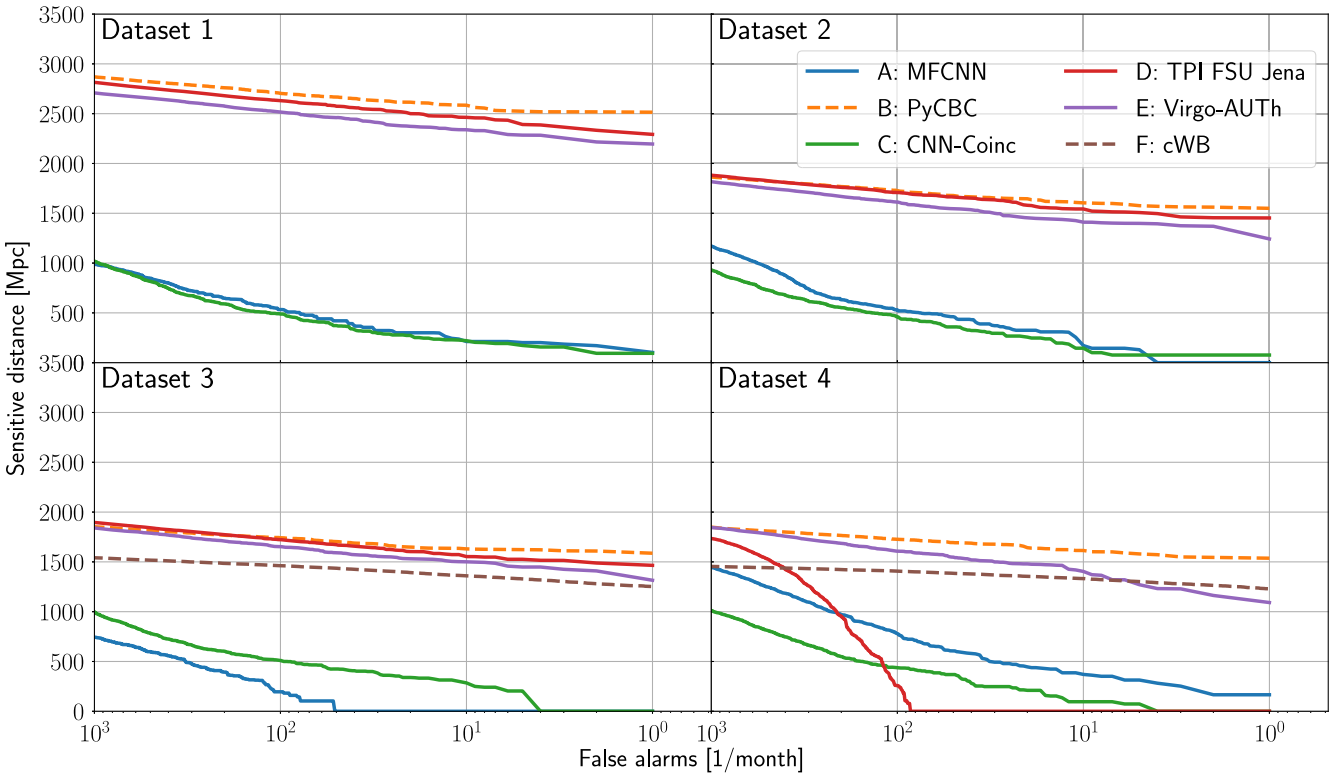

FIG. 3. The sensitive distances of all submissions and all four datasets as functions of the FAR. The sensitive distances are calculated using only the data from the foreground file. The FAR is determined from the false positives on that data. Submissions that made use of a machine learning algorithm at their core are shown with solid lines, others with dashed lines. This figure differs from Fig. 2 as the algorithms from MFCNN and CNN-Coinc behave differently on the foreground and the background.

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 [76] 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 inherently 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 nonzero 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 nonspinning 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 nonstationary 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 [63,65,67,70].

B. 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 [86]. 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 $D_{c,\mathrm{eff}}$, τ_0 versus mass-ratio q, and the effective precession spin χ_p [128] 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 [129,130]. 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 $M_c/D_{c,\mathrm{eff}}$ 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 Fig. 4 we show the found injections from dataset 1 in the M_c-$D_{c,\mathrm{eff}}$-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

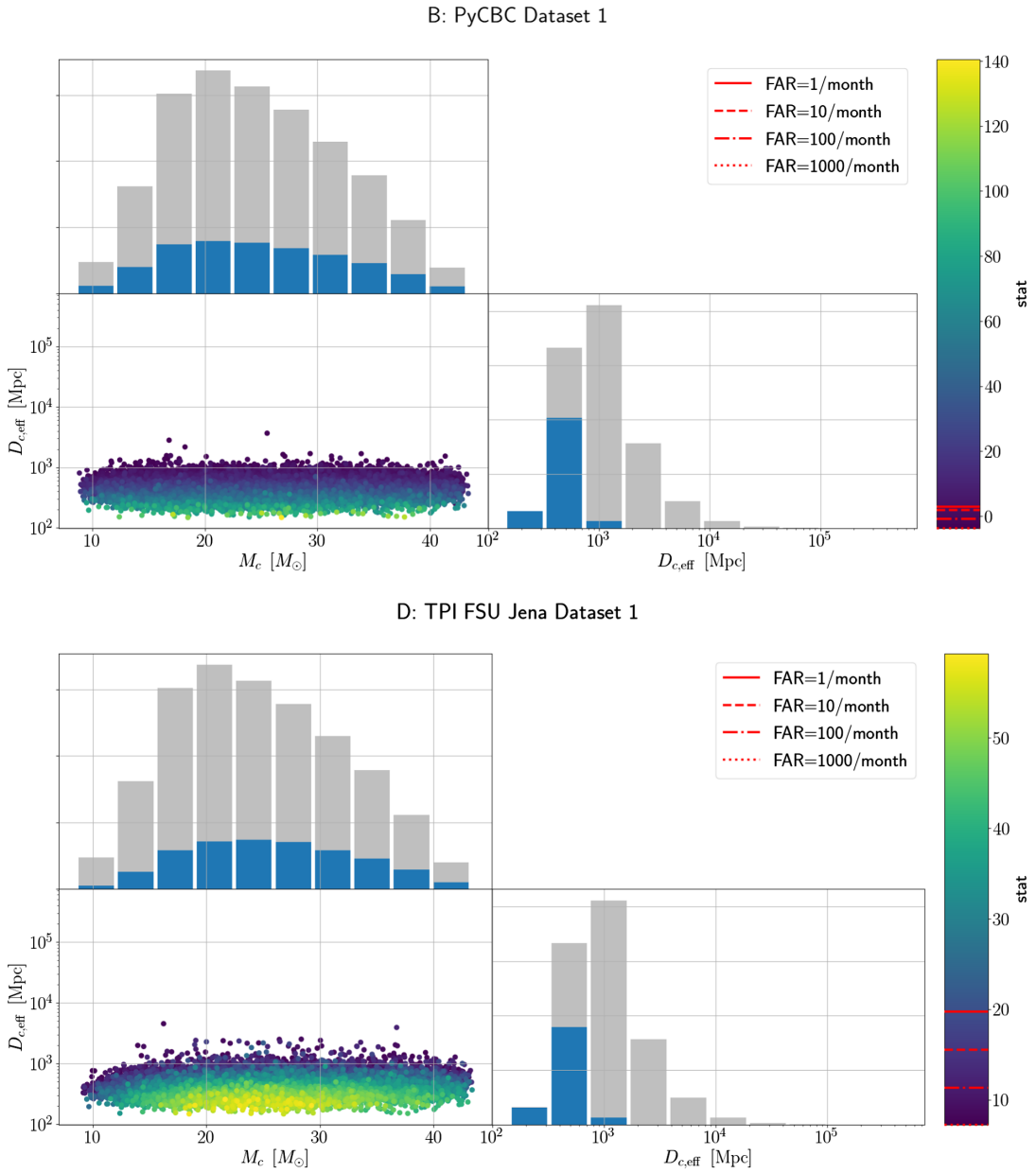

FIG. 4. The injections from dataset 1 identified by the PyCBC and TPI FSU Jena submissions with a FAR $\leq 10^3$ per month in the chirp-mass M_c versus decisive effective chirp-distance $D_{c,\mathrm{eff}}$ plane. The blue bars in the histograms show the one-dimensional marginal distributions of the 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 color bar highlight the thresholds on the "stat" to achieve different FARs.

be observed for all submissions. Furthermore, all signals with large $D_{c,\mathrm{eff}}$ are missed by both searches, showing that sensitivity estimates are not limited by the injected population. From Fig. 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 $M_\odot$ to 35 $M_\odot$. 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 Fig. 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

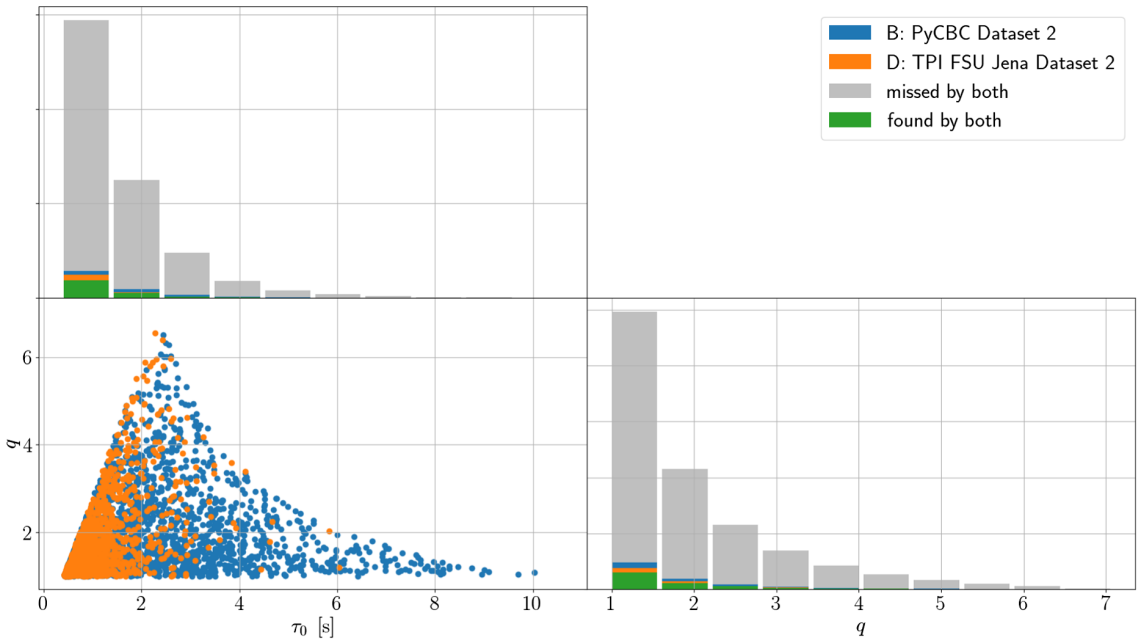

FIG. 5. The injections from dataset 2 identified by the PyCBC and TPI FSU Jena submissions with a FAR $\leq 10^3$ per month in the signal duration τ_0 versus mass ratio q plane. The scatter plot shows injections that are found only by one of the two algorithms. Injections that are missed or found by both are only shown in the 1D marginal distributions.

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 Fig. 6 we compare the θ_{jn} and χ_p values of the injections from dataset 3 that are found by one algorithm

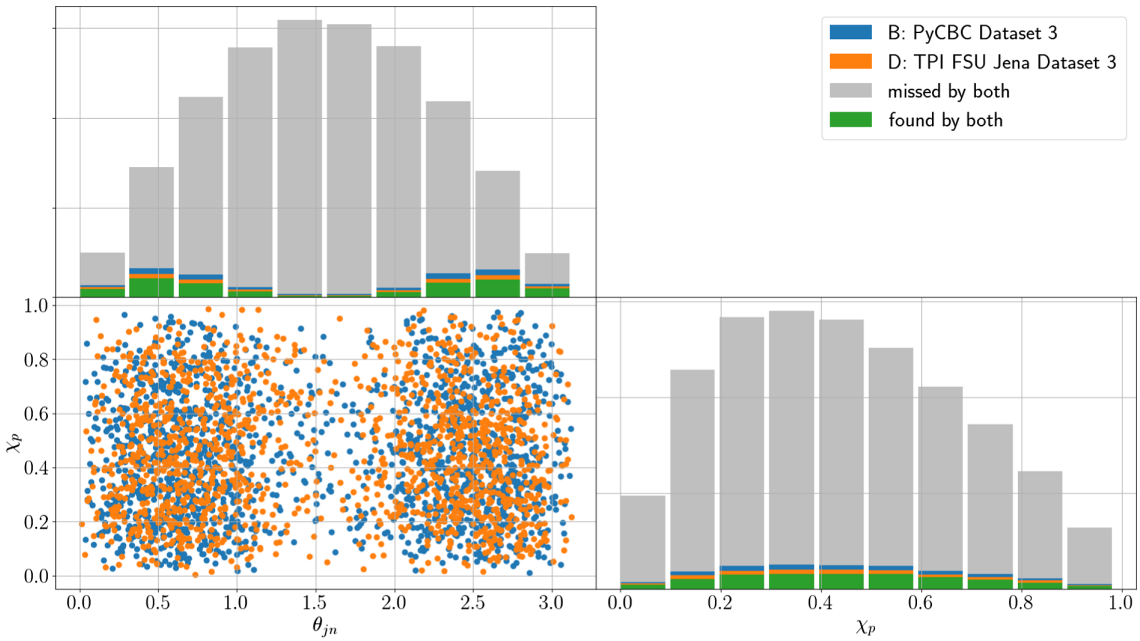

FIG. 6. The injections from dataset 3 identified by the PyCBC and TPI FSU Jena submissions with a FAR $\leq 10^3$ per month in the inclination to spin axis θ_{jn} to χ_p plane. The scatter plot shows injections that are found only by one of the two algorithms. Injections that are missed or found by both are only shown in the 1D marginal distributions.

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 nonprecessing 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.

C. Run-times

All run-times in this section are given in terms of wall-clock times obtained on equivalent hardware, which is listed in Table II. The run-times 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 IV. 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 run-times are heavily dependent on the amount of optimization of the algorithms. The main objective for this challenge was the sensitivity and not the run-time.

The PyCBC and cWB submissions are exceptions as their run-times 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 preprocessing 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 $\approx$11.5 h.

For dataset 1 the PyCBC submission has a run-time 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 run-time 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 preprocessing 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 [131]. One advantage of machine learning algorithms is that they make use of well optimized libraries such as PyTorch [97] or TensorFlow [132] 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 run-times 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 [92], 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 reevaluations 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.

VI. CONCLUSIONS

In this paper we have presented the results of the first Machine Learning Gravitational Wave Search Mock Data Challenge. 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 [92] pipeline and the unmodeled coherent wave burst search [22,118]. 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 are 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 [14] 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 [65,68,133]. We also did not receive submission based on many other neural network architectures that have been used in the past, such as autoencoders [74,81,82,134], inception networks [47,69], or two-dimensional convolutions that analyze time-frequency decompositions [70]. We hope that some of these approaches will be adapted 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, subsolar mass black holes could be targeted. Existing modeled searches in those regions make use of several million templates and are computationally limited [135]. 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.

ACKNOWLEDGMENTS

We want to thank Narenraju Nagarajan and Pascal Müller for their valuable scientific input and contributions to the code of this challenge. We acknowledge the Max Planck Gesellschaft and the Atlas cluster computing team at Albert-Einstein Institut (AEI) Hannover for support. O. Z. thanks the Carl Zeiss Foundation for the financial support within the scope of the program line

"Breakthroughs." The MFCNN team members would like to acknowledge that the submission was supported by the Peng Cheng Laboratory Cloud Brain (No. PCL2021A13). Z. C. was supported by the National Key Research and Development Program of China Grant No. 2021YFC2203001, the NSFC (No. 11920101003 and No. 12021003), and CAS Project for Young Scientists in Basic Research YSBR-006. The Virgo-AUTh team members would like to acknowledge the support provided by the IT Center of the Aristotle University of Thessaloniki (AUTh) throughout the progress of this work, as results presented in this work have been produced, in part, using the AUTh High Performance Computing Infrastructure and Resources, and thank the COST network CA17137 "G2Net" for support. P. I. acknowledges support by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No. 759253. The cWB team gratefully acknowledges the computational resources provided by LIGO-Virgo. This material is based upon work supported by NSF's LIGO Laboratory, which is a major facility fully funded by the National Science Foundation. This research has made use of data, software and/or web tools obtained from the Gravitational Wave Open Science Center, a service of LIGO Laboratory, the LIGO Scientific Collaboration and the Virgo Collaboration. The work by S. K. was supported by NSF Grants No. PHY 1806165 and No. PHY 2110060. This publication is based upon work from COST Action CA17137, supported by COST (European Cooperation in Science and Technology). E. A. H. is supported by Laboratory Directed Research and Development (LDRD) funding from Argonne National Laboratory, provided by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-06CH11357, and the U.S. National Science Foundation Grants No. OAC-2209892 and No. OAC-1931561. C. M. is supported by the Science and Technology Research Council (Grant No. ST/V005634/1) and the European Cooperation in Science and Technology (COST) action CA17137. F. O. was supported by the Max Planck Society's Independent Research Group Programme. This research has made use of data or software obtained from the Gravitational Wave Open Science Center (gw-openscience.org), a service of LIGO Laboratory, the LIGO Scientific Collaboration, the Virgo Collaboration, and KAGRA. LIGO Laboratory and Advanced LIGO are funded by the United States National Science Foundation (NSF) as well as the Science and Technology Facilities Council (STFC) of the United Kingdom, the Max-Planck-Society (MPS), and the State of Niedersachsen/Germany for support of the construction of Advanced LIGO and construction and operation of the GEO600 detector. Additional support for Advanced LIGO was provided by the Australian Research Council. Virgo is funded, through the European Gravitational Observatory (EGO), by the French Centre National de Recherche Scientifique (CNRS), the Italian Istituto Nazionale di Fisica Nucleare (INFN) and the Dutch Nikhef, with contributions by institutions from Belgium, Germany, Greece, Hungary, Ireland, Japan, Monaco, Poland, Portugal, Spain. The construction and operation of KAGRA are funded by Ministry of Education, Culture, Sports, Science and Technology (MEXT), and Japan Society for the Promotion of Science (JSPS), National Research Foundation (NRF) and Ministry of Science and ICT (MSIT) in Korea, Academia Sinica (AS) and the Ministry of Science and Technology (MoST) in Taiwan.

[1] B. P. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Phys. Rev. Lett. **116**, 061102 (2016).
[2] J. Aasi *et al.* (LIGO Scientific Collaboration), Classical Quantum Gravity **32**, 074001 (2015).
[3] F. Acernese *et al.* (Virgo Collaboration), Classical Quantum Gravity **32**, 024001 (2015).
[4] B. P. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Phys. Rev. Lett. **119**, 161101 (2017).
[5] B. P. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Phys. Rev. X **9**, 031040 (2019).
[6] A. H. Nitz, C. Capano, A. B. Nielsen, S. Reyes, R. White, D. A. Brown, and B. Krishnan, Astrophys. J. **872**, 195 (2019).
[7] A. H. Nitz, T. Dent, G. S. Davies, S. Kumar, C. D. Capano, I. Harry, S. Mozzon, L. Nuttall, A. Lundgren, and M. Tápai, Astrophys. J. **891**, 123 (2020).
[8] T. Venumadhav, B. Zackay, J. Roulet, L. Dai, and M. Zaldarriaga, Phys. Rev. D **100**, 023011 (2019).
[9] T. Venumadhav, B. Zackay, J. Roulet, L. Dai, and M. Zaldarriaga, Phys. Rev. D **101**, 083030 (2020).
[10] R. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Phys. Rev. X **11**, 021053 (2021).
[11] R. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), arXiv:2108.01045.
[12] A. H. Nitz, C. D. Capano, S. Kumar, Y.-F. Wang, S. Kastha, M. Schäfer, R. Dhurkunde, and M. Cabero, Astrophys. J. **922**, 76 (2021).
[13] R. Abbott *et al.* (LIGO Scientific, Virgo, KAGRA Collaborations), arXiv:2111.03606.
[14] A. H. Nitz, S. Kumar, Y.-F. Wang, S. Kastha, S. Wu, M. Schäfer, R. Dhurkunde, and C. D. Capano, arXiv:2112.06878.

[15] B. P. Abbott *et al.* (KAGRA, LIGO Scientific, Virgo Collaborations), Living Rev. Relativity **23**, 3 (2020).
[16] C. Cahillane and G. Mansell, Galaxies **10**, 36 (2022).
[17] C. Messick *et al.*, Phys. Rev. D **95**, 042001 (2017).
[18] T. Dal Canton, A. H. Nitz, B. Gadre, G. S. Davies, V. Villa-Ortega, T. Dent, I. Harry, and L. Xiao, Astrophys. J. **923**, 254 (2021).
[19] T. Adams, D. Buskulic, V. Germain, G. M. Guidi, F. Marion, M. Montani, B. Mours, F. Piergiovanni, and G. Wang, Classical Quantum Gravity **33**, 175012 (2016).
[20] B. Allen, W. G. Anderson, P. R. Brady, D. A. Brown, and J. D. E. Creighton, Phys. Rev. D **85**, 122006 (2012).
[21] S. Klimenko, S. Mohanty, M. Rakhmanov, and G. Mitselmakher, Phys. Rev. D **72**, 122002 (2005).
[22] S. Klimenko *et al.*, Phys. Rev. D **93**, 042004 (2016).
[23] R. Lynch, S. Vitale, R. Essick, E. Katsavounidis, and F. Robinet, Phys. Rev. D **95**, 104046 (2017).
[24] T. Akutsu *et al.* (KAGRA Collaboration), Nat. Astron. **3**, 35 (2019).
[25] I. Harry, S. Privitera, A. Bohé, and A. Buonanno, Phys. Rev. D **94**, 024012 (2016).
[26] I. Harry, J. Calderón Bustillo, and A. Nitz, Phys. Rev. D **97**, 023004 (2018).
[27] R. Dhurkunde and A. H. Nitz, Phys. Rev. D **106**, 103035 (2022).
[28] A. M. Deiana *et al.*, Front. Big Data **5**, 787421 (2022).
[29] M. AlQuraishi, Curr. Opin. Chem. Biol. **65**, 1 (2021), mechanistic Biology * Machine Learning in Chemical Biology.
[30] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh, Nature (London) **559**, 547 (2018).
[31] L. Gray, T. Klijnsma, and S. Ghosh, arXiv:2003.08013.
[32] M. Cranmer, A. Sanchez-Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho, arXiv:2006.11287.
[33] T. E. Smidt, M. Geiger, and B. K. Miller, Phys. Rev. Res. **3**, L012002 (2021).
[34] M. P. Oxley, M. Ziatdinov, O. Dyck, A. R. Lupini, R. Vasudevan, and S. V. Kalinin, npj Comput. Mater. **7**, 65 (2021).
[35] M. Dax, S. R. Green, J. Gair, M. Deistler, B. Schölkopf, and J. H. Macke, arXiv:2111.13139.
[36] M. Zevin *et al.*, Classical Quantum Gravity **34**, 064003 (2017).
[37] D. George, H. Shen, and E. A. Huerta, Phys. Rev. D **97**, 101501(R) (2018).
[38] S. Bahaadini, V. Noroozi, N. Rohani, S. Coughlin, M. Zevin, J. R. Smith, V. Kalogera, and A. Katsaggelos, Information Sciences (NY) **444**, 172 (2018).
[39] A. J. K. Chua and M. Vallisneri, Phys. Rev. Lett. **124**, 041102 (2020).
[40] H. Gabbard, C. Messenger, I. S. Heng, F. Tonolini, and R. Murray-Smith, Nat. Phys. **18**, 112 (2022).
[41] D. Chatterjee, S. Ghosh, P. R. Brady, S. J. Kapadia, A. L. Miller, S. Nissanke, and F. Pannarale, Astrophys. J. **896**, 54 (2020).
[42] M. Dax, S. R. Green, J. Gair, J. H. Macke, A. Buonanno, and B. Schölkopf, Phys. Rev. Lett. **127**, 241103 (2021).
[43] A. McLeod, D. Jacobs, C. Chatterjee, L. Wen, and F. Panther, arXiv:2201.11126.
[44] C. Dreissigacker, R. Sharma, C. Messenger, R. Zhao, and R. Prix, Phys. Rev. D **100**, 044009 (2019).
[45] F. Morawski, M. Bejger, and P. Ciecieląg, Mach. Learn. Sci. Tech. **1**, 025016 (2020).
[46] A. L. Miller *et al.*, Phys. Rev. D **100**, 062005 (2019).
[47] C. Dreissigacker and R. Prix, Phys. Rev. D **102**, 022005 (2020).
[48] B. Beheshtipour and M. A. Papa, Phys. Rev. D **101**, 064009 (2020).
[49] B. Beheshtipour and M. A. Papa, Phys. Rev. D **103**, 064027 (2021).
[50] T. S. Yamamoto, A. L. Miller, M. Sieniawska, and T. Tanaka, Phys. Rev. D **106**, 024025 (2022).
[51] C. Chatterjee, L. Wen, K. Vinsen, M. Kovalam, and A. Datta, Phys. Rev. D **100**, 103025 (2019).
[52] M. Cabero, A. Mahabal, and J. McIver, Astrophys. J. Lett. **904**, L9 (2020).
[53] S. Jadhav, N. Mukund, B. Gadre, S. Mitra, and S. Abraham, Phys. Rev. D **104**, 064051 (2021).
[54] M. J. Williams, J. Veitch, and C. Messenger, Phys. Rev. D **103**, 103006 (2021).
[55] T. Mishra, B. O'Brien, V. Gayathri, M. Szczepanczyk, S. Bhaumik, I. Bartos, and S. Klimenko, Phys. Rev. D **104**, 023014 (2021).
[56] D. Lopez, V. Gayathri, A. Pai, I. S. Heng, C. Messenger, and S. K. Gupta, Phys. Rev. D **105**, 063024 (2022).
[57] T. Mishra, B. O'Brien, M. Szczepańczyk, G. Vedovato, S. Bhaumik, V. Gayathri, G. Prodi, F. Salemi, E. Milotti, I. Bartos, and S. Klimenko, Phys. Rev. D **105**, 083018 (2022).
[58] C. McIsaac and I. Harry, Phys. Rev. D **105**, 104056 (2022).
[59] S. Khan and R. Green, Phys. Rev. D **103**, 064015 (2021).
[60] P. Nousi, S.-C. Fragkouli, N. Passalis, P. Iosif, T. Apostolatos, G. Pappas, N. Stergioulas, and A. Tefas, arXiv:2107.04312.
[61] S.-C. Fragkouli, P. Nousi, N. Passalis, P. Iosif, N. Stergioulas, and A. Tefas, arXiv:2203.08434.
[62] D. George and E. A. Huerta, Phys. Rev. D **97**, 044039 (2018).
[63] D. George and E. A. Huerta, Phys. Lett. B **778**, 64 (2018).
[64] H. Gabbard, M. Williams, F. Hayes, and C. Messenger, Phys. Rev. Lett. **120**, 141103 (2018).
[65] T. D. Gebhard, N. Kilbertus, I. Harry, and B. Schölkopf, Phys. Rev. D **100**, 063015 (2019).
[66] P. G. Krastev, Phys. Lett. B **803**, 135330 (2020).
[67] P. G. Krastev, K. Gill, V. A. Villar, and E. Berger, Phys. Lett. B **815**, 136161 (2021).
[68] W. Wei, A. Khan, E. A. Huerta, X. Huang, and M. Tian, Phys. Lett. B **812**, 136029 (2021).
[69] M. B. Schäfer, F. Ohme, and A. H. Nitz, Phys. Rev. D **102**, 063015 (2020).
[70] W. Wei and E. A. Huerta, Phys. Lett. B **816**, 136185 (2021).
[71] W. Wei, E. A. Huerta, M. Yun, N. Loutrel, M. A. Shaikh, P. Kumar, R. Haas, and V. Kindratenko, Astrophys. J. **919**, 82 (2021).
[72] E. A. Huerta *et al.*, Nat. Astron. **5**, 1062 (2021).
[73] V. Skliris, M. R. K. Norman, and P. J. Sutton, arXiv:2009.14611.

[74] F. Morawski, M. Bejger, E. Cuoco, and L. Petre, Mach. Learn. Sci. Technol. **2**, 045014 (2021).
[75] M. B. Schäfer, O. Zelenka, A. H. Nitz, F. Ohme, and B. Brügmann, Phys. Rev. D **105**, 043002 (2022).
[76] M. B. Schäfer and A. H. Nitz, Phys. Rev. D **105**, 043003.
[77] A. Khan and E. A. Huerta, Phys. Lett. B **835**, 137505 (2022).
[78] W.-H. Ruan, H. Wang, C. Liu, and Z.-K. Guo, arXiv: 2111.14546.
[79] P. Chaturvedi, A. Khan, M. Tian, E. A. Huerta, and H. Zheng, Front. Artif. Intell. **5**, 828672 (2022).
[80] G. Baltus, J. Janquart, M. Lopez, H. Narola, and J.-R. Cudell, Phys. Rev. D **106**, 042002 (2022).
[81] E. A. Moreno, B. Borzyszkowski, M. Pierini, J.-R. Vlimant, and M. Spiropulu, Mach. Learn. Sci. Technol. **3**, 025001 (2022).
[82] P. Bacon, A. Trovato, and M. Bejger, arXiv:2205.13513.
[83] E. Cuoco *et al.*, Mach. Learn. Sci. Technol. **2**, 011002 (2021).
[84] E. A. Huerta and Z. Zhao, *Handbook of Gravitational Wave Astronomy* (Springer, New York, 2021).
[85] E. G. O. EGO, G2net gravitational wave detection Kaggle challenge (2021), https://www.kaggle.com/competitions/g2net-gravitational-wave-detection/overview.
[86] M. Schäfer and O. Zelenka, MLGWSC-1 github repository (2021), https://github.com/gwastro/ml-mock-data-challenge-1.
[87] The HDF Group, Hierarchical Data Format, version 5 (1997-2022), https://www.hdfgroup.org/HDF5/.
[88] G. Pratten, C. García-Quirós, M. Colleoni, A. Ramos-Buades, H. Estellés, M. Mateu-Lucena, R. Jaume, M. Haney, D. Keitel, J. E. Thompson, and S. Husa, Phys. Rev. D **103**, 104056 (2021).
[89] L. S. Collaboration, LIGO Algorithm Library—LALSuite, free software (GPL) (2018), 10.7935/GT1W-FZ16.
[90] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016).
[91] R. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), SoftwareX **13**, 100658 (2021).
[92] S. A. Usman *et al.*, Classical Quantum Gravity **33**, 215004 (2016).
[93] B. Abbott *et al.* (LIGO Scientific Collaboration), Phys. Rev. D **77**, 062002 (2008).
[94] H. Wang, S. Wu, Z. Cao, X. Liu, and J.-Y. Zhu, Phys. Rev. D **101**, 104003 (2020).
[95] H. Wang, Mfcnn docker image (2022), https://hub.docker.com/r/iphysresearch/gwmlsc.
[96] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, arXiv:1512.01274.
[97] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (Neural Information Processing Systems, San Diego, 2019), pp. 8024–8035.
[98] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, arXiv:1905.02244.
[99] A. H. Nitz, A. Lenon, and D. A. Brown, Astrophys. J. **890**, 1 (2020).
[100] A. H. Nitz, T. Dal Canton, D. Davis, and S. Reyes, Phys. Rev. D **98**, 024050 (2018).
[101] L. Nuttall *et al.*, Classical Quantum Gravity **32**, 245005 (2015).
[102] M. Cabero, A. Lundgren, A. H. Nitz, T. Dent, D. Barker, E. Goetz, J. S. Kissel, L. K. Nuttall, P. Schale, R. Schofield, and D. Davis, Classical Quantum Gravity **36**, 155010 (2019).
[103] D. Davis and M. Walker, Galaxies **10**, 12 (2022).
[104] B. Allen, Phys. Rev. D **71**, 062001 (2005).
[105] A. H. Nitz, Classical Quantum Gravity **35**, 035016 (2018).
[106] A. H. Nitz, T. Dent, T. Dal Canton, S. Fairhurst, and D. A. Brown, Astrophys. J. **849**, 118 (2017).
[107] G. S. Davies, T. Dent, M. Tápai, I. Harry, C. McIsaac, and A. H. Nitz, Phys. Rev. D **102**, 022004 (2020).
[108] S. Husa, S. Khan, M. Hannam, M. Pürrer, F. Ohme, X. Jiménez Forteza, and A. Bohé, Phys. Rev. D **93**, 044006 (2016).
[109] S. Ioffe and C. Szegedy, in *Proceedings of the International Conference on Machine Learning* (PMLR, 2015), pp. 448–456, arXiv:1502.03167.
[110] M. Schäfer, Cnn-coinc github repository (2022), https://github.com/MarlinSchaefer/cnn-coinc.
[111] P. Welch, IEEE Trans. Audio Electroacoust. **15**, 70 (1967).
[112] N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, IEEE Trans. Neural Netw. Learn. Syst. **31**, 3760 (2019).
[113] K. He, X. Zhang, S. Ren, and J. Sun, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Institute of Electrical and Electronics Engineers, New York, 2016), pp. 770–778.
[114] P. Nousi, A. E. Koloniari, N. Passalis, P. Iosif, N. Stergioulas, and A. Tefas, arXiv:2211.01520.
[115] M. Drago, S. Klimenko, C. Lazzaro, E. Milotti, G. Mitselmakher, V. Necula, B. O'Brian, G. A. Prodi, F. Salemi, M. Szczepanczyk, S. Tiwari, V. Tiwari, G. V, G. Vedovato, and I. Yakushin, SoftwareX **14**, 100678 (2021).
[116] S. Klimenko, G. Vedovato, V. Necula, F. Salemi, M. Drago, R. Poulton, E. Chassande-Mottin, V. Tiwari, C. Lazzaro, B. O'Brian, M. Szczepanczyk, S. Tiwari, and V. Gayathri, cwb pipeline library: 6.4.1 (2021), 10.5281/zenodo.5798976.
[117] cWB Development Team, coherent WaveBurst Homepage (2021), https://gwburst.gitlab.io/.
[118] S. Klimenko and G. Mitselmakher, Classical Quantum Gravity **21**, S1819 (2004).
[119] T. Chen and C. Guestrin, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16 (ACM, New York, NY, USA, 2016), pp. 785–794, 10.1145/2939672.2939785.
[120] C. Talbot and E. Thrane, Astrophys. J. **856**, 173 (2018).
[121] A. Bohé *et al.*, Phys. Rev. D **95**, 044028 (2017).
[122] K. Chandra, J. Calderón Bustillo, A. Pai, and I. Harry, Phys. Rev. D **106**, 123003 (2022).

[123] J. Aasi *et al.* (LIGO Scientific, Virgo Collaborations), Classical Quantum Gravity **32**, 115012 (2015).
[124] B. P. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Classical Quantum Gravity **33**, 134001 (2016).
[125] B. P. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Classical Quantum Gravity **37**, 055002 (2020).
[126] D. Davis, M. Trevor, S. Mozzon, and L. K. Nuttall, Phys. Rev. D **106**, 102006 (2022).
[127] B. P. Abbott *et al.* (LIGO Scientific, Virgo Collaborations), Phys. Rev. D **93**, 122004 (2016); **94**, 069903(A) (2016).
[128] P. Schmidt, F. Ohme, and M. Hannam, Phys. Rev. D **91**, 024043 (2015).
[129] T. Cokelaer, Phys. Rev. D **76**, 102004 (2007).
[130] M. Maggiore, *Gravitational Waves. Vol. 1: Theory and Experiments*, Oxford Master Series in Physics (Oxford University Press, New York, 2007).
[131] R. Dhurkunde, H. Fehrmann, and A. H. Nitz, Phys. Rev. D **105**, 103001 (2022).
[132] M. Abadi *et al.*, TensorFlow: Large-scale machine learning on heterogeneous systems (2015), software available from www.tensorflow.org.
[133] A. Schmitt, K. Fu, S. Fan, and Y. Luo, in *Proceedings of the 2nd International Conference on Computer Science and Software Engineering*, CSSE 2019 (Association for Computing Machinery, New York, NY, USA, 2019), pp. 73–78, 10.1145/3339363.3339377.
[134] H. Shen, D. George, E. A. Huerta, and Z. Zhao, in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2019), 10.1109/ICASSP.2019.8683061.
[135] A. H. Nitz and Y.-F. Wang, Phys. Rev. D **106**, 023024 (2022).