institut für windenergiesysteme

Leibniz Universität Hannover

Fakultät für Bauingenieurwesen
und Geodäsie

Model updating of wind turbine blade cross sections with invertible neural networks

PREPRINT

Submitted to:
Wind Energy Journal
Submission Date:
24. April 2021

Pablo Noever-Castelos[1,*]
Lynton Ardizzone[2]
Claudio Balzani[1]

[1]*Leibniz University Hannover, Institute for Wind Energy Systems, Hanover, Germany*
[2]*Heidelberg University, Visual Learning Lab, Heidelberg, Germany*

[*]**Corresponding author:**
Pablo Noever-Castelos
research@iwes.uni-hannover.de

Abstract

Fabricated wind turbine blades have unavoidable deviations from their designs due to imperfections of the manufacturing processes. Model updating is a common approach to enhance model predictions and therefore improve the numerical blade design accuracy compared to the built blade. An updated model can provide a basis for a digital twin of the rotor blade including the manufacturing deviations. State of the art in structural model updating are classical optimization algorithms most often combined with reduced order or surrogate models. However, these deterministic methods suffer from high computational costs and a missing probabilistic evaluation.

This study approaches the model updating task by inverting the model through the application of Invertible Neural Networks, which allow for inferring a posterior distribution of the input parameters from given output parameters, without costly optimization or sampling algorithms. In our use case, rotor blade cross sections are updated to match given cross sectional parameters. To this end, a sensitivity analysis of the input and output parameters first selects relevant features in advance to then set up and train the Invertible Neural Network.

The trained network predicts with outstanding accuracy most of the cross sectional input parameters for different radial positions, i.e. the posterior distribution of the features show a narrow width. At the same time, it identifies some parameters that are hard to recover accurately or contain intrinsic ambiguities. Hence, we demonstrate that invertible neural networks are highly capable for structural model updating.

1 Introduction

Wind turbine blades are huge and complex structures that are exposed to extreme load conditions. Thus, an accurate blade design is of fundamental importance for the turbine's safety and reliability. As for most engineering structures, primarily numerical models build the design basis for rotor blades. However, manufacturing deviations lead to a mismatch in structural behaviour of the numerically designed and rotor blades built in real life.[16] These deviations may be crucial even within the allowed tolerances and material parameter uncertainties. Consequently, enhancing virtual models by means of model updating is an important aspect of a modern blade design procedure. Where model updating seeks to correct the inaccurate parameters of the numerical model in order to improve test result predictions.[33] This method is either applied for calibrating the model with conducted real life tests [23, 27] or to detect damages in terms of structural health monitoring.[44] The updated model provides a basis for a digital twin of the rotor blade built.[43]

Model updating most commonly takes the form of an optimization problem: this optimization can either directly manipulate the modelling parameters (e.g. material properties, layup, etc.) or take corrective actions in the final model itself (e.g. stiffness or mass matrix of a beam model) [53]. For both approaches, metaheuristic algorithms such as genetic or particle swarm algorithms are commonly used to solve the optimization problem.[6] Such deterministic model updating algorithms (e.g. global pattern search [18]) have been applied successfully in the field of rotor blade damage detection. But all these algorithms yield exactly one result for the model parameters and do not cover possible result ambiguity, which can emerge due to a lack of sensitive output parameters. This uncertainty worsens the user's confidence about the updated model parameters, as more than one configuration may yield the given output results.[32] Depending on the algorithm it even may get stuck in local optima and depend on the randomness of the starting samples.[12] Bayesian inference algorithms solve this issue by predicting posterior distributions for the updated parameters, which lets the user estimate the prediction confidence. Popular methods for this are Bayesian model updating [46] or Approximate Bayesian Computation [49], among others.

All the aforementioned approaches for model updating suffer from the same general drawback: the prohibitively high computational cost of repeatedly simulating the physical model. This is especially severe for the probabilistic algorithms such as Bayesian model updating, where techniques like Markov Chain Monte Carlo sampling are needed. Practitioners try to avoid this problem by using surrogate models, that are faster to compute than the full physical model, to cut down on the computational costs.[45] These surrogate models can take the form of reduced order models [51] or other reduction techniques such as the response-surface method [27]. However, the surrogate model approach in turn sacrifices physical input-output linkage of the original model, and may lead to a loss in accuracy depending on the abstraction level and the model complexity itself.[50]

Machine learning techniques, specifically artificial neural networks (ANNs), can help address these issues of model updating in various ways. Most importantly, they can be trained to map the relationship between input and output parameters highly accurate, without knowledge of the physical connections.[8] In this way, they can serve as surrogate models that may be dramatically faster to compute or more accurate than other types of surrogates.[31, 47, 48, 26] They have also been successfully applied as surrogates in Bayesian model updating.[14, 54] However, combining ANNs and model updating algorithms in such a way requires a lot of implementation effort, tuning, and software engineering. Even then, using ANNs as surrogates may not achieve the desired speed-up, as they do not remove the fundamental limitations of having to compute the surrogate model many times, which is part of optimiziation-based or sampling-based model updating. In principle, ANNs could also be trained to directly predict the desired parameters,

circumventing the need for an optimization procedure all together. While they are orders of magnitude faster than any traditional model updating techniques, the main problem is that they lack indications of confidence, uncertainty, or goodness of fit, and are hard to rigorously verify. Due to this, standard ANNs are rarely used in this direct way for the purpose of model updating.

Within this difficult setting, we present the main idea of this paper: we use Invertible Neural Networks (INNs) as probabilistic models to directly produce a posterior distribution of the input parameters. During training, the network receives the model parameters as inputs, as would be the case with a surrogate model ANN. At test time however, the network can be inverted to directly produce samples from the posterior, without having to perform additional algorithms. This approach offers a significant potential speed-up over traditional model updating techniques, even ones using ANNs as surrogate models. At the same time, we obtain a full Bayesian posterior that allows among other things determining confidence intervals and uncovering ambiguities, in the same way that is otherwise reserved for computationally expensive Bayesian model updating algorithms. In contrast, this is not possible with existing direct ANN approaches or standard optimization-based model updating procedures.

Our experiments clearly demonstrate the practical applicability and benefit of these INNs in the research field of structural rotor blade model updating: The INN predicts highly accurate material and layup parameters based on cross sectional beam properties, as well as offering verifiable uncertainty estimates, and identifying some ambiguous and unrecoverable parameters.

To the best of our knowledge, ANNs have not yet been applied for structural model updating of wind turbine blades, especially not in the form of probabilistic models such as INNs. Instead, the major application of ANNs in the topic of wind turbines is the field of controls (e.g. model predictive control [17], adaptive control [21], yaw control [42], and aerodynamic coefficient prediction for control [13]), and for condition or structural health monitoring considering fault or damage prediction.[3, 29, 30, 25, 36] INNs have been introduced relatively recently, even in the field of machine learning itself [9, 10, 22, 41, 20], but have seen rapidly growing research attention in the last years. They have been successfully applied in a broad field of application, commonly in image processing but also in scientific studies. In this paper, we specifically adapt and apply the conditional Invertible Neural Network (cINN) [1] implemented in the FrEIA Framework [52].

Section 2 of this paper covers the overall workflow description, with explanations on the feature selection method based on a sensitivity analysis. The approach and architecture of a cINN is briefly addressed in the sections 2.2 and 2.3, respectively. Subsequently the feature selection results are presented in section 3. The cINN parameter definition, training and evaluation is reported in section 4, followed by the conclusion in section 5.

2 Model Updating Methodology with Invertible Neural Networks

This section describes the methods used in this investigation to analyse the input and output parameters of the model updating procedure and how a neural network is structured and trained for inverse problems. In this publication we will restrict the problem to rotor blade cross section analysis and evaluate the capability for structural model updating of invertible neural networks on a first level in wind turbine blade design processes. Figure 1 illustrates the overall workflow for this study, which the following subsections will discuss in more detail. Shortly summarized the approach consists of a data preprocessing step in form of a sensitivity analysis to identify relevant input and output features of the model. The selected input features are then randomly sampled. Based on these samples, cross sectional properties of the wind turbine blade at a particular radial position is calculated and filtered according to the feature selection. The workflow splits

these sample sets of input and output features into training, validation and testing subsets for the cINN (the validation set is used to check progress of the training and tune the network settings. The test set is only used for the final evaluation of the method, to avoid biasing the results). The data generation is based on a rotor blade model within the in-house modelling tool MoCA (Model Creation and Analysis Tool for Wind Turbine Rotor Blades) and its interface to BECAS (BEam Cross section Analysis Software) [5].

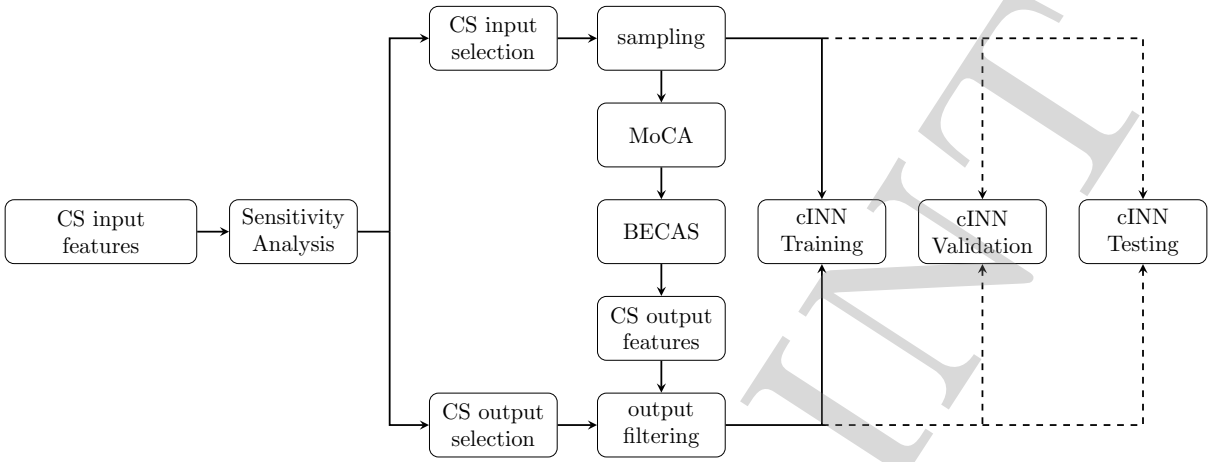

Figure 1: Overall workflow of this study. Based on all cross sectional input features a sensitivity analysis is performed to determine the relevant in- and output features. From these training, validation, and test sets are sampled for the conditional invertible neural network (cINN).

2.1 Sensitivity Analysis of Blade Cross Sectional Properties

Data preprocessing plays an important role to build a proper dataset specially for neural networks, but also for machine learning problems in general.[39] This contribution focuses on a sensitivity analysis to perform feature selection.[19] The feature selection technique reduces the number of input features to a subset which has a significant impact on the output features, based on the assumption that some data contains irrelevant or redundant information.[7] Additionally the output features are reduced by all insensitive components, as these can not be expressed with respect to the given input features.

In the example of cross sectional model updating we will be focusing on material and geometrical composite layup parameters. The material parameters include Young's modulus E_{11} (for anisotropy additionally: E_{22}, E_{33}) and density ρ. The geometrical layup parameters are described by the layup division point locations on the cross section's circumference, that subdivide the composite layup of the cross section. However, overall blade shell geometry is assumed to be well known, as 3D laser scanning can offer an accurate measurement of the blade outer shell/ blade mould.[28] Figure 2 illustrates the sensitivity analysis for a simplified example of three input features x: Young's Modulus E, density ρ, and one division point P, as well as three output features y: Mass M, Stiffness K, and Area A.

During the sensitivity analysis all selected input features x are varied individually. From all created parameter subsets the cross sectional property response is calculated with MoCA and BECAS for a particular blade radius. All these subsets are than concatenated to a full database, labelled as *CS results* in Figure 2. Then each output feature y is standardized to $\bar{y} = 0$ and $\sigma = 1$ over the full database and denoted as $\hat{y}$. This simplifies the sensitivity evaluation of each feature, as $\hat{y}$ describes the output feature's deviation magnitude for each sample in relation to all other samples.

After standardization the full database is split again into the subset, i.e. variation of one input feature x. In Figure 2 denoted as σ is the calculation of the standard deviation $\sigma_{y,x}$ over each subset's output feature,

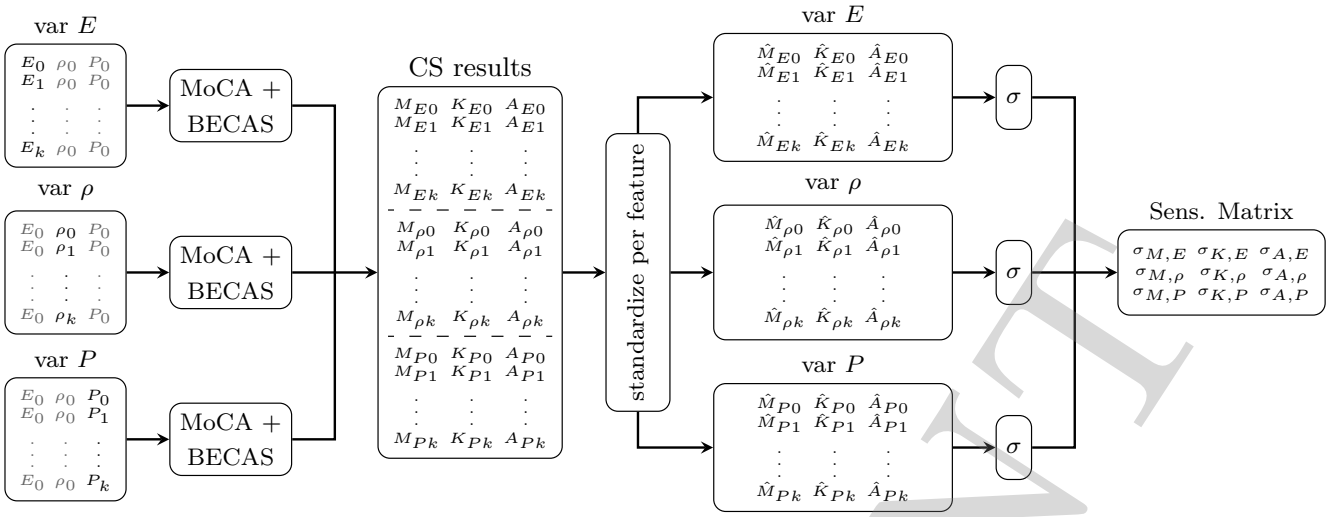

Figure 2: The feature selection process based on a sensitivity analysis applied to a simplified cross sectional (CS) example with three input features Young's modulus E, density ρ and division point position P. The algorithm varies all individually and calculates their corresponding CS characteristics, here exemplary the mass M, stiffness K and area A. After global feature standardization, splitting into the previous sets and computing the standard deviation, the process returns a reduced sensitivity matrix. This can be used for feature selection.

which is then collected in the Sensitivity Matrix. Through this, all input features x, have a single value for each output feature y showing the input feature's impact on that respective output feature. Finally by defining a threshold λ the sensitivity analysis identifies irrelevant input features, in case $\sigma_{y,x} < \lambda$ for all y. On the other hand an insensitive output feature y is discarded, if $\sigma_{y,x} > \lambda$ applies for all x. Additionally the algorithm reduces all linearly dependent output features to one, as the others do not include further information for the training process of the neural network.

2.2 Invertible Neural Networks for Inverse Problems

The general setting described in the introduction is shared across many fields in engineering and natural science: the problem is well understood and modelled in the forward process, i.e. the observed response y can be readily calculated based on some parameters x that describe a system (from mechanics, physics, chemistry, medicine). However, scientists are commonly interested in the corresponding *inverse problem*, i.e. computing possible states x of the underlying system given observations y. Computing this inverse can be a highly challenging task, and common approaches such as classical model updating have some intrinsic shortcomings, as was briefly laid out in the introduction: Firstly, they are often computationally very expensive, as the forward process has to be computed or simulated many times to fit a set of system parameters $\hat{x}$ that matches a given set of observations. Secondly, especially for safety-relevant applications, obtaining a single guess $\hat{x}$ is not sufficient: ideally, any ambiguities in the solution as well as its uncertainty should be captured and precisely quantified. This can be fulfilled by a (Bayesian) posterior distribution $p_x(x \mid y)$.[38] The posterior quantifies the probability that any system state x could have led to the observations y, and allows producing confidence intervals, or discovering ambiguous or unrecoverable system parameters.[2]

An approach that alleviates both of these difficulties, and has seen growing adoption in recent years, is the use of conditional Invertible Neural Networks (cINNs) to reliably model the full posterior distribution in a computationally efficient way. Such networks were first successfully applied to image processing such as inpainting [9], colorization of grayscale images [1], synthetic image generation [22], but more recently in

many scientific fiels such as astro-physics [24], particle physics [4], medical imaging [15], and recently in epidemiology [37]. In short, cINNs rely on a simple reference distribution $p_z(z)$ called the *latent distribution*, most commonly a Gaussian. The cINN f then conditionally transforms and reshapes between the posterior $p_x(x \mid y)$ and the latent distribution $p_z(z)$. (see Fig. 3).

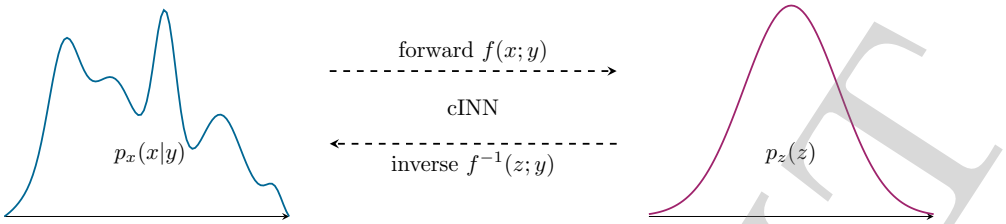

Figure 3: Schematic illustration of the principle of a cINN. The cINN f conditionally transforms and reshapes between the posterior $p_x(x \mid y)$ and the latent distribution $p_z(z)$.

From this construction, the posterior that the network represents can be exactly computed through the change-of-variables formula as follows [9]:

$$p_x(x \mid y) \quad = \quad p_z\big(f(x;y)\big) \left| \det \left(\frac{\delta f}{\delta x} \right) \right| \tag{1}$$

Here, $\det\left(\frac{\delta f}{\delta x}\right)$ denotes the determinant of the models Jacobian, $\det(J)$ for short from here on. Similarly, samples from the posterior can be drawn by first sampling z from the latent distribution $p_z(z)$, and using the *inverted* cINN to transform them to the domain of the posterior: $x = f^{-1}(z;y)$. As with many classic probabilistic modelling techniques, the cINN can be trained through maximum likelihood training. This means that given existing pairs of (x_i, y_i), the models posterior $p_x(x \mid y)$ will match the true posterior of the inverse problem $p^*(x \mid y)$ if the average log-likelihood of the models posterior is maximized, or as done in practice, the negative log likelihood (NLL) is minimized. Together with the change-of-variables formula, and a Gaussian latent distribution $p_z(z) \propto \exp(\|z\|^2/2)$, we arrive at the following objective:

$$\mathcal{L}_{\text{NLL}} = \mathbb{E}\Big[-\log\big(p(x_i \mid y_i)\big)\Big] = \mathbb{E}\left[\frac{\|f(x_i;y_i)\|^2}{2} - \log\left|\det(J_i)\right| \right] + const. \tag{2}$$

For a more detailed explanation and derivation of the objective function, see e.g. [1].

2.3 Conditional Invertible Neural Network's Architecture and Training

From the previous section, we can conclude that the neural network we use to represent f must be invertible and have a way to readily compute the Jacobian determinant. In the following, we describe the implementation of the cINN architecture, that satisfies these requirements. In general, the cINN consists of a sequence of so-called coupling blocks, specifically affine coupling blocks in our case. To this end, the (unconditional) coupling blocks introduced in RealNVP [10] can be extended to include the condition y, shown below in Figure 4.

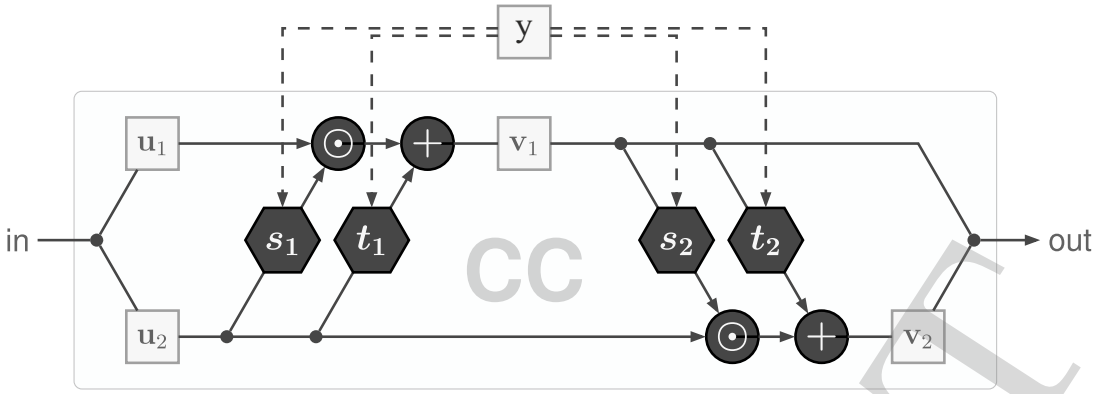

Figure 4: Structure of a conditional affine coupling block (CC).[1]

This block first splits the input data u into $[u_1, u_2]$ and applies affine transformation according to the following functions:

$$v_1 = u_1 \odot \exp(s_1(u_2, y)) + t_1(u_2, y) \quad (3)$$

$$v_2 = u_2 \odot \exp(s_2(v_1, y)) + t_2(v_1, y) \quad (4)$$

The results $[v_1, v_2]$ are concatenated afterwards to v. Inverting the set of equations yields these inverse operation:

$$u_2 = (v_2 - t_2(v_1, y)) \oslash \exp(s_2(v_1, y)) \quad (5)$$

$$u_1 = (v_1 - t_1(u_2, y)) \oslash \exp(s_1(u_2, y)) \quad (6)$$

The internal functions s_j and t_j always take as input the corresponding variables u_2 or v_1 and additionally the conditional data y. As these functions must not be inverted they can be replaced by any arbitrary mathematical expression, in our case by shallow standard neural networks that will be referred to as subnetworks. A big advantage of this coupling block is the simplicity to compute the logarithm of the Jacobian determinant being the sum of s_1 and s_2 over the inputs dimension.[10] In addition to the plain coupling blocks, we include a number of technical improvements common for invertible network architectures, such as fixed permutations between variables. In order to improve generalization of the cINN we apply drop-out layers in the subnetworks, as well as L2 weight regularization [35]. Gradient clipping avoids exploding gradients in back propagation [40] and an optimizer's learning rate scheduler improves the convergence [35].

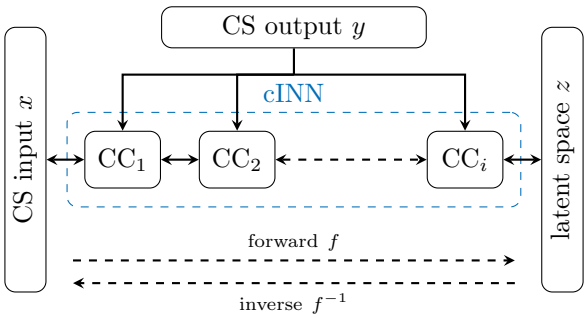

Figure 5: The conditional invertible neural network (cINN) structure applied to cross sectional model updating. in the forward path f Cross sectional (CS) input features x are processed over a sequential concatenation of conditional coupling (CC) blocks, which represents the cINN. The CS output features y contribute as coupling block conditions. And the cINN result is a latent space z. This path can be inverted, which is defined as f^{-1}.

3 Input and Output Feature Selection

The sensitivity analysis and feature selection as described in section 2.1 will focus in this study particularly on the cross section at a radial position of R = 6 m of the SmartBlades2 DemoBlade. This cross section is depicted in Figure 6 as a BECAS output with material assignment and division point (P) location. In a cross section, division points divide the shell into different sections with a constant material layup, or define sub-component positions such as the web location.

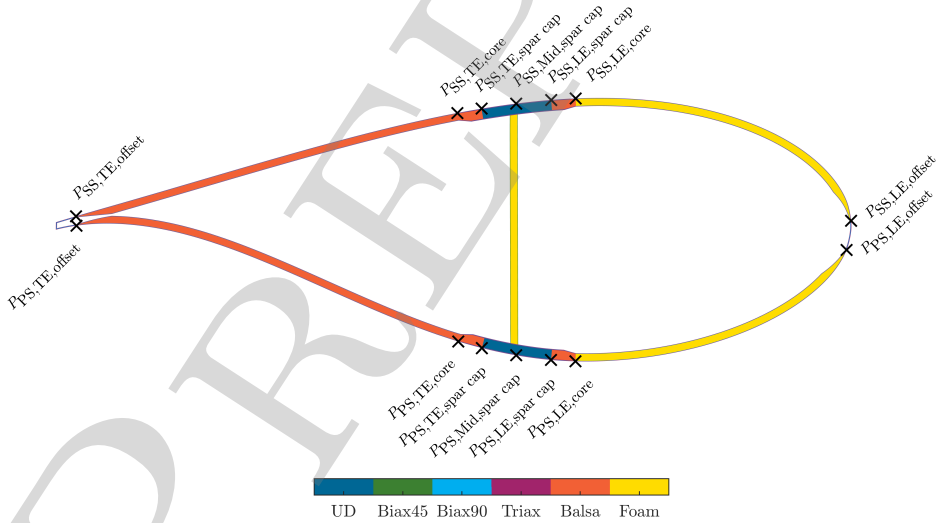

Figure 6: This is a cross section of the SmartBlades2 DemoBlade at a radial position of R = 6 m. The division points on the circumference divide the blade shell into sections of equal material layup as follows: $P_{SS,TE,offset}$ to $P_{SS,TE,core}$ is the suction side (SS) sandwich panel located at the trailing edge (TE); $P_{SS,TE,spar\ cap}$ to $P_{SS,LE,spar\ cap}$ denotes the location of the spar cap; this if flanked by balsa transition pieces in-between $P_{SS,TE,core}$ to $P_{SS,LE,core}$; followed by the sandwich panel located to the leading edge (LE) from $P_{SS,LE,core}$ to $P_{SS,LE,offset}$. The pressure side (PS) is build accordingly. As the outer face laminates are hard to identify due to their small thickness, the layup is show in Table A.1 ind the appendix A

Additionally to the cross sectional view, Table A.1 in the Appendix A contains the layup of each cross sectional sub-component at R = 6 m for a full insight to the analysed structure. This may enhance the interpretation of the following sensitivity analysis in this section. The spar cap is prefabricated with balsa

transition pieces on each side of the uni-directional glass fibre (UD) material and trimmed to the correct size, before placing it in the blade mould.

The input feature variation is selected upon manufacturing tolerances for materials and layup of wind turbine blades. In case of the SmartBlades2 DemoBlade manufacturing admits in general a tolerance of approximately maximmum ±5 % deviation for material parameters, such as densities and Young's moduli. The ply positioning tolerances in cross sectional direction, i.e. division point locations, depend on the material; valid tolerances for core material are ±5 mm, whereas spar cap and web location may vary ±5-10 mm maximum. In order to account for even higher inaccuracies the analysis range was extended for each parameter as stated in Table 1. Under consideration that the spar cap is prefabricated all related positions varied together, i.e. all suction side division points from $P_{SS,TE,core}$ to $P_{SS,LE,core}$ are moved simultaneously, same holds for the pressure side respectively.

Table 1: Parameter variation range for sensitivity analysis and neural network training. The prefabricated spar cap is varied synchronous for each shell side.

Parameter	Attribute	Max. Variation
UD	E_{11}, E_{22}, E_{33}, ρ	± 10%
Biax 45°	E_{11}, E_{22}, E_{33}, ρ	± 10%
Biax 90°	E_{11}, E_{22}, E_{33}, ρ	± 10%
Triax	E_{11}, E_{22}, E_{33}, ρ	± 10%
Balsa	E_{11}, ρ	± 10%
Foam	E_{11}, ρ	± 10%

Parameter	Attribute	Max. Variation	Note
$P_{SS,TE,offset}$	Location	± 10 mm	
$P_{SS,TE,core}$	Location	± 15 mm	Prefabr.: varied synchr.
$P_{SS,TE,spar\ cap}$	Location	± 15 mm	Prefabr.: varied synchr.
$P_{SS,Mid,spar\ cap}$	Location	± 15 mm	Prefabr.: varied synchr.
$P_{SS,LE,spar\ cap}$	Location	± 15 mm	Prefabr.: varied synchr.
$P_{SS,LE,core}$	Location	± 15 mm	Prefabr.: varied synchr.
$P_{SS,LE,offset}$	Location	± 10 mm	
$P_{PS,LE,offset}$	Location	± 10 mm	
$P_{PS,LE,core}$	Location	± 15 mm	Prefabr.: varied synchr.
$P_{PS,LE,spar\ cap}$	Location	± 15 mm	Prefabr.: varied synchr.
$P_{PS,Mid,spar\ cap}$	Location	± 15 mm	Prefabr.: varied synchr.
$P_{PS,TE,spar\ cap}$	Location	± 15 mm	Prefabr.: varied synchr.
$P_{PS,TE,core}$	Location	± 15 mm	Prefabr.: varied synchr.
$P_{PS,TE,offset}$	Location	± 10 mm	

After generating the model in MoCA and processing it with BECAS, the output features in Table 2 are available. These include cross sectional locations of shear, elastic, area, and mass centre, as well as total mass, total area, inertias and principal axis orientation. However, probably the most important output are the stiffness and mass matrices which serve as input for finite element beam models.

Following the sensitivity algorithm described in section 2.1 a sensitivity matrix is computed based on the parameter variation listed in Table 1 and the cross sectional output variables in Table 2. The full sensitivity matrix is given in the Appendix A in the Tables A.2-A.4. A threshold value $\lambda = 0.25$ was chosen to identify irrelevant features, which are than greyed out in the matrix. This led to discarding the input features E_{22} and E_{33} of each glass fibre laminate as well as the Young's Modulus E of both core materials. As the prefabricated spar cap was moved simultaneously the algorithm additionally rejects 4 of 5 linearly dependent division

Table 2: BECAS cross sectional output parameters

Variable	Description
SC_x, SC_y	Shear centre (SC) coordinates
EC_x, EC_y	Elastic centre (EC) coordinates
M_{total}	Total mass
CoG_x, CoG_y	Centre of Gravity (CoG) coordinates
I_{xx}, I_{yy}, I_{xy}	Mass moment of inertia
A_x, A_y	Area centre cordinates
A_{xx}, A_{yy}, A_{xy}	Area moment of inertia
A_{total}	Total areas
$\alpha_{PC,Ref}$	Orientation Principal Axis
$\alpha_{PC,ECent}$	Orientation Principal Axis w.r.t. EC

Variable	Description
$\begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} & K_{15} & K_{16} \\ & K_{22} & K_{23} & K_{24} & K_{25} & K_{26} \\ & & K_{33} & K_{34} & K_{35} & K_{36} \\ & & & K_{44} & K_{45} & K_{46} \\ & & & & K_{55} & K_{56} \\ & & & & & K_{66} \end{bmatrix}$	Stiffness Matrix
$\begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} & M_{15} & M_{16} \\ & M_{22} & M_{23} & M_{24} & M_{25} & M_{26} \\ & & M_{33} & M_{34} & M_{35} & M_{36} \\ & & & M_{44} & M_{45} & M_{46} \\ & & & & M_{55} & M_{56} \\ & & & & & M_{66} \end{bmatrix}$	Mass Matrix

points, keeping the $P_{XX,Mid,spar\ cap}$ as representative for each shell sides. Regarding the output, the algorithm sorts out several insensitive stiffness (K_{13}, K_{23}, K_{14}, K_{24}, K_{15}, K_{25}) and mass features (M_{12}, M_{13}, M_{23}, M_{14}, M_{24}, M_{15}, M_{25}, M_{36}, M_{46}, M_{56}). Lastly, due to linear dependencies M_{11} was kept but M_{total}, M_{22}, M_{33} was discarded and additionally I_{xx}, I_{yy}, I_{xy}, M_{34}, M_{35}, were sorted out in favour of M_{44}, M_{55}, M_{45}, M_{16}, M_{26}, respectively. This feature selection yields the final reduced sensitivity matrix in Table 3. All remaining 14 input features are used to generate training and validation/test samples for the cINN, while the selected 33 output features from the respective samples are passed as conditions to the cINN.

Table 3: Reduced Sensitivity Matrix of Cross Section at R=6 m. The greyed values are below the threshold $\lambda = 0.25$. A maximum of ± 10 % variation applies for material input features, ± 15 mm for the spar cap division points $P_{XX,Mid,spa\ cap}$ and ± 10 mm for the other division points.

	SC_x	SC_y	EC_x	EC_y	CoG_x	CoG_y	$Area_x$	$Area_y$	A_{xx}	A_{yy}	A_{xy}	$Area_{total}$	$\alpha_{PC,Ref}$	$\alpha_{PC,EC}$	K_{11}	K_{12}	K_{22}	K_{33}	K_{34}	K_{44}	K_{35}	K_{45}	K_{55}	K_{16}	K_{26}	K_{66}	M_{11}	M_{44}	M_{45}	M_{55}	M_{16}	M_{26}	M_{66}
$E_{11,UD}$	0.2	0.8	3.6	3.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.3	2.3	2.0	0.2	2.1	3.8	0.1	0.1	4.0	1.5	4.4	0.7	0.2	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ρ_{UD}	0.0	0.0	0.0	0.0	3.4	2.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.7	0.0	1.4	3.9	0.0	2.8	0.7
$E_{11,Biax90}$	0.2	0.3	1.5	1.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.9	0.7	0.1	0.8	1.4	2.2	2.5	1.2	0.5	0.8	0.3	0.1	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ρ_{Biax90}	0.0	0.0	0.0	0.0	1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.6	2.0	0.7	1.1	1.3	1.4	2.0
$E_{11,Triax}$	0.0	0.4	2.1	1.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.3	1.4	1.0	0.1	1.4	2.5	3.5	4.0	1.6	0.9	1.5	0.4	0.3	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ρ_{Triax}	0.0	0.0	0.0	0.0	1.1	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.6	2.9	1.2	1.9	1.9	1.7	3.0
ρ_{Balsa}	0.0	0.0	0.0	0.0	1.1	3.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.2	3.1	1.6	1.3	3.9	1.5	3.0
ρ_{Foam}	0.0	0.0	0.0	0.0	0.8	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.1	0.4	0.1	0.7	0.9	1.0	0.5
$P_{SS,TE,offset}$	0.4	0.0	0.1	0.0	0.1	0.3	0.8	2.2	2.6	1.1	0.9	2.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.5	0.4	0.1	0.3	0.1	0.0	0.3	0.1	0.3
$P_{SS,Mid,spar\ cap}$	2.8	3.3	1.6	1.8	2.4	0.9	0.9	1.5	1.5	0.8	2.8	0.4	2.6	2.8	3.6	3.3	3.6	0.0	1.6	0.1	1.5	3.0	0.1	3.4	4.0	3.0	0.0	0.0	2.7	0.1	0.7	2.4	0.0
$P_{SS,LE,offset}$	0.1	0.0	0.1	0.0	0.2	0.1	1.9	1.9	1.4	2.2	1.0	2.3	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.2	0.9	0.0	0.1	0.1	0.0	0.1	0.2	0.0
$P_{PS,LE,offset}$	0.0	0.1	0.1	0.0	0.4	0.1	3.9	1.9	1.4	3.7	1.6	2.4	0.0	0.0	0.1	0.0	0.2	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.1	1.0	0.0	0.0	0.1	0.0	0.1	0.4	0.0
$P_{PS,Mid,spar\ cap}$	3.8	3.3	0.4	1.8	0.9	1.0	1.2	1.6	1.5	1.2	3.2	0.2	2.8	2.5	2.0	3.3	1.4	0.0	1.6	0.1	0.4	3.2	0.0	3.2	2.2	3.0	0.1	0.0	3.0	0.0	0.7	0.8	0.0
$P_{PS,TE,offset}$	0.5	0.0	0.1	0.0	0.2	0.3	0.4	2.3	2.7	0.7	0.5	2.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.8	0.5	0.1	0.3	0.1	0.0	0.3	0.2	0.3

4 Invertible Neural Network Structure, Training and Evaluation

After having identified the significant in- and output features of the model, in this upcoming section we seek appropriate cINN hyper-parameters to subsequently train and evaluate the invertible neural network. Furthermore, the network is selected with respect to its computational training costs and applicability on other related scenarios.

4.1 Identifying Network Hyper-Parameters

While the network parameters (i.e. the network weights) are produced by the training process, the network size and structure, length of the training procedure, and other settings, have to be set by the user beforehand. These are known as hyper-parameters, and we describe our choices in the following. All the programming is done within Pytorch [34] including the FrEIA Framework for INNs.[52] As previously stated, the cINN is a sequence of conditional affine coupling blocks (CC), with subnetworks acting as internal functions. The subnetworks are represented by standard feed-forward neural networks consisting of a number of hidden layers (network depth), each with a certain number of nodes (network width). Every hidden layer is followed by a drop-out layer to improve generalisation, and an activation layer. To find a set of well-performing hyper-parameters for the cINN, we trained networks with various different depths and widths as depicted in Table 4. The hyper-parameter tuning revealed that shallow but wide subnetworks are favourable for this application. The AdaGrad [11] optimization algorithm gave the best and fastest convergence, which is finally improved by a learning rate scheduler. Due to huge magnitude differences between the features all passed samples are standardized per feature ($\bar{x} = 0$ and $\sigma_x = 1$) to equalize the contribution magnitude of each one. For further information on artificial neural network's terminology please refer for example to [8].

Table 4: Final cINN hyper-parameter set. Note, the parameters where chosen subjectively, involving the number of epochs and the learning rate and its scheduler, while focusing more on computationally cheaper hyper-parameter sets may also achieve reasonably good accuracy with lower computational costs.

model	No. CC	Subnet nodes	Subnet layer	Activation Function	Drop-out rate	Optimizer	Learning rate	Batch size	Epochs	Samples	Training time
0	2	100					0.2				65 mins
1	4	200	1	PReLu	0.05	AdaGrad	0.2	32	1000	20,000	132 mins
2	8	400					0.2				199 mins
3	16	800					0.15				466 mins

In order to determine the necessary network depth of the cINN we will evaluate the four trained models from Table 4 against their prediction quality of the input feature's posterior. If not otherwise stated all given results will be shown as standardized values to improve direct comparability between the features. Considering that all input features were sampled uniformly within their respective symmetric maximum variation $\pm x_{\text{max}}$ from Table 1, the standard deviation is defined as:

$$\sigma_x \quad = \quad |x_{\text{max}}| \cdot \sqrt{\frac{1}{3}} \tag{7}$$

Equation (7) helps to estimate the real range of the respective input feature's posterior distribution. Furthermore, to enhance the understanding of the upcoming discussion, we use Figure 7 to explain the interpretation of the results for two exemplarily chosen features: $E_{11,\text{Biax90}}$ and ρ_{Biax90}. The two left graphs represent the posterior distribution of the respective features after evaluating the cINN inversely. Each sample is a set of input and output feature. This original input feature value is the so called ground truth, which the cINN tries to predict as accurately as possible. Therefore, all given results are related to the

ground truth value of each sample feature. This enables the comparison of different samples with varying ground truth values and it lets the reader recognize the accuracy of the prediction at a glance.

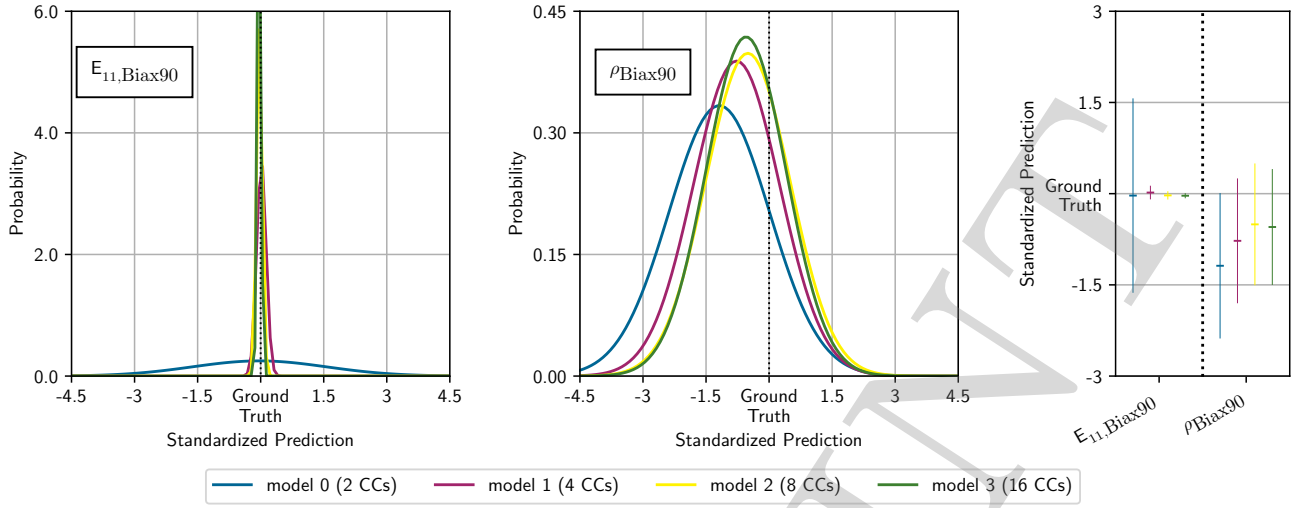

Figure 7: Exemplary standardized prediction for the two input features $E_{11,Biax90}$ and ρ_{Biax90} of one sample computed with all four models from Table 4. The left two graphs show the predicted distribution of the corresponding feature x. The right graph summarizes the same results as error bars with $1 \cdot \sigma_x$ width around the predicted mean value $\bar{x}$. The results are related to the ground truth value.

What is striking about the left graph is the improved prediction with increasing network depth. For all models the mean value is relatively close to the ground truth ($-0.11 \leq \bar{E}_{11,Biax90} \leq 0.02$), though the standard deviation of the shallowest model ($\sigma_{E_{11,Biax90}} = 1.59$) is much higher than of the deepest ($\sigma_{E_{11,Biax90}} = 0.04$). That means the shallowest model has a poor prediction confidence for $E_{11,Biax90}$ compared to the deeper models. In contrast, the input feature ρ_{Biax90} is predicted very similarly by all models. Here, the width of the posteriors is generally also much higher, indicating that ρ_{Biax90} can not be recovered with as great of a precision as $E_{11,Biax90}$, even for the deeper and more powerful cINN architecture. Another interesting fact is that the posteriors are all approximately Gaussian (as opposed to having multiple peaks, skewed shape, etc.). This could help justify even simpler methods in future, that may only provide Gaussian uncertainty estimates. Without producing the full non-parametric posteriors first with the cINN, such simplifying assumptions could not be made. Both first graphs can be summarized as presented in the right graph. There, the prediction moves to the y-axis and for each feature the posterior is depicted as error bars with $1 \cdot \sigma_x$ width around the mean value $\bar{x}$, making it easy to compare several features and models at a glance.

Having explained how to interpret the inverted model results, we will now move on to define the best network depth from the given models in Table 4. Therefore, Figure 8 shows all four model's prediction of each input feature's posterior for 10 randomly chosen samples. It is directly apparent from this figure that most of the features are predicted extremely accurate with a high confidence by the deeper models, except for both densities of the glass fibre plies Biax90 and Triax.

These two density features can not be recovered accurately enough by none of the given models due to an ambiguity resulting from their relatively similar mass contribution and quasi identical position in the cross section. Recovering from Figure 6 above and Table A.1 in the Appendix A, the Triax and Biax90 are placed directly upon each other in the shell sandwich laminate with a similar nominal thickness. Thus, they counteract each other, i.e. if one density raises the other diminishes to achieve together the same total weight and inertia contribution. This behaviour is clearly approved by Figure 9, where the mean value of the learned posterior of the samples is scattered along a thin line against each other, i.e. the features

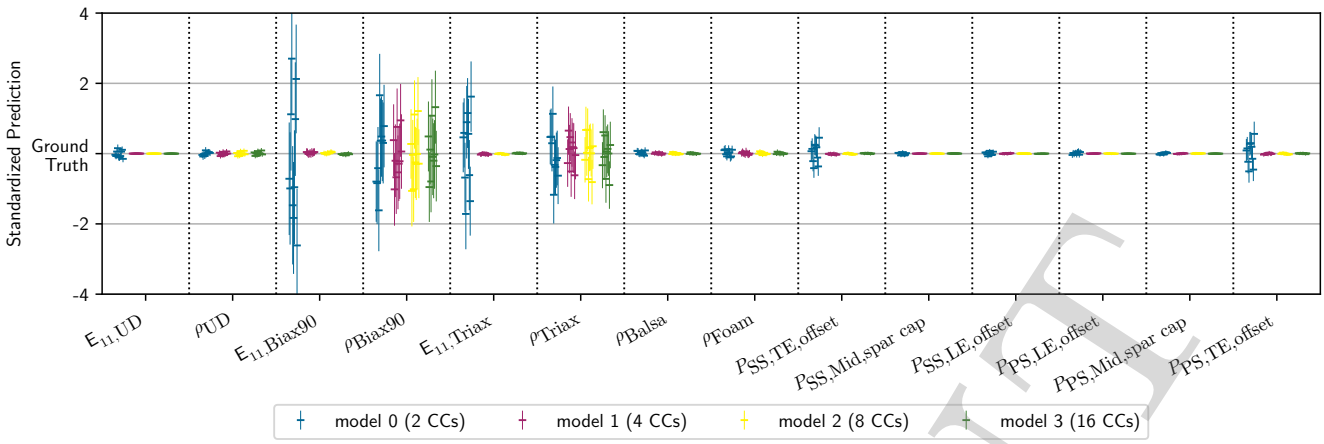

Figure 8: Input feature prediction showed for 10 random samples of four different models with full output as conditional features. The four invertible neural networks increase in depths.

are negatively proportional and highly correlated R^2=0.9989. From this, we can conclude that the cINN has correctly detected the ambiguity and represents it accordingly in the posterior. However, it is able to quite precisely predict a merged density of both values as following results will show. Another interesting point, due to a greater nominal thickness (+38 %) and its existence in the prefabricated Spar Cap the Triax' is slightly more dominant, which reflects in a marginally better prediction and thus narrower posterior distribution compared to the Biax90 (cf. Figure 8).

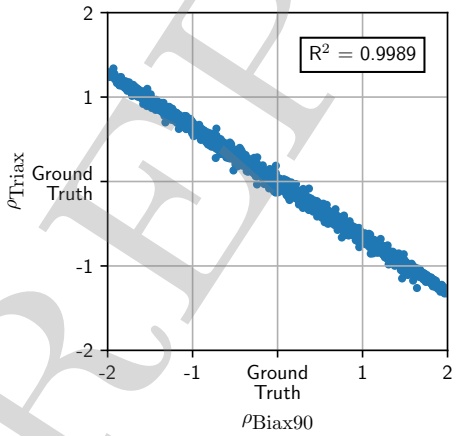

Figure 9: Counteraction of the mean predicted densities ρ_{Biax90} and ρ_{Triax}. The highly correlating samples show that the cINN correctly detects this ambiguity.

As stated above, the overall inference performance of the models is strikingly accurate, specially from model 1 (4 CCs) on. Figure 8 shows similar result qualities for the models 1-3, with only little improvements in the standard deviation in each step. In addition Table 4 describes a computational time increase by 50 % from model 1 to 2 and approx. 130 % from model 2 to 3. Thus, the authors decided to choose the model 2 (8 CCs) hyper-parameter set as cINN design. In addition to the depth selection, the necessary training sample size and number of epochs for a fixed learning rate of 0.2 is analysed to further computational cost cut down. All scenarios showed satisfying posteriors, therefore a sample size of 8,000 and 1,000 epochs was chosen as trade-off between computational time (108 mins) and accuracy. Table 5 summarizes the final cINN hyper-parameter set. To considering the complete process to create a cINN, the sample generation has to be taken into account, which is a significant cost driver for classical iterative model updating techniques.

On a 40 node computing cluster, generating 8,000 samples with MoCA and BECAS with the given mesh density of 500 elements per circumference takes approx. 30 mins.

Table 5: Final cINN hyper-parameter set. Note, the parameters where chosen subjectively, involving the number of epochs and the learning rate and its scheduler, while focusing more on computationally cheaper hyper-parameter sets may also achieve reasonably good accuracy with lower computational costs.

No. CC	Subnet nodes	Subnet layer	Activation Function	Drop-out rate	Optimizer	Learning rate	Batch size	Epochs	Samples	Training time
8	400	1	PReLu	0.05	AdaGrad	0.2	32	1000	8,000	108 mins

4.2 Invertible Network Evaluation

After having defined our cINN, this section will continue with the evaluation of its performance and applicability for different scenarios. First of all we will recap the input feature prediction of the finally chosen model and its training process. From the data in the left graph of Figure 10, the good prediction quality is again apparent, except for the two counteracting input features: ρ_{Biax90} and ρ_{Triax}. The latter two features are mostly unrecoverable due to an inherent ambiguity of the problem, as discussed above. During training, the negative log likelihood loss curve on the right graph is monotonically decreasing with little steps every 100th epoch, where the scheduler reduces the learning rate by 20 % of the actual rate. The validation loss is even lower due to the averaging effect of the drop-out layers mimicking multiple trained models.[40] However, both seem to have nearly converged to their optimum. In order to extend

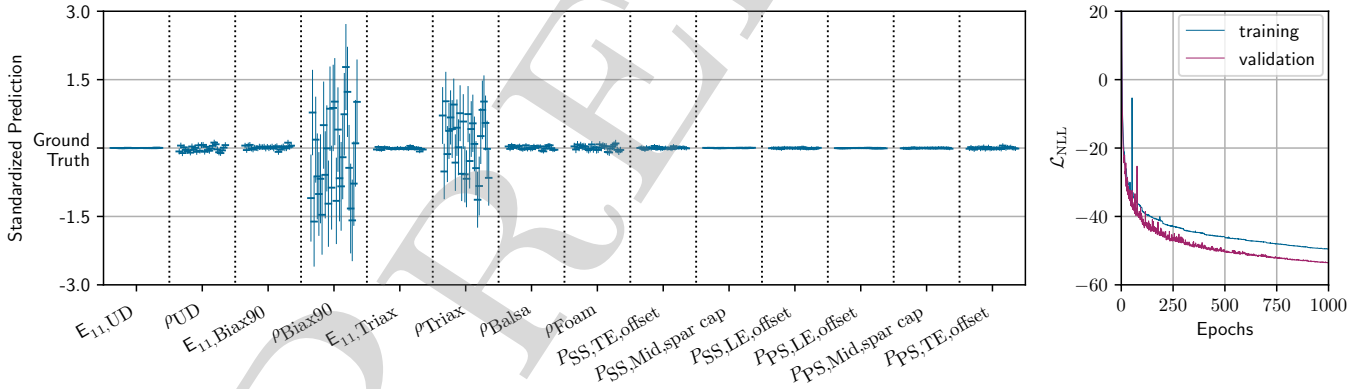

Figure 10: Final model: Input feature prediction on the left graph for 30 random samples with full output as conditional features. Right graph depicts the negative log likelihood loss curve of the cINN for training and validation samples.

the analysis of the posterior distribution, the correlation of an input feature prediction (mean value $\bar{(x)}$) against its ground truth value can be investigated. Therefore we use the coefficient of determination (R^2). Figure B.1 in the Appendix B compares for 10,000 random samples the linear correlation of each inferred input feature, also stating each feature's R^2-score, which in most cases is $R^2 > 0.998$ except for the two previously discussed densities. The computed correlation affirms the previous outstanding predictions for a wide range of samples. Next, these predicted mean values are used to recalculate the output features with MoCA and BECAS to evaluate its accuracy. All recalculated values match extremely accurately the ground truth values as proved by the given R^2-scores, which are all approx. 1 with a roundoff error at the

5th decimal digit. Interestingly, the inaccuracy of ρ_{Biax90} and ρ_{Triax} seem to cancel each other out, due to their counteraction. Considering the proximity of both laminates and that both are infused together, a merged or averaged density for both materials could improve the prediction of such a parameter in future applications.

So far we have demonstrated an excellent model updating capability of the finally designed cINN for a cross section at R = 6 m. Hereafter, the same cINN will be trained for cross sections at the following positions R = 9 m, 12 m, 15 m. Figure 11 presents the posterior predictions for all four radial locations considering 10 random samples. Here again, the predictions are outstanding, except for ρ_{Balsa}. This rests upon the fact that balsa is replaced by Foam in the trailing edge panels after R = 6 m and only appear in the transition pieces of the spar cap, as shown exemplarily in Figure B.2 in the Appendix B. Hence the contribution to any mass property and sensitivity is comparatively low to gain enough information to recover this input feature. The wide posteriors produced by the cINN show that it has correctly understood and modeled this uncertainty, instead of having the same high confidence as for the other parameters.

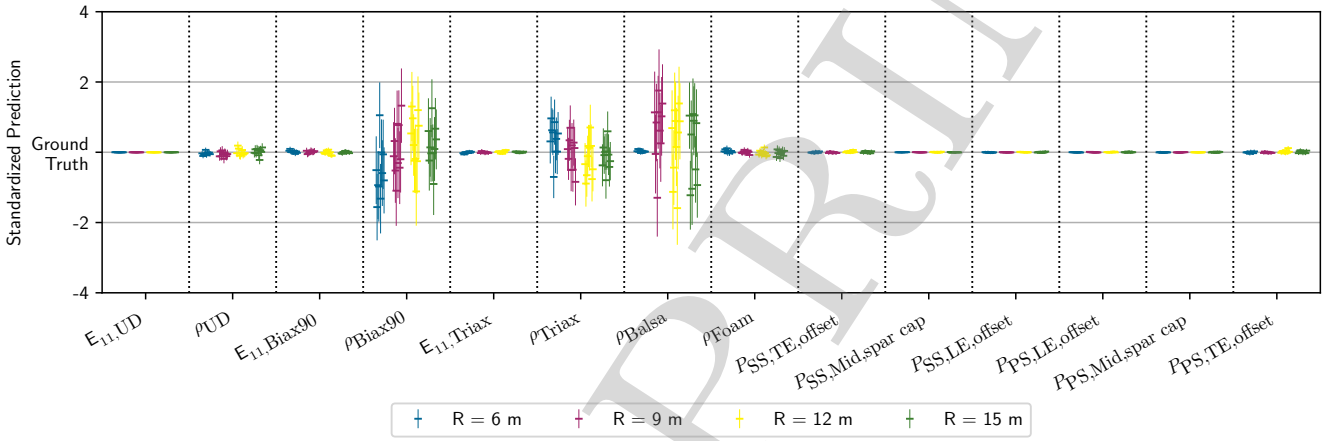

Figure 11: Input feature prediction showed for 10 random samples at four different radial positions with full output as conditional features. The four cINNs were each trained individually for their respective radius.

As a final aspect, considering a future case of a finite element beam model updating, the possible updated features from it are the stiffness and mass matrix. Therefore analysing the prediction quality trained only with these two matrices as conditional features indicates if the basic material and layup input features can be subsequently inferred. Figure 12 shows the already familiar posterior prediction graph for the four previously analyzed cross sectional positions only considering the stiffness and mass matrix features. The overall width of the posterior distributions increase slightly for all cross sections and features, but is still reasonably accurate. Only the original cINN for the cross section at R = 6 m noticeably looses accuracy at several features ($E_{11,Biax90}$, $E_{11,Triax}$, $P_{SS,TE,offset}$, $P_{SS,LE,offset}$). This only presents the easiest and most straight forward way of inferring the input posteriors, though recovering the full output parameter set from the stiffness and mass matrix before inferring the input features is also possible with a few calculations.[5]

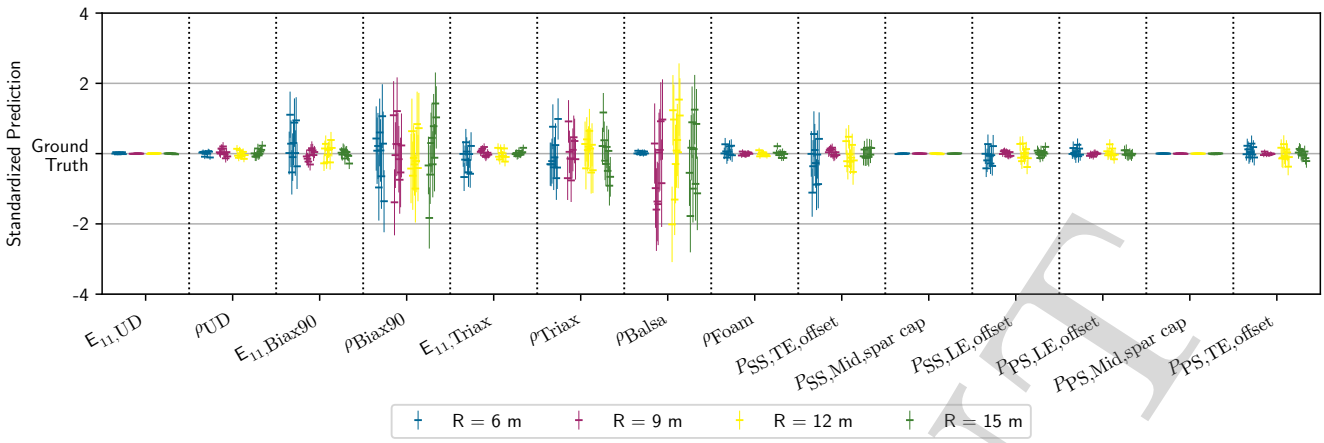

Figure 12: Input feature prediction showed for 10 random samples at four different radial positions only considering the stiffness and mass matrix as conditional features. The four cINNs were each trained individually for their respective radius.

5 Conclusion

This investigation set out to reveal the capability of invertible neural networks to be successfully applied in the field of wind turbine blade structural model updating. The study was based on an example of blade cross sections.

The paper performs feature selection using a sensitivity analysis that yields a sensitivity matrix. This analysis included material parameter (Young's modulus and density) and layup variation (Cross sectional layup division points). All parameters were varied within extended manufacturing tolerances. Based upon the sensitivity matrix and a given threshold value, the significant in and output features were identified. Furthermore, the general architecture and principals of a conditional invertible neural network (cINN) are explained. Subsequently the authors investigated the necessary cINN structure considering the trade-off between computational time and prediction accuracy. A cINN with shallow feedforward subnetworks was selected that took approx. 140 mins for sample generation and training on a computing cluster equipped with 40 CPUs and an NVIDIA Tesla P100 GPU.

10,000 samples based on randomly varied input feature sets were generated with MoCA and BECAS for testing. The cINN inferred strikingly accurate input feature values from the given test samples, except for two ambiguous glass fibre reinforced plastic density values. The results revealed that the ambiguity rests upon the two densities counteracting each other. However, a recalculation of output parameters from the inferred values affirmed again correct predictions. The only inaccuracies of the densities cancelled each other out. Another advantage over classical model updating techniques is that cINNs generate posterior distribution and not single values. This gives the user an instrument to evaluate the model's confidence on the predicted value and to reveal unrecoverable parameters. These findings were further confirmed by investigating cross sections at different radial positions, showing similar accurate results. The study found the posterior for the density of balsa was very wide for the other radii, which is due to a significant reduced balsa application in these cross sections and thus lower contribution to the mass related output features, and therefore a source of uncertainty correctly captured by the cINN. Additionally this paper studied a reduced output feature set, training the model only with stiffness and mass matrix as conditional feature. This scenario becomes relevant whenever a finite beam model updating can predict these values and a further inference to the material and layup level is desired. Here, the standard deviation of the posterior

distribution increases slightly, i.e. the confidence of the prediction diminishes. However, the results are still satisfactory.

Concluding, this study was able to show that cINNs present a serious alternative to classical model updating. The example of wind turbine blade cross sectional model updating proved outstanding performance for cINNs in this research field, which the authors will extend to more complex applications such as blade finite element beam models in their current and future studies.

Acknowledgments

This work was supported by the compute cluster, which is funded by the Leibniz University Hannover, the Lower Saxony Ministry of Science and Culture (MWK) and the German Research Association (DFG).

Funding

The authors acknowledge the financial support by the Federal Ministry for Economic Affairs and Energy of Germany in the projects SmartBlades2 (grant number 0324032C) and ReliaBlade (grant number 0324335B).

Abbreviations

The following abbreviations are used in this manuscript:

ANN	Artificial neural network
BECAS	Beam Cross section Analysis Software
CC	Conditional coupling block
cINN	Conditional invertible neural network
CS	Cross section
EC	Elastic centre
FrEIA	Framework for Easily Invertible Architectures
CoG	Centre of gravity
INN	Invertible neural network
K	Stiffness-matrix
LE	Leading edge
MoCA	Model Creation and Analysis Tool for Wind Turbine Rotor Blades
M	Mass-matrix
NLL	Negative log likelihood
PS	Pressure side
SC	Shear centre
SS	Suction side
TE	Trailing edge
UD	Uni-directional

References

[1] ARDIZZONE, Lynton ; LÜTH, Carsten ; KRUSE, Jakob ; ROTHER, Carsten ; KÖTHE, Ullrich: Guided Image Generation with Conditional Invertible Neural Networks. (2019). http://arxiv.org/pdf/1907.02392v3

[2] ARDIZZONE, Lynton ; MACKOWIAK, Radek ; ROTHER, Carsten ; KÖTHE, Ullrich: Training Normalizing Flows with the Information Bottleneck for Competitive Generative Classification. In: *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)* Bd. 33, 2018, S. 7828–7840

[3] BANGALORE, Pramod ; TJERNBERG, Lina B.: An approach for self evolving neural network based algorithm for fault prognosis in wind turbine. In: *2013 IEEE Grenoble Conference*, IEEE, 2013. – ISBN 978–1–4673–5669–5, S. 1–6

[4] BELLAGENTE, Marco ; BUTTER, Anja ; KASIECZKA, Gregor ; PLEHN, Tilman ; ROUSSELOT, Armand ; WINTERHALDER, Ramon ; ARDIZZONE, Lynton ; KÖTHE, Ullrich: Invertible networks or partons to detector and back again. In: *SciPost Physics* 9 (2020), Nr. 5. http://dx.doi.org/10.21468/SciPostPhys.9.5.074. – DOI 10.21468/SciPostPhys.9.5.074

[5] BLASQUES, José P. ; STOLPE, Mathias: Multi-material topology optimization of laminated composite beam cross sections. In: *Composite Structures* 94 (2012), Nr. 11, S. 3278–3289. http://dx.doi.org/10.1016/j.compstruct.2012.05.002. – DOI 10.1016/j.compstruct.2012.05.002. – ISSN 02638223

[6] BRUNS, Marlene ; HOFMEISTER, Benedikt ; PACHE, Dorian ; ROLFES, Raimund: Finite Element Model Updating of a Wind Turbine Blade—A Comparative Study. Version: 2019. http://dx.doi.org/10.1007/978-3-319-97773-7. In: *EngOpt 2018 Proceedings of the 6th International Conference on Engineering Optimization*. Cham : Springer International Publishing, 2019. – DOI 10.1007/978–3–319–97773–7, S. 569–580

[7] CHEN, Zhihao ; MENZIES, Tim ; PORT, Dan ; BOEHM, Barry: Feature subset selection can improve software cost estimation accuracy. In: *ACM SIGSOFT Software Engineering Notes* 30 (2005), Nr. 4, S. 1–6. http://dx.doi.org/10.1145/1082983.1083171. – DOI 10.1145/1082983.1083171. – ISSN 0163–5948

[8] CHOLLET, François: *Deep learning with Python*. Shelter Island, NY : Manning, 2018 (Safari Tech Books Online). http://proquest.safaribooksonline.com/9781617294433. – ISBN 9781617294433

[9] DINH, Laurent ; KRUEGER, David ; BENGIO, Yoshua: NICE: Non-linear Independent Components Estimation. (2014). http://arxiv.org/pdf/1410.8516v6

[10] DINH, Laurent ; SOHL-DICKSTEIN, Jascha ; BENGIO, Samy: Density estimation using Real NVP. (2016). http://arxiv.org/pdf/1605.08803v3

[11] DUCHI, John ; HAZAN, Elad ; SINGER, Yoram: Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. In: *Journal of Machine Learning Research* (2011), Nr. 12, S. 2121–2159

[12] FUCHS, Caro ; SPOLAOR, Simone ; NOBILE, Marco S. ; KAYMAK, Uzay: A Swarm Intelligence Approach to Avoid Local Optima in Fuzzy C-Means Clustering. In: *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, 2019. – ISBN 978–1–5386–1728–1, S. 1–6

[13] GAMBIER, Adrian ; BEHERA, Anshu: Modelling the aerodynamic coefficients of wind turbines by using neural networks for control design purposes. In: *Journal of Physics: Conference Series*

1037 (2018), S. 032032. http://dx.doi.org/10.1088/1742-6596/1037/3/032032. – DOI 10.1088/1742–6596/1037/3/032032. – ISSN 1742–6588

[14] GOLLER, B. ; BROGGI, M. ; CALVI, A. ; SCHUËLLER, G. I.: A stochastic model updating technique for complex aerospace structures. In: *Finite Elements in Analysis and Design* 47 (2011), Nr. 7, S. 739–752. http://dx.doi.org/10.1016/j.finel.2011.02.005. – DOI 10.1016/j.finel.2011.02.005. – ISSN 0168874X

[15] GRÖHL, Janek ; SCHELLENBERG, Melanie ; DREHER, Kris ; MAIER-HEIN, Lena: Deep learning for biomedical photoacoustic imaging: A review. In: *Photoacoustics* 22 (2021), S. 100241. http://dx.doi.org/10.1016/j.pacs.2021.100241. – DOI 10.1016/j.pacs.2021.100241. – ISSN 2213–5979

[16] GUNDLACH, J. ; GOVERS, Y.: Experimental modal analysis of aeroelastic tailored rotor blades in different boundary conditions. In: *Journal of Physics: Conference Series* 1356 (2019), S. 012023. http://dx.doi.org/10.1088/1742-6596/1356/1/012023. – DOI 10.1088/1742–6596/1356/1/012023. – ISSN 1742–6588

[17] HAN, Bing ; KONG, Xiaofang ; ZHANG, Zhiwen ; ZHOU, Lawu: Neural network model predictive control optimisation for large wind turbines. In: *IET Generation, Transmission & Distribution* 11 (2017), Nr. 14, S. 3491–3498. http://dx.doi.org/10.1049/iet-gtd.2016.1989. – DOI 10.1049/iet–gtd.2016.1989. – ISSN 1751–8695

[18] HOFMEISTER, Benedikt ; BRUNS, Marlene ; ROLFES, Raimund: Finite element model updating using deterministic optimisation: A global pattern search approach. In: *Engineering Structures* 195 (2019), S. 373–381. http://dx.doi.org/10.1016/j.engstruct.2019.05.047. – DOI 10.1016/j.engstruct.2019.05.047. – ISSN 01410296

[19] HUANG, Jianglin ; LI, Yan-Fu ; XIE, Min: An empirical analysis of data preprocessing for machine learning-based software cost estimation. In: *Information and Software Technology* 67 (2015), S. 108–127. http://dx.doi.org/10.1016/j.infsof.2015.07.004. – DOI 10.1016/j.infsof.2015.07.004. – ISSN 09505849

[20] JACOBSEN, Jörn-Henrik ; SMEULDERS, Arnold ; OYALLON, Edouard: i-RevNet: Deep Invertible Networks. In: *ICLR 2018 - International Conference on Learning Representations* (2018). http://arxiv.org/pdf/1802.07088v1

[21] JAFARNEJADSANI, Hamidreza ; PIEPER, Jeff ; EHLERS, Julian: Adaptive Control of a Variable-Speed Variable-Pitch Wind Turbine Using Radial-Basis Function Neural Network. In: *IEEE Transactions on Control Systems Technology* 21 (2013), Nr. 6, S. 2264–2272. http://dx.doi.org/10.1109/TCST.2012.2237518. – DOI 10.1109/TCST.2012.2237518. – ISSN 1063–6536

[22] KINGMA, Durk P. ; DHARIWAL, Prafulla: Glow: Generative Flow with Invertible 1x1 Convolutions. In: CURRAN ASSOCIATES, INC. (Hrsg.): *Advances in Neural Information Processing Systems*. 2018

[23] KNEBUSCH, Johannes ; GUNDLACH, Janto ; GOVERS, Yves: A systematic investigation of common gradient based model updating approaches applied to high-fidelity test-data of a wind turbine rotor blade. In: *Proceedings of the XI International Conference on Structural Dynamics*, EASDAthens, 2020, S. 2159–2174

[24] KSOLL, Victor F. ; ARDIZZONE, Lynton ; KLESSEN, Ralf ; KOETHE, Ullrich ; SABBI, Elena ; ROBBERTO, Massimo ; GOULIERMIS, Dimitrios ; ROTHER, Carsten ; ZEIDLER, Peter ; GENNARO, Mario: Stellar parameter determination from photometry using invertible neural networks. In: *Monthly Notices of the*

Royal Astronomical Society 499 (2020), Nr. 4, S. 5447–5485. http://dx.doi.org/10.1093/mnras/staa2931. – DOI 10.1093/mnras/staa2931. – ISSN 0035–8711

[25] LU, Yang ; SUN, Liping ; ZHANG, Xinyue ; FENG, Feng ; KANG, Jichuan ; FU, Guoqiang: Condition based maintenance optimization for offshore wind turbine considering opportunities based on neural network approach. In: *Applied Ocean Research* 74 (2018), S. 69–79. http://dx.doi.org/10.1016/j.apor.2018.02.016. – DOI 10.1016/j.apor.2018.02.016. – ISSN 01411187

[26] LU, Yong ; TU, Zhenguo: A two-level neural network approach for dynamic FE model updating including damping. In: *Journal of Sound and Vibration* 275 (2004), Nr. 3-5, S. 931–952. http://dx.doi.org/10.1016/S0022-460X(03)00796-X. – DOI 10.1016/S0022–460X(03)00796–X. – ISSN 0022460X

[27] LUCZAK, Marcin ; MANZATO, Simone ; PEETERS, Bart ; BRANNER, Kim ; BERRING, Peter ; KAHSIN, Maciej: Updating Finite Element Model of a Wind Turbine Blade Section Using Experimental Modal Analysis Results. In: *Shock and Vibration* 2014 (2014), S. 1–12. http://dx.doi.org/10.1155/2014/684786. – DOI 10.1155/2014/684786. – ISSN 1070–9622

[28] MAGERRAMOVA, Liubov ; VASILYEV, Boris ; KINZBURSKIY, Vladimir: Novel Designs of Turbine Blades for Additive Manufacturing. In: *Proceedings of ASME Turbo Expo 2016: Turbomachinery Technical Conference and Exposition*, 2016

[29] MALIK, Hasmat ; MISHRA, Sukumar: Application of Probabilistic Neural Network in Fault Diagnosis of Wind Turbine Using FAST, TurbSim and Simulink. In: *Procedia Computer Science* 58 (2015), S. 186–193. http://dx.doi.org/10.1016/j.procs.2015.08.052. – DOI 10.1016/j.procs.2015.08.052. – ISSN 18770509

[30] MALIK, Hasmat ; MISHRA, Sukumar: Artificial neural network and empirical mode decomposition based imbalance fault diagnosis of wind turbine using TurbSim, FAST and Simulink. In: *IET Renewable Power Generation* 11 (2017), Nr. 6, S. 889–902. http://dx.doi.org/10.1049/iet-rpg.2015.0382. – DOI 10.1049/iet–rpg.2015.0382. – ISSN 1752–1416

[31] MARWALA, Tshilidzi: *Finite-element-model updating using computional intelligence techniques: Applications to structural dynamics*. London : Springer, 2010. – ISBN 9781849963220

[32] MOJTAHEDI, A. ; LOTFOLLAHI YAGHIN, M. A. ; HASSANZADEH, Y. ; ETTEFAGH, M. M. ; AMINFAR, M. H. ; AGHDAM, A. B.: Developing a robust SHM method for offshore jacket platform using model updating and fuzzy logic system. In: *Applied Ocean Research* 33 (2011), Nr. 4, S. 398–411. http://dx.doi.org/10.1016/j.apor.2011.05.001. – DOI 10.1016/j.apor.2011.05.001. – ISSN 01411187

[33] MOTTERSHEAD, J. E. ; FRISWELL, M. I.: Model Updating In Structural Dynamics: A Survey. In: *Journal of Sound and Vibration* 167 (1993), Nr. 2, S. 347–375. http://dx.doi.org/10.1006/jsvi.1993.1340. – DOI 10.1006/jsvi.1993.1340. – ISSN 0022460X

[34] PASZKE, Adam ; GROSS, Sam ; MASSA, Francisco ; LERER, Adam ; BRADBURY, James ; CHANAN, Gregory ; KILLEEN, Trevor ; LIN, Zeming ; GIMELSHEIN, Natalia ; ANTIGA, Luca ; DESMAISON, Alban ; KÖPF, Andreas ; YANG, Edward ; DEVITO, Zach ; RAISON, Martin ; TEJANI, Alykhan ; CHILAMKURTHY, Sasank ; STEINER, Benoit ; FANG, Lu ; BAI, Junjie ; CHINTALA, Soumith: PyTorch: An Imperative Style, High-Performance Deep Learning Library. (2019). http://arxiv.org/pdf/1912.01703v1

[35] PATTERSON, Josh ; GIBSON, Adam: *Deep learning: A practitioner's approach*. First edition. Beijing and Boston and Farnham and Sebastopol and Tokyo : O'Reilly, 2017. – ISBN 978–1491914250

[36] QIU, Binbin ; LU, Yang ; SUN, Liping ; QU, Xianqiang ; XUE, Yanzhuo ; TONG, Fushan: Research on the damage prediction method of offshore wind turbine tower structure based on improved neural network. In: *Measurement* 151 (2020), S. 107141. http://dx.doi.org/10.1016/j.measurement.2019.107141. – DOI 10.1016/j.measurement.2019.107141. – ISSN 02632241

[37] RADEV, Stefan T. ; GRAW, Frederik ; CHEN, Simiao ; MUTTERS, Nico T. ; EICHEL, Vanessa M. ; BÄRNIGHAUSEN, Till ; KÖTHE, Ullrich: Model-based Bayesian inference of disease outbreak dynamics with invertible neural networks. (2020). http://arxiv.org/pdf/2010.00300v3

[38] RADEV, Stefan T. ; MERTENS, Ulf K. ; VOSS, Andreas ; ARDIZZONE, Lynton ; KOTHE, Ullrich: BayesFlow: Learning Complex Stochastic Models With Invertible Neural Networks. In: *IEEE transactions on neural networks and learning systems* PP (2020). http://dx.doi.org/10.1109/TNNLS.2020.3042395. – DOI 10.1109/TNNLS.2020.3042395

[39] RASCHKA, Sebastian ; MIRJALILI, Vahid: *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Third edition. Birmingham : Packt, 2019. – ISBN 1789955750

[40] RAVICHANDIRAN, Sudharsan: *Hands-on deep learning algorithms with python: Master deep learning algorithms with extensive math by implementing them using tensorflow*. Birmingham and Mumbai : Packt Publishing, 2019. – ISBN 9781789344158

[41] REZENDE, Danilo ; MOHAMED, Shakir: Variational Inference with Normalizing Flows. In: *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France, 2015

[42] SAENZ-AGUIRRE, Aitor ; ZULUETA, Ekaitz ; FERNANDEZ-GAMIZ, Unai ; LOZANO, Javier ; LOPEZ-GUEDE, Jose: Artificial Neural Network Based Reinforcement Learning for Wind Turbine Yaw Control. In: *Energies* 12 (2019), Nr. 3, S. 436. http://dx.doi.org/10.3390/en12030436. – DOI 10.3390/en12030436

[43] SAYER, F. ; ANTONIOU, A. ; GOUTIANOS, S. ; GEBAUER, I. ; BRANNER, K. ; BALZANI, C.: ReliaBlade Project: A Material's Perspective towards the Digitalization of Wind Turbine Rotor Blades. In: *IOP Conference Series: Materials Science and Engineering* 942 (2020), S. 012006. http://dx.doi.org/10.1088/1757-899X/942/1/012006. – DOI 10.1088/1757–899X/942/1/012006

[44] SCHRÖDER, Karsten ; GROVE, Saskia ; TSIAPOKI, Stavroula ; GEBHARDT, Cristian G. ; ROLFES, Raimund: Structural Change Identification at a Wind Turbine Blade using Model Updating. In: *Journal of Physics: Conference Series* 1104 (2018), S. 012030. http://dx.doi.org/10.1088/1742-6596/1104/1/012030. – DOI 10.1088/1742–6596/1104/1/012030. – ISSN 1742–6588

[45] SERNA, Alberto ; BUCHER, Christian: Advanced Surrogate Models for Multidisciplinary Design Optimization. In: *6th Weimar Optimization and Stochastic Days 2009*, 2009

[46] SUN, Hao ; BÜYÜKÖZTÜRK, Oral: Bayesian model updating using incomplete modal data without mode matching. In: *Health Monitoring of Structural and Biological Systems 2016*, SPIE, 2016 (SPIE Proceedings), S. 98050D

[47] SUNG, Heejun ; CHANG, Seongmin ; CHO, Maenghyo: Reduction method based structural model updating method via neural networks. In: *AIAA Scitech 2020 Forum*. Reston, Virginia : American Institute of Aeronautics and Astronautics, 2020. – ISBN 978–1–62410–595–1

[48] SUNG, Heejun ; CHANG, Seongmin ; CHO, Maenghyo: Efficient Model Updating Method for System Identification Using a Convolutional Neural Network. In: *AIAA Journal* (2021), S. 1–10. http://dx.doi.org/10.2514/1.J059964. – DOI 10.2514/1.J059964. – ISSN 0001–1452

[49] SUNNÅKER, Mikael ; BUSETTO, Alberto G. ; NUMMINEN, Elina ; CORANDER, Jukka ; FOLL, Matthieu ; DESSIMOZ, Christophe: Approximate bayesian computation. In: *PLoS Comput Biol* 9 (2013), Nr. 1, S. e1002803

[50] TREHAN, Sumeet ; CARLBERG, Kevin T. ; DURLOFSKY, Louis J.: Error modeling for surrogates of dynamical systems using machine learning. In: *International Journal for Numerical Methods in Engineering* 112 (2017), Nr. 12, S. 1801–1827. http://dx.doi.org/10.1002/nme.5583. – DOI 10.1002/nme.5583. – ISSN 00295981

[51] VAN DAMME, Christopher I. ; ALLEN, Matthew S. ; HOLLKAMP, Joseph J.: Updating Geometrically Nonlinear Reduced-Order Models Using Nonlinear Modes and Harmonic Balance. In: *AIAA Journal* 58 (2020), Nr. 8, S. 3553–3568. http://dx.doi.org/10.2514/1.J058698. – DOI 10.2514/1.J058698. – ISSN 0001–1452

[52] VISUAL LEARNING LAB HEIDELBERG ; GITHUB (Hrsg.): *FrEIA - Framework for Easily Invertible Architectures*. https://github.com/VLL-HD/FrEIA. Version: 2021

[53] YANG, Y. B. ; CHEN, Y. J.: A new direct method for updating structural models based on measured modal data. In: *Engineering Structures* 31 (2009), Nr. 1, S. 32–42. http://dx.doi.org/10.1016/j.engstruct.2008.07.011. – DOI 10.1016/j.engstruct.2008.07.011. – ISSN 01410296

[54] YIN, Tao ; ZHU, Hong-Ping: An efficient algorithm for architecture design of Bayesian neural network in structural model updating. In: *Computer-Aided Civil and Infrastructure Engineering* 35 (2020), Nr. 4, S. 354–372. http://dx.doi.org/10.1111/mice.12492. – DOI 10.1111/mice.12492. – ISSN 1093–9687

A Sensitivity Matrix

The full sensitivity matrix contains the impact of each input feature on the output feature for a cross section at R = 6 m. The values represent the standard deviation as described in section 2.1. The greyed values are below the threshold $\lambda = 0.25$. All Gray highlighted rows and columns are rejected by the feature selection.

Table A.1: Layup of each subcomponent of the DemoBlade cross section at R = 6 m.

Shell		
Laminate	No. of plies	nom. Thickness
Triax	1	0.9 mm
Biax 0° / 90°	1	0.65 mm
Foam / Balsa	1	20 mm
Biax 0° / 90°	1	0.65 mm
Triax	1	0.9 mm

Prefabricated Spar Cap		
Laminate	No. of plies	nom. Thickness
Triax	1	0.9 mm
UD / Balsa	32	26.2 mm
Triax	1	0.9 mm

Web		
Laminate	No. of plies	nom. Thickness
Biax ±45°	2	1.3 mm
Foam	1	20 mm
Biax ±45°	2	1.3 mm

Table A.2: Full Sensitivity Matrix of Cross Section at R = 6 m - Part A: Cross sectional properties.

	SC_x	SC_y	EC_x	EC_y	$Mass_{total}$	CoG_x	CoG_y	I_{xx}	I_{yy}	I_{xy}	$Area_x$	$Area_y$	A_{xx}	A_{yy}	A_{xy}	$Area_{total}$	$\alpha_{PC,Ref}$	$\alpha_{PC,EC}$
$E_{11,UD}$	0.2	0.8	3.6	3.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.3	2.3
$E_{22,UD}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$E_{33,UD}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ρ_{UD}	0.0	0.0	0.0	0.0	2.7	3.4	2.1	0.0	3.9	1.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$E_{11,Biax90}$	0.2	0.3	1.5	1.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.9
$E_{22,Biax90}$	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1
$E_{33,Biax90}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ρ_{Biax90}	0.0	0.0	0.0	0.0	1.6	1.0	0.5	2.0	1.1	0.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$E_{11,Triax}$	0.1	0.4	2.1	1.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.3	1.4
$E_{22,Triax}$	0.1	0.1	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1
$E_{33,Triax}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ρ_{Triax}	0.0	0.0	0.0	0.0	2.6	1.1	0.5	2.9	1.9	1.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
E_{Balsa}	0.1	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1
ρ_{Balsa}	0.0	0.0	0.0	0.0	2.2	1.1	3.4	3.1	1.3	1.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
E_{Foam}	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ρ_{Foam}	0.0	0.0	0.0	0.0	1.1	0.8	2.0	0.4	0.7	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$P_{SS,TE,offset}$	0.4	0.0	0.1	0.0	0.1	0.1	0.3	0.3	0.0	0.1	0.8	2.2	2.6	1.1	0.9	2.3	0.0	0.0
$P_{SS,TE,core}$	2.8	3.3	1.6	1.8	0.0	2.4	0.9	0.0	0.1	2.7	0.9	1.5	1.5	0.8	2.8	0.4	2.6	2.8
$P_{SS,TE,spar\ cap}$	2.8	3.3	1.6	1.8	0.0	2.4	0.9	0.0	0.1	2.7	0.9	1.5	1.5	0.8	2.8	0.4	2.6	2.8
$P_{SS,Mid,spar\ cap}$	2.8	3.3	1.6	1.8	0.0	2.4	0.9	0.0	0.1	2.7	0.9	1.5	1.5	0.8	2.8	0.4	2.6	2.8
$P_{SS,LE,spar\ cap}$	2.8	3.3	1.6	1.8	0.0	2.4	0.9	0.0	0.1	2.7	0.9	1.5	1.5	0.8	2.8	0.4	2.6	2.8
$P_{SS,LE,core}$	2.8	3.3	1.6	1.8	0.0	2.4	0.9	0.0	0.1	2.7	0.9	1.5	1.5	0.8	2.8	0.4	2.6	2.8
$P_{SS,LE,offset}$	0.1	0.0	0.1	0.0	0.0	0.2	0.1	0.1	0.0	0.1	1.9	1.9	1.4	2.2	1.0	2.3	0.0	0.0
$P_{PS,LE,offset}$	0.0	0.1	0.1	0.0	0.0	0.4	0.1	0.0	0.0	0.1	3.9	1.9	1.4	3.7	1.6	2.4	0.0	0.0
$P_{PS,LE,core}$	3.8	3.3	0.4	1.8	0.1	0.9	1.0	0.0	0.0	3.0	1.2	1.6	1.5	1.2	3.2	0.2	2.8	2.5
$P_{PS,LE,spar\ cap}$	3.8	3.3	0.4	1.8	0.1	0.9	1.0	0.0	0.0	3.0	1.2	1.6	1.5	1.2	3.2	0.2	2.8	2.5
$P_{PS,Mid,spar\ cap}$	3.8	3.3	0.4	1.8	0.1	0.9	1.0	0.0	0.0	3.0	1.2	1.6	1.5	1.2	3.2	0.2	2.8	2.5
$P_{PS,TE,spar\ cap}$	3.8	3.3	0.4	1.8	0.1	0.9	1.0	0.0	0.0	3.0	1.2	1.6	1.5	1.2	3.2	0.2	2.8	2.5
$P_{PS,TE,core}$	3.8	3.3	0.4	1.8	0.1	0.9	1.0	0.0	0.0	3.0	1.2	1.6	1.5	1.2	3.2	0.2	2.8	2.5
$P_{PS,TE,offset}$	0.5	0.0	0.1	0.0	0.1	0.2	0.3	0.3	0.0	0.1	0.4	2.3	2.7	0.7	0.5	2.4	0.0	0.0

Table A.3: Full Sensitivity Matrix of Cross Section at R = 6 m - Part B: Stiffness Matrix.

	K_{11}	K_{12}	K_{22}	K_{13}	K_{23}	K_{33}	K_{14}	K_{24}	K_{34}	K_{44}	K_{15}	K_{25}	K_{35}	K_{45}	K_{55}	K_{16}	K_{26}	K_{36}	K_{46}	K_{56}	K_{66}
$E_{11,UD}$	2.0	0.2	2.1	0.0	0.0	3.8	0.0	0.0	0.1	0.1	0.0	0.0	4.0	1.5	4.4	0.7	0.2	0.0	0.0	0.0	1.2
$E_{22,UD}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$E_{33,UD}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ρ_{UD}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$E_{11,Biax90}$	0.7	0.1	0.8	0.0	0.0	1.4	0.0	0.0	2.2	2.5	0.0	0.0	1.2	0.5	0.8	0.3	0.1	0.0	0.0	0.0	0.5
$E_{22,Biax90}$	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.1	0.1	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
$E_{33,Biax90}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ρ_{Biax90}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$E_{11,Triax}$	1.0	0.1	1.4	0.0	0.0	2.5	0.0	0.0	3.5	4.0	0.0	0.0	1.6	0.9	1.5	0.4	0.3	0.0	0.0	0.0	0.7
$E_{22,Triax}$	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.1	0.1	0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0
$E_{33,Triax}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ρ_{Triax}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
E_{Balsa}	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0
ρ_{Balsa}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
E_{Foam}	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0
ρ_{Foam}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$P_{SS,TE,offset}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.4
$P_{SS,TE,core}$	3.6	3.3	3.6	0.0	0.0	0.0	0.0	0.0	1.6	0.1	0.0	0.0	1.5	3.0	0.1	3.4	4.0	0.0	0.0	0.0	3.0
$P_{SS,TE,spar\ cap}$	3.6	3.3	3.6	0.0	0.0	0.0	0.0	0.0	1.6	0.1	0.0	0.0	1.5	3.0	0.1	3.4	4.0	0.0	0.0	0.0	3.0
$P_{SS,Mid,spar\ cap}$	3.6	3.3	3.6	0.0	0.0	0.0	0.0	0.0	1.6	0.1	0.0	0.0	1.5	3.0	0.1	3.4	4.0	0.0	0.0	0.0	3.0
$P_{SS,LE,spar\ cap}$	3.6	3.3	3.6	0.0	0.0	0.0	0.0	0.0	1.6	0.1	0.0	0.0	1.5	3.0	0.1	3.4	4.0	0.0	0.0	0.0	3.0
$P_{SS,LE,core}$	3.6	3.3	3.6	0.0	0.0	0.0	0.0	0.0	1.6	0.1	0.0	0.0	1.5	3.0	0.1	3.4	4.0	0.0	0.0	0.0	3.0
$P_{SS,LE,offset}$	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.9
$P_{PS,LE,offset}$	0.1	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.1	0.1	0.0	0.0	0.0	1.0
$P_{PS,LE,core}$	2.0	3.3	1.4	0.0	0.0	0.0	0.0	0.0	1.6	0.1	0.0	0.0	0.4	3.2	0.0	3.2	2.2	0.0	0.0	0.0	3.0
$P_{PS,LE,spar\ cap}$	2.0	3.3	1.4	0.0	0.0	0.0	0.0	0.0	1.6	0.1	0.0	0.0	0.4	3.2	0.0	3.2	2.2	0.0	0.0	0.0	3.0
$P_{PS,Mid,spar\ cap}$	2.0	3.3	1.4	0.0	0.0	0.0	0.0	0.0	1.6	0.1	0.0	0.0	0.4	3.2	0.0	3.2	2.2	0.0	0.0	0.0	3.0
$P_{PS,TE,spar\ cap}$	2.0	3.3	1.4	0.0	0.0	0.0	0.0	0.0	1.6	0.1	0.0	0.0	0.4	3.2	0.0	3.2	2.2	0.0	0.0	0.0	3.0
$P_{PS,TE,core}$	2.0	3.3	1.4	0.0	0.0	0.0	0.0	0.0	1.6	0.1	0.0	0.0	0.4	3.2	0.0	3.2	2.2	0.0	0.0	0.0	3.0
$P_{PS,TE,offset}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.8	0.0	0.0	0.0	0.5

Table A.4: Full Sensitivity Matrix of Cross Section at R = 6 m - Part C: Mass Matrix.

	M_{11}	M_{12}	M_{22}	M_{13}	M_{23}	M_{33}	M_{14}	M_{24}	M_{34}	M_{44}	M_{15}	M_{25}	M_{35}	M_{45}	M_{55}	M_{16}	M_{26}	M_{36}	M_{46}	M_{56}	M_{66}
$E_{11,UD}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$E_{22,UD}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$E_{33,UD}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ρ_{UD}	2.7	0.0	2.7	0.0	0.0	2.7	0.0	0.0	0.0	0.0	0.0	0.0	2.8	1.4	3.9	0.0	2.8	0.0	0.0	0.0	0.7
$E_{11,Biax90}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$E_{22,Biax90}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$E_{33,Biax90}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ρ_{Biax90}	1.6	0.0	1.6	0.0	0.0	1.6	0.0	0.0	1.3	2.0	0.0	0.0	1.4	0.7	1.1	1.3	1.4	0.0	0.0	0.0	2.0
$E_{11,Triax}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$E_{22,Triax}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$E_{33,Triax}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ρ_{Triax}	2.6	0.0	2.6	0.0	0.0	2.6	0.0	0.0	1.9	2.9	0.0	0.0	1.7	1.2	1.9	1.9	1.7	0.0	0.0	0.0	3.0
E_{Balsa}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ρ_{Balsa}	2.2	0.0	2.2	0.0	0.0	2.2	0.0	0.0	3.9	3.1	0.0	0.0	1.5	1.6	1.3	3.9	1.5	0.0	0.0	0.0	3.0
E_{Foam}	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ρ_{Foam}	1.1	0.0	1.1	0.0	0.0	1.1	0.0	0.0	0.9	0.4	0.0	0.0	1.0	0.1	0.7	0.9	1.0	0.0	0.0	0.0	0.5
$P_{SS,TE,offset}$	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.3	0.3	0.0	0.0	0.1	0.1	0.0	0.3	0.1	0.0	0.0	0.0	0.3
$P_{SS,TE,core}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	0.0	0.0	2.4	2.7	0.1	0.7	2.4	0.0	0.0	0.0	0.0
$P_{SS,TE,spar\ cap}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	0.0	0.0	2.4	2.7	0.1	0.7	2.4	0.0	0.0	0.0	0.0
$P_{SS,Mid,spar\ cap}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	0.0	0.0	2.4	2.7	0.1	0.7	2.4	0.0	0.0	0.0	0.0
$P_{SS,LE,spar\ cap}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	0.0	0.0	2.4	2.7	0.1	0.7	2.4	0.0	0.0	0.0	0.0
$P_{SS,LE,core}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7	0.0	0.0	0.0	2.4	2.7	0.1	0.7	2.4	0.0	0.0	0.0	0.0
$P_{SS,LE,offset}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.0	0.2	0.1	0.0	0.1	0.2	0.0	0.0	0.0	0.0
$P_{PS,LE,offset}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.4	0.1	0.0	0.1	0.4	0.0	0.0	0.0	0.0
$P_{PS,LE,core}$	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.7	0.0	0.0	0.0	0.8	3.0	0.0	0.7	0.8	0.0	0.0	0.0	0.0
$P_{PS,LE,spar\ cap}$	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.7	0.0	0.0	0.0	0.8	3.0	0.0	0.7	0.8	0.0	0.0	0.0	0.0
$P_{PS,Mid,spar\ cap}$	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.7	0.0	0.0	0.0	0.8	3.0	0.0	0.7	0.8	0.0	0.0	0.0	0.0
$P_{PS,TE,spar\ cap}$	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.7	0.0	0.0	0.0	0.8	3.0	0.0	0.7	0.8	0.0	0.0	0.0	0.0
$P_{PS,TE,core}$	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.7	0.0	0.0	0.0	0.8	3.0	0.0	0.7	0.8	0.0	0.0	0.0	0.0
$P_{PS,TE,offset}$	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.0	0.3	0.3	0.0	0.0	0.2	0.1	0.0	0.3	0.2	0.0	0.0	0.0	0.3

B Evaluation of cINN

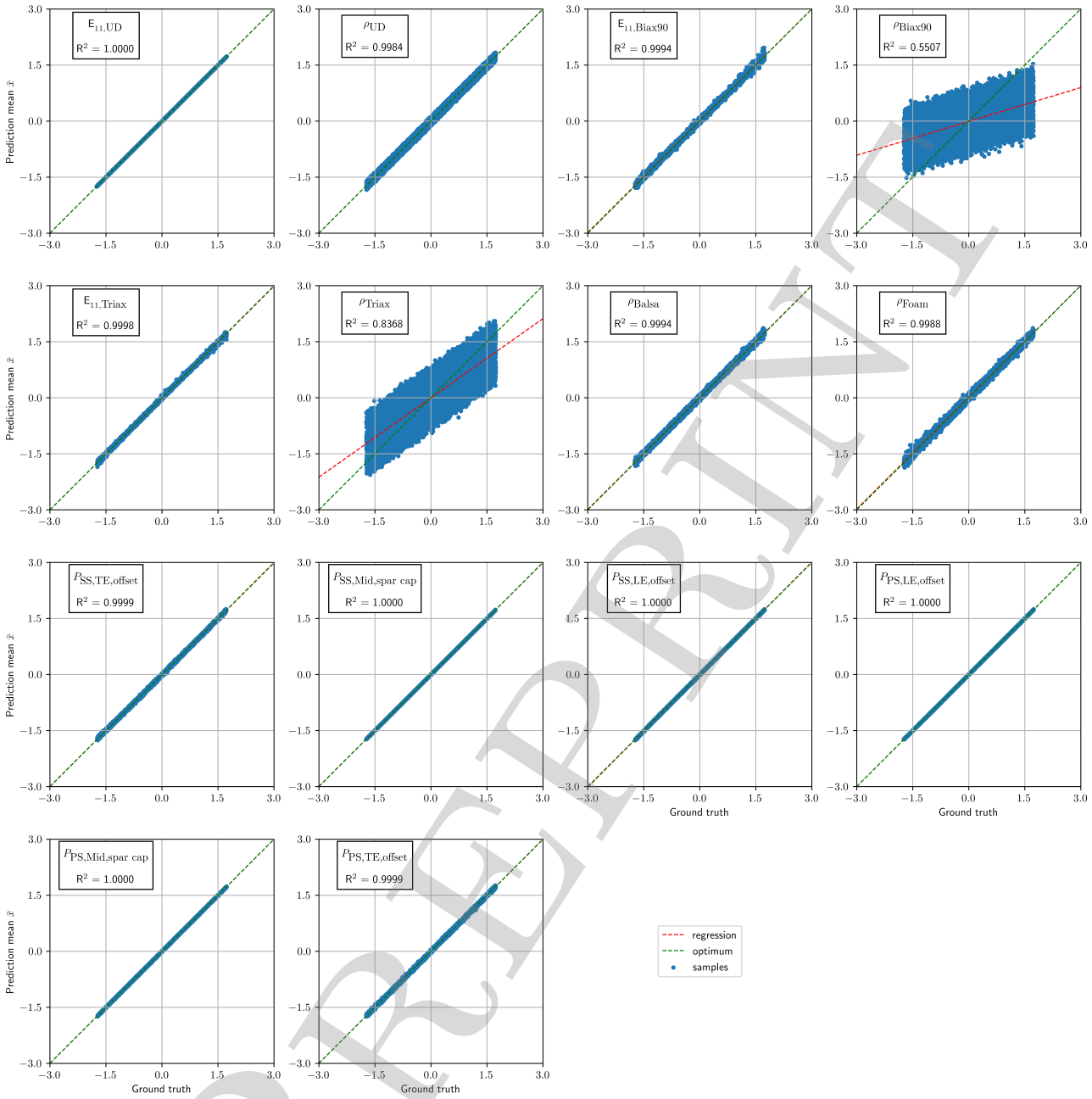

Figure B.1: Correlation between input feature prediction and ground truth measured with R^2. Optimum values are located on $f(x) = m \cdot x$.

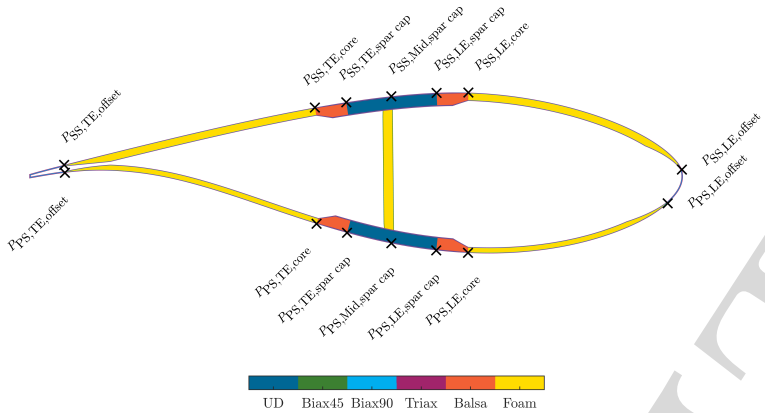

Figure B.2: Cross Section of the SmartBlades2 DemoBlade at a radial position of R = 12 m