ROSS-ML: A MACHINE LEARNING OPEN-SOURCE SOFTWARE FOR ROTORDYNAMICS

Luan da Silva Prado

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Mecânica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Mecânica.

Orientador: Thiago Gamboa Ritto

Rio de Janeiro
Julho de 2021

ROSS-ML: A MACHINE LEARNING OPEN-SOURCE SOFTWARE FOR ROTORDYNAMICS

Luan da Silva Prado

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA MECÂNICA.

Orientador: Thiago Gamboa Ritto

Aprovada por: Prof. Thiago Gamboa Ritto
Prof. Américo Barbosa da Cunha Junior
Prof. Daniel Alves Castello

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2021

Prado, Luan da Silva

ROSS-ML: A Machine Learning Open-source software for Rotordynamics/Luan da Silva Prado. – Rio de Janeiro: UFRJ/COPPE, 2021.

XII, 67 p.: il.; 29, 7cm.

Orientador: Thiago Gamboa Ritto

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Mecânica, 2021.

Referências Bibliográficas: p. 58 – 67.

1. Rotordynamics. 2. Neural Networks. 3. Machine Learning. I. Ritto, Thiago Gamboa. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Mecânica. III. Título.

A Ele, a Ela e a eles todos

Thanks

Agradeço aos meus pais pelas circunstâncias, ao meu orientador pela confiança e ao amigo Rodrigo por toda paciência e ajuda.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ROSS-ML: UM SOFTWARE ABERTO DE MACHINE LEARNING VOLTADO À ROTODINÂMICA

Luan da Silva Prado

Julho/2021

Orientador: Thiago Gamboa Ritto

Programa: Engenharia Mecânica

Neste trabalho, desenvolve-se um ambiente de aprendizado de máquina voltado à rotodinâmica baseado na biblioteca Keras. Este ambiente possui uma série de recursos de pré-processamento, tais como a redução de recursos por meio de árvores decisoras,diversos métodos de normalização dos dados e uma vasta quantidade de funções de otimização, permitindo um amplo repertório para a obtenção do melhor modelo possível. Além disso, o ambiente dispõe de um módulo de pós-processamento que permite avaliar os resultados sob diferentes perspectivas, o que facilita a tomada de decisão. Após apresentar todas as ferramentas disponíveis, treina-se uma rede neural com dados de um modelo de mancal hidrodinâmico e comparam-se os resultados entre a rede neural e o modelo físico.Esta comparação mostra que além de rápida a rede neural é um excelente modelo substituto.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ROSS-ML: A MACHINE LEARNING OPEN-SOURCE SOFTWARE FOR ROTORDYNAMICS

Luan da Silva Prado

July/2021

Advisor: Thiago Gamboa Ritto

Department: Mechanical Engineering

This work develops a Keras-based machine learning environment for rotor dynamics. This environment has many pre-processing resources, as Decision-Trees feature reduction, many scalers, and a vast number of optimizers, allowing a broad repertoire to obtain the best model possible. Besides that, the environment has a post-processing module that allows model evaluation from different perspectives. After presenting all the available tools, a neural network is trained with data from a hydrodynamic bearing model, and the aftermath is compared with the physical model. This comparison shows that besides computational performance, neural networks are excellent surrogate models as well.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Rotor dynamics is a multidisciplinary area that involves Tribology, Strength of Materials, Fluid, and Thermodynamics. Normally, the governing equations of the complete problem do not have an analytical solution, which justifies the use of techniques as Finite Differences and Finite Elements, often available in both commercial and open-source software.

In general, most software of dynamic analysis uses Finite Element Method (FEM) [1] and Computational Fluid Dynamics (CFD) [2] as its core. To run any analysis, programs as ROMAC, XLTRC2, PERMAS, NASTRAN, ANSYS, and Dyrobes need an underlying geometry, a structured mesh, boundary conditions, and the physical parameters to build a model.

Those programs, however, still do not allow the user to import data-driven models and use them combined with CFD or FEM analysis. Besides that, commercial programs assume that all the variables in place are deterministic, making users who want stochastic analysis need to implement their models.

In this work, a program will be developed to address those questions by using machine learning methods, which allows the user to generate models with either experimental data or computational simulations and use them at any other software in Python environment or even in commercial programs by generating a tabular output. Furthermore, this software can be combined with ROSS [3], an open-source code for rotor dynamics, and produce even better results.

Machine Learning methods encompass many techniques and algorithms, from a simple linear regression [4, 5] to intricate neural network structures [6, 7].In the

past few years, artificial neural networks (ANNs) showed versatility, being able to perform tasks as facial recognition[8] and autonomous vehicle control[9]. However, to perform such feats, the training set must be huge since the error surface in neural networks with large degrees of freedom tends to be highly non-convex and non-smooth, which compromises the search of a reliable minimum [10].

The main advantage of machine learning techniques is the creation of reusable and cheaper surrogate models with minimum codification, which allows the development of new tools with an outstanding speed; However, machine learning models rely heavily on data, fact that compromises the reliability of models in which noise is predominant, the information is doubtful or must be guessed. Moreover, depending on the model, the amount of data needed to obtain reliable results in high dimensional data is prohibitive, and, because of that, users must know the problem to choose only essential features to perform the training.

When this is not possible, the solution is data augmentation or feature reduction. The first case looks to the big picture to unravel intrinsic patterns to generate new points [11] while the second maps data into lower spaces trying to keep its structure unchanged [12], searches for relevant directions in data [13] or attributes to each label a score based on its performance in a simple model and choose the only highest ones [5].

1.2 Objectives

The objectives of this work are threefold. The first is to create a pipeline that systematically builds a neural network with design flexibility. The second is to develop a post-processing procedure to interpret the results and aid in the decision taking process and the last objective is to show the neural network in a simple example of uncertainty quantification.

1.3 Organization

This work starts initially discussing the main topics of Rotor dynamics, Neural Networks, and related work combining both in Chapter 2. After setting out the fundamentals, Chapter 3 describes the methodology used to construct the software, while Chapter 4 exemplifies the program usage in detail. Finally, Chapter 5 concludes the work and proposes further implementations.

Chapter 2

Bibliographic Review

2.1 Rotors

Rotating machinery is composed of shafts, disks, and bearings. To correctly operate, many metrological procedures must be carried out, from adjusting to alignment. If all practical considerations are in a model, the computations become cumbersome or even infeasible. This kind of issue is avoided by choosing only important parameters to compose the model.

The addition of an uncertainty layer to the mathematical model recovers the real-world problem; however, the solution obtained is now in terms of confidence intervals, whose size depends on the computational effort available to calculate it.

Rotor dynamics starts formally in 1869 [14] when it was discussed the relationship between centrifugal and restoring forces. In that time, it was concluded that exists a threshold of operation, which would be impossible to operate above. This conclusion was refuted in 1924 [15] with a flexible rotor that worked about seven times the critical speed; however the term "critical speed" is held until nowadays.

At the first glance, the researchers' focus was predicting the critical speeds. In 1894, it was derived an empirical formula to the lowest critical speed of a multirotor system and coined this term due to the abrupt change in the system behavior[16]. Then, in 1924, was created a diagram representing critical speed about the cross points of natural frequency curves and the straight lines proportional to the rotational speed, the so-called Campbell Diagram [17]. After the discoveries of 1894 [16], in 1921 was proposed a method to calculate natural frequencies and mode shapes in torsional vibrations [18].

After empirical and heuristic developments, a fundamental theory was established in 1919 [19] and considerably improved from 1924 to 1969 [15, 20–23]. Bearings have been studied in problems as internal friction of shaft materials causing self-excited vibration in 1924 [24], oil whip in 1925 [25], nonlinear resonances due to ball bearings in 1955 [26], journal bearings in 1966 [27]. Seals have been studied due to steam whirl, a self-excited vibration phenomenon, that has been already explained in turbines in 1958 [28] and compressors [29] in 1965.

In terms of modeling, Finite Element Method [1], [30] is used to describe shaft, disks, and standard roller bearings. When journal bearings and seals are present, proper CFD schemes are used to determine the pressure field and with it determine both stiffness and damping coefficients in order to assembly elements to obtain the equation

$$[M]\ddot{x}(t) + [C(\Omega)]\dot{x}(t) + [K(\Omega)]x(t) = f(t),$$

where $[M], [C], [K], x(t)$ and $f(t)$ are mass matrix, damping matrix, stiffness matrix, displacement and force vector, respectively.

Concerning notation, terms in brackets are matrices, terms in keys are vectors, and all the other terms can be either vectors or scalars, depending on the context they are used. In statistics, the notation is the same as used in [5, 31, 32]. In dynamics, vector fields are described in bold and scalars in usual font.

Besides classical modeling, uncertainty analysis is also carried out. Generally, Monte Carlo [31] techniques are used in conjunction with a probabilistic model to achieve this objective. However, this scheme imposes another loop in the computation, which increases greatly the computational effort required to obtain the results. So a surrogate model is needed to turn the analysis less prohibitive.

2.2 Fluid Bearings and Seals

In turbomachinery, oil-lubricated fluid film bearings are often used due to adequate load support, good damping characteristics, and absence of wear if correctly designed. Its purpose is to produce a low friction motion between two solid surfaces with relative motion. The lubricant or fluid between the surfaces can be a solid, liquid, or gas [33, 34].

When correctly designed and operated, they can support both static and dynamic loads and, thus, their effects on the performance are quite relevant. The bearings reviewed here are the ones with a full film separating the mechanical surfaces. The

term film refers to the fluid thickness, say gap or clearance, separating the surfaces is several orders of magnitude smaller than the width and length [33].

In terms of operational principles, Fluid film bearings can be hydrodynamic, hydrostatic, or hybrid. In Hydrodynamic fluid film bearings, also called self-acting bearings, there is relative motion between two mechanical surfaces with a wedge. The fluid is drawn into the film generating hydrodynamic pressures able to withstand an externally applied load [33, 35].

Hydrodynamic bearings have advantages as long life, high coefficients of damping and stiffness, the ability to withstand heavy loads, the absence of an external source of pressure, and fluid flow drag into the convergent gap in the direction of the surface where relative motion happens. On the other hand, disadvantages include thermal effects that can affect the performance when film thickness or available flow rate is insufficient, the requirement of a surface with a relative motion to generate load support, the potential to induce hydrodynamic instability, and the induction of large drag torque, which brings power losses, and potential surface damage at start-up.

In Hydrostatic fluid film bearings, also named externally-pressurized bearings, there is an external source of pressurized fluid that forces the lubricant or fluid between the surfaces, providing their separation and the ability to withstand a load without contact with the surface [33–35].

As advantages of this arrangement, one could cite the capacity to support massive loads, long life, very large coefficients, and the load independence from both film thickness and lubricant viscosity. However, due to the external source of pressure, this type of bearing needs auxiliary equipment, larger installation, a fluid filtration system, which impacts on costs, loss of performance with contamination, potential to induce hydrodynamic instability, and pneumatic hammer instability for highly compressible fluids, in other words, the loss of damping at low and high frequencies of operation [33, 35].

Besides fluid film bearings, complementary mechanisms are used to improve rotor performance, as Squeeze film dampers and radial seals. The first works effectively under compression and are used to reduce the vibration amplitudes and to isolate structural components, while the second separate regions of high and low pressure, minimizing the leakage and improving the machine efficiency in the process of extracting or delivering power to a fluid [33].

2.3 Neural Networks

To reduce the computational cost of cumbersome CFD calculations, one can use surrogate models . As examples, one can cite response surfaces, kriging, gradient-enhanced kriging (GEK) [36], radial basis function, support vector machines [37], space mapping [38] , artificial neural networks [6] and Bayesian networks [39].

To properly build a surrogate model, $\hat{f}$, of a black-box expensive model, f, a sampling plan $X = (x_1, x_2, ..., x_n)$ is required. The first step of the sampling plan is the correct choice of X, which depends on which features affects the output significantly when modified. With this information, the most relevant features are sampled to represent the design space accurately. After this, f is learned from data pairs $[(x_1, y_1), (x_2, y_2), (x_3, y_3), ..., (x_n, y_n)]$, reason why this process is called supervised learning. To sum up, this procedure is a search across the space of conceivable functions $\hat{f}$ that would replicate f at the specified design. The main caveat in obtaining data pairs is the computational cost involved to calculate the y_i in each sample. As a good practice, to avoid further efforts is recommended to scale the input into the unit cube $[0, 1]^k$.

An adequate function must fit the data parsimoniously since it cannot be too refined, capturing each detail of data, including noise, or too gross, being unable to represent correctly$f$ at all. When the first issue occurs, the function $\hat{f}$ is over-fitting the data, which completely captures each detail at the design space, but as a trade-off generalizes poorly. On the other hand, when $\hat{f}$ cannot even capture the details at the design space, it occurs under-fitting [5] (Fig 2.1 [40])

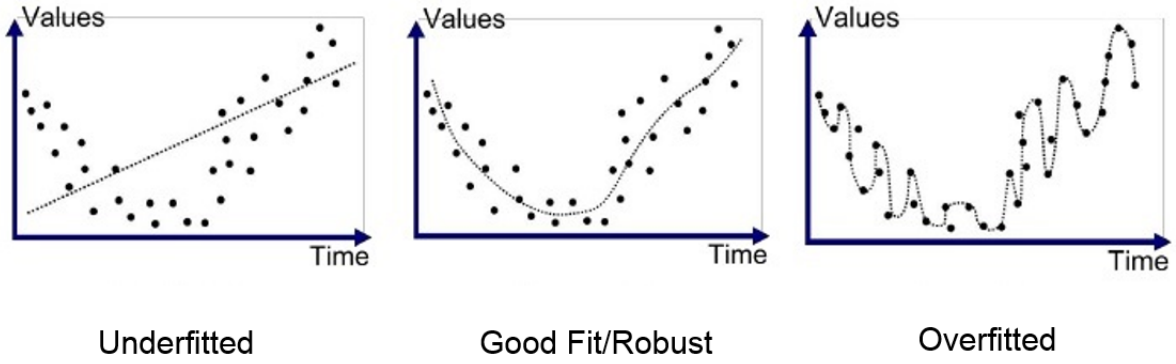


Figure 2.1: Under-fit (left figure), Proper fit (center figure) and Over-fit (right figure). Note that under-fitted model is unable to catch any characteristic of data, while the overfitted one captures excessive detail in data, which can be misleading due to the presence of noise.

By completing the first step, we obtained the data necessary to perform the learning process and a generic model $\hat{y}(x, w)$. The next step is to choose the set of parameters w which best fits the data. The most common techniques are Maximum Likelihood Estimation (MLE) [31] and Cross-Validation (CV) [37].

Given a set of parameters w and the model $\hat{y}(x, w)$, is possible to compute data set probability $\{(x_1, y_1 + \epsilon), (x_2, y_2 + \epsilon), (x_3, y_3 + \epsilon), ..., (x_n, y_n + \epsilon)\}$ that resulted from $\hat{y}$, with ϵ considered small and with a constant margin around each point. If the error ϵ is considered independent and randomly distributed following a normal distribution and the points are independent and identically distributed, the data set can be modeled as:

$$f(y|\hat{y}(x,w)) = \frac{1}{(2\pi\sigma^2)^{n/2}} \prod_{i=1}^{n} \left\{ exp\left[-\frac{1}{2} \left(\frac{y_i - \hat{y}(x_i, w)}{\sigma} \right)^2 \right] \right\}$$

In order to obtain the best w, one should maximize the above expression with respect of w, or similarly:

$$\min_w \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}(x_i, w)}{\sigma} \right)^2$$

.Being σ a constant, the expression reduces to:

$$\min_w \sum_{i=1}^{n} \left(y_i - \hat{y}(x_i, w) \right)^2,$$

which is the so-called ordinary least squares criterion. One of the weakness of this approach is the fact of considering the error independent, what does not hold when heteroscedasticity is present. This term refers to the phenomenon of variable standard deviation across the data, which results in a non-optimum w obtained with ordinary least squares.

To solve this question, methods as Weighted Least Squares and Generalized Least Squares arise. The first supposes that each error has a variance, and the last goes further and adds the hypothesis that the errors are correlated.To further details, check [31].

To avoid using the least-squares strategy in heteroscedastic data, one must perform a hypothesis test to verify it. As examples of heteroscedasticity tests, one can cite the White test [41] and Breusch-Pagan test [42] or even scatter plots or standardized error plot.

Another procedure used is cross-validation, method that split data into q equal or almost subsets, then perform the fit always removing one of the subsets. During the training process, the model is evaluated by a loss function that is used to evaluate the error between the values in data set and the predicted ones. In mathematical terms, we have the following:

$$\epsilon_{cv} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{f}^{-\zeta(i)}(x, w)),$$

where $\hat{f}^{-\zeta(i)}$ is the function calculated in every point except the ones in the subset $\zeta(i)$. By using the mean squared error, one obtains:

$$\epsilon_{cv} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{f}^{-\zeta(i)}(x, w))^2.$$

Note that this expression and its results don't depend on the hypothesis of homocedasticity; however, the right choice of the loss function is of main concern, being a good choice the one that takes into account the topology of the design space.

To validate the trained model, is it chosen a subset of say, 25% to 30% of the data set, to be compared to the trained model [5] by using an adequate model. The most common ones are mean squared error(MSE), mean absolute error (MAE), and correlation coefficient (R^2). Another alternative is to use graphic methods as q-q plots or conduct a hypothesis test, as described in chapter 3.

As stated before, neural networks are a type of surrogate model, so it inherits all the considerations described above to achieve proper results. Besides being a surrogate model, neural networks are a bio-inspired model that can acquire and retain knowledge by using a set of processing units, artificial neurons, connected by the means of several interconnections, implemented by vectors and matrices.

From a historical perspective, [43] composed a bio-inspired mathematical model that resulted in the first conception of an artificial neuron. [44] proposed a training procedure that was based on hypothesis and observations of neuro-physiologic nature. [45] proposed the first neurocomputer, the Mark I Perceptron, crafting the basic model of perceptron. [46] proposed the ADALINE (adaptative linear element). [47] published a book concerning the perceptron theory and [48], [49] and [50] proposed the use of reverse gradients, self-organizing maps, and adaptive resonance theory, respectively. As examples of modern application, we can cite facial recognition[8] and autonomous vehicle control[9].

To obtain reliable results, a large amount of data is necessary to perform the training, which is directly proportional to the task complexity. An example of this is the autonomous car from Tesla, which involved 48 neural networks and 70000 hours of GPU of training [51]. A very pertinent question arises in the neural network area: Is it possible to train neural networks when data is insufficient? The answer is: it depends. If the task has similarity with another, Transfer Learning techniques [52] can be performed, but if the task is unique and data is scarce, Manifold Learning [53] is recommended.

Before describing the methodology,it is worth to briefly discuss what is exactly Manifold Learning and which areas it encompasses. Manifold Learning is a multidisciplinary area that involves General Topology , Differential Geometry and Statistics. Good references in Topology include [54–61], Differential Geometry include[62–65] and Statistics include [66] and [32].

The main focus of Manifold Learning is the information extraction of Manifolds, which are a generalization of vector spaces in two, three, or higher dimensions. To properly develop the intuition behind the manifold, imagine an ant crawling on a guitar body. From the ant perspective, due to its tiny size, the guitar seems flat and featureless, although its shape is curved. A manifold is a topological space that locally looks flat and featureless and behaves like a Euclidean Space; however different from Euclidean Spaces, topological spaces do not have the concept of distance.

To clarify what is a locally Euclidean space, some definitions are necessary. A topological space X is said to be locally Euclidean if there exists an integer $d \geq 0$ such that around every point in X, there is a local neighborhood that is homeomorphic, that is, there is an invertible continuous map $g : X \to Y$, to an open subset in Euclidean space $\mathbb{R}^d$.[53]

Besides Euclidean space, another space in Manifold learning is the Hausdorff space. A topological space X is a Hausdorff space if every pair of distinct points has a corresponding pair of disjoint neighborhoods. Almost all spaces are Hausdorff, including the real line $\mathbb{R}$ with the standard metric topology. Also, subspaces and products of Hausdorff spaces are Hausdorff. X is second-countable if its topology has a countable basis of open sets[53].

In terms of techniques of Manifold Learning, one can cite Spectral Embedding Methods, like Isomap[67] and Local Linear Embedding, LLE,[68], Laplacian Eigenmaps[69], Diffusion Maps [12], Hessian Eigenmaps[70], Nonlinear PCA [71], Manifold Sampling[72] and Normal Bundle Bootstrap [73]. With the aforementioned methods, it was possible to use manifold learning from image recognition [74] to human motion[75].

2.4 Related work

The combination of neural networks with Rotor dynamics is a recent field in engineering with a crescent number of publications. To cite a few, [76] used Neural networks to diagnose rotating machines based on their orbit and [77] goes further and develops automated fault detection based on the transient analysis. Recently, [78] reviewed the machine learning techniques and showed how these tools use in the context of rotor analysis. Concerning bearings, [79] focused on fault diagnosis of ball bearings using artificial neural networks (ANN) and support vector machine (SVM) and [80] used a Convoluted Neural Network and Dynamical Time Warping to detect race faults in roller bearings.

The present work creates an environment that allows the creation of neural networks for regression and provides several tools to evaluate the results from different perspectives, allowing a robust and reliable analysis of the model's performance. Chapter 3 resumes the theory behind the software and highlights complementary literature for both theoretical and practical purposes.

Chapter 3

Program Methodology

This chapter discusses the software itself. Its core is the Keras library, which allows an all-purpose, fast and scalable solution. The software developed is focused on regression, but it can also be used in classification, with just a few modifications.

To correctly use this code, theoretical key-concepts of neural networks must be understood. To complete this task, books as [7],[6] and [81] should be used.[82] and [83] are good practical references.

The first step to build an acceptable solution is the proper understanding of inputs and outputs. To obtain acceptable results, one must avoid the aphorism: "garbage in, garbage out". What does exactly this phrase mean in the context of neural networks? In terms of output, "garbage" is an overfitted or an under fitted result, and concerning inputs, "garbage" is a data set with high dimension(many columns) and few samples (few rows) that produces poor results, the so-called curse of dimensionality, which can be troublesome to the optimization algorithm to achieve a proper result.

To avoid it, two options are commonly used: obtaining more data or feature reduction. The first option is not always feasible, since data can be expensive to get or simply unavailable due to limited budget, and the second is to perform feature reduction, by using techniques as Decision Trees [84][5], Adjusted Mutual Info-Score [85] or F-test [86].

To build a proper solution, the instructions needed to solve this kind of problem are structured in the form of a pipeline, which is shown in figure 3.1.

The first two steps, *Set Labels* and *Set Features* in Fig. 3.1, are straightforward and encompasses the data set input and its divisions in labels and features. Feature

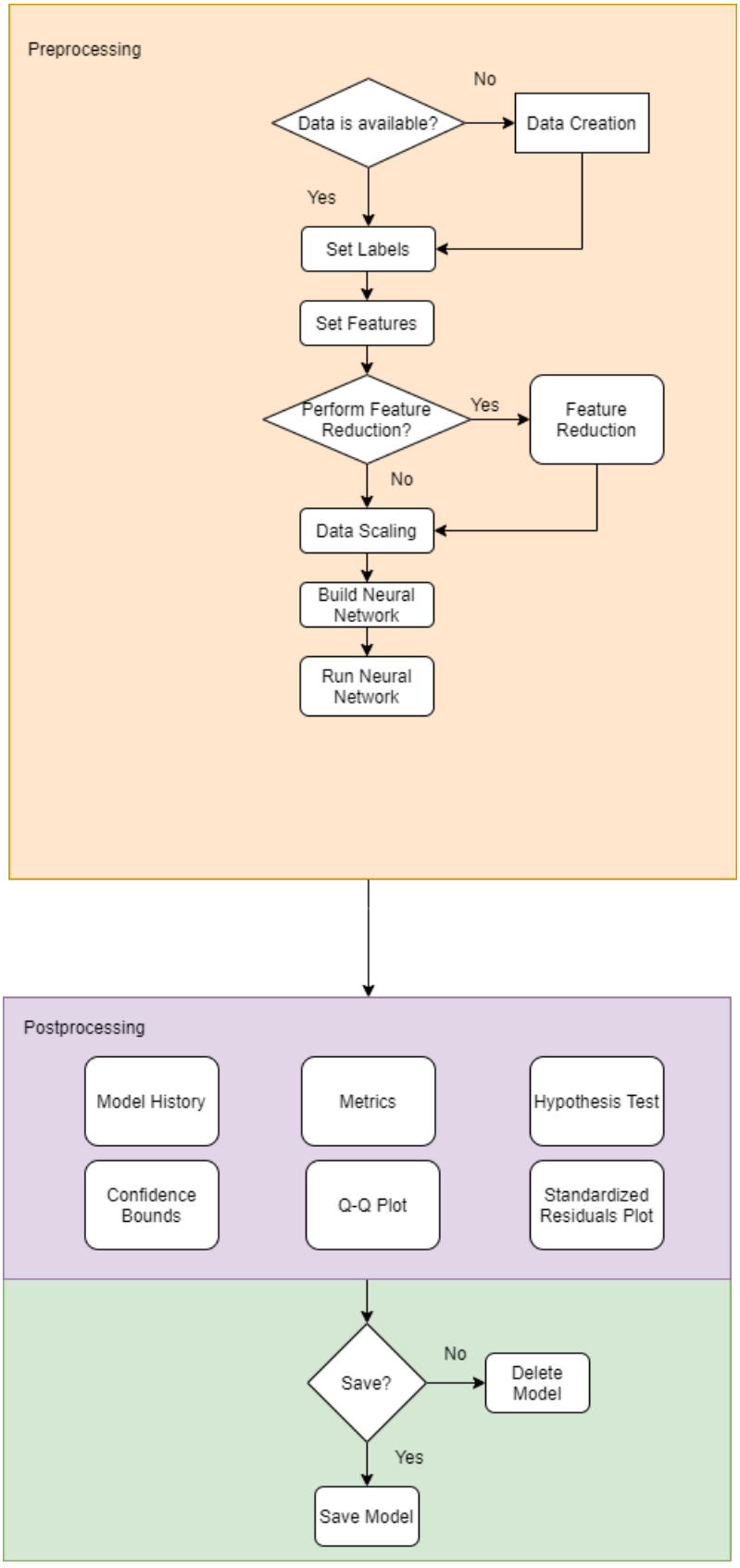


Figure 3.1: A Pipeline to build a Neural Network based surrogate model

reduction (lit. in Fig. 3.1) is an optional step recommended only when the number of features is high and the number of samples is insufficient, which is discovered by running a model with all the available features and checking the loss function at training. If the loss function is too high and achieves saturation too fast, this means that the samples given are unable to describe precisely the underlying topology and, because of that, the optimizer stays stuck on a local minimum instead of continuing the search of a global minimum.

The scaling process,referred as *Data Scaling* in Fig.3.1, is the most important before the model construction since it is possible to encapsulate the weights on the same scale, turning the algorithm more stable and increasing its performance. There are several ways to scale data, being the most popular one the normalization, which basically shifts the data by its mean and scaling it by its standard deviation.Mathematically speaking, classical normalization performs the following:

$$Z = \frac{X - \mu}{\sigma},$$

where μ is the mean and σ is the standard deviation. This process keeps the transformed data with the same distribution as the brute data; however, this kind of transformation is insensitive to outliers, which could compromise the optimization process. Another similar process of scaling is using the minimum and maximum to keep the data set in interval [0,1], that is:

$$Z = \frac{X - X_{min}}{X_{max} - X_{min}}.$$

Note that in this case, the scaling is also insensitive to outliers. To solve this issue is possible to scale the data by shifting it with the median and dividing it by the IQR range, the difference between the third and the first quartiles, that is

$$Z = \frac{X - Q_{50}}{Q_{75} - Q_{25}},$$

being Q_{50} the median,Q_{75} and $Q_{25}$the third and first quartiles, respectively. The quartiles are robust to outliers since the calculations involve the value position, not the values themselves. As an example, it suffices to note that both $(1, 10, 500)$ and $(-100000, 10, 100000)$ has the same median but different means. Note that this process of scaling only works well in normal distributions or in symmetric data.

Another form of scaling is to perform non-linear transformations, as Box-Cox or Yeo-Johnson transformations that aims to stabilize the variance and to transform the feature distribution into a normal-like distribution. With a normal-like distribution

in hands, is possible to perform all the above techniques to convert the interval to a more appropriate one.

Besides all the techniques aforementioned, is also possible to perform quantile transformation, which converts each feature in the data set on a Uniform distribution at the [0,1] interval by using the inverse of the Empirical Cumulative Density Function (ECDF) to map the data distribution into a standard uniform with support [0,1]. To be properly converted, a considerable amount of data needs to be provided, in order to guarantee the transformation accuracy. With quantile transformation is also possible to map the converted uniform in any other distribution by knowing its CDF.

After scaling the data, the next step is to build an adequate Neural Network (*Build Neural Network in Fig. 3.1*), which main ingredients are namely the activation functions, the number of neurons per layer, and the number of layers. In regression problems, Rectified Linear Unit, ReLU in short, is the most used activation function, due to its non-saturating gradient, which greatly increases the convergence of stochastic gradient descent [87] compared to the sigmoid and hyperbolic tangent functions [88]. As a reminder, the ReLU function is given by:

$$ReLU(x) = max(0, x).$$

Note that the ReLU function is easier to implement in comparison with other activation functions, which turns the computational cost cheaper. A comparison between activation function and architectures can be found in [89].

The number of neurons can be determined in a straightforward way for both first and the last layer. These layers must have the same number of neurons as features and labels respectively. The hidden layers can be both single layers or multiple layers since they are both universal approximations and can approximate arbitrarily well any continuous function of n variables on a compact domain. The difference between them is the number of neurons required to perform the task. In a shallow network, is necessary a huge quantity of neurons since the abstraction must happen in a single layer, while deep networks need fewer neurons but more layers to perform the abstraction, which allows more versatile networks with fewer neurons and faster computations [90],[91].

As a rule of thumb, when a hidden layer scheme is used it is recommended to use a decreasing number of neurons per layer, decreasing from the number of inputs to the number of outputs in the last layer. The optimal set-up actually is obtained

by trial-and-error by using as a compass the loss function per epoch, the moment in which the weights are actualized. If a new set-up makes a relevant change, this will be reflected on the loss per epoch, which also allows the analyst to verify the presence of over-fitting in the algorithm.

After building the ANN model, the next step is to choose how the weights will be actualized,referred as *Run Neural Network* in Fig. 3.1. To do so, a Loss function and an optimizer must be chosen. A list of all the available loss functions can be found at [92] and a comprehensible explanation can be found in [5]. The loss function determines the underlying topology to be optimized, which also affects the optimizer's performance.

In Keras, several loss functions are implemented for both regression and classification. In regression problems, losses as Mean Squared Error, Mean Absolute Error, Log-Cosh, among many others. In classification, loss functions as Binary Cross-Entropy, KL-Divergence are available. The loss function choice depends on the problem tackled and the user should choose it with care. In this work, the loss function used will be always Mean Squared Error due to its interpretability.

The main idea behind the most common optimizers is the gradient descent. Gradient Descent is an optimization scheme that aims to minimize a function by taking the steepest descent, ie the gradient, in each iteration, that is

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k),$$

where α^k is the step size on iteration k and $\nabla f(x^k)$ is the gradient at the same iteration. The step size can be optimized at each iteration:

$$\alpha^k = arg\min_{\alpha}(f(x^k - \alpha \nabla f(x^k)).$$

Note, however, that optimizing the step size would require a bracket search, which increases the computational process cost. In the machine learning community, the step size is normally called the learning ratio and is a prescribed value.

Gradient descent is not a reliable method in any topology, since it has hard times with narrow valleys because of the number of iterations required to achieve its floor. To avoid this problem methods as Conjugate Gradient can be used and standard Gradient Descent can be improved with the addition of momentum or Nesterov momentum, an improvement of classical momentum method.

Momentum methods surely reduce the iterations to achieve a minimum; however, they have as the main issue the fact that all x components are updated with the same learning rate, which is a problem when the gradients are sparse. To solve these questions, several methods were developed as Adagrad, RMSProp, Adadelta, Adam, and Nadam, which can be found in detail in [87]. Due to its ease to deal with sparsity, which happens often in real problems, the aforementioned methods are commonly used as optimizers in ANN regression.

Besides choosing an optimizer and a loss function, ANNs also requires a batch size and many epochs to perform the training, since it uses Stochastic Gradient Descent. Batch size is the number of samples used in each iteration, that will give ANN the gradient information and an epoch is the number of passes of the entire training data set that the machine learning algorithm has completed. Nowadays, a good practice is to choose the batch size as an integer division of data and the number of epochs chosen must be long enough to avoid under-fitting and sufficiently short to prevent over-fitting, which is still figured out by trial-and-error and post-processing heuristics.

Concerning post-processing, complementary procedures are created to explain as many training aspects as possible. The first step is to evaluate Model History, which will detail how was the procedure in terms of training loss and validation loss per epoch. If the first is significantly less than the second, the model was over-fitted(Fig. 3.2[93]) and if both values are high the model was under-fitted (Fig. 3.3[94]).

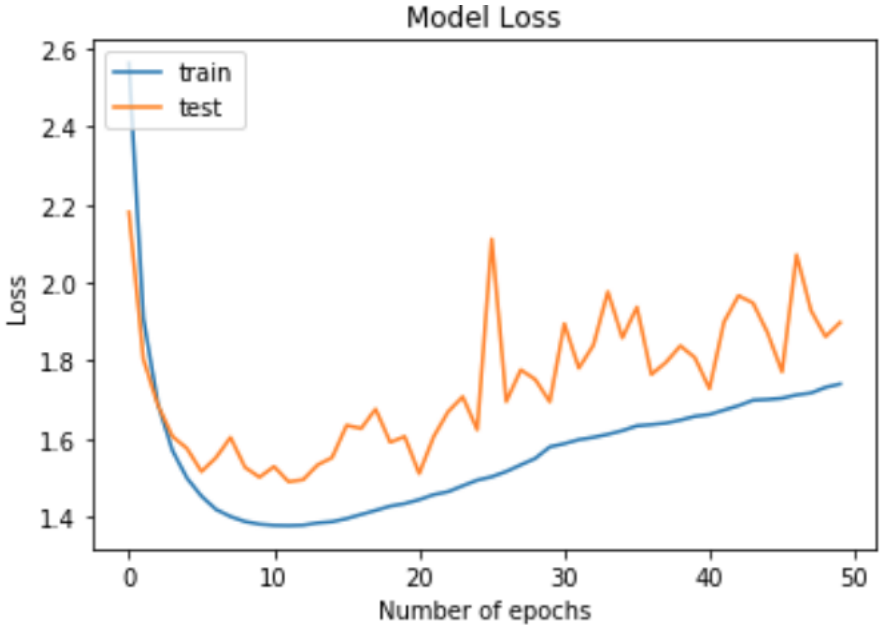


Figure 3.2: Loss function per epoch suggesting overfiting.

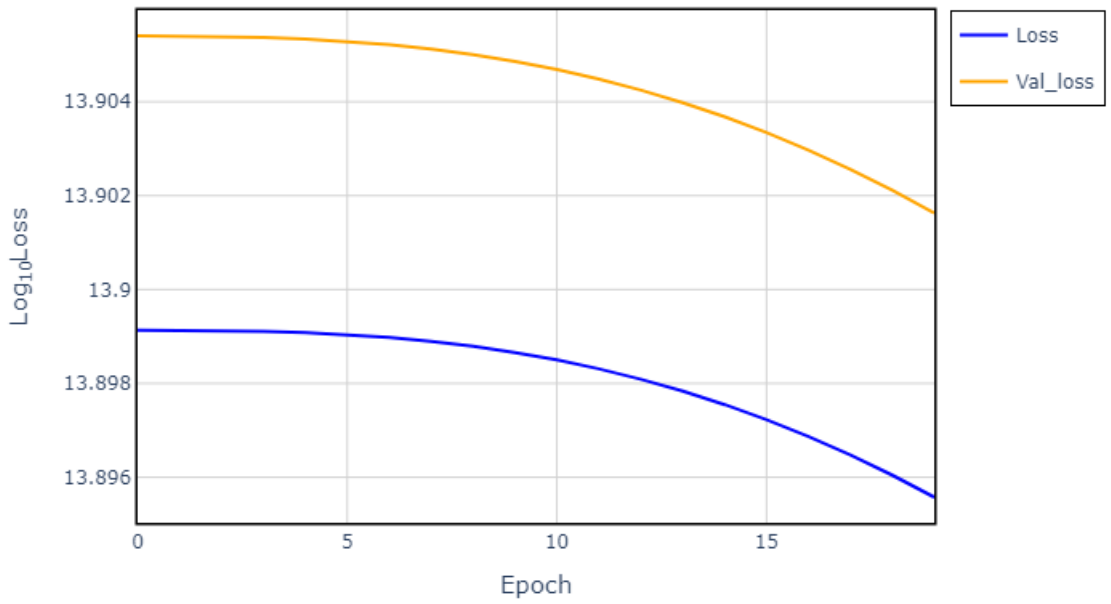


Figure 3.3: Loss function per epoch suggesting undefitting

To prevent over-fitting, some strategies could be used, as early stopping [95], regularization [37] or Dropout [96]. The first technique stops the optimization process if the loss function stops decreasing after a defined number of epochs. Regularization is a technique that adds to the loss function a regularizer term that penalizes complex solutions. Dropout is a technique that at each iteration utilizes in some layers a random fewer number of neurons than the total available, obtaining thus a simpler model.

Besides the above strategies, regularization techniques as Lasso, Ridge, Elastic-Net [5] and SR3 [97] arise to impose restrictions on the solution spaces and obtain feasible solutions. Lasso, also called L1 regression, induces sparsity in the weights parameter space, Ridge, also called L2 regression, equalizes the weights, while Elastic-Net is a compromise between L1 and L2. SR3 is an evolution of Elastic-Net, which aims to improve its performance and has greater flexibility.

Underfit can be avoided by increasing the number of epochs necessary to training, by denoising data or use a complex model by increasing the features or increasing the number of layers, in the ANN context.

Besides Model History, is possible to evaluate the model in a statistical fashion. Consider that the data set is actually a sample from a unreachable population, whose distribution must be at least bounded by a confidence interval. A good model will reside inside the confidence band of each feature. To build a non-parametric confidence band,first define the empirical distribution function, given by:

$$\hat{F}_n = \frac{\sum_{i=1}^{n} I(X_i \leq x)}{n},$$

where

$$I(X_i \leq x) = \begin{cases} 1 & ,X_i \leq x \\ 0 & ,\text{otherwise} \end{cases} \tag{3.1}$$

with the distribution function, to obtain non-parametric $1-\alpha$ confidence band by defining:

$$L(x) = max(\hat{F}_n(x) - \epsilon_n, 0)$$

$$U(x) = min(\hat{F}_n(x) + \epsilon_n, 1),$$

where

$$\epsilon_n = \sqrt{\frac{1}{2n} log\left(\frac{2}{\alpha}\right)}$$

To define a confidence bound, it suffices to use the DKW inequality which states the following: "*Let* $X_1, ..., X_n$ F*.Then, for any* $\epsilon > 0$, $P\left(\sup_x |F(x) - \hat{F}_n(x)|\right) \leq 2e^{-2n\epsilon^2}$". Thus, for any F:

$$P(L(x) \leq F(x) \leq U(x)) \geq 1 - \alpha,$$

which is a $1-\alpha$ confidence band. Is important to note that the greater the sample, the narrower is the confidence band. With a few samples, the confidence band tends to be too broader and thus is necessary to adopt another strategy.

To obtain a even more reliable result, further analysis must be performed. Besides confidence bands, another approach is to verify if the distribution provided by the Neural Networks is the same that the one that came from data set. Such task is done with Kolmogorov Hypothesis Test, which uses the distance between the empirical distribution functions of the train and test functions, in order to verify if the model could also sample data from the population, which reflects the surrogate model reliability. The test statistic of the Kolmogorov Test is

$$D_{n,m} = sup_x |F_{1,n}(x) - F_{2,m}(x)|.$$

For large samples the null hypothesis of both distribution being equal is rejected at level α if $D_{n,m} > c(\alpha)\sqrt{\frac{n+m}{nm}}$, being $c(\alpha) = \sqrt{\frac{1}{2}\log\left(\frac{2}{\alpha}\right)}$. If all features resides on the confidence bands and all the Features pass the Kolmogorov Test, the results are consistent. However, if all the features resides on the confidence bands but the Features does not pass the Kolmogorov Test, a Welch Test should be conducted, in order to verify if at least the distribution has the same mean. To perform a Welch test, one needs to calculate both the t statistic and the degrees of freedom, given respectively by:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{s_1^2/N_1 + s_2^2/N_2}}$$

$$\nu = \frac{(s_1^2/N_1 + s_2^2/N_2)^2}{\frac{s_1^4}{N_1^2\nu_1} + \frac{s_2^4}{N_2^2\nu_2}},$$

where $\bar{X}$,s_i,N_i,$\nu_i = N_i - 1$ are respectively the sample mean, sample standard deviation, number of samples and the number of degrees of freedom of the ith variable. With these two parameter, namely t and ν, the null hypothesis of same mean is checked by using the T-Student distribution.

Aside all the numerical techniques to check model reliability, is also possible to perform graphical methods, such QQ plot and Standardized Error plot. QQ Plot shows the surrogate model results plotted against the target-model ones. If the surrogate model succeeds, its results will lie close the line with 45 degree slope (Fig. 3.4), else, the results will be distant from this line or even present a non-linear behavior (Fig. 3.5).

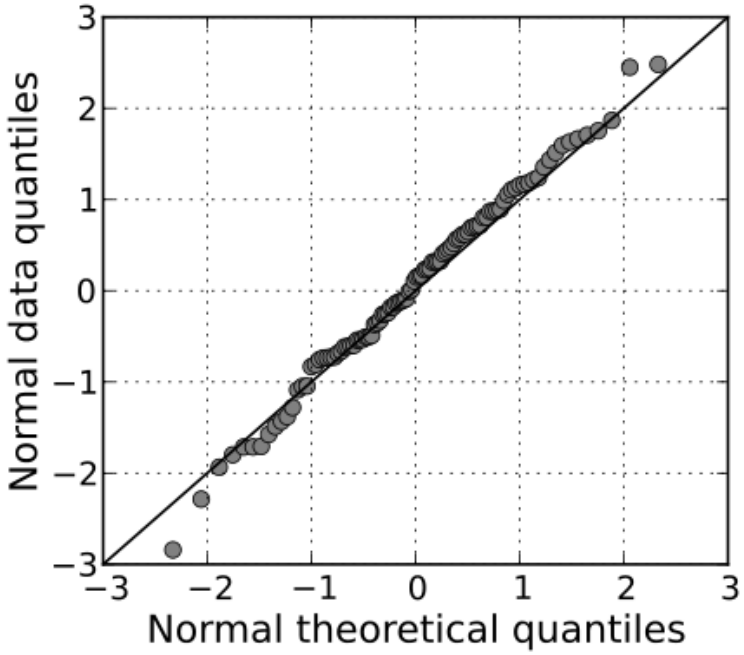


Figure 3.4: A example of a well-trained model. Note that QQ-plot shows that the values are close enough of the target-model. Source: Wikipedia

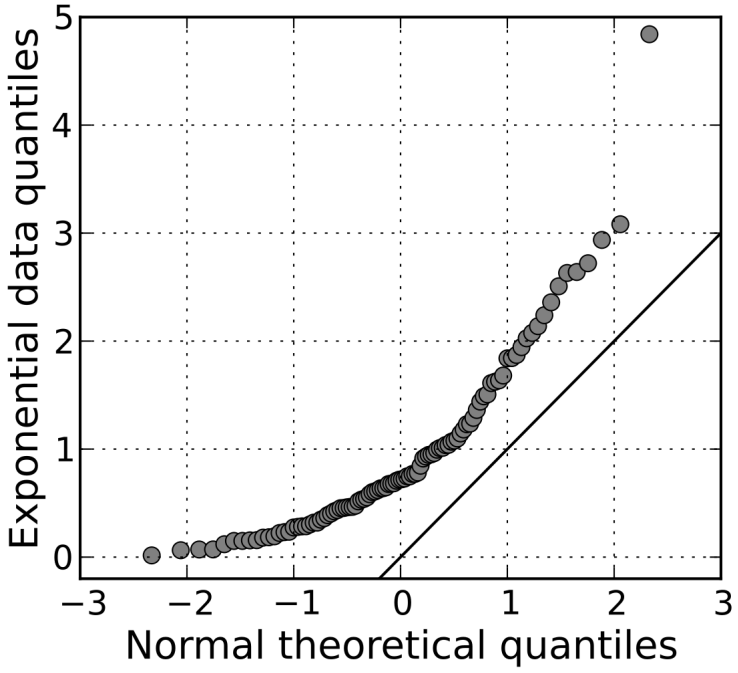


Figure 3.5: A example of a poorly trained model. Note that in this case, the trained values are distant from the target-model and present non-linear behavior. Source: Wikipedia

As a complement to the QQ-plot, is common to visualize also a standardized residual plot(Fig.3.6[98]) that consists of plotting the standardized error,namely $\frac{y-\hat{y}}{\sigma_{(y-\hat{y})}}$, against the test values,y. The main characteristic of this plot is to exhibit the dispersion at each point of the reference model, which helps to find how accurate has the training across the training set range. If the variance is a function of the values, this shows that the training does not have significance at this range, which suggests that this region requires more samples.

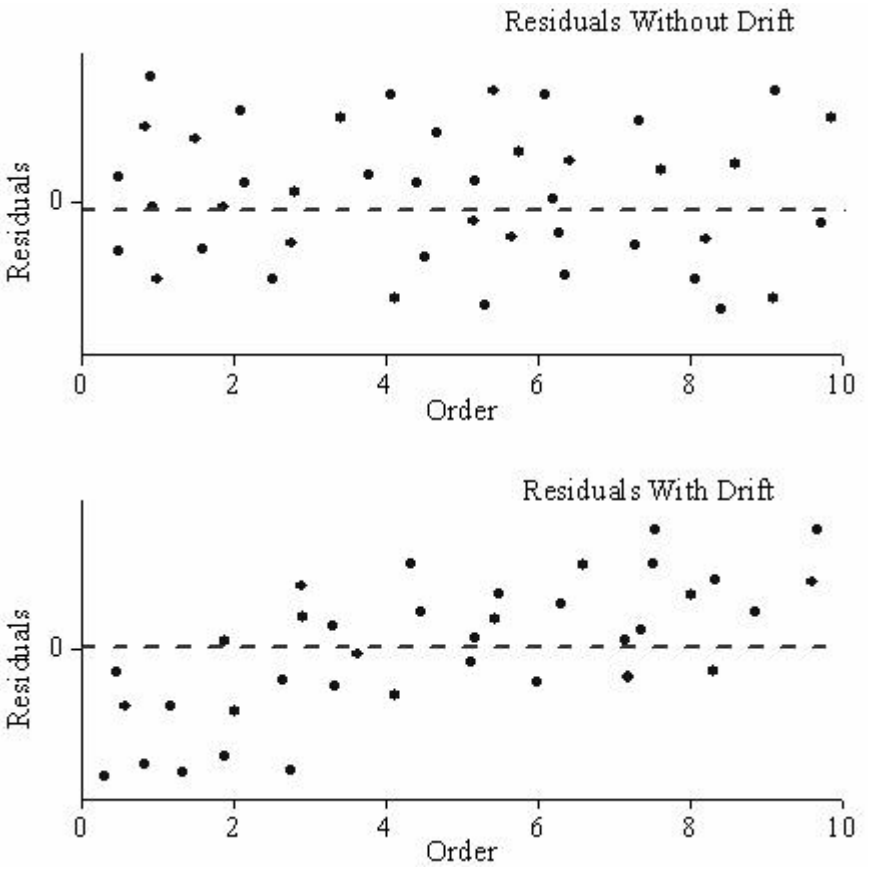


Figure 3.6: A example of the plot of the standardized residuals. Note that the first case shows a model hypothetically well trained at every range, while the second model suggests an underestimation at the lower values and overestimation at higher values. To solve this kind of problem, the model must be reviewed or a new data set must be taken.

.

In cases when data is insufficient, the target model must be revisited with more inputs. To obtain more of them, a copula strategy can be performed without any greater risk. There are many approaches to obtain the marginals and it will depend on how much information is known.

The scenarios covered here are three:

- the user only has minimal knowledge about the subject, knowing very little about the marginals;
- the user does not have complete data, but knows the marginals probability density functions (pdfs) and supports;
- the user has a reasonable amount data, but do not know which distribution to fit in each marginal;

To solve the first case, a solution is to use the available information to infer what inputs should be. One approach is to use the principle of Maximum Entropy(MaxEnt), which states that the current state of knowledge is represented by the probability distribution function (pdf) with the largest entropy [99]. Another approach is using

When this strategy is used, new data can be generated from the discovered probability density functions by using rejection sampling; however, the generated data is uncorrelated. A straightforward way to generate correlated data keeping as marginal de pdfs of MaxEnt method is to generate data from a multivariate normal distribution, $N(0, \rho)$, being ρ as a user-defined correlation matrix, map the marginals into (0,1) Uniforms by using Cumulative Distribution Function and then remap these marginals into those obtained with MaxEnt by using empirical percent point function (ppf), obtained from the inverse empirical cumulative distribution function [31] from each one of the uncorrelated marginals. Schematically, we have this strategy as described in figure 3.8.

The second case is quite straightforward since the user knows which distributions to use, being his only concern on how to generate correlated samples of dependent variables, if any. The suggestion is to initially use the same strategy as above; however, the Achilles' heel of this approach is the presence of asymmetry between correlations, which was one of the Financial Crisis'(2007-2008) causes. Aside from the coupling using normal distribution is possible to use other functions to do so, as shown in figure 3.9[100, 101]. An extensive explanation of this topic can be found in [102]

When a reasonable amount of data is available, the marginals can be obtained with the aid of Kernel Density Estimation [31], a non-parametric method, or by fitting beta distribution on continuous and unimodal data using Maximum Likelihood

Estimation (MLE) [31] and by re-sampling the discrete data following the proportion of each outcome. The main drawback of non-parametric is the tuning kernel parameter. In literature, several procedures have been developed to solve this issue, being the most prominent ones [103] and [104]. To by-pass this problem, the aforementioned MLE strategy can work, due to the versatility of beta distribution (Fig. 3.7) and the ease to re-sample data from discrete distributions, if this were the case. If there is a need of even more versatile distributions, one should use Generalized Lambda Distributions [105].

Before coupling the marginals, one should quantify how good the fit was by calculating the Kullback-Leibler divergence between data and the fitted samples, in other words:

$$D_{KL}(P||Q) = -\sum_{k} p_k \log\left(\frac{q_k}{p_k}\right),$$

where P is the reference, Q is the proposed model and p_k and q_k their respective samples. A good fit should result into a value close to zero, which implies that $Q \approx P$

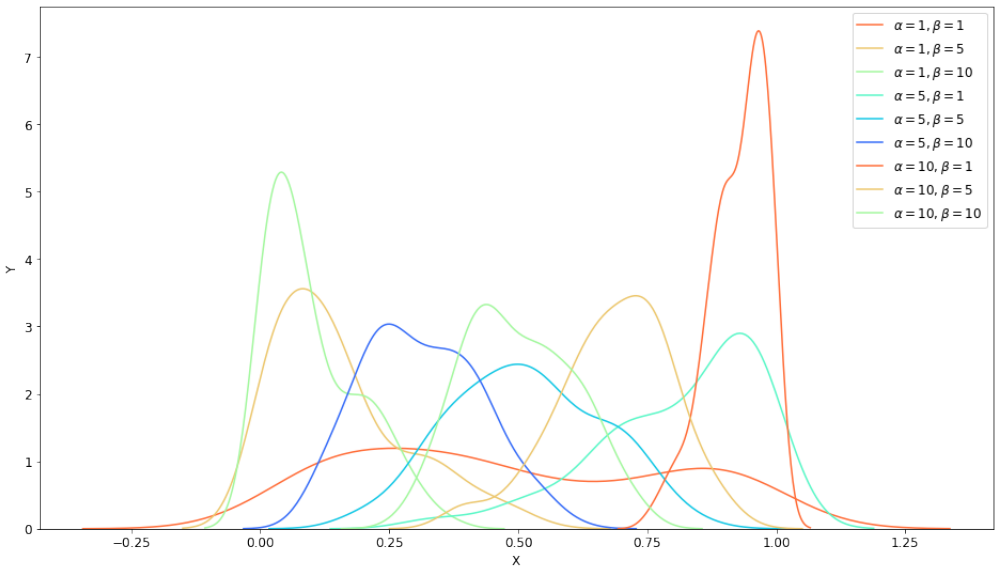


Figure 3.7: Beta distributions.

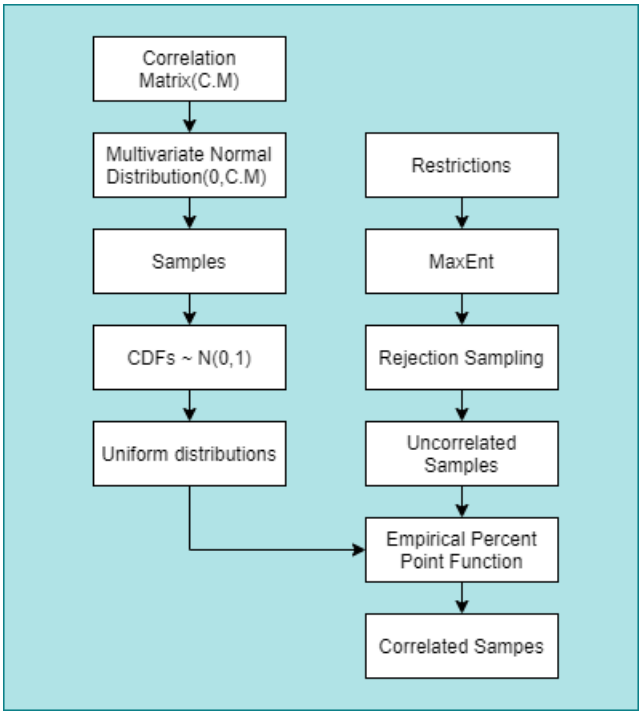


Figure 3.8: schematic showing how to generate correlated data using the Maximum Entropy principle with just the information of marginals

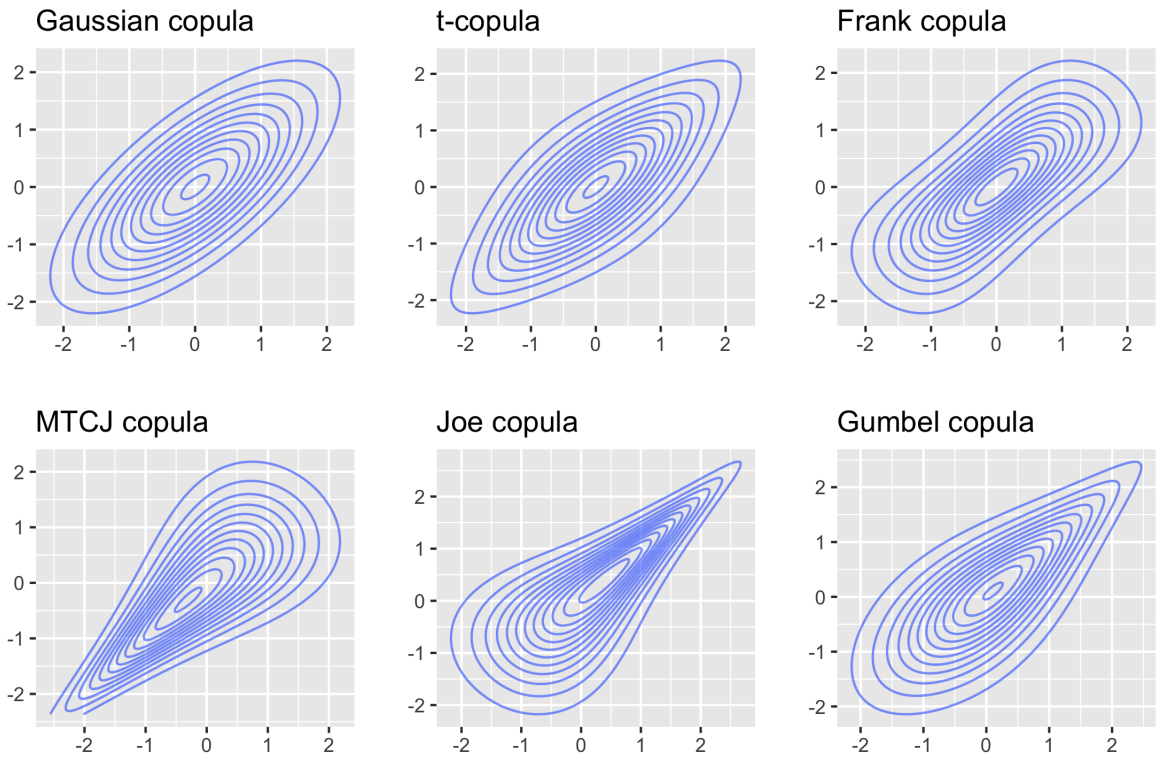


Figure 3.9: Different functions that couple marginal distributions

Chapter 4

Program usage

This chapter shows how to use the program efficiently by showing examples of each program's functionalities. The first step is to subdivide the data set into features and labels, the former being the inputs and the latter the outputs.

Before constructing the neural network itself, is possible to perform feature reduction, which is recommended when the number of features is considerably high and optimization cannot achieve a reliable minimum, which compromises model's accuracy.

To exemplify the effects of feature reduction, a neural network with the architecture 20:16:16:8 (Fig. 4.1) and is trained with a database with 20 features(Fig. 4.2) and its results are compared with another network trained with 15 (Fig. 4.3) and 3 (Fig. 4.4) features,respectively. Is worth noting that the input layer have always the number of neurons equivalent to the reduced number of features.

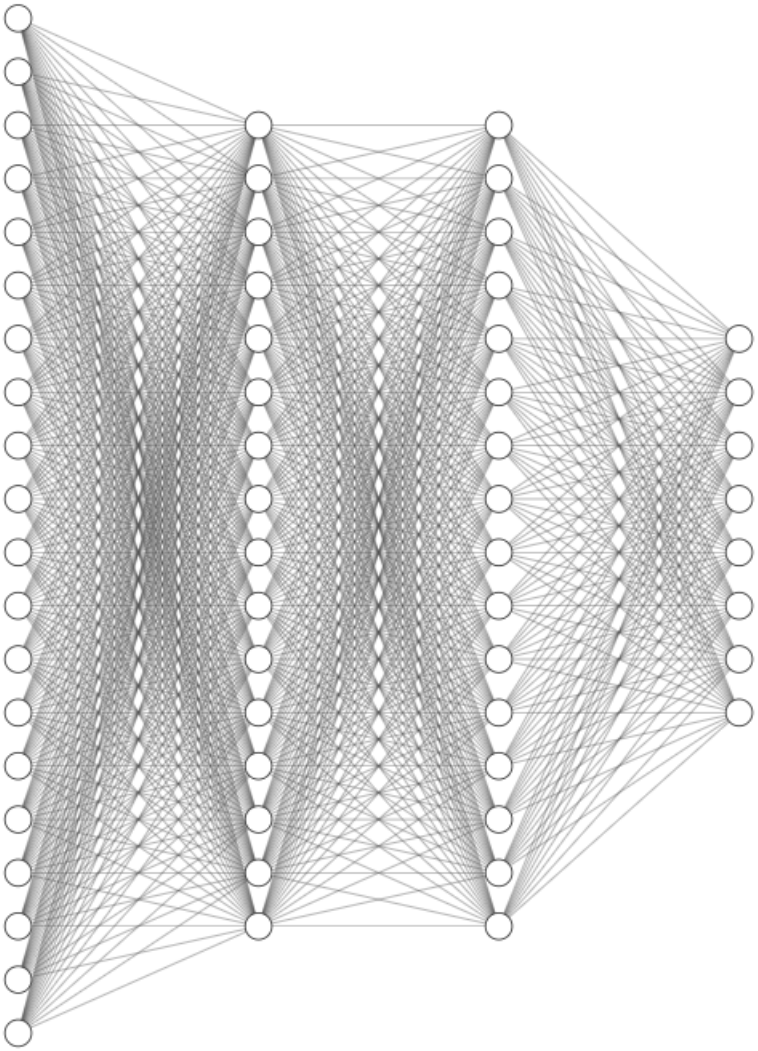


Figure 4.1: Architecture used in example

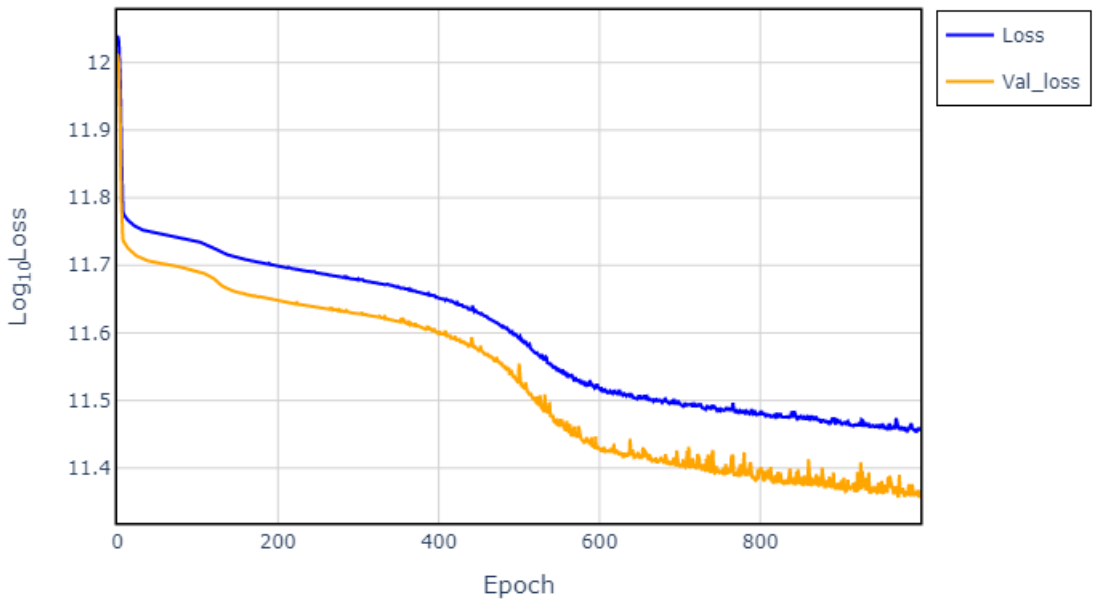


Figure 4.2: Loss function per epoch obtained using all the features available to train the model

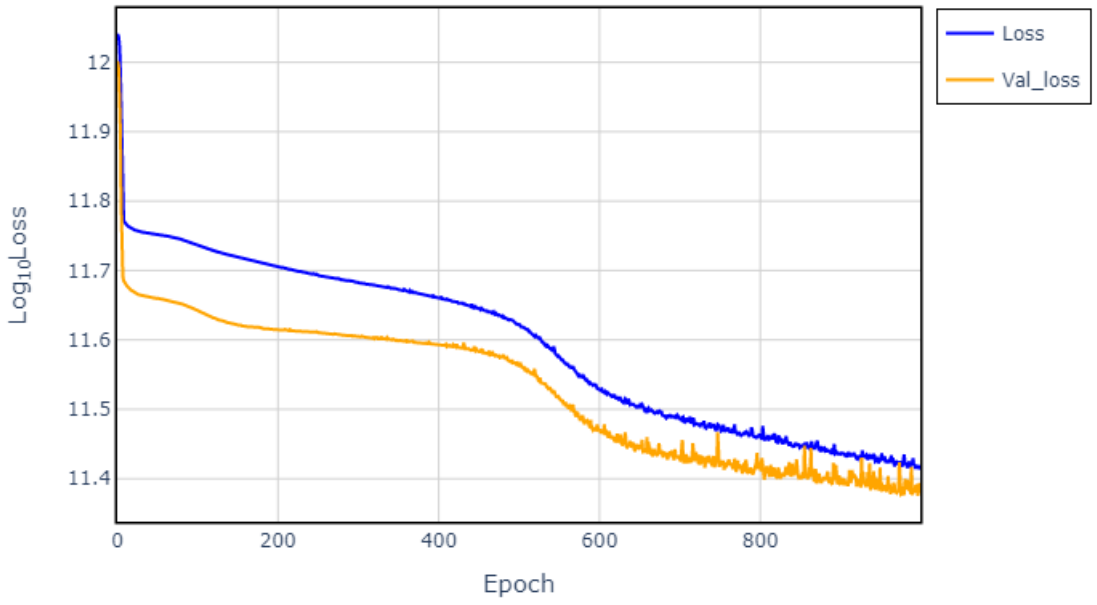


Figure 4.3: Loss function obtained using 15 features to train the model

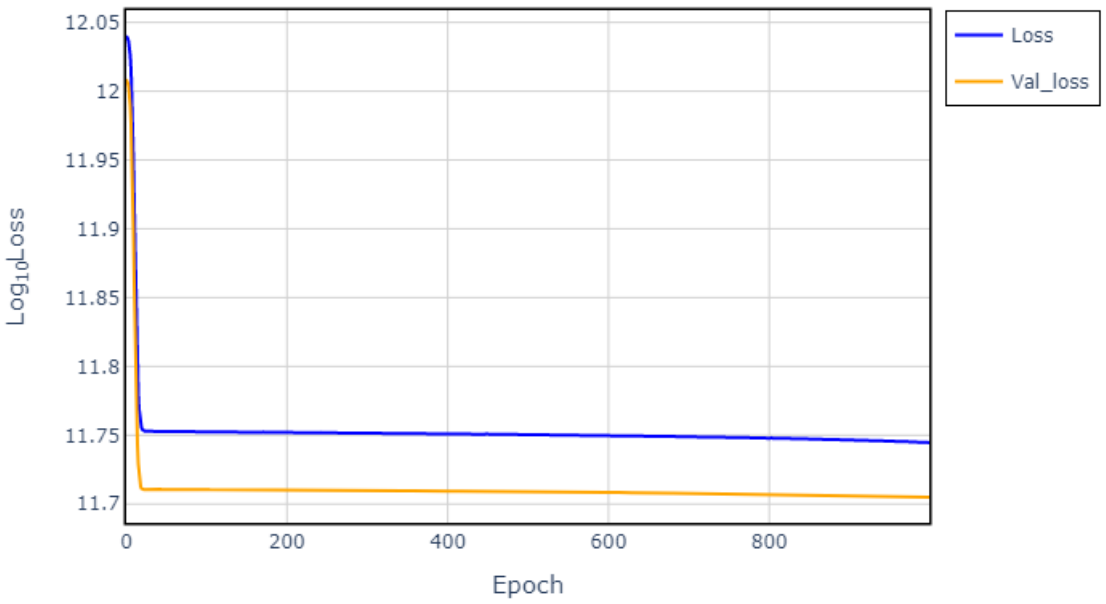
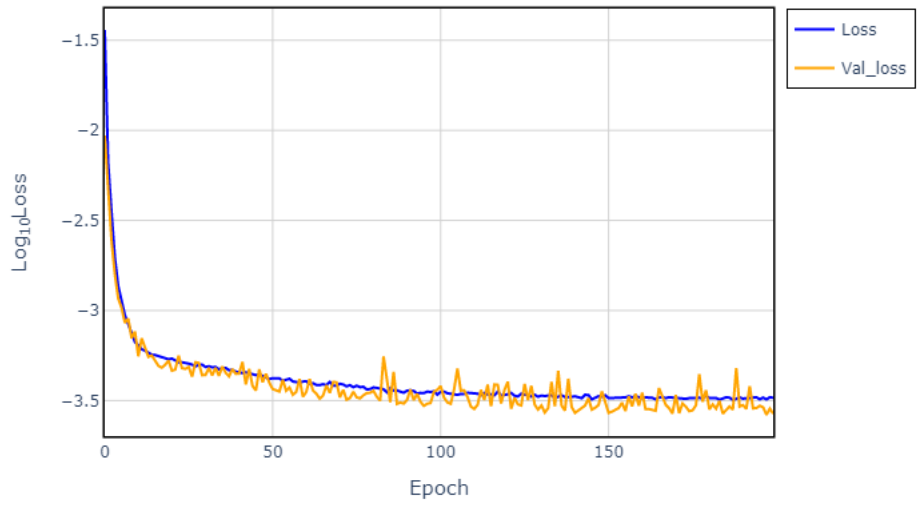


Figure 4.4: Loss function obtained using 3 features to train the model

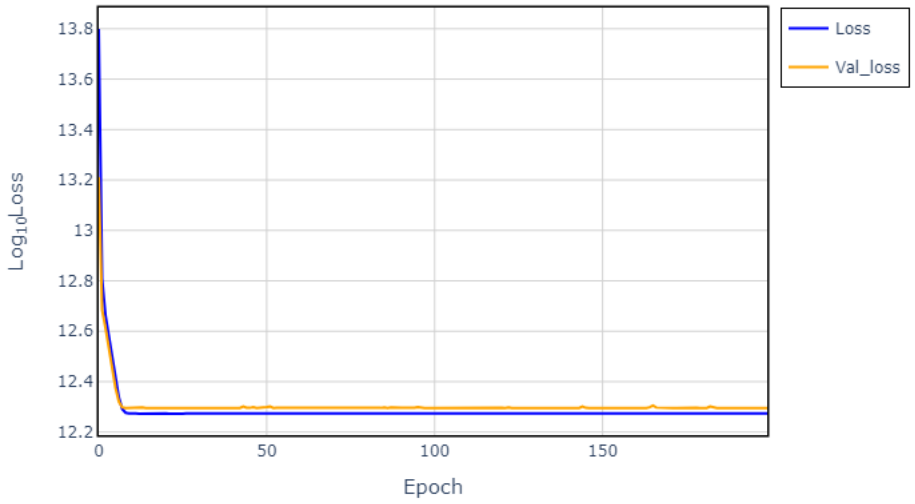
Note that the use of fewer features does not change significantly the model performance until 3 features. After that, it turns out that the optimization algorithm gets stuck on a local minimum, making further learning impossible.

With the selected features, the next step is scaling. In this example, both inputs and outputs were scaled by Scikit-Learn MinMaxScaler, which maps the data set into the [0,1] interval based on data amplitude. Scaling reduces the computational effort in obtaining a minimum in the optimization process. A comparison between a scaled and unscaled set is shown in figure 4.5

The next step is to build the neural network by setting the number of hidden layers used, as both input and output layers are defined by labels and features: After building ANN, the next set is to run in by defining the number of division of data set, namely batch size, and the number of iterations in which the weights are updated, namely the epochs: By standard, the optimizer used is Adam and the loss function to be minimized is the Mean Squared Error. Both can be changed to any other optimizer or loss-function available in the Keras environment. To choose properly the optimizer, [87] is a good reference and loss-functions are widely discussed on [5, 31].



(a)



(b)

Figure 4.5: Difference between a scaled model and a unscaled one. Note that scaling allows a better minimization of loss function, while unscaled data has higher error and a worst minimization.

After training, the results can be further analyzed by post-processing procedures. In ROSS-ML, the available analysis is:

- Model History
- Metrics
- Hypothesis Tests

- Confidence Bounds
- Q-Q Plot
- Standardized Residuals Plot

Model history, as stated before describes the evolution of loss function per epoch, which allows us to diagnose the optimization process. Metrics describe the model's performance quantitatively, being the following ones implemented on ROSS-ML:

- MAE: $\frac{\sum_i |Y_i - \hat{Y}_i|}{n}$
- MSE: $\frac{\sum_i (Y_i - \hat{Y}_i)^2}{n}$
- R^2: $1 - \frac{\sum_i (Y_i - \hat{Y}_i)^2}{\sum_i (Y_i - \bar{Y}_i)^2}$
- R^2 adjusted: $1 - 1 - R^2 \frac{n-1}{n-p-1}$
- Explained Variance:$1 - \frac{Var(Y - \hat{Y})}{Var(Y)}$

MSE is a measure of the quality of an estimator—it is always non-negative, and values closer to zero are better. This measure is recommended when the overall values are greater than one, since squaring values less than this threshold always gives low values, which can be misleading.

This question is solved by using MAE. This metric looks only at the absolute error and thus penalize points with a low order of magnitude more than MSE. However, distant points are less penalized than closer ones when compared to the MSE, which is troublesome since it induces a low risk, which is misleading in the decision process.

R^2, adjusted R^2, and explained variance aim to quantify the quality of fit. The first aims to compare both the spread around the fitted curve and that around the mean. A good model is the one that has significantly less variance in comparison to the second centered moment of data itself. Adjusted R^2 corrects its counterpart by accounting the phenomenon of automatic increasing of R^2 when extra explanatory variables are added to the model and adjusted variance compares directly the second centered moment of the error distribution and data distribution. So, The closest these metrics are to one, the better.

The aforementioned metrics should be used with care since they cannot tell objectively if the model is accurate enough to a specific application. Due to that, MSE and MAE are also called risk functions [31] and the use of R^2-like metrics alone can be also misleading. [106].

To complement these metrics, graphics and statistical analysis are available in ROSS-ML. As described in Chapter 3 KS-Test and Welch-test can be used. The first, as described before, tests if the surrogate model could reproduce the same distributions as the physical model, while the last only verifies the equality between the first moments. To use the Welch test properly is recommended to have a normal distribution or at least a non-skewed distribution. If this condition cannot be achieved, the test loses its power and to amend that the significance level should be changed or the distribution should be transformed by using Box-Cox or Yeo-Johnson transforms for example.

As stated on Chapter 3, the available graphics besides model history are:

- Q-Q plot (Fig. 4.6)
- Standardized Residuals Plot (Fig. 4.7)
- Confidence bounds (Fig. 4.8)

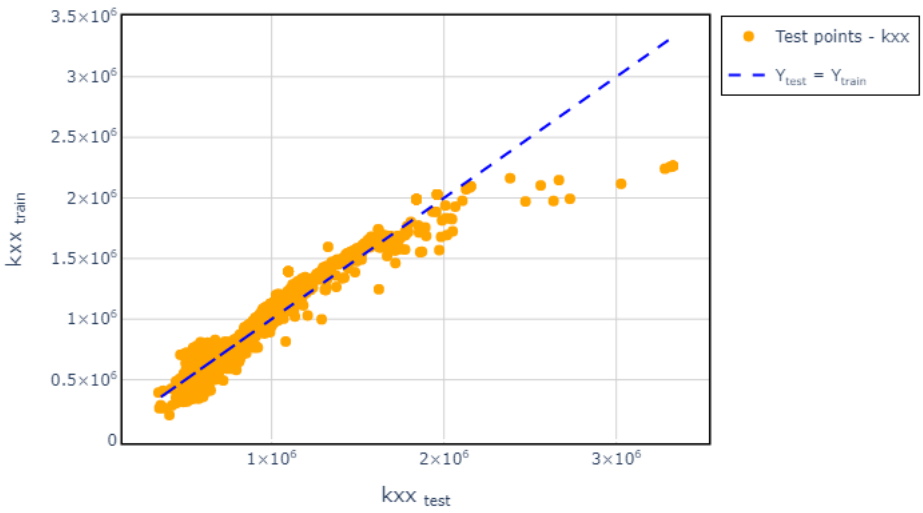


Figure 4.6: A quantile-quantile plot obtainable with ROSS-ML

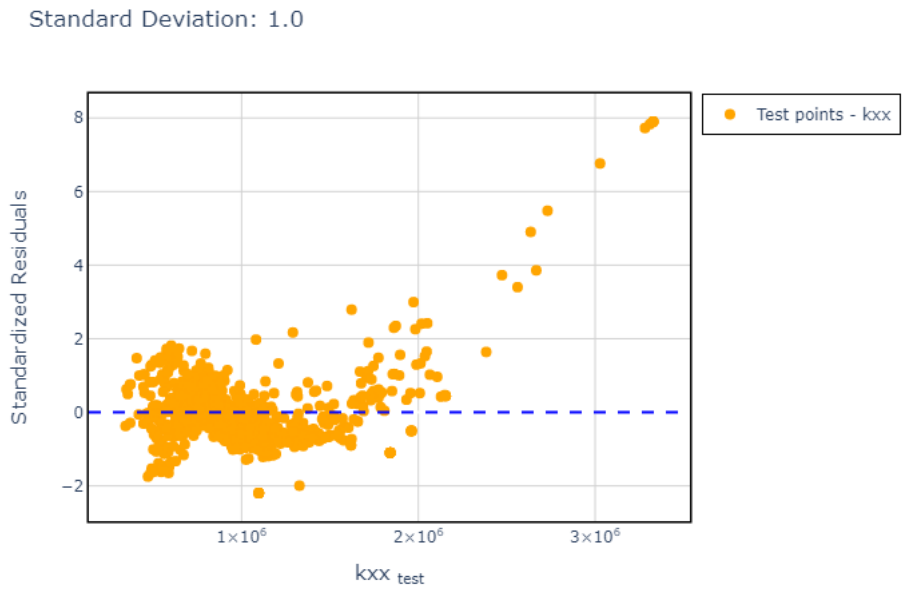


Figure 4.7: A standardized error plot obtainable with ROSS-ML

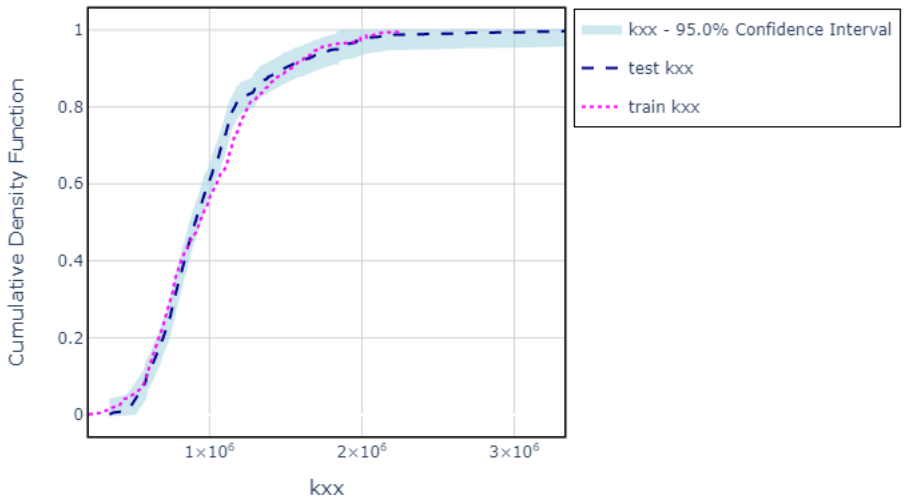


Figure 4.8: A 95% confidence interval obtainable with ROSS-ML

According to [107], the Q-Q plot is a graphical technique for determining if two data sets come from populations with the same distribution. The closer the points are from a 45-degree reference line, the more similar are the distributions. The greater the distance from this reference line, the greater is the difference of the underlying distributions.

Q-Q plots are quite advantageous, since it can compare samples with different sizes and also allows the analysis of many distributional aspects of the results, namely shifts in location, scale, changes in symmetry, outlier detection and presence of multimodality. In a Q-Q plot, if two data sets from populations whose distributions are different just in location, the points will be shifted from the reference line (Fig. 4.9). If the difference lies on scale, the slope will be different from 45 degrees (Fig. 4.10). When the distributions are different, the scattered curve shows a non-linear behavior (Fig. 4.11)

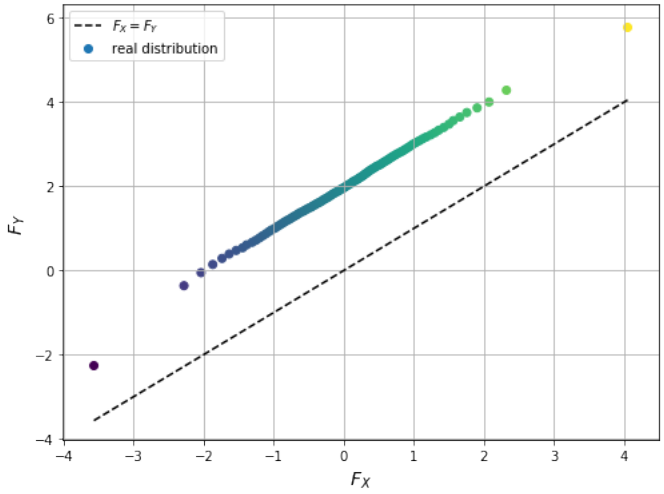


Figure 4.9: When the difference lies on location, the scattered points are shifted from the reference line.

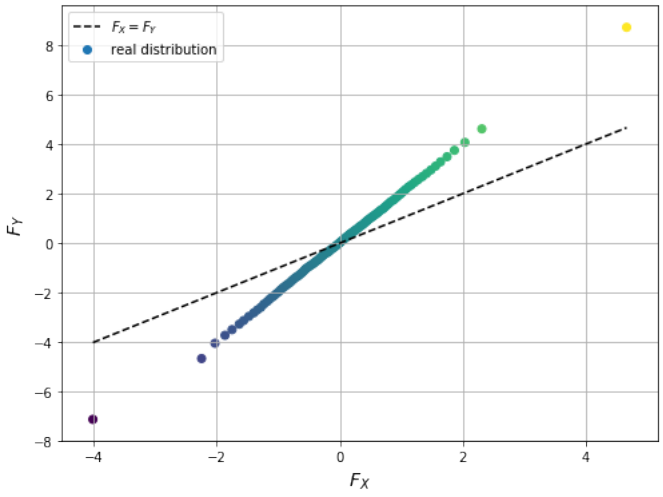


Figure 4.10: When the difference lies on scale, the scattered points has a different slope from the reference line.

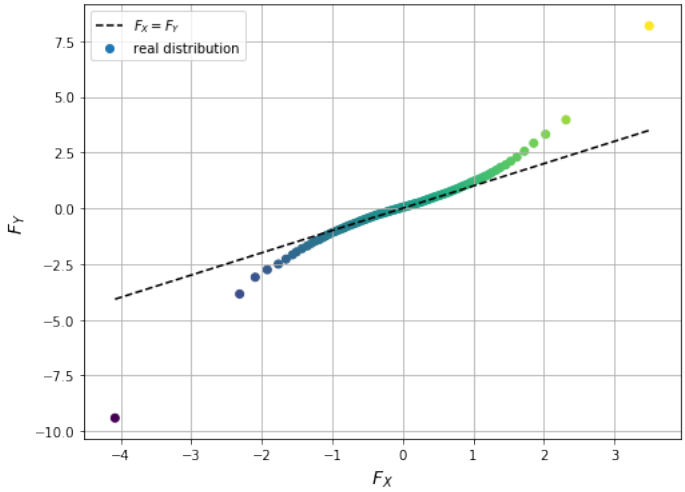


Figure 4.11: When the difference lies on distribution, the scattered points has a non-linear pattern

A standardized residuals plot (SEP) is a tool to assess the sufficiency of the functional part of the model. Normally, in regression the models follow the structure:

$$Y = f(X) + \epsilon,$$

where $f(X)$ is the adjusted model and ϵ is the associated error. When the model fits data adequately, ϵ presents a random behavior towards zero, often supposed normal. On the other hand, an inadequate model presents a systematic structure in its error, which suggest that the model can be further improved or that ϵ has complex structure. As an example of SEP use is shown in figures 4.12 and 4.13

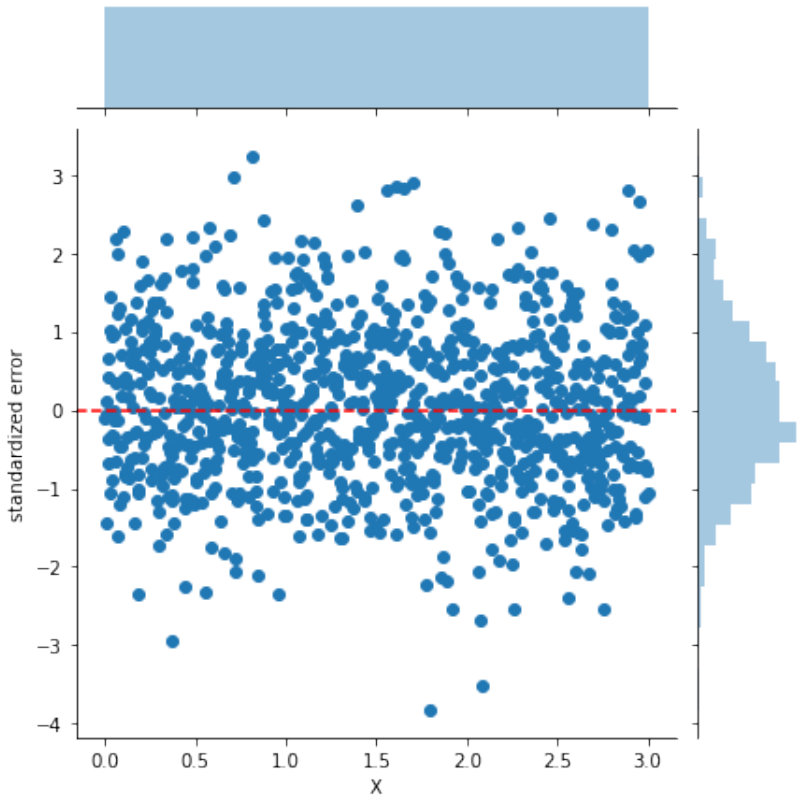


Figure 4.12: When the model is adequate, the residual behaves itself as a random variable towards 0. In this example , the random points have been generated by adding noise with the form $N(0, 3)$ to the function $y = 2x + 4$, which has been chosen as the model to fit data. Note that in this case the residuals shows clearly a random behavior.

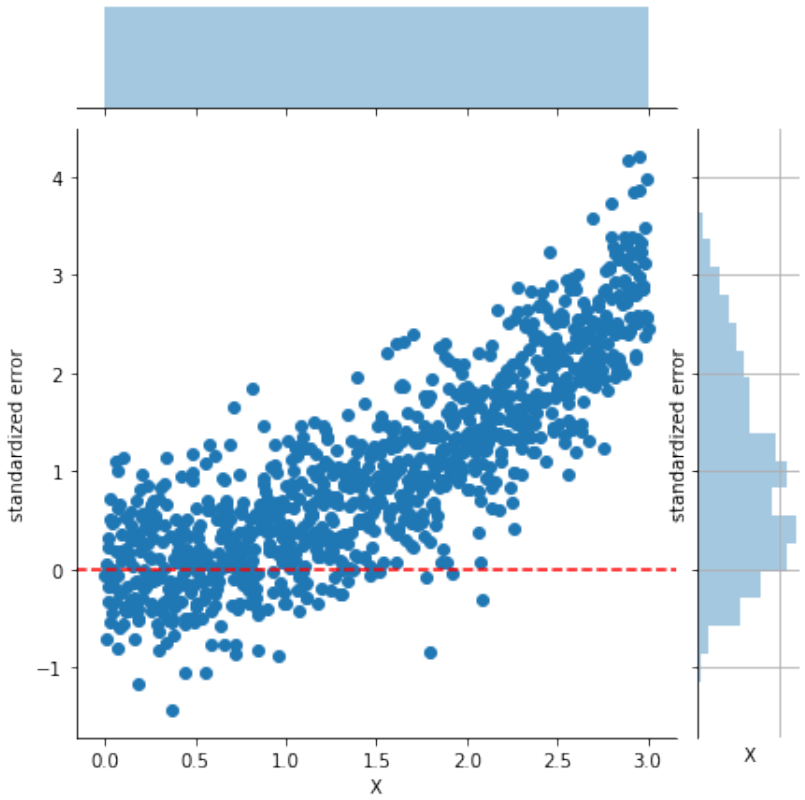


Figure 4.13: When inadequate, the residuals shows a systematic behavior. This case use as data the same set as before, differing only in the chosen model is now $y = 2x^2 + 2x + 4$. Thus, the residuals have the form $\epsilon = 2x^2 + N(0, 3)$ and now present a systematic behavior

Confidence intervals are tool to identify the range of values that is likely to include a population value with a confidence level. In ROSS-ML, this tool generates a confidence interval around each variable from the test set, which tells us the possible whereabouts of another sample 95% of the time. So it is reasonable to think that if a model generates good predictors, the predicted values should lie in the confidence intervals since both test sets and the predicted values would come from the same distribution.

With all ROSS-ML's features explained, it is time for an example. Here, it will be used the journal bearing equation [34, 108–111] available on ROSS, in which the governing equation of the pressure field is a simplification from the Navier-Stokes:

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v}\right) = \nabla\cdot\sigma\,, \tag{4.1}$$

where $\mathbf{v} = (u, v, w)$ is the velocity field, ρ is the fluid density,σ is Cauchy's tensor and t is time variable. The usual hypotheses of Newtonian incompressible fluid and laminar flow are considered. In cylindrical coordinates, considering a small gap

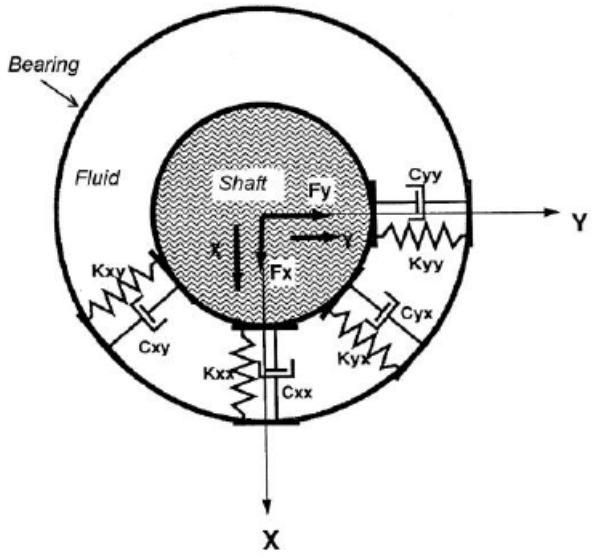


Figure 4.14: Coefficients obtained by integrating the pressure field over the area

between to shaft and the bearing, we arrive at the system of differential equations [110, 111]:

$$\begin{aligned} -\frac{\partial p}{\partial z} + \mu\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)\right] = 0\,, \\ -\frac{1}{r}\frac{\partial p}{\partial \theta} + \mu\left[\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial rw}{\partial r}\right)\right] = 0\,, \end{aligned} \tag{4.2}$$

where p is the two-dimensional pressure field, u is the speed in the axial (z) direction, w is the speed in the tangential (θ) direction, r is the radial direction, and μ is the fluid viscosity. Adding the continuity equation, including the boundary conditions, and discretizing the pressure field by means of the finite difference method, it is possible to obtain an approximation to the pressure distribution. All the development can be found in [110, 111], and the codes are in ROSS [3].

By integrating the pressure field over the journal bearing area, the forces in the x and y directions are obtained. The bearing coefficients (Fig.4.14 [112]) (stiffness-k and damping-c) can then be computed:

$$\begin{aligned} k_{xx} &= \frac{F_x - F_{xx}}{\Delta x} \quad , \quad k_{xy} = \frac{F_x - F_{xy}}{\Delta x} \\ k_{yx} &= \frac{F_y - F_{yx}}{\Delta y} \quad , \quad k_{yy} = \frac{F_y - F_{yy}}{\Delta y} \\ c_{xx} &= \frac{F_x - F_{xx}}{\Delta \dot{x}} \quad , \quad c_{xy} = \frac{F_x - F_{xy}}{\Delta \dot{x}} \\ c_{yx} &= \frac{F_y - F_{yx}}{\Delta \dot{y}} \quad , \quad c_{yy} = \frac{F_y - F_{yy}}{\Delta \dot{y}} \end{aligned} \tag{4.3}$$

Although accurate, this model is too slow for applications such as uncertainty

analysis. To solve this issue, a surrogate model is build with aid of ROSS-ML. To generate the data set, the strategy chosen is as it follows: let P_{ar} be the nominal value of each input parameter of the physics-based model, and p_{er} a perturbation factor of 0.01. To generate the data set, 100 independent samples has been sampled from Uniform random variables with support $P_{ar}[(1 - p_{er}), (1 + p_{er})]$ were used to all variables but the stator and the rotor radius, which respect $P_{ar}[1, (1 + p_{er})]$ and $P_{ar}[(1 - p_{er}), 1]$, respectively, to guarantee geometrical compatibility. The four nominal input parameters are: shaft rotational speed $\Omega = 523.6$ RPM, viscosity $\mu = 2.405 \times 10^{-2}$ Pa.s, fluid density $\rho = 860$ kg/m^3, rotor radius $r_r = 45$ mm. The stator radius is computed as $r_s = r_r + c$, and the eccentricity is computed as $\epsilon = r_s - r_r$. The clearance and load are considered constants, respectively, 194.6×10^{-6} m and 622 N.

To achieve good results, one can use either shallow or deep neural networks [90]. In the first case, the architecture effectiveness is evaluated by gradually increase the number of neurons and computing metrics as R^2 or Akaike Information Criterion (AIC). In the second case, this strategy turns to be infeasible since the number of possible scenarios considerably increases. To address this problem techniques as pruning [113] are used to reduce the architecture.

As an example of shallow neural networks architecture selection, the number of neurons will be gradually increased and R^2 is computed for each architecture 4.16. Here, the best architecture for shallow networks is 6:7:8.

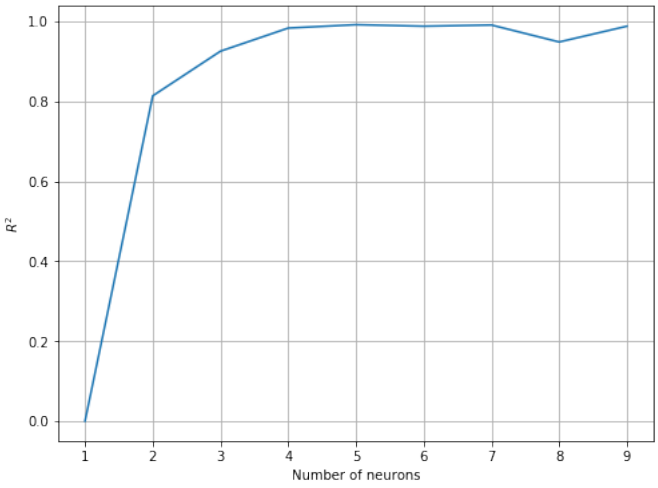


Figure 4.15: Checking the best architecture of a shallow network. Note that 7 neurons already give a good result

Even though the shallow arrangement presented a good R^2, the KS-Test has shown inconsistencies in representing both damping and stiffness coefficients at a level of

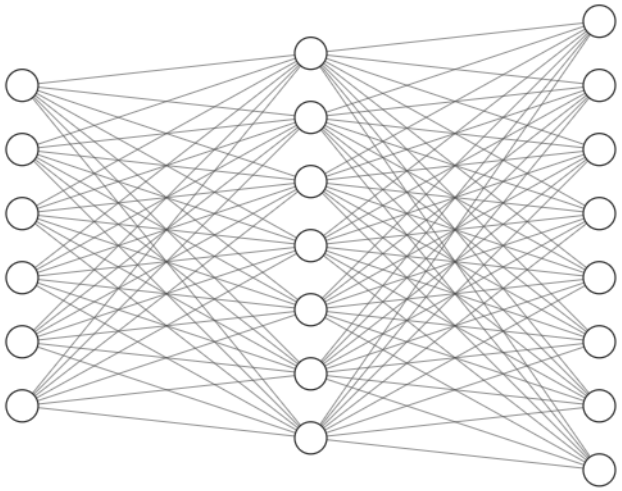


Figure 4.16: The best architecture a priori for a shallow arrangement

significance 5% (Tab. 4.1), which compromises the surrogate model.

Coefficient	p-value	status
k_{xx}	0.024	Reject $\mathcal{H}_0$
k_{xy}	0.094	Not Reject $\mathcal{H}_0$
k_{yx}	0.008	Reject $\mathcal{H}_0$
k_{yy}	0.282	Not Reject $\mathcal{H}_0$
c_{xx}	0.024	Reject $\mathcal{H}_0$
c_{xy}	0.046	Reject $\mathcal{H}_0$
c_{yx}	0.105	Not Reject $\mathcal{H}_0$
c_{yy}	0.024	Reject $\mathcal{H}_0$

Table 4.1: KS Test applied to model results.

As the shallow network fails in representing the model, a deep neural network with architecture 6:8:8:8:8 (Fig.4.17) is chosen to substitute it. After training it without dropout, it is time to check its performance. First, as usual, one should verify the loss function graphic (Fig. 4.18):

As stated in Fig. 4.18, the model overfits; however, if one looks closely at epoch 150 to 200 the loss, the validation loss starts to approximate from training loss, which suggests that increasing the number of epochs could solve the problem. Unfortunately, this is not the case, as stated in the figure 4.20.

As the number of epochs increasing did not solve the problem (Fig. 4.20), it is time to change the approach. Now, we will reduce the number of features to just

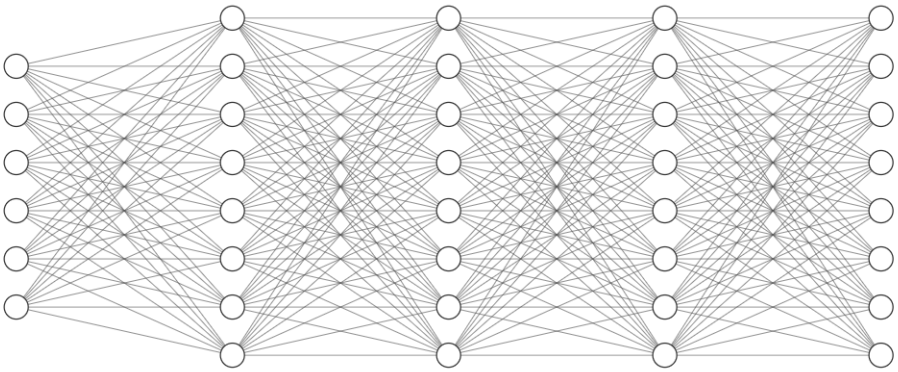


Figure 4.17: Deep Neural Network used to substitute the shallow model

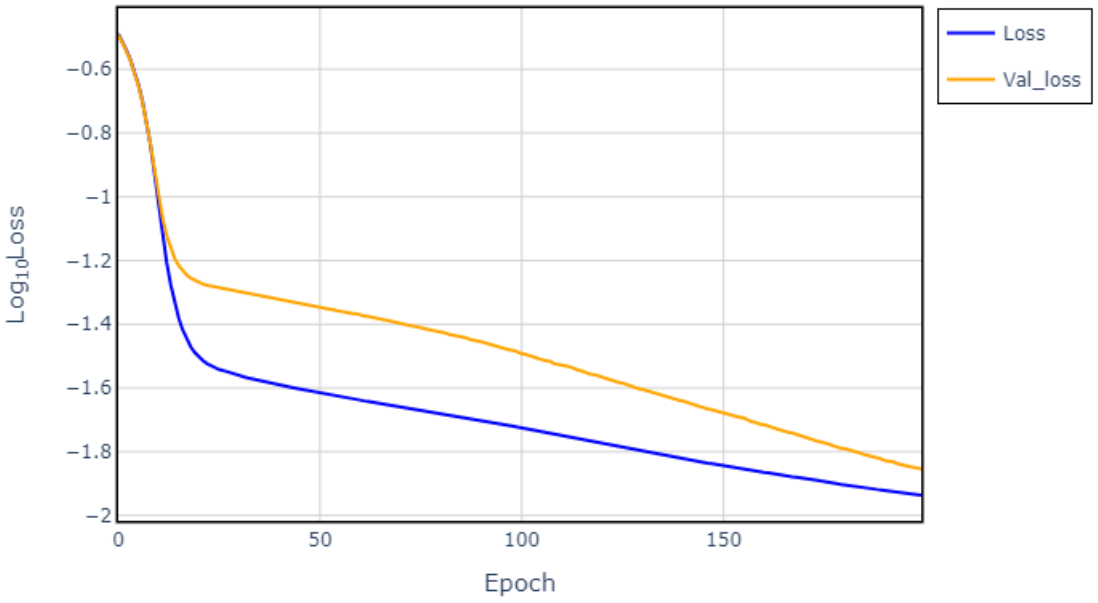


Figure 4.18: Loss function of trained model. Note the presence of overfit after epoch 25

3 and add a dropout of 0.05 to each layer. By doing this, we solve the problem of over-fitting (Fig. 4.20)

After removing overfit from the model, it is time to see how good this model is. To do that, one should analyze the available metrics, since Hypothesis tests are not recommended here, since the number of samples is too small. For this model, the metrics are as presented in table 4.2:

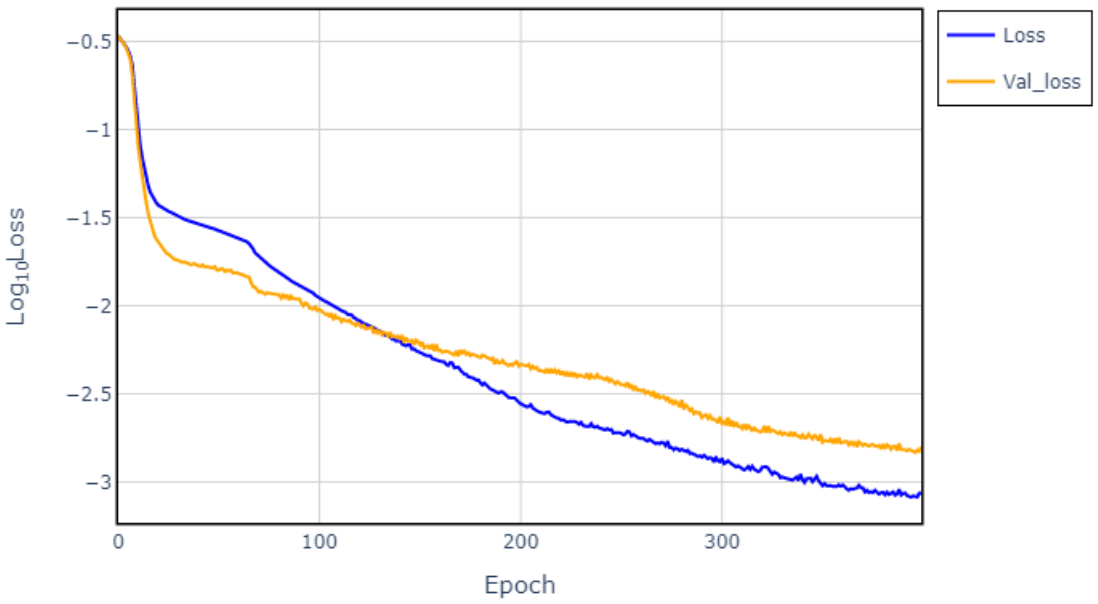


Figure 4.19: Loss function of trained model. Note the presence of overfit after epoch 25 and the sudden change in function behavior from epoch 150 and beyond

Metric	Value
MAE	0.024
MSE	0.001
R^2	0.964
Adjusted R^2	0.893
Explained Variance	0.966

Table 4.2: Database metrics. Note that in spite of good R^2, its adjusted counterpart presented bad results.

By observing the metrics, we note that the adjusted R^2 returned a value less than 0.95, which suggests that this model despite solving the over-fitting problem still is unable to be an adequate surrogate model. Now, one way to solve this problem is to get more data; however, this procedure is very expensive due to the cumbersome computations. In this case, we still can obtain more data, not from the physical model, but from the underlying structure of present data, as stated by [11].

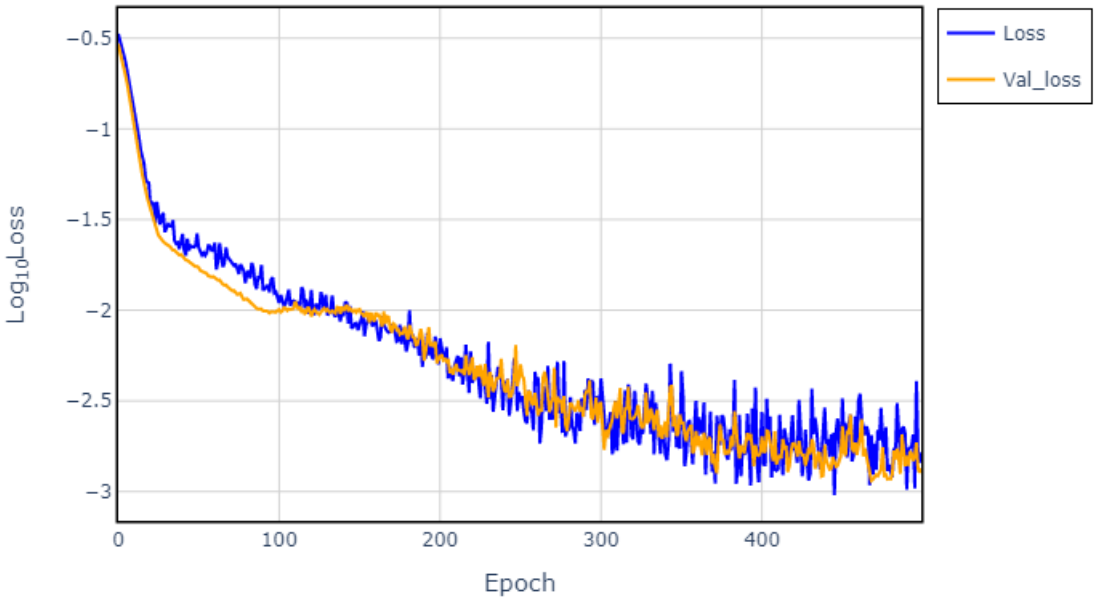


Figure 4.20: Loss function of trained model with feature reduction and addition of dropout layers. Note that overfit was removed from the model; however, as a drawback, the loss function becomes noisy

With this data augmentation technique it is possible to sample as many new samples as possible. Here, we will sample 10,000 new samples with the aid of this technique and retrain the neural network. With this new data set, the ANN is trained with the same architecture as before but without dropout layers. The result is shown in figure 4.21

This model with expanded data has the results shown on table 4.3.

Metric	Value
MAE	0.010
MSE	$<1 \times 10^{-}3$
R^2	0.989
Adjusted R^2	0.989
Explained Variance	0.990

Table 4.3: Expanded data set metrics. Note that the adjusted R^2 improved in comparison to the original data set

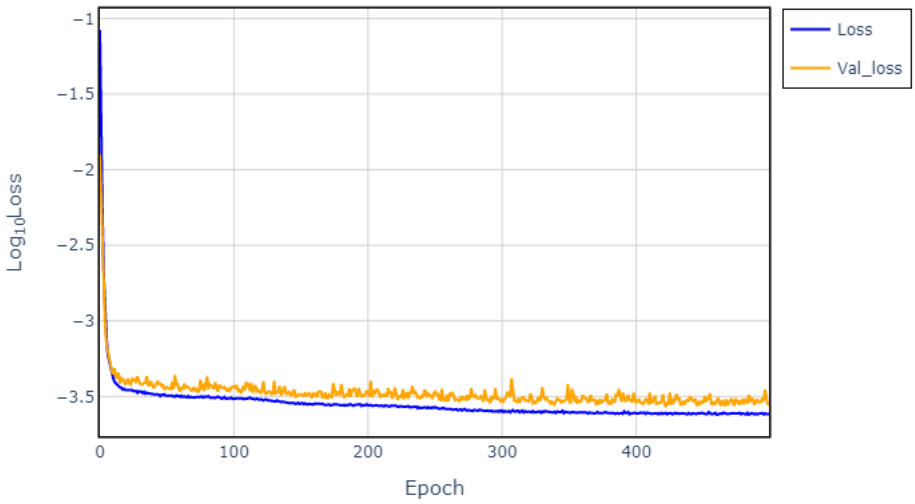


Figure 4.21: new database's loss function. Note that both training loss and validation loss have become less noisy than the architecture with dropout and a higher loss was obtained

Note that the adjusted R^2 improved considerably, this happens due to the number of samples being significantly higher than the number of variables, which implies $R^2 \approx$ adjusted R^2. Those results allow us to carry out further analysis. The next step is to evaluate graphically all the coefficients in terms of Q-Q Plots 4.22 and 4.23:

To complement the Q-Q plots, standardized error plots are used. This graph shows the standardized residuals, that is $\epsilon = \frac{ytrain - y_{test}}{\sqrt{Var(y_{train} - y_{test})}}$, in relation to each point of the test set (Figs. 4.24 and 4.25) :

As the figures 4.24 and 4.25 shown, the ANN model tends to both overestimate and underestimate the coefficients. To verify if this matter affects considerably the performance, the results will be now analyzed in terms of non-parametric 99% confidence intervals (CI) constructed over the Empirical Cumulative Density Function(Figs. 4.26 and 4.27). Those CI will be used to evaluate graphically if the ANN result could be also considered a plausible output from the physical model.

Even though the surrogate model presented both underestimation of main coefficients and overestimation of cross coefficients, the observation of confidence intervals shows us that this model still can generate compatible results with the physical model since the obtained result lied on the 99% confidence interval.

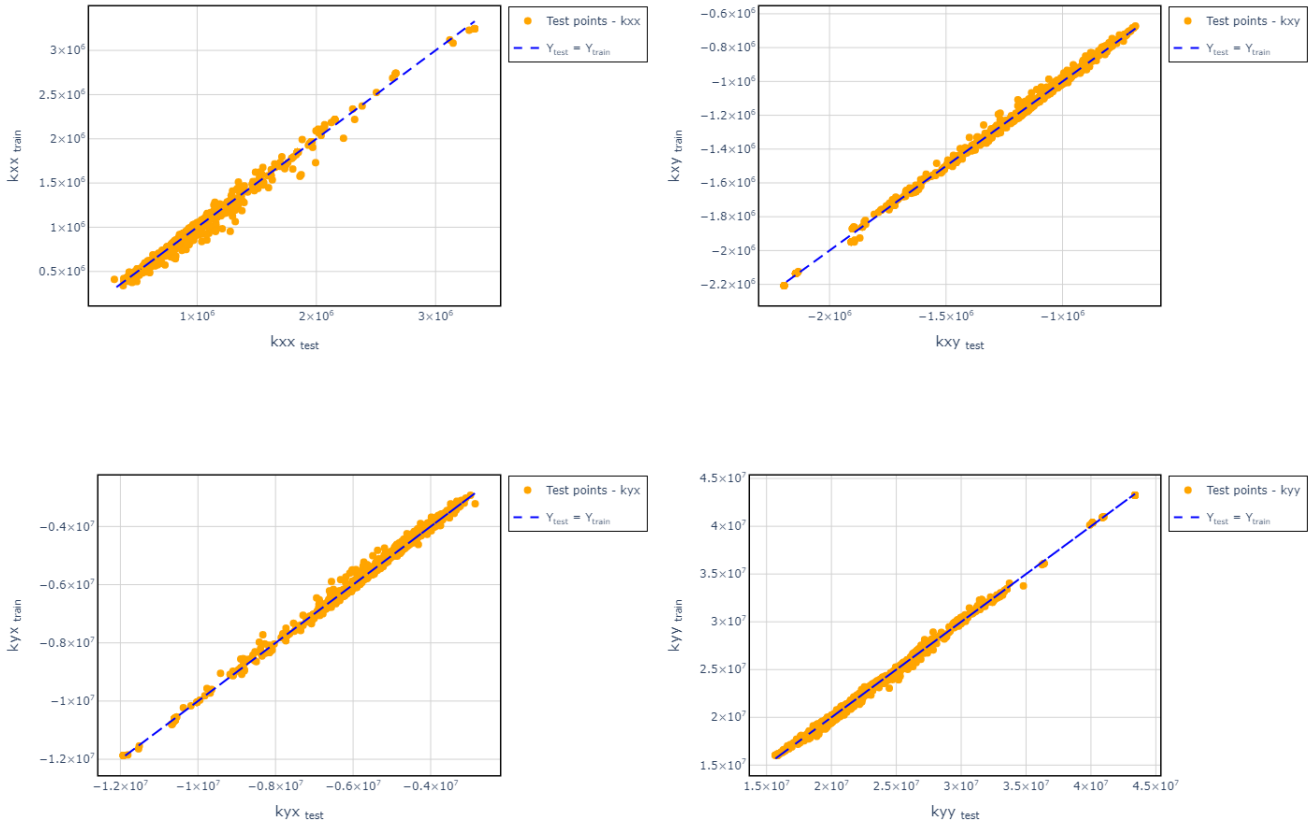


Figure 4.22: Q-Q plots of stiffness coefficients

This graphical result can be verified also in terms of the Kolmogorov-Smirnov Hypothesis Test, which null hypothesis, $\mathcal{H}_0$, is that both ANN and physical model are samples from the same distribution. To test this statement, we choose a significance level of 0.05 and run this test to each coefficient. The results are shown in table 4.4.

Coefficient	p-value	status
k_{xx}	0.910	Not Reject $\mathcal{H}_0$
k_{xy}	0.884	Not Reject $\mathcal{H}_0$
k_{yx}	0.997	Not Reject $\mathcal{H}_0$
k_{yy}	0.952	Not Reject $\mathcal{H}_0$
c_{xx}	0.855	Not Reject $\mathcal{H}_0$
c_{xy}	0.716	Not Reject $\mathcal{H}_0$
c_{yx}	0.855	Not Reject $\mathcal{H}_0$
c_{yy}	0.459	Not Reject $\mathcal{H}_0$

Table 4.4: KS Test applied to model results.

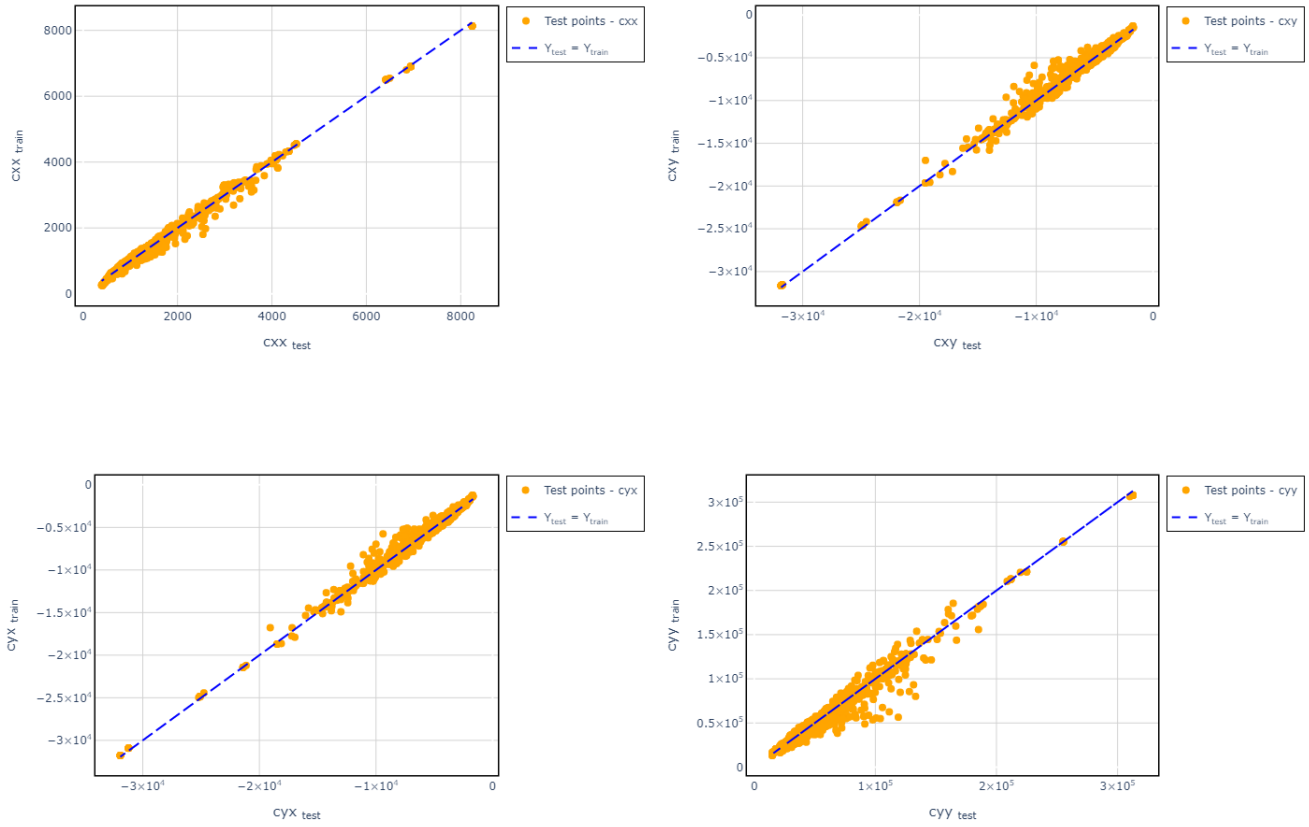


Figure 4.23: Q-Q plots of damping coefficients

Observing the hypothesis test results it is noted that we cannot reject the null hypothesis for any coefficient, which means that there is not any statistical evidence against the considered hypothesis of both samples come from the same distribution. This, however, is not a factual proof and further analysis must be carried out. To do additional analysis, we generate a new 2000 more points from the physical model and run the KS-Test again and the results are shown in table 4.5.

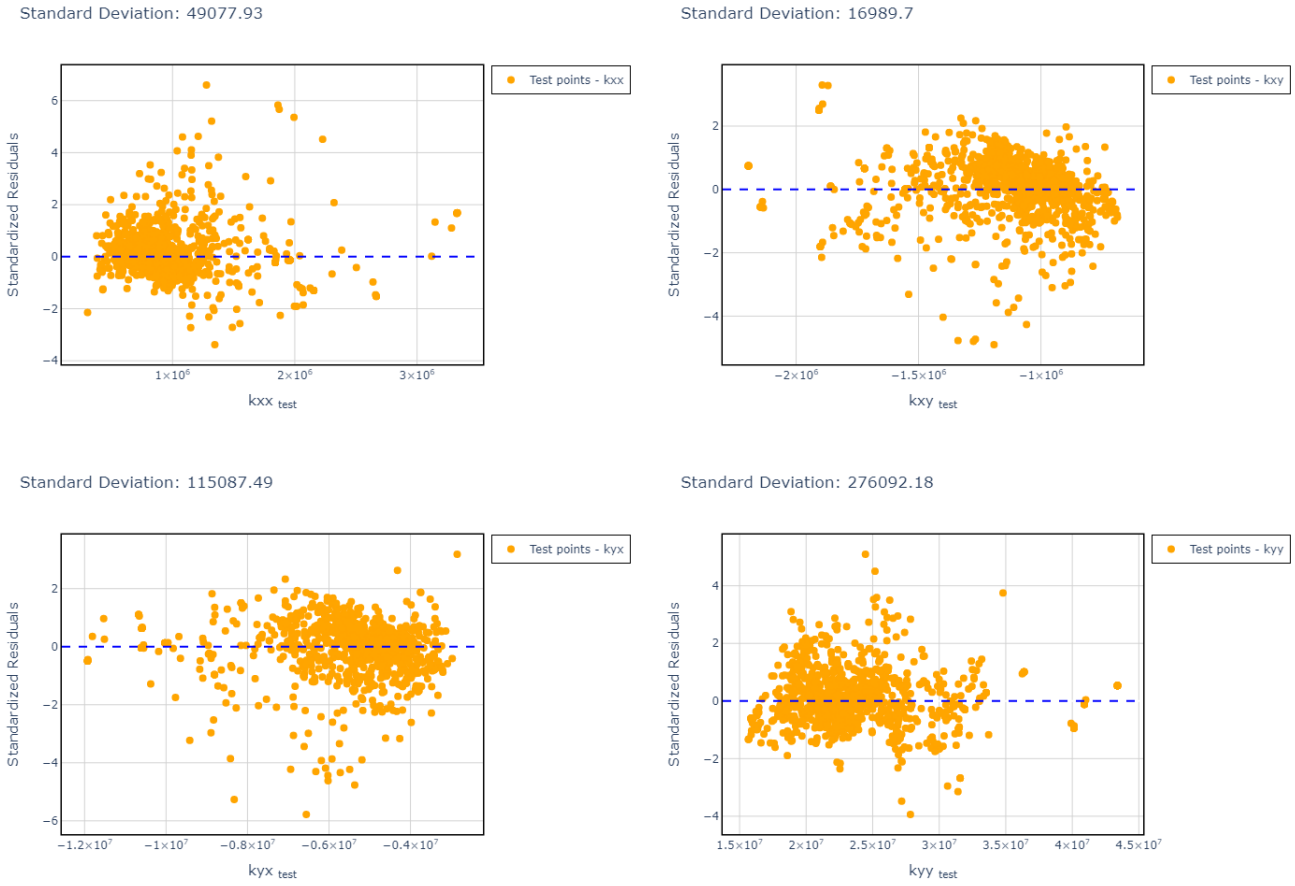


Figure 4.24: Standardized residuals of the stiffness coefficients. Realize that the k_{ii} coefficients are underestimated while the cross coefficients, k_{ij}, tend to be overestimated. In addition, heteroscedasticity is also present

Coefficient	p-value	status
k_{xx}	0.127	Not Reject $\mathcal{H}_0$
k_{xy}	0.860	Not Reject $\mathcal{H}_0$
k_{yx}	0.760	Not Reject $\mathcal{H}_0$
k_{yy}	0.136	Not Reject $\mathcal{H}_0$
c_{xx}	0.004	Reject $\mathcal{H}_0$
c_{xy}	0.001	Reject $\mathcal{H}_0$
c_{yx}	0.001	Reject $\mathcal{H}_0$
c_{yy}	< 0.001	Reject $\mathcal{H}_0$

Table 4.5: KS Test applied to model results with new data. Note that now none of damping coefficients passed in the test

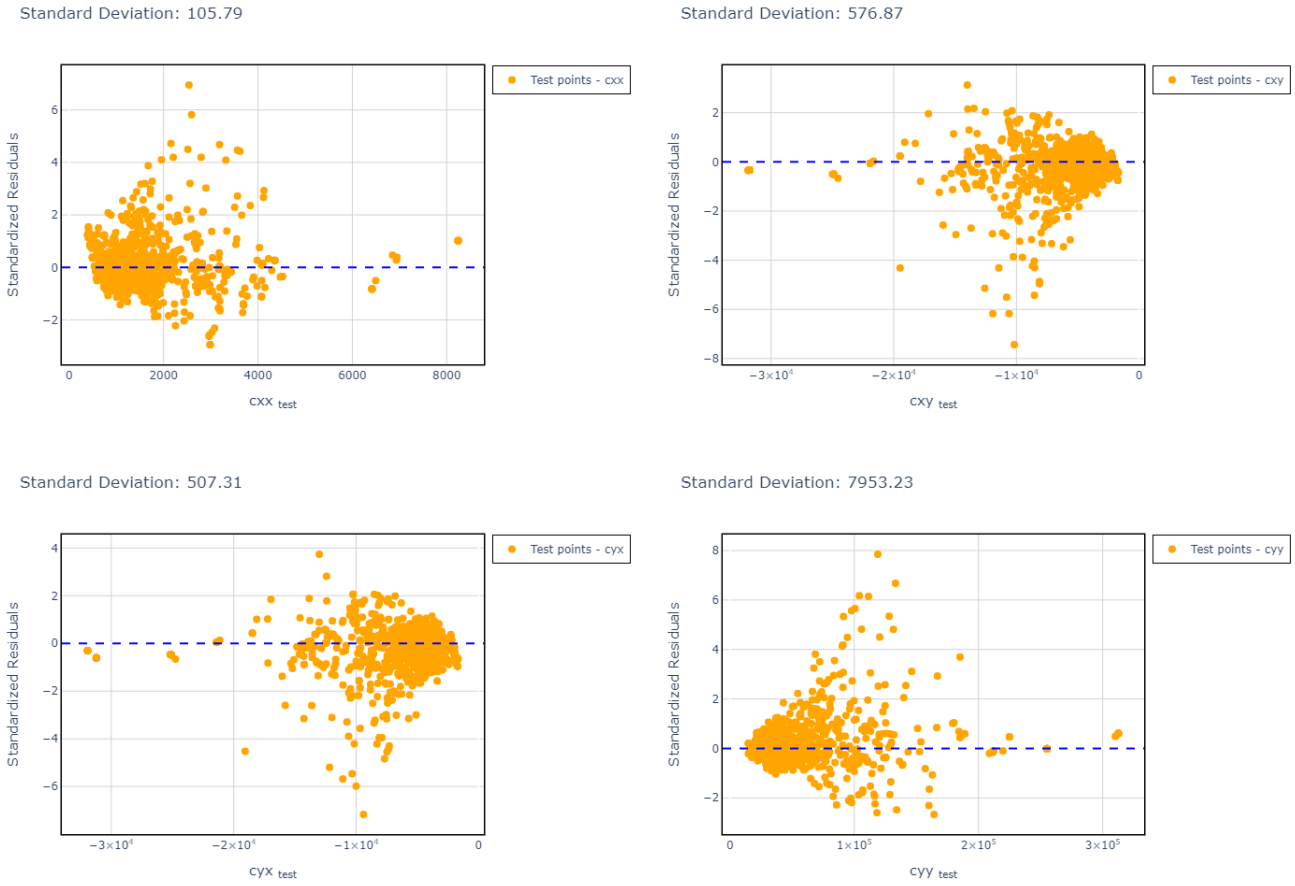


Figure 4.25: Standardized residuals of the damping coefficients. Realize that this case has the same behavior as before, underestimating c_{ii} and overestimating c_{ij}. Note that as stiffness heteroscedasticity is also present

Even though damping coefficients did not pass by the null hypothesis, this not means necessarily that the surrogate model should be disposed off since the Hypothesis tests only tell that this result is statistically insignificant, but this does not mean that it cannot be scientifically significant. To verify this last statement, both the new database created from the physical model and the ANN one are used to obtain the dynamic response of a rotor in terms of its FRF,Phase and Time response (Figs. 4.29,4.30,4.31). For this simulation, we use a rotor with a shaft diameter of 50 mm and two disks with 280 mm of diameter each. The left disk is 625 mm far from the left bearing while the right disk is 750 mm far from the right bearing. The frequency response is extracted from node 4, which is in the y direction. Phase response is extracted from the same node and in time response is obtained by applying harmonic forces to the node 4 in both directions x and y.

Note that even though the KS Test warnings about the quality of the generated damping coefficients, the surrogate model presented a good result in comparison

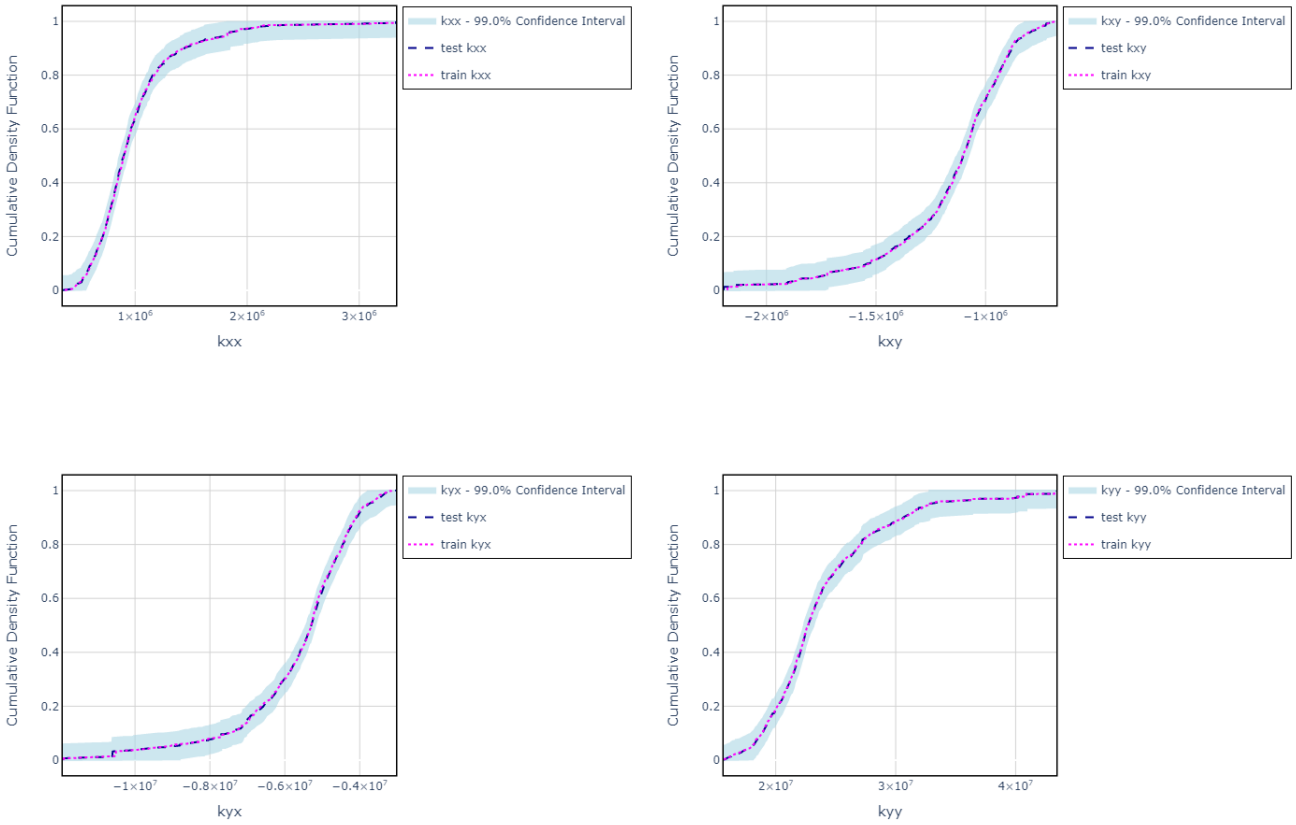


Figure 4.26: A 99% confidence interval to each stiffness coefficient. Note that all the results distributions lies inside it

with its physical counterpart. The effect of a less precise damping was only missing the amplitude by about 1% in natural frequency. This error is negligible since the presented scale in Fig. 4.29 is log-scale, which means that this error is even smaller.

The analysis of Fig. 4.30 shows that the physical bearing model and neural networks bearing presented similar results, which suggests that the training procedure was successful in producing accurate responses at the desired range. Time response (4.31) presented results as good as the last two results, being almost impossible to distinguish both responses by the naked eye.

Real-world rotors are not as simple as 4.28. An example of this complexity is a rotor with overlapping shaft elements, as represented in 4.32. In this particular case, uncertainty propagation starts to differ reasonably, both in FRF and Phase as shown in Figs 4.29 and 4.30. Note that even though the last model presented a worse performance if compared with the former, it still is quite accurate.

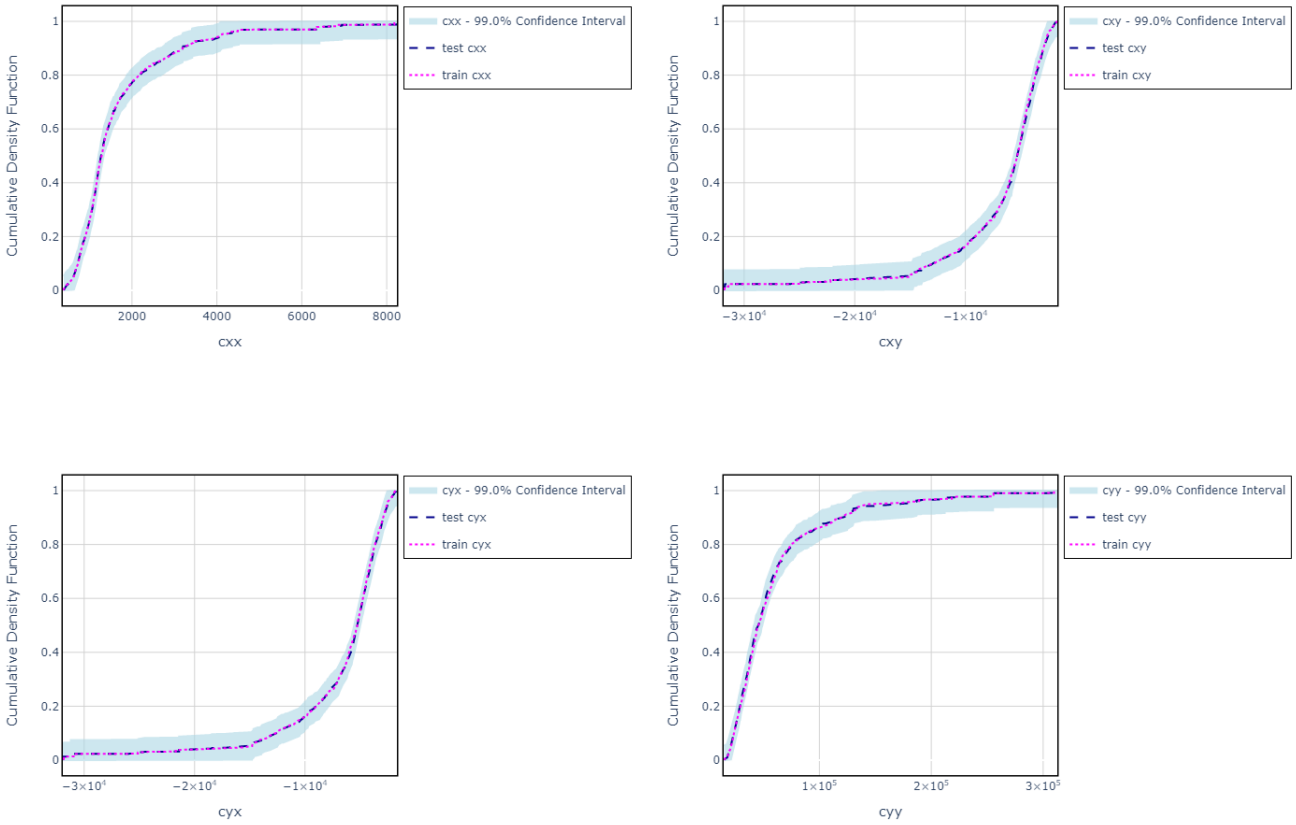


Figure 4.27: A 99% confidence interval to each damping coefficient. Note that all the results distributions lies inside it

The maintenance of the performance is due to the sampling plan choice. This step is crucial to obtain a good surrogate since the selected points must cover the parameter space well enough to get the best outputs possible. With both good inputs and outputs, supervised learning is productive, and optimization achieves reasonable results. To a broad discussion about sampling plans, please check [87].

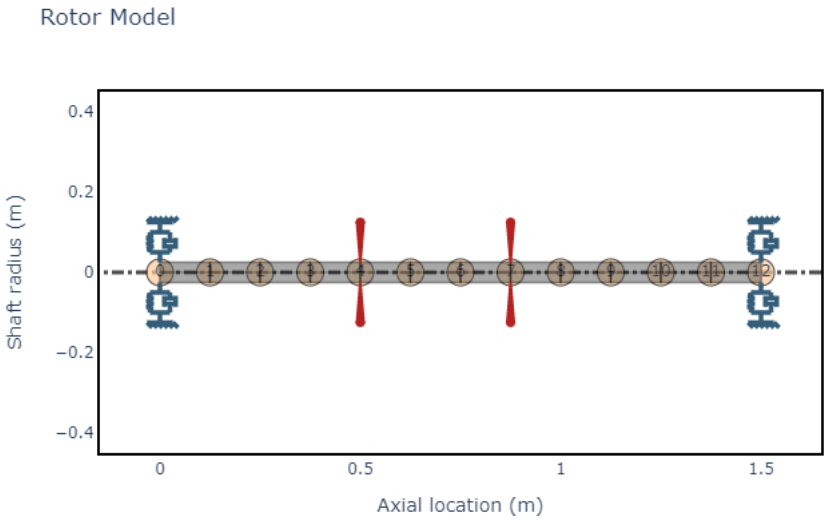


Figure 4.28: Rotor used to test the ANN model

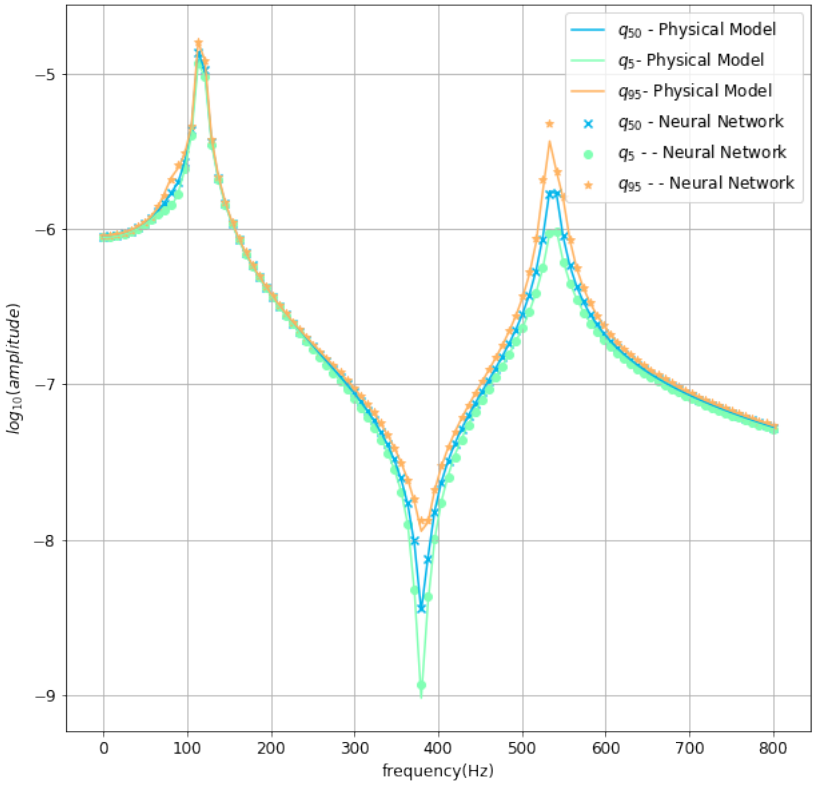


Figure 4.29: FRF obtained from physical bearing model and from ANN bearing. Note that despite of the damping coefficients were rejected on KS-Test, the error due to them is negligible

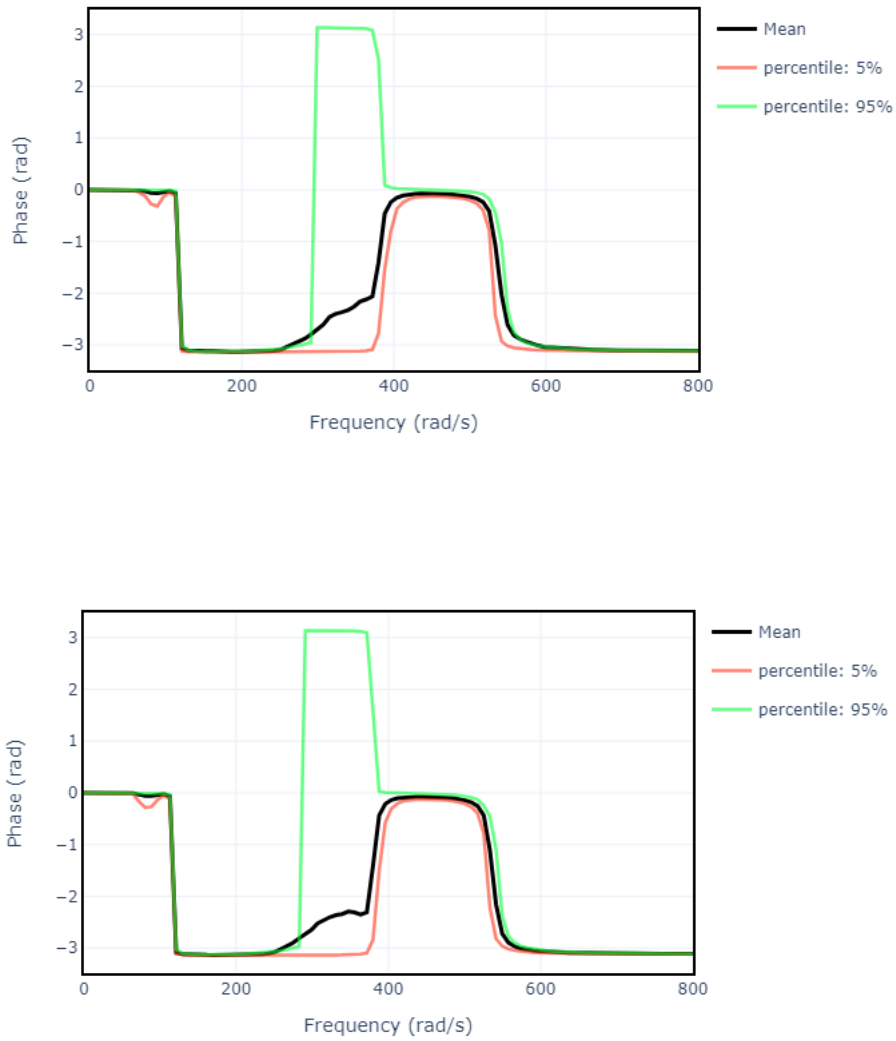


Figure 4.30: Phase obtained with the ANN bearing (up) and with the physical model bearing (down). Note that both models present the same overall behavior, with little differences

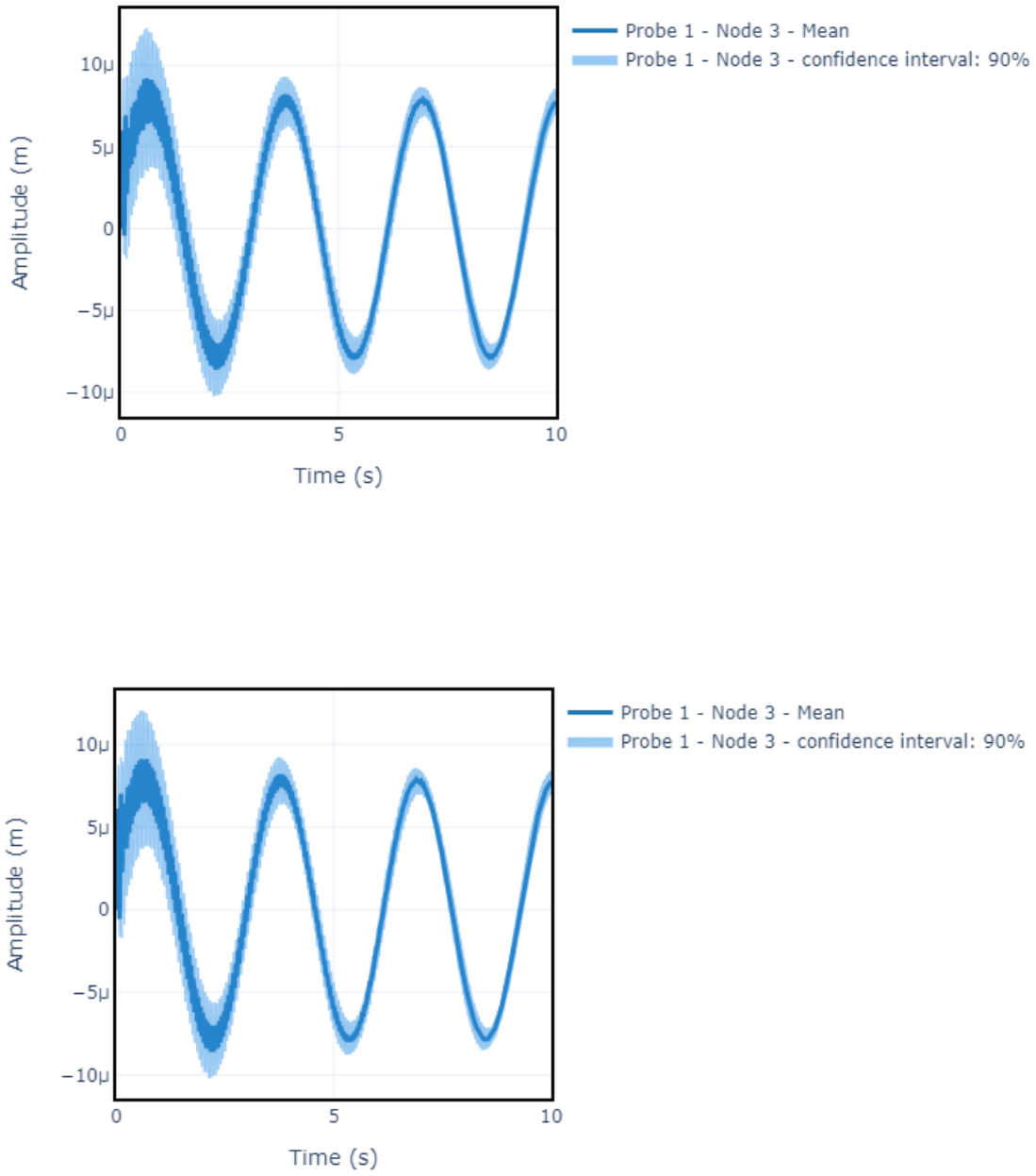


Figure 4.31: Time response obtained with the ANN bearing (up) and with the physical model bearing(down) . Note that both models present the same overall behavior, being almost indistinguishable

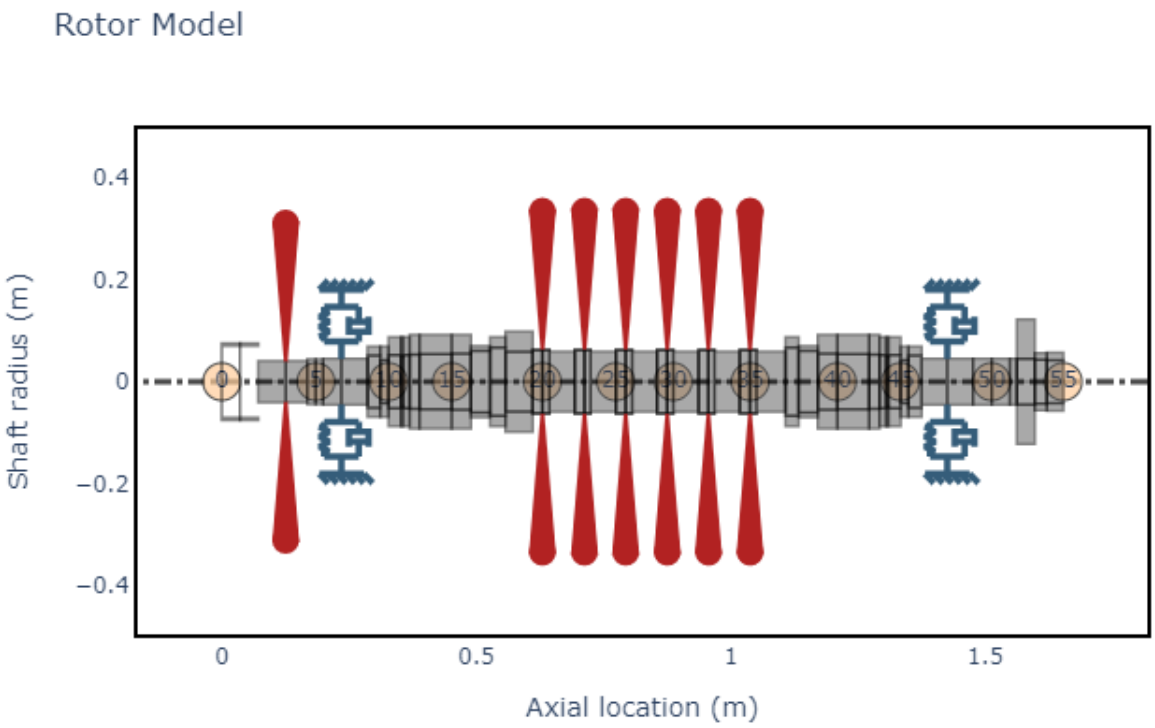


Figure 4.32: A rotor commonly found in the real world applications

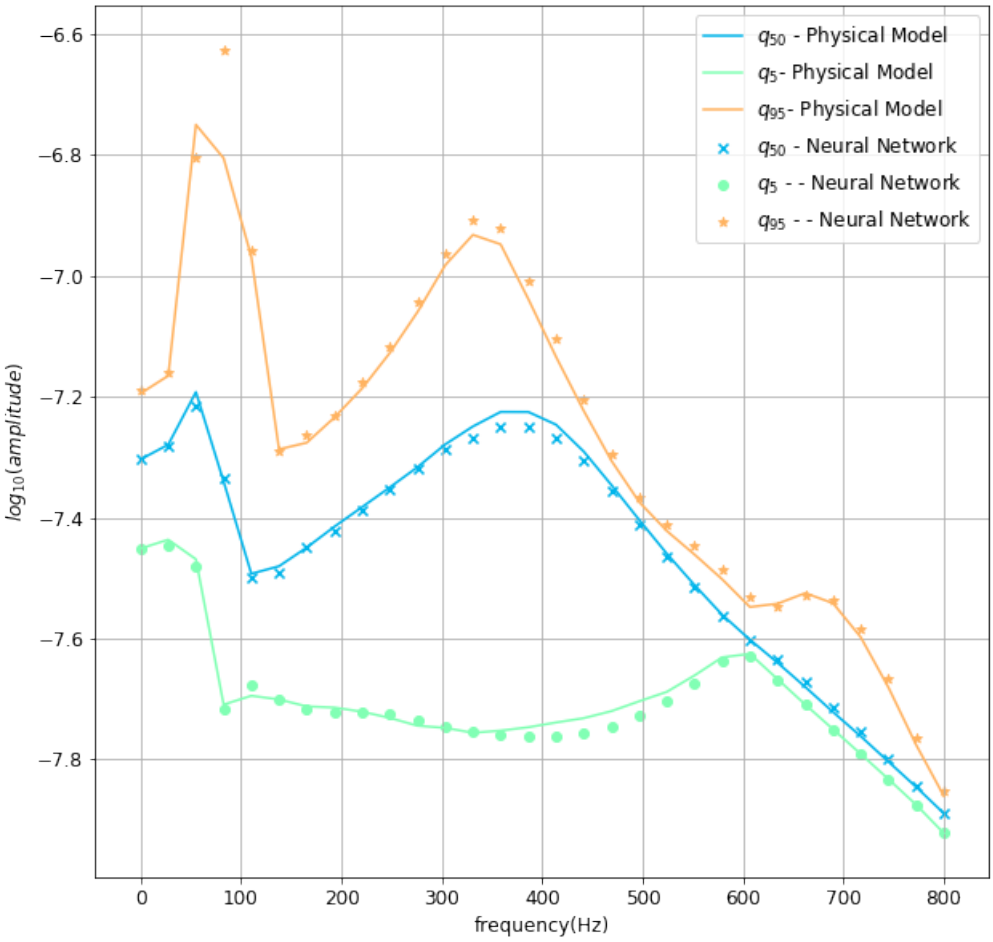


Figure 4.33: FRF of the real rotor. Here we note clearly discrepancies between both intervals. Still, this model maintains a good accuracy.

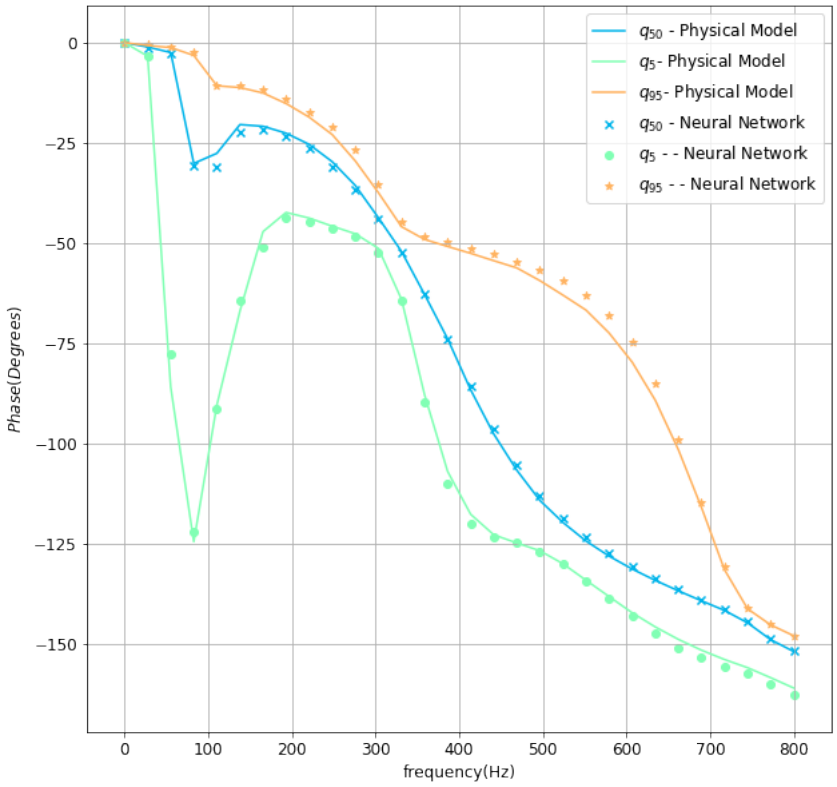


Figure 4.34: Phase obtained with the ANN bearing (dots) and with the physical model bearing (solid line). Note that both models present the same overall behavior, with more apparent differences than the former case

Chapter 5

Conclusions and Future Works

This work presented a routine to systematic implement neural networks to generate surrogate models for regression and uncertainty quantification. The features developed here allows the users to perform a robust analysis and obtain reliable results inside the design space.

As a reminder, our first step was to gather data, which is obtainable by either experiments, physical models, or artificially by using previous knowledge as leverage. Next, we have subdivided data into features (inputs) and labels (outputs). If we have an excessive number of features, its reduction is recommended to avoid parameters excess and both over-fit and lack of performance. With the correct number of labels and features, our next step was to scale data to ease the optimization process and equalize the weights. With all data pre-processed, we have initially built a shallow arrangement and verified by brute force the best one, which was insufficient for our purposes, which obliged us to choose a deep neural networks architecture. In this case, verify the best architecture is infeasible. So, the best approach is to check the weights before the training and perform techniques as pruning. Then, we have defined the loss function and the optimizers, directly tied to the learning process. Next, we evaluated the model performance and improved its results by using a data augmentation technique. Finally, the comparison between the neural network and its physical counterpart has shown that the former outperformed the latter in speed while keeping the accuracy.

Even obtaining good results,there are many questions still unanswered as the number of neurons per layer of the number of layers to accomplish a task since neural networks have advanced quicker in practice than theory. Recently, active sub-spaces solutions show promise in answering those questions.[114–117].

According to [118], the active subspace is a set of directions in a multivariate function's input space, in which the perturbation of the entries along these directions changes the function's output more, on average, then perturbing the inputs in orthogonal directions. This tool converts the entries in the most relevant dimensionless quantities, enabling the neural network to tackle multiple related problems with a single training.

Another application possible is architecture optimization [119]. After building neural networks, active sub-spaces scan the architecture and suggest a new one based on the active neurons to perform a testing task. The new architecture has the first N layers of the original neural network, an active sub-space layer that mapped the neurons into the reduced space with a linear combination and a polynomial chaos expansion layer that maps the active subspace layer into the outputs.

With active sub-spaces is also possible to evaluate the neural network robustness by creating a adversarial attack input that maximizes the output error [119]. The most robust architecture is the one that has the minimum error produced from an adversarial attack. Another strategy possible is to retrain the neural network with an augmented data set [11] generated around input generated by adversarial attack. With this tactics combined a even more robust architecture can be achieved.

Concerning Structures, is possible to impose physical restrictions on the loss function, in the so called Physics Inspired Neural Networks, shortly PINNs. This kind of neural network is capable of solving both forward and inverse problems [120], improving even more ROSS-ML performance.

References

[1] ZIENKIEWICZ, O. C., TAYLOR, R. L., ZHU. *The Finite Element Method.* 6 ed. Oxford, Elsevier Butterworth-Heinemann, 2005.

[2] VERSTEEG, H., MALALASEKERA, W. "An introduction to computational fluid dynamics : the finite volume method / H. K. Versteeg and W. Malalasekera." *SERBIULA (sistema Librum 2.0)*, 03 2021.

[3] TIMBÓ, R., MARTINS, R., BACHMANN, G., et al. "ROSS - Rotordynamic Open Source Software", *Journal of Open Source Software*, v. 5, pp. 2120, abr. 2020. doi: 10.21105/joss.02120.

[4] WEISBERG, S. *Applied Linear Regression.* Wiley, 1985.

[5] HASTIE, T., TIBISHIRANI, R., FRIEDMAN. *Elements of Statistical Learning.* Springer, 2009.

[6] HAYKIN, S. *Neural Networks and Learning Machines.* Pearson, 2009.

[7] DA SILVA, I. N., SPATTI, D. H., FLAUZINO, R. A., et al. *Artificial Neural Networks: A Practical Course.* Springer Publishing Company, Incorporated, 2016. ISBN: 3319431617.

[8] LAWRENCE, S., GILES, C., TSOI, A. C., et al. "Face recognition: a convolutional neural-network approach", *Neural Networks, IEEE Transactions on*, v. 8, n. 1, pp. 98–113, jan. 1997. doi: 10.1109/72.554195. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=554195&tag=1>.

[9] KOCIC, J., JOVICIC, N. S., DRNDAREVIC, V. "An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms." *Sensors*, v. 19, n. 9, pp. 2064, 2019. Disponível em: <http://dblp.uni-trier.de/db/journals/sensors/sensors19.html#KocicJD19>.

[10] KHAMARU, K., WAINWRIGHT, M. J. "Convergence guarantees for a class of non-convex and non-smooth optimization problems." In: *ICML*, v. 80, *Proceedings of Machine Learning Research*, pp. 2606–2615. PMLR, 2018.

[11] PRADO, L., RITTO, T. "Data driven Dirichlet sampling on manifolds", *arXiv*, dez. 2020.

[12] NADLER, B., LAFON, S., COIFMAN, R., et al. "Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker-Planck operators", *Adv Neural Inf Process Syst*, v. 18, jul. 2005.

[13] PEARSON, K. "On Lines and Planes of Closest Fit to Points in Space", *Philosophical Magazine*, v. 2, pp. 559–572, nov. 1900. doi: 10.1080/14786440109462720.

[14] RANKINE, W. "On the centrifugal force of rotating shafts", *Engineering*, v. 27, pp. 249–249, 1869.

[15] STODOLA, A. *Dampf und Gas-Turbinen.* Verlag von Julius Springer, 1924.

[16] DUNKERLEY, S. "On the whirling and vibration of shaft", *Philos. Trans. R. Soc*, v. 185, pp. 279–359, 1894.

[17] CAMPBELL, W. "The Protection of steam-turbine disk wheels from axial vibration", *Trans. ASME*, v. 46, pp. 31–160, 1924.

[18] HOLZER, H. *Die Berechnung der Drehschwingungen.* CRC Press, 2012.

[19] JEFFCOTT, H. "The lateral vibration of loaded shafts in the neighborhood of a whirling speed", *Philos. Mag. A*, v. 37, pp. 304–315, 1919.

[20] BISHOP, R. E. D. "Vibration of rotating shafts", *J. Mech. Eng*, v. 1, pp. 50–65, 1959.

[21] BISHOP, R., GLADWELL, G. "The vibration and balancing of an unbalanced flexible rotor", *J. Mech. Eng.*, v. 1, pp. 66–77, 1959.

[22] BISHOP, R. E. D., PARKINSON. "Second order vibration of flexible shafts", *Philos. Trans. R. Soc. Lond. , Ser. A*, v. 1095, n. 259, pp. 1–31, 1965.

[23] ESHLEMAN, R., EUBANKS. "On the critical speeds of a continuous rotor", *Trans. ASME, J. Eng. Ind.*, v. 91, n. 4, pp. 1180–1188, 1969.

[24] NEWKIRK, B. L. "Shaft whipping", *Gen. Electr. Rev.*, v. 27, n. 3, pp. 169–178, 1924.

[25] NEWKIRK, B. L., TAYLOR, H. D. “Shaft whirling due to oil action in journal bearings”, *Gen. Electr. Rev.*, v. 28, n. 7, pp. 559–568, 1925.

[26] YAMAMOTO, T. “On the critical speed of a shaft of sub-harmonic oscillation”, *Trans. JSME*, v. 21, n. 111, pp. 853–858, 1955.

[27] EHRICH, F. F. “Subharmonic Vibration of Rotors in Bearing Clearance”, *ASME*, v. 66, 1966.

[28] THOMAS, J. “Instabile Eigenschwingungeri von Turbinenlaufern, Angefacht durch die Spaltstromungen, in Stoptbuchsen und Beschauflungen”, *"AEG-Sonderdruck"*, pp. 1039—-1063, 1958.

[29] ALFORD, J. “Protecting turbomachinery from self-excited rotor whirl”, *Trans. ASME, J. Eng. Power*, v. 87, n. 4, pp. 333–344, 1965.

[30] COOK, R. D., MALKUS, D. S., PLESHA, M. E., et al. *Concepts and Applications of Finite Element Analysis*. Fourth edition ed. New York, John Wiley, 2001.

[31] WASSERMAN, L. *All of Statistics*. Springer Texts in Statistics, 2004.

[32] DEGROOT, M., SCHERVISH. *Probability and Statistics*. Addison-Wesley, 2002.

[33] SAN ANDRÉS, L. “Modern Lubrication Theory, Gas Film Lubrication,”. `https://rotorlab.tamu.edu/me626/default.htm`, 2010. Accessed: 2021-03-29.

[34] ISHIDA, Y., YAMAMOTO. *Linear and Nonlinear Rotordynamics: A Modern Treatment with Applications*. Wiley-VCH, 2012.

[35] MUSZYNSKA, A. *Rotordynamics*. Taylor & Francis, 05 2005. ISBN: 9780429133565. doi: 10.1201/9781420027792.

[36] FORRESTER, A. I. J., KEANE, A., SOBESTER, A. *Engineering Design Via Surrogate Modelling: A Practical Guide*. Willey, 2008.

[37] BISHOP, C. M., SVENSEN, M. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[38] ECHEVERRIA, D., LAHAYE, D., HEMKER, P. “Space Mapping and Defect Correction”. v. 5, pp. 157–176, jan. 2008. ISBN: 978-3-540-78840-9. doi: 10.1007/978-3-540-78841-6_8.

[39] BEN-GAL, I. "Bayesian Networks", *Encyclopedia of Statistics in Quality and Reliability*, mar. 2008.

[40] BHANDE, A. "What is underfitting and overfitting in machine learning and how to deal with it", *Medium*, 2018.

[41] WHITE, H. "A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity", *Econometrica*, v. 48, pp. 237–268, jan. 1980.

[42] BREUSCH, T., TSANG, R., PAGAN, A. "A Simple Test for Heteroscedasticity and Random Coefficient Variation", *Econometrica*, v. 47, pp. 1287–94, fev. 1979.

[43] PITTS, W. "Logical calculus of the ideas immanent in nervous activity", *J Bull Math Biophys*, v. 5, pp. 115–133, jan. 1993.

[44] ATTNEAVE, F., HEBB, D. "The Organization of Behavior; A Neuropsychological Theory", *The American Journal of Psychology*, v. 63, pp. 633, out. 1950. doi: 10.2307/1418888.

[45] ROSENBLATT, F. "The perceptron: A probabilistic model for information storage and organization in the brain [J]", *Psychol. Review*, v. 65, pp. 386 – 408, dez. 1958. doi: 10.1037/h0042519.

[46] WIDROW, B., HOFF, M. "Adaptive Switching Circuits", *Neurocomputing.*, pp. 126–134, jan. 1960.

[47] MINSKY, M., PAPERT, S. *Perceptrons: An Introduction to Computational Geometry*. jan. 2017. ISBN: 9780262343930. doi: 10.7551/mitpress/11301.001.0001.

[48] WERBOS, P., JOHN, P. "Beyond regression : new tools for prediction and analysis in the behavioral sciences", 1974.

[49] KOHONEN, T. "Self-organised formation of topologically correct feature map", *Biological Cybernetics*, v. 43, pp. 59 – 69, jan. 1982.

[50] CARPENTER, G. "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine", *Computer Vision, Graphics, and Image Processing*, v. 37, pp. 54–115, nov. 1986. doi: 10.1016/S0734-189X(87)80014-2.

[51] "Autopilot AI". `https://www.tesla.com/autopilotAI`, 2020. Accessed: 2021-03-27.

[52] TAN, C., SUN, F., KONG, T., et al. "A Survey on Deep Transfer Learning." In: *ICANN (3)*, v. 11141, *Lecture Notes in Computer Science*, pp. 270–279. Springer, 2018.

[53] MA, Y., FU, Y. *Manifold Learning Theory and Applications*. CRC Press, 2012.

[54] LIMA, E. L. *Elementos de Topologia Geral*. 1 ed. São Paulo, Editora USP, 1970.

[55] KELLEY, J. L. *General Topology*. 2 ed. New Jersey, Princeton,NJ, 1975.

[56] WILLARD, S. *General Topology*. Reading,MA, Addison-Wesley, 1988.

[57] BOURBAKI, N. *General Topology (2 volumes)*. New York, Springer-Verlag, 1989.

[58] MENDELSON, B. *Introduction to Topology*. 3 ed. New York, Dover Publicatiosn, 1990.

[59] STEEN, L. A. *Counterexamples in Topology*. New York, Dover Publications, 1999.

[60] JAMES, I. M. *History of Topology*. Netherlands, Elsevier B. V., 1999.

[61] LEE, J. *Introduction to Smooth Manifolds*. Springer, jan. 2003. ISBN: 0387954953. doi: 10.1007/978-0-387-21752-9.

[62] SPIVAK, M. *Calculus on Manifolds*. New York, Benjamin, 1965.

[63] KREYZIG, E. *E. Differential Geometry*. Dover Publications, 1991.

[64] KUHNEL, W. *Differential Geometry: Curves–Surfaces–Manifolds*. 2 ed. Providence,RE, American Mathematical Society, 2000.

[65] PRESSLEY, A. *Elementary Differential Geometry*. 2 ed. New York, Springer-Verlag, 2010.

[66] CASELLA, G., BERGER, R. L. *Statistical Inference*. Duxbury Press, 2002.

[67] TENENBAUM, J., SILVA, V., LANGFORD, J. "A Global Geometric Framework for Nonlinear Dimensionality Reduction", *Science (New York, N.Y.)*, v. 290, pp. 2319–23, jan. 2001. doi: 10.1126/science.290.5500.2319.

[68] ROWEIS, S., SAUL, L. "Nonlinear Dimensionality Reduction by Locally Linear Embedding", *Science (New York, N.Y.)*, v. 290, pp. 2323–6, jan. 2001. doi: 10.1126/science.290.5500.2323.

[69] BELKIN, M., NIYOGI, P. "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering", *Advances in Neural Information Processing System*, v. 14, abr. 2002.

[70] DONOHO, D., GRIMES, C. "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. Proc. National Academy of Science (PNAS), 100, 5591-5596", *Proceedings of the National Academy of Sciences of the United States of America*, v. 100, pp. 5591–6, jun. 2003. doi: 10.1073/pnas.1031596100.

[71] SCHÖLKOPF, B., SMOLA, A., MÜLLER, K.-R. "Nonlinear Component Analysis as a Kernel Eigenvalue Problem", *Neural Computation*, v. 10, pp. 1299–1319, jul. 1998. doi: 10.1162/089976698300017467.

[72] SOIZE, C., GHANEM, R. G. "Probabilistic learning on manifolds constrained by nonlinear partial differential equations for small datasets." *CoRR*, v. abs/2010.14324, 2020. Disponível em: <`http://dblp.uni-trier.de/db/journals/corr/corr2010.html#abs-2010-14324`>.

[73] ZHANG, R., GHANEM, R. "Normal-Bundle Bootstrap", *SIAM Journal on Mathematics of Data Science*, v. 3, pp. 573–592, 01 2021. doi: 10.1137/20M1356002.

[74] MURASE, H., NAYAR, S. "Learning and recognition of 3D objects from appearance", *IEEE Workshop on Qualitative Vision*, pp. 39–50, 01 1993.

[75] URTASUN, R., FLEET, D., HERTZMANN, A., et al. "Priors for people tracking from small training sets". v. 1, pp. 403– 410 Vol. 1, nov. 2005. ISBN: 0-7695-2334-X. doi: 10.1109/ICCV.2005.193.

[76] JEONG, H., PARK, S., WOO, S., et al. "Rotating Machinery Diagnostics Using Deep Learning on Orbit Plot Images", *Procedia Manufacturing*, v. 5, pp. 1107–1118, dez. 2016. doi: 10.1016/j.promfg.2016.08.083.

[77] BARSZCZ, T., ZABARYLLO, M. "Concept of automated fault detection of large turbomachinery using Machine Learning on transient data", *Diagnostyka*, v. 20, pp. 63–71, dez. 2018. doi: 10.29354/diag/100399.

[78] KORNAEV, A., SAVIN, L., KORNAEV, N., et al. "Machine learning for rotating machines: simulation, diagnosis and control". v. 32, jun. 2020. doi: 10.21595/vp.2020.21549.

[79] KANKAR, P., SHARMA, CHANDRA, S., et al. "Fault diagnosis of ball bearings using machine learning methods", *Expert Syst. Appl.*, v. 38, pp. 1876–1886, 03 2011. doi: 10.1016/j.eswa.2010.07.119.

[80] SOBIE, C., FREITAS, C., NICOLAI, M. "Simulation-driven machine learning: Bearing fault classification", *Mechanical Systems and Signal Processing*, v. 99, pp. 403–419, 01 2018. doi: 10.1016/j.ymssp.2017.06.025.

[81] GOODFELLOW, I. J., BENGIO, Y., COURVILLE, A. *Deep Learning*. Cambridge, MA, USA, MIT Press, 2016. `http://www.deeplearningbook.org`.

[82] HAGAN, M. T. *Neural Network Design*. E-book, 1988.

[83] GUERON, A. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly, 2019.

[84] BREIMAN, L., FRIEDMAN, J., OLSHEN, R., et al. *Classification and Regression Trees*. 2 ed. Belmont, CA, Wadsworth, 1984.

[85] VINH, N., EPPS, J., BAILEY, J. "Information theoretic measures for clusterings comparison", *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pp. 2837–2854, dez. 2020.

[86] LOMAX, R. G., H.-V. *Statistical Concepts*. Routledge, 2012.

[87] KOCHENDERFER, M., WHEELER, T. A. *Algorithms for Optimization*. MIT Press, 2019.

[88] KRIZHEVSKY, A., SUTSKEVER, I., HINTON, G. "ImageNet Classification with Deep Convolutional Neural Networks", *Neural Information Processing Systems*, v. 25, jan. 2012. doi: 10.1145/3065386.

[89] MISHKIN, D., MATAS, J. "CNN-Advances-2016-CMP", mar. 2017.

[90] POGGIO, T., MHASKAR, H., ROSASCO, L., et al. "Why and When Can Deep – but Not Shallow – Networks Avoid the Curse of Dimensionality", nov. 2016.

[91] HINTON, G., OSINDERO, S., TEH, Y.-W. "A Fast Learning Algorithm for Deep Belief Nets", *Neural computation*, v. 18, pp. 1527–54, ago. 2006. doi: 10.1162/neco.2006.18.7.1527.

[92] "Keras website". `https://keras.io/api/losses/`. Accessed: 2021-03-27.

[93] BROWNLEE, J. "How to use Learning Curves to Diagnose Machine Learning Model Performance". https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/, 2020. Accessed: 2021-06-04.

[94] FASTAI. "Determining when you are overfitting, underfitting, or just right?" https://forums.fast.ai/t/determining-when-you-are-overfitting-underfitting-or-just-right/7732, 2020. Accessed: 2021-03-28.

[95] LEE, C., OVERFITTING, G., CARUANA, R., et al. "Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping", *Advances in Neural Information Processing Systems 13, NIPS 2000*, mar. 2001.

[96] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, v. 15, pp. 1929–1958, jun. 2014.

[97] ZHENG, P., ASKHAM, T., BRUNTON, S. L., et al. "A Unified Framework for Sparse Relaxed Regularized Regression: SR3", *IEEE Access*, v. 7, pp. 1404–1423, 2019. doi: 10.1109/ACCESS.2018.2886528.

[98] "Residual Plot Analysis". https://www.originlab.com/doc/Origin-Help/Residual-Plot-Analysis. Accessed: 2021-03-27.

[99] SOIZE, C. "Construction of probability distributions in high dimension using the maximum entropy principle: Applications to stochastic processes, random fields and random matrices", *International Journal for Numerical Methods in Engineering*, v. 76, pp. 1583 – 1611, dez. 2008. doi: 10.1002/nme.2385.

[100] PILLOW, T. "Introduction to Copulas". https://medium.com/@timothypillow/introduction-to-copulas-ad1a3b83a297, 2020. Accessed: 2021-03-28.

[101] CHANG, B. "Copula: A Very Short Introduction". https://bochang.me/blog/posts/copula/, maio 2019.

[102] NELSEN, R. *An Introduction to Copulas*. Springer, jan. 2006. ISBN: 978-0-387-28659-4. doi: 10.1007/0-387-28678-0.

[103] SILVERMAN, B. *Density Estimation for Statistics and Data Analysis*. fev. 2018. ISBN: 9781315140919. doi: 10.1201/9781315140919.

[104] SHEATHER, S., JONES, M. "A Reliable Data-Based Bandwidth Selection Method for Kernel Density Estimation", *Journal of the Royal Statistical Society: Series B (Methodological)*, v. 53, pp. 683–690, jul. 1991. doi: 10.1111/j.2517-6161.1991.tb01857.x.

[105] DEDDUWAKUMARA, D., PRENDERGAST, L., STAUDTE, R. "An efficient estimator of the parameters of the generalized lambda distribution", *Journal of Statistical Computation and Simulation*, v. 91, pp. 1–19, 08 2020.

[106] ACHEN, C. "What Does "Explained Variance" Explain?: Reply", *Political Analysis*, v. 2, jan. 1990. doi: 10.1093/pan/2.1.173.

[107] NIST. "Quantile-Quantile Plot". `https://www.itl.nist.gov/div898/handbook/eda/section3/qqplot.htm#`, 2012. Accessed: 2021-03-27.

[108] DOWSON, D. "A generalized Reynolds equation for fluid-film lubrication", *International Journal of Mechanical Sciences*, v. 4, n. 2, pp. 159–170, 1962.

[109] PINKUS, O. "The Reynolds Centennial: A Brief History of the Theory of Hydrodynamic Lubrication", *Journal of Tribology*, v. 109, n. 1, pp. 2–15, 1987.

[110] MOTA, J. A., VALÉRIO, J. V., RANGEL, F. M. "Simplifications of Navier-Stokes in Journal Bearing Simulation". In: *Anais do XIII Encontro Acadêmico de Modelagem Computacional*, pp. 76–85, fev. 2020.

[111] MOTA, J. A., VALÉRIO, J. V., RITTO, T. G., et al. "Modeling of hydrodynamic bearings with a novel boundary parameterization approach", 04 2021.

[112] MIRANDA, W., MACHADO, L., FARIA, M. "Some Insights into the Dynamic Response of Flexible Rotors Supported on Fluid Film Bearings". jan. 2005. doi: 10.13140/2.1.4142.9449.

[113] KARNIN, E. "Simple procedure for pruning back-propagation trained neural networks", *Neural Networks, IEEE Transactions on*, v. 1, pp. 239 – 242, jul. 1990. doi: 10.1109/72.80236.

[114] CONSTANTINE, P., ROSARIO, Z., IACCARINO, G. "Data-driven dimensional analysis: algorithms for unique and relevant dimensionless groups", ago. 2017.

[115] BROUWER, A., EISENBERG, M. "The underlying connections between identifiability, active subspaces, and parameter space dimension reduction", *arxiv*, fev. 2018.

[116] TRIPATHY, R., BILIONIS, I. "Deep Active Subspaces: A Scalable Method for High-Dimensional Uncertainty Propagation". ago. 2019. doi: 10.1115/DETC2019-98099.

[117] JIANG, X., HU, X., LIU, G., et al. "A generalized active subspace for dimension reduction in mixed aleatory-epistemic uncertainty quantification", *Computer Methods in Applied Mechanics and Engineering*, v. 370, pp. 113240, out. 2020. doi: 10.1016/j.cma.2020.113240.

[118] CONSTANTINE, P. G. *Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies*. USA, Society for Industrial and Applied Mathematics, 2015. ISBN: 1611973856.

[119] CUI, C., ZHANG, K., DAULBAEV, T., et al. "Active Subspace of Neural Networks: Structural Analysis and Universal Attacks", *SIAM Journal on Mathematics of Data Science*, v. 2, pp. 1096–1122, jan. 2020. doi: 10.1137/19M1296070.

[120] RAISSI, M., PERDIKARIS, P., KARNIADAKIS, G. "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations", *Journal of Computational Physics*, v. 378, 11 2018. doi: 10.1016/j.jcp.2018.10.045.