Fillipe Rocha Leonel Esteves

Design of a Floating Offshore Structure by a Deep Neural Network

São Paulo, Brazil

2022

Fillipe Rocha Leonel Esteves

Design of a Floating Offshore Structure by a Deep Neural Network

Revised version

Master's Dissertation submitted to the Escola Politécnica of the University of São Paulo in partial fulfillment of the requirements for the degree of Master of Science.

Field of Study:
Naval Architecture and Ocean Engineering (3010)

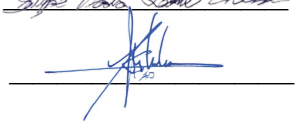
Supervisor:
Prof. Ph.D. Kazuo Nishimoto

São Paulo, Brazil
2022

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 08 de agosto de 2022

Assinatura do autor: 

Assinatura do orientador: 

Catalogação-na-publicação

Esteves, Fillipe Rocha Leonel
Design of a floating offshore structure by a deep neural network / F. R. L. Esteves -- versão corr. -- São Paulo, 2022.
80 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Naval e Oceânica.

1.REDES NEURAIS 2.ESTRUTURAS OFFSHORE FLUTUANTES 3.APRENDIZADO COMPUTACIONAL 4.INTELIGÊNCIA ARTIFICIAL 5.SUPERFÍCIES DE RESPOSTA I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Naval e Oceânica II.t.

To my wife Sheila and my
daughter Maria Luísa, with love.

Acknowledgements

"Not unto us, O Lord, not unto us, but to Thy name give the glory, for thy mercy, and for thy truth's sake." Psalm 115:1

I thank my family, especially my parents: Valmir and Silvana, and my sister Gabriella for supporting and investing efforts during my academic-professional training. Also to my grandparents: Waldevino, Luiza, Elmo, and Maria, who always believed in education to guarantee a better future for the family.

To my wife Sheila, for her companionship and support in academic activities. Also, for the endless scholarly planning and engineering discussions. Also to her family: José Otávio, Eliana and Bruna for their support.

I would acknowledge Fr. Marcos Funchal for providing me with doctrinal and ascetical orientation. He helped me find something great and new in the simplicity of my habitual work.

My sincere gratitude to my supervisors: Ph.D. Bernardo Andrade and Ph.D. Kazuo Nishimoto, for their trust in me in cutting-edge research and the guidance during the development of this project. Also, to Professor Ph.D. Henrique Gaspar for his comments on the conclusion of this work. To my friend and colleague at the Optimization and Design Laboratory (Loopin), Pedro Bulla, for the fruitful discussions since our undergraduate research.

To the Institute for Technological Research (IPT) and Fundação de Apoio ao Instituto de Pesquisas Tecnológicas (FIPT), for providing the necessary computing infrastructure. To the IPT researchers who provided me an inspiring and innovative research environment: Ph.D. João Dantas, Ph.D. André Kogishi, Ph.D. Carlos Padovezi, and M.Sc. Maria Gandara. To my colleagues at IPT, especially the multiusuário's staff: Gustavo Gomes, M.Sc. Rodolfo Chreim, M.Sc. Eduardo Katsuno, M.Sc. Felipe Castro, Ana Gilda, Cláudio Trancoso, José Marcos, Bruno Cheng, Patrick Queiroz, and Igor Coutinho.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Ad maiorem Dei gloriam

Abstract

Esteves, F. R. L. **Design of a Floating Offshore Structure by a Deep Neural Network**. 2022. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2022

The Deep Neural Network (DNN) is a machine learning algorithm that principle is to concatenate nonlinear operations involving matrices. These artificial networks can achieve reasonable transformations of input to output data by updating a matrix of randomly initialized weights. It is necessary to provide a training dataset to minimize a loss function during the network training. Validation and test procedures guarantee the quality of the trained network. The offshore design requires complex modeling that reflects the nature of the ocean environment. To produce a mapping of the hydrodynamic response of the offshore system, an extensive volume of simulations is often necessary, which elevates the computational cost of the design process. At this point, the opportunity to converge the deep learning potentialities and the challenges of offshore design emerges. This work proposes a framework to assess deep neural networks used as response surfaces of the semi-submersible platform dynamic models in waves: a mass-spring-damper model and an analytical hydrodynamic model validated with reference data. The low computation cost of these models allowed the generation of large datasets. The N-dimensional response hypersurface in each case is a combination of input parameters. An appropriate study elucidated the correct parameters definition of the DNN: the number of layers and the number of neurons per layer, targeting the configuration that provides the minimum mean squared error. The response surface represented by the DNN can easily be coupled to an optimization algorithm that evaluates hundreds of viable solutions and finds the optimal design. Using neural networks as a response surface has excellent cost-benefit in preliminary design dynamic modeling, in cases where the available time before the optimization tasks is long enough to prepare a training dataset, and in cases subjected to requisites updates throughout the conceptual design phase.

Keywords: Deep Neural Networks. Offshore Design. Response Surface.

Resumo

Esteves, F. R. L. **Design of a Floating Offshore Structure by a Deep Neural Network**. 2022. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2022

As Redes Neurais Profundas são um algoritmo de aprendizado de máquina que tem como princípio concatenar operações não lineares envolvendo matrizes. Essas redes artificiais podem realizar transformações entre dados de entrada e de saída atualizando uma matriz de pesos, inicializados aleatoriamente. É necessário fornecer um conjunto de dados de treinamento para minimizar uma função de perda, durante o treinamento da rede. Os procedimentos de validação e teste garantem a qualidade da rede treinada. O projeto offshore requer uma modelagem complexa que reflete a natureza do ambiente oceânico. Para produzir um mapeamento da resposta hidrodinâmica do sistema offshore, muitas vezes é necessário um grande volume de simulações, o que eleva o custo computacional do projeto. Neste ponto, surge a oportunidade de convergir as potencialidades do *deep learning* e os desafios do projeto *offshore*. Assim sendo, este trabalho propõe um método para avaliar redes neurais profundas, usadas como superfícies de resposta dos modelos dinâmicos de plataforma semissubmersível em ondas: um modelo massa-mola-amortecedor e um modelo hidrodinâmico analítico, validado com dados de referência. O baixo custo computacional desses modelos permitiu a geração de grandes conjuntos de dados. A hipersuperfície de resposta N-dimensional em cada caso é uma combinação de parâmetros de entrada. Um estudo adequado permitiu a correta definição dos parâmetros da rede neural: o número de camadas e o número de neurônios por camada, visando a configuração que fornecesse o mínimo erro quadrático médio. A superfície de resposta representada pela rede neural pode ser facilmente acoplada a um algoritmo de otimização que avalie centenas de soluções viáveis e encontre o projeto ótimo. O uso de redes neurais como superfície de resposta tem excelente custo-benefício na modelagem dinâmica durante o projeto preliminar de plataformas *offshore*, nos casos em que o tempo disponível antes das tarefas de otimização é longo o suficiente para preparar um conjunto de dados de treinamento e nos casos sujeitos a atualizações de requisitos ao longo da fase de projeto conceitual.

Palavras chave: Redes Neurais. Projeto Oceânico. Superfície de Resposta.

List of Figures

List of Tables

List of abbreviations and acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	American Petroleum Institute
CAD	Computer Aided Design
CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
DNV	Det Norske Veritas
DRL	Deep Reinforcement Learning
FEA	Finite Element Analysis
FPSO	Floating Production Storage and Offloading
FPU	Floating Production Unit
GPU	Graphical Processing Unit
IMO	International Maritime Organization
JONSWAP	Joint North Sea Wave Project
MDP	Markov Decision Process
RAO	Response Amplitude Operator
RL	Reinforcement Learning
RNN	Recurrent Neural Network
TPU	Tensor Processing Unit
USP	University of São Paulo

List of symbols

α	Wave incidence angle
ω	Wave frequency
$\omega_{n,h}$	Heave natural frequency
ρ	Specific mass of water
A	Wave amplitude
A_{WL}	Waterline area
CS	Column Spacing
g	Acceleration of gravity
T	Draft
H_p	Pontoon Height
W_c	Column Width

Contents

1 Introduction

1.1 Artificial Intelligence and Ocean Engineering

Machine learning algorithms can process arbitrarily large and complex datasets to automate human intellectual tasks through mathematical statistics and computer science. The industry and researchers had been skeptical about the new developments in machine learning, driven by deep learning (DL) maturation since the 2000s. This skepticism reflects the past artificial intelligence (AI) winters: during the 1970s and between the late 1980s and early 1990s when failed attempts occured. In opposition to other exciting moments in the AI community, there are currently more resources to support the development of these algorithms. Big data and higher computing capacity, including Graphical Processing Units (GPU) and Tensor Processing Units (TPU), stand out as the supporting resources for new developments on deep neural networks (DNNs) (BRUNTON; NOACK; KOUMOUTSAKOS, 2019). For five years, skepticism has turned into real applications, and artificial intelligence is even more present in simple daily applications such as digital human resources and lawyers, virtual assistants, medical diagnosis, and Q&A bots. But AI is also leading the most advanced engineering applications: the autopilots for Tesla electric vehicles (Tesla..., 2021), the Boston Dynamics highly-mobile robots (Boston..., 2021), the Emirates Team New Zealand maneuvering simulator that afforded them America's Cup 2021 (America's..., 2021), generative design applications for images, voice, and 3D objects generation (NVIDIA, 2021).

The principle of this learning process is to update a randomly initialized matrix of weights until the dot product of the weights to the input data can reasonably represent a known output after being subjected to a nonlinear function. These product operations are successively repeated in many layers adding the "deep" characteristic of the neural network. The numbers of layers and elements inside them are closely related to the order of the learned problem. This technique is a powerful tool for automatic feature identification, much simpler to use than the sophisticated methods used before the introduction of DNNs. It is necessary to provide a training dataset with known inputs and outputs, used to minimize a pre-defined loss function using an optimizer. Validation and test procedures guarantee the quality of the network and prevent overfitting.

State-of-the-art AI applications are variations of Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Deep Reinforcement Learning (DRL). An expressive part of AI algorithms currently employs convolutional networks as a strategy to feature extraction in computer vision (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Recurrent neural networks are used in natural language processing (MIKOLOV et al.,

2010), and reinforcement learning is naturally related to robotics (KOBER; BAGNELL; PETERS, 2013). In the field of engineering design, Yonekura and Hattori (2019) combined the image processing capabilities from deep learning (using Convolutional Neural Networks on CFD contours) and reinforcement learning (updating the angle of attack of foils using the Deep Q-Networks algorithm) to propose an optimization framework for industrial turbine design. The authors affirm that the framework is task-independent, only requiring objective and design variables updates to extend it to other optimization tasks. For fluid dynamics applications, Wang and Wang (2021) observed that there are three notable uses of Artificial Neural Networks (ANN): provide a direct input-output response surface, optimize the existing physical models, or a combination of both strategies. Brunton, Noack and Koumoutsakos (2019) enumerate possible applications of different machine learning techniques to fluid mechanics fundamental research: experimental data processing, reduced-order modeling, shape optimization, turbulence closure models, and flow control. Baiges et al. (2020) and Haghighat et al. (2020) also present the use of ANN for reduced-order modeling and solutions in solid mechanics. Chaves, Tancredi and Andrade (2013) develop a framework to find the optimal structural design of small vessels, making use of the ANN as response surfaces.

In ocean engineering, researchers have worked with Deep Neural Networks as response surfaces to represent the system performance during the conceptual phase. The offshore design divides into three traditional phases of the so-called design spiral: conceptual design, basic design, and detailed design (API, 2007). An early estimation of the main geometric characteristics occurs during the conceptual design when the engineer employs complex models that reflect the nature of the ocean environment to predict the platform behavior. To map the hydrodynamic response of the offshore system, an extensive volume of simulations is often necessary during both the conceptual and basic design phases. It elevates the computational cost of the design process as a penalty for achieving the best hydrodynamic and economical solutions in the early design stages. Complex engineering has modified the conceptual design during the past decade by considering operational aspects and value perception along the whole life-cycle of the system. More recently, a large volume of operational data from sensors installed in similar platforms has impacted the early design, introducing the digital twin concept. At this point, the opportunity to converge the deep learning potentialities and the challenges of modern offshore design emerges.

An example of the application of Deep Neural Networks in ocean engineering is the building up of a neural network to represent the response of a Floating, Production, Storage, and Offloading unit (FPSO) due to the action of waves, winds, and current (SAEID, 2006). Zhang et al. (2017) presented a similar development for a semi-submersible unit but using a traditional response surface method instead. Lønnum (2018) assessed the application of deep learning in foreseeing the stochastic met-ocean generators for simulation-driven

design. Christiansen et al. (2013) proposed a hybrid method that combines finite element analysis (FEM) and artificial neural networks to perform dynamic analyses of mooring line systems. Venzon, Tancredi and Andrade (2014) employed the artificial neural network as a response surface to speed up the optimization of a semi-submersible hull in multidirectional waves.

1.2 Objectives

The objective of this work is to investigate the application of Deep Learning algorithms to represent the hydrodynamic response of an offshore platform during the conceptual design phase. This assessment includes an evaluation of the method, including its benefits, drawbacks, and challenges.

1.3 Document Outline

This thesis is organized as follows: Chapter 1 provides a brief introduction to Deep Learning and a general understanding of the state of art, specifically for engineering design. Chapter 2 introduces the main advances in offshore systems design along the past fifty years and connects the actual needs of offshore engineering to the artificial intelligence potentialities. Chapter 3 reviews Deep Learning: its fundamentals, general structure, and algorithms. Chapter 4 introduces the case study, presents the physical modeling of the system and the generation of the dataset. Appendix A shows the numerical experiments that corroborates the hydrodynamic model. Chapter 5 shows the training, validation, and testing methods used to build the deep learning model and results in the response hypersurface. Finally, chapter 6 discusses the benefits, drawbacks, and challenges of the Deep Learning application in offshore design, formalizes a framework for this AI application, and suggests future work and research.

2 Offshore Systems Design

Engineering design is a decision process dedicated to finding a solution in a set of alternatives. Chernoff and Moses (1986) established the basis of the modern decision theory, a formal model that employs statistics over observations and experiments about the state of nature. Other relevant design schools: Catalogue Design, which uses data of various proven solutions (PAHL; BEITZ, 1984); Robust Design which can accommodate extreme variations in design parameters and process variables via Information Axiom (SUH, 1998); Decision-Based Design, which embodies the concept of concurrent engineering design for the life cycle (MISTREE; MUSTER, 1990), Building-Block Design, an integrated approach to ship synthesis (ANDREWS, 2006) and; Risk-Based Design (PAPANIKOLAOU, 2009). More recently, optimization techniques have gained importance, and the decision process can deal with multi-objective models (NOWACKI, 2019). There is not only a single solution but a Pareto boundary that introduces a set of viable and optimal solutions. Cost and risk are also relevant, especially for large and expensive projects like offshore systems. In this way, the initial design has to fulfill the technical requirements and to set up from a perspective of the economic balance. There are also techniques to deal with uncertainty and complexity in early design.

Offshore structures are large platforms providing the necessary facilities and equipment for the exploration and production of oil and natural gas in a marine environment; wind, wave, and tidal energy farms; or seafood confined production (aquaculture). The design of these systems depends on the size, location, and water depth of the development. The O&G production facilities are either floating platforms or platforms placed directly on the seabed.

The platforms placed on the seabed can be fixed (rigidly or elastically), bottom supported (with or without skirts and piles, providing transverse resistance), or compliantly restrained, according to Dhanak and Xiros (2016). The use of these platforms is restricted by the water depth. The floating offshore platforms consist of SPARS, Tension Leg Platforms (TLPs), Semi-Submersibles (SS), and FPSOs.

2.1 Conceptual design

The conceptual design of an offshore structure defines its preliminary dimensions and capabilities, a process similar to the conceptual design of a ship. The cost of wrong decisions during this phase is comparatively high, considering the possibility of inefficient performance and the risk involved in each system's life cycle. A poor-quality design can reflect negatively on operational expenditures (OPEX) as the downtime increases in

production units. Other expensive consequences of mistakes in the preliminary design are catastrophic situations like sinking, explosions, oil spills, and fire on a production unit. These accidents represent losses of millions of dollars, damage to the environment, and risk to human lives. For this reason, special attention is given to the operational aspects and behavior of an offshore system: Traditional literature, for instance, Patel (1989) and Mccormick (2009), discusses the dynamics of different types of offshore structures with a high level of formalization from fluid mechanics and structural analysis. The stochastic nature of the sea environment leads to complex relations between the hull form and the behavior of the offshore system. The behavior, in turn, impacts the performance of the system in different circumstances, mainly governed by environmental aspects as waves, winds, sea currents, and the occurrence of storms. It is particularly difficult to choose the best concept during the preliminary design phase because there is a limited knowledge supporting decisions at this point (GASPAR, 2013), and the available time to work on this solution is limited, once it is often just a pre-contract phase (HALKYARD, 2005). The conceptual or preliminary design is a dynamic discipline that has developed from a rational selection of the main dimensions to simulation-driven and systems engineering approaches in the last decades.

The offshore industry keeps some particularities, differing from the marine industry in various aspects. For example, the offshore design tends to be more customized than the ship design, which occasionally works with standardized developments. Halkyard (2005) classifies the floating structures into permanent or mobile facilities. Permanent facilities are structures moored in place for a few decades (typically, 30 years) and demand a higher surviving capacity in harsh environmental conditions. The mobile units are suitable for temporary functions like drilling and construction after moving to different places.

As mentioned before, the floating systems are employed to support oil and gas production factories, drilling and construction units, wind energy generators, wave and tidal energy systems, aquaculture cages. Halkyard (2005) highlights that the ocean engineer "must understand all of the systems supported by the hull, and be prepared to include their effects in his modeling and design". It includes subsea equipment like O&G risers and umbilicals that depend on the subsea field architecture. El-Reedy (2012) relates the offshore structure design to the field development of an O&G project via multi-criteria selection techniques, considering for it the type of the platform, the topside facilities layout, well locations, subsea equipment, reservoir engineering, storage, and offloading systems. Clauss, Lehmann and Östergaard (1992) emphasize that the main parameters in the design of offshore structures for oil production are the reservoir size, the water depth, and the environmental conditions. They present the conceptual design together with hydromechanical analysis targeting transparent structures in wave conditions. Clauss, Lehmann and Östergaard (1988) expanded the design to arrangements dimensioning, structural evaluation, and analysis. Chakrabarti (2005) suggests statistical analysis for

floating systems, dividing it into short-crest and long-crest responses. Summarily, operational parameters for offshore engineering are closely related to environmental conditions and the type of equipment connected to the system.

According to Watson (1998), the isolated treatment of these different disciplines belongs to naval architecture. The designer must preserve the essence of all these subjects to synthesize a vessel concept that satisfies all the shipowner's requirements. Then, the design is a separate field of study among the many disciplines that develop along its pipeline. Regarding the offshore design, Clauss and Birk (1996) presented the shape optimization of different structures: gravity base, TLP, and semi-submersible. In Birk and Clauss (2001) and Birk (2008), rational seakeeping criteria provided the objectives set for the optimization of the pontoon section of a semi-submersible. Tancredi (2009) introduced the use of surface responses in the optimization of marine systems. Using artificial neural networks as response surfaces reduced the time spent processing several simulations. The author simulated the physical models in distributed systems.

The most common activities during the conceptual design phase of offshore structures are the weight and stability definition, the calculation of dynamic responses of the offshore platform, and fatigue in the mooring system. Academia has published different levels of modeling and design of ocean systems. Most works present the improvement of concepts of semi-submersible platforms using simplified models. For example, Leite, Vasconcellos and Nishimoto (1992) developed a method to generate and compare hull forms, Leite and Nishimoto (1992) studied the heave minimization considering the installation of rigid risers, Nishimoto et al. (1992) also worked on the design of a minimum heave platform, Nishimoto and Leite (1993) studied the influence of lateral keels and blisters to minimize heave. Conti, Andrade and Birk (2008) studied the effect of lower and upper parts of semi-submersible columns for heave response improvement, and Venzon, Tancredi and Andrade (2014) optimized a semi-submersible hull to improve the seakeeping in multidirectional waves. Deep and ultra-deep developments also present installation and mooring challenges as much as local environmental conditions (HOLMAGER, 2014). Therefore, Nishimoto, Fucatu and Masetti (2001) developed a time domain simulator for anchored FPSO units. In this work, we are particularly interested on the dynamics topic of the first spiral loop, i.e., the preliminary design.

2.2 The role of AI in the Offshore Design

As highlighted by Brunton, Noack and Koumoutsakos (2019), machine learning can support fluid dynamics in experimental/numerical data processing and reduced-order modeling. It is valuable in model-based design. Tancredi (2009) presents multiple examples of engineering design applications: from structural analysis of simple beams to a catamaran

hull. The advanced capability of computational processing and the possibility of using modern neural networks improved the quality of response surfaces produced by these algorithms.

The most advanced neural networks can train an agent to perform optimized operations that can be desirable to a high-level design, as in America's... (2021). These networks can learn policies to design strategies and help in shape optimization, for example. Table 1 shows multi-level applications of AI in ocean engineering, for different hydrodynamic models and types of neural networks. This table provides a general understanding of the AI potentialities on the offshore design subject. It also clarifies the choice of the thesis adopted pathway, described in the next sections of this document. The first column on the left side refers to the scope of this work, replacing the dynamic model of the semi-submersible platform with a deep neural network. This can also be done for FPSO designs, as mentioned in the second column. The processing cost of these options is variable, depending on the chosen analysis method. The next column refers to cases where CFD results are the basis of the analysis method, for example, to determine the airgap. Convolutional Neural Networks can represent the velocity and pressure fields. In the fourth column, it is proposed that the Reinforcement Learning algorithm could act as the agent looking for optimal and innovative designs, changing the floating system geometry iteratively. The last column merges all these ideas in a Deep Reinforcement Learning strategy. Dozens of other strategies could join AI and offshore design together. The brainstorming presented in table 1 is merely illustrative and portrays the range of applications at different challenging levels.

Table 1 – AI applications in ocean engineering.

Type	Model/Value		Model/Value	Value	Value/Policy
ANN	Deep Learning		Deep Learning	Reinforcement	DRL
Method	Dense Layers		CNN	Q-Learning	Q, CNN, Dense
Structure	Semi-sub	FPSO	Air-gap	All	All
Input	Geometry, met-ocean	Geometry, met-ocean	CFD contour	Geometry, met-ocean	All
Output	Heave, GM, airgap	Heading, excursion	Min airgap, efficiency	6 DoF response	All
Extra Output	6 DoF response	6 DoF response	-	-	-
Hydrodynamic models – Basic	Response model	1st order	-	-	-
Hydrodynamic models – Advanced	Panel Method, Morison, Wamit, Current, Wind, Mooring	Strip Theory, Jones, 2nd order	Finite-Volume	Wamit, Finite-Volume	Wamit, Finite-Volume
Validation	Cross- validation, Iterations, N layers	Cross- validation, Iterations, N layers	Verification & Validation, Cross-validation	Verification & Validation, Cross-validation	Verification & Validation, Cross-validation

Source: Author

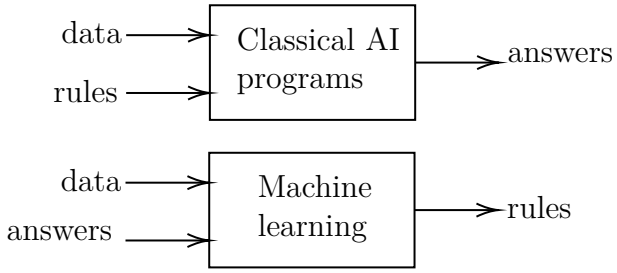
3 Artificial Intelligence

The idea behind artificial intelligence (AI) is to automate intellectual tasks that were exclusively performed by humans. These tasks are essentially "think" and "act". In some cases, it is desirable to perform thinking activities, such as learning, decision-making, and problem-solving. In other circumstances, the intelligent algorithm has to act in order to operate autonomously, pursue or create a goal, perceive its environment, and often adapt to new changes. Artificial intelligence is able not only to reproduce human performance but also to surpass it, achieving ideal/rational levels of functionality (RUSSELL; NORVIG, 2010). This rational approach is a combination of mathematics, physics, computer science, and engineering.

The fact of algorithms reach ideal levels of functionality is a great revolution in its paradigms, since by the beginning of the AI research, during the 1950s, experts believe that the so-called symbolic AI could achieve human-level performance only with the implementation of a large set of rules to manipulate data. In this approach, achieve the solution of well-defined and logical problems is the easiest task, once traditional computer programs are designed to perform fast arithmetic and explicitly follow instructions (BUDUMA; LOCASCIO, 2017).

Artificial intelligence congregates the machine learning algorithms that can learn to perform a specific task. While classical AI programs receive rules and data to output answers, in machine learning algorithms, the engineer inputs data and answers to output a model of the world, as shown in figure 1. Thus, a machine-learning system is trained rather than explicitly programmed (CHOLLET, 2017).

Figure 1 – Comparison between classical AI program and machine learning schemes.



Source: Adapted from Chollet (2017)

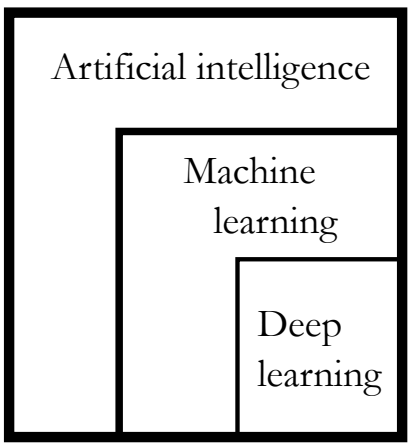
3.1 Machine Learning

Supervised learning is a mapping of input data to known targets or annotations informed by humans, as stated by Chollet (2017). It represents most part of the problems in machine learning: Binary classification, multiclass classification, and scalar regression (CHOLLET, 2017). It is also possible to create self-supervised models (for example, autoencoders) by generating the labels from the input data. Another approach is to find transformations of the input data without the help of any target, it is unsupervised learning. The main applications of unsupervised learning are dimensionality reduction, clustering, data visualization, data compression, data denoising, and the search for correlations in data analytics (CHOLLET, 2017). Finally, in reinforcement learning, an agent receives information about its environment and learns to choose actions that will maximize a pre-determined reward (CHOLLET, 2017).

All of these machine learning strategies are related to mathematical statistics and deal with large and complex datasets. The classical statistics becomes impracticable considering the data size and complexity, leading machine learning to be less theoretical and more oriented to engineering problems.

To represent or encode data using machine learning it is required: input data points, examples of expected outputs, and a measure of the distance between the predicted and expected outputs, that is the feedback signal. In this context, the automatic search for a good representation is the learning process of the model. The representation is selected in a predefined space of possibilities. A pioneering method to verify the quality of learning and originality of an AI code is the Turing test, in which a human player enters into a conversation, in natural language, with another human and a machine, separately. If the judge is unable to safely distinguish them, the machine passes the test.

Figure 2 – AI, machine learning and deep learning sets.



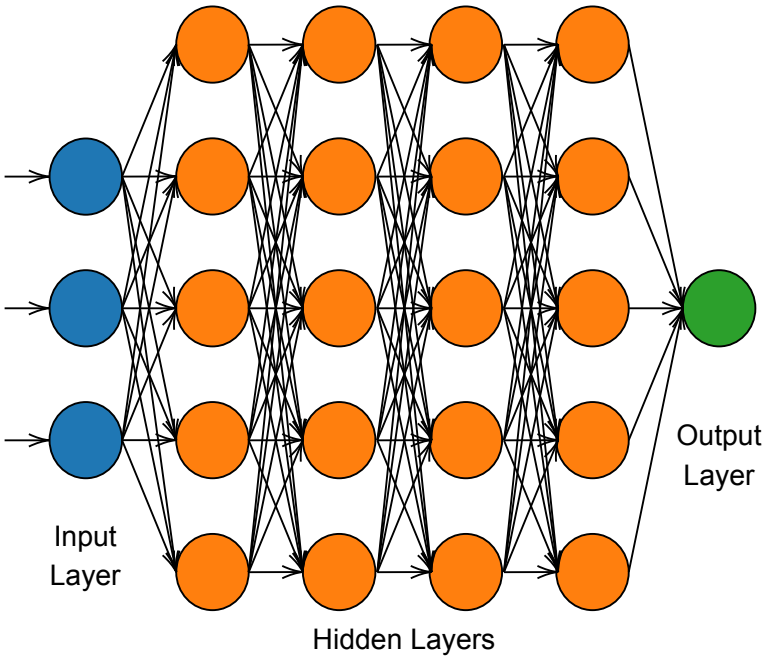
Source: Adapted from Chollet (2017)

According to Chollet (2017), one of the most popular types of machine learning in

present days is deep learning. The author points out that just as machine learning is a subset of artificial intelligence techniques, deep learning is one of the machine learning algorithms (see figure 2). However, most part of industrial machine learning algorithms are not deep learning, because it is not the best-suited model for a great part of applications. Probabilistic models, as the Naive Bayes and the logistic regression, have useful classification applications. Kernel method, Support Vector Machine (SVM), Decision Trees, and Random Forest are also classification algorithms.

3.2 Deep Learning

Figure 3 – Traditional representation of the deep learning pipeline



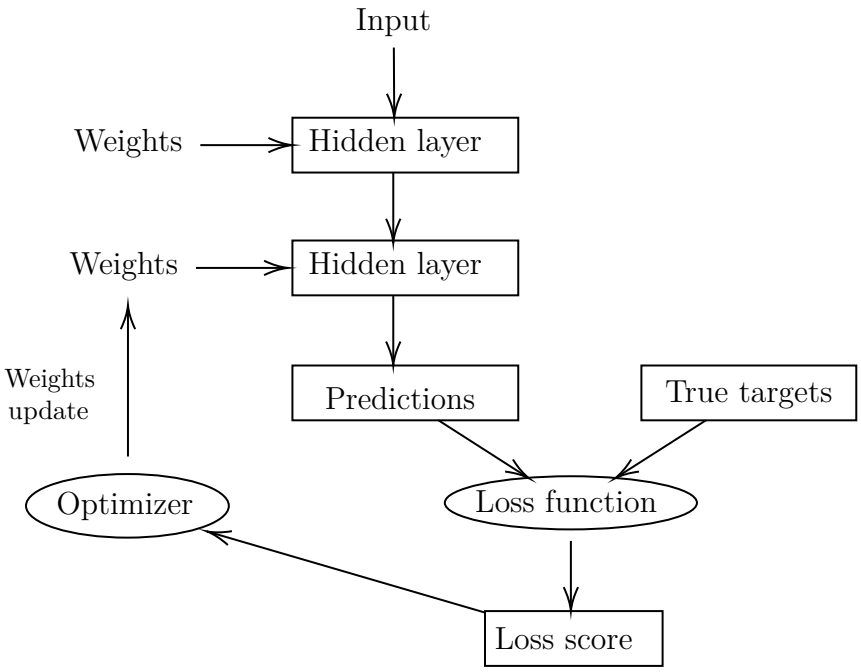
Source: Author

Deep learning is a type of machine learning. Its strategy is to add multiple operations layers to successively increase the representations meaning. The deep neural network possess multiple information stages that act as filters, reducing large amounts of vector data. There are always input and output layers in this type of model, while intermediary layers are named hidden layers. The input is parametrized in each hidden layer by a dot product to the weights matrix. Initially, these weights are assigned to random values. A loss/objective function is defined to compute the distance score between the target and the predicted outputs. The learning process is related to a back-propagation algorithm that adjusts the values of the weights with an optimizer. The distance score is used as a feedback signal in this process, which is represented in the figure 3 flowchart. A variation with only one or two hidden layers is called shallow learning. Previous machine learning methods required manual representations of the data features. Deep learning automates this step and allows

a joint learning process for all layers, in the opposite of a greedy process. Finally, the application of deep learning is particularly recommended for perceptual problems, like computer vision and natural language problems.

In figure 3 we can see the traditional representation of the matrix operations performed by a deep neural network and the data flow inside it. Each circle represents a neuron as the one presented in figure 5. In the next subsections we describe in details the foundation elements of these networks and some algorithms developed to improve training efficiency and the quality of final results. The whole procedure is summarized in the flowchart presented in figure 4.

Figure 4 – Deep learning algorithm with two hidden layers



Source: Adapted from Chollet (2017)

According to Russell and Norvig (2010), the main tasks performed by AI systems are natural language processing, knowledge representation, automated reasoning, machine learning to fit and extrapolate patterns, computer vision, and robotics. In these examples, all the combinations between *think/act* and *human-level/ideal level* oppositions are well represented. Most of these problems are quite easy to solve for human brains but require complex computations of intelligent machines. The learning process for humans occurs in most cases by example or by reinforcement, that is a reward or correction. Deep learning is likewise a subset of artificial intelligence in that learning occurs from example, not by formula.

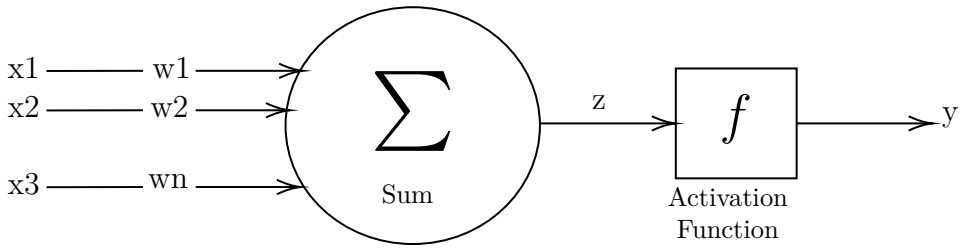
3.2.1 Neuron

A neuron is a basic unit of the neural network, "optimized to receive information from other neurons, process this information, and send it in a unique way" (BUDUMA; LOCASCIO, 2017). The sum of these signals, strengthened or weakened according to some criteria (activation function), constitutes the new propagated signal. This unit can be understood as a biomimetic model, compared to a biological neuron that performs similar operations in human brains. Actually, McCulloch and Pitts (1990) proposed the artificial neuron taking a number of vectorized inputs $x_1, x_2, ..., x_n$ and multiplying by weights $w_1, w_2, ..., w_n$. The problem can include a bias $b_1, b_2, ..., b_n$, producing a logit y given by equation 3.1. The outputs are also vectorized representations. A graphical representation of the neuron can be visualized in figure 5.

$$y = f(\sum_{i=0}^{n} w_i x_i + b_i) \tag{3.1}$$

An early type of neural network was the perceptron. Used for classification problems, it employs a binary function to divide a Cartesian coordinate plane in two. A linear perceptron is composed of a single neuron, but single neurons are more powerful when using different activation functions (BUDUMA; LOCASCIO, 2017). As indicated in the previous chapter, a deep neural network is composed by multiple layers of multiple neurons. The layers between the input and output layers are the hidden layers, that automatically learn the data features, when submitted to the optimizer. There must have fewer neurons in hidden layers than the number of inputs when it is desirable to force a compressed representation of the data. It is not necessary to connect all inputs to an output of a prior layer (see the Dropout technique in section 3.2.12).

Figure 5 – The neuron model



Source: Author

3.2.2 Activation

Nonlinearity is imposed on the model by activation functions. The most common are the sigmoid, tanh, and rectified linear unit (ReLU), although the linear perceptron activates in a binary function, as exposed before. The sigmoid neuron has equation f as

3.2. This imposes output close to zero for small logit and output close to one for very large logit. It is an S-shape function ranging from 0 to 1, as shown in figure 6.

$$f(z) = \frac{1}{1 + e^{-z}} \implies f'(z) = f(z)(1 - f(z)) \tag{3.2}$$

tf.math.sigmoid(z)

Another S-shape function is the hyperbolic tangent, differing the range from sigmoid. $f(z) = tanh\,(z)$ ranges from -1 to 1 and the expression is 3.3.

$$f(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \implies f'(z) = 1 - f(z)^2 \tag{3.3}$$

tf.math.tanh(z)

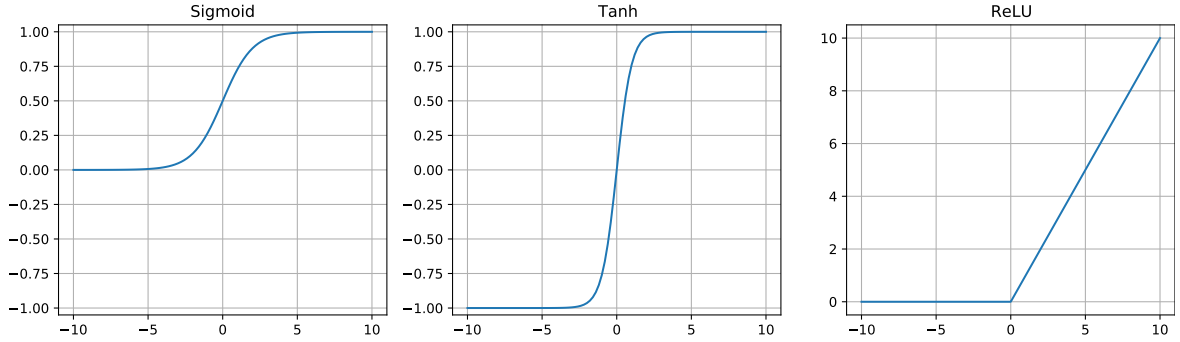
The neuron containing a rectified linear unit (ReLU) uses the function 3.4 and is valuable for multiple tasks, especially computer vision.

$$f(z) = max(0, z) \implies g'(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

tf.nn.relu(z)

Representation of the sigmoid, tanh and ReLU activation functions are plotted in figure 6.

Figure 6 – Sigmoid, tanh and ReLU activation functions



Source: Author

3.2.3 Feed-Forward Neural Networks

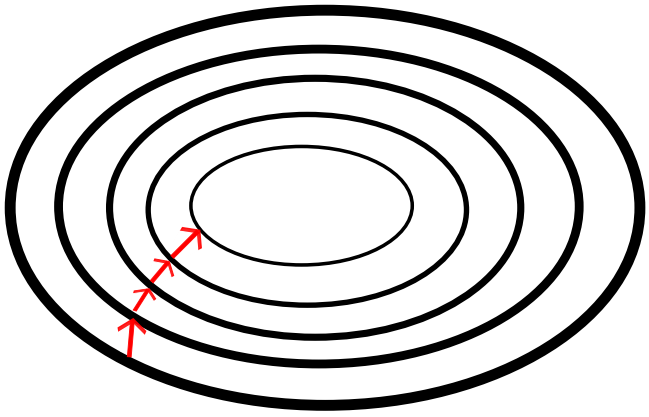
The training process consists in optimize the parameter vectors (weights). The intelligent selection of a training set provides effectiveness to the neural network, but it is necessary an accurate approach to train the ANN. The first choice, considering the linear neurons, could be the inclusion of the error function in the problem and treat all of them as a system of equations. Although it is a theoretically adequate approach, there is a generalization loss when we include the nonlinear functions. A better strategy to tackle the training process is the gradient descend.

$$E = \frac{1}{2}\sum_{i}\left(t^{(i)} - y^{(i)}\right)^2 \tag{3.5}$$

3.2.4 Gradient descend

To minimize the squared error function in equation 3.5 we employ the gradient (calculus concept) once the steepest descent is perpendicular to the contours in the plot of the error as a function of the weights. The strategy consists of randomly initialize the weights; evaluate error and gradient at the current position, and change it value in the direction of the steepest descend (BUDUMA; LOCASCIO, 2017).

Figure 7 – Graphic representation of the gradient descend



Source: Adapted from Buduma and Locascio (2017)

The algorithm requires an additional parameter to carry out the training process, the learning rate. The definition of this hyperparameter depends on the steepness of the surface. If we underestimate the learning rate, the training process longs more than necessary. On the other hand, taking a too large learning rate, the solution may diverge away from the minimum (BUDUMA; LOCASCIO, 2017). At this point, it is possible to adopt adaptive learning rates. The evaluation of the gradient provides the step size to

update each weight. The partial derivative of the error function shown in equation 3.6 is called the delta rule. It concerns each of the weights.

$$\Delta w_k = -\epsilon x_k^{(i)} \left(t^{(i)} - y^{(i)}\right)^2 \tag{3.6}$$

3.2.5 Back-propagation

The training of a multilayer neural network is better performed with the algorithm of backpropagation, proposed by Rumelhart, Hinton and Williams (1986). By definition, deep layers introduce the black-box modeling, suppressing the previous feature engineering. The features represented in hidden layers are unknown, so the backpropagation algorithm aims to compute the change velocity of the error by evaluating the error derivatives along with a single training. The knowledge of all the partial derivatives leads to the determination of the error change concerning each weight.

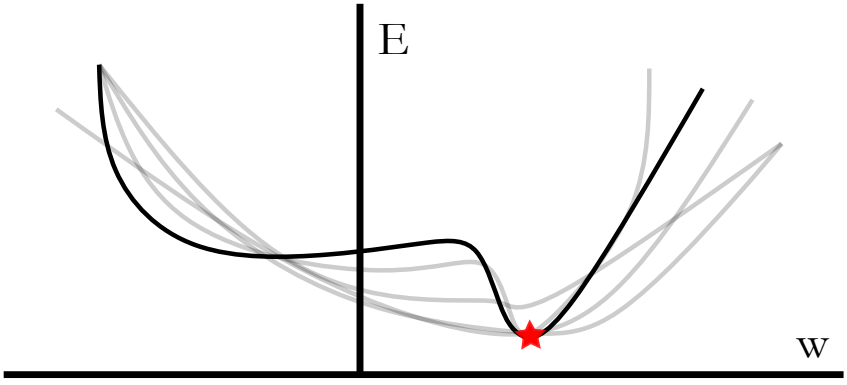
$$\Delta w_{ij} = -\sum_{k\ \epsilon\ dataset} \epsilon y_i^{(k)} y_j^{(k)} \left(1 - y_j^{(k)}\right) \frac{\partial E^{(k)]}}{\partial y_j^{(k)}} \tag{3.7}$$

Where the subindex j refers to layer j and the subindex i refers to layer i that is below j.

3.2.6 Stochastic gradient descend

For more complex error surfaces, the batch gradient descend can lead to saddle points. It produces erroneous results. The Stochastic Gradient Descend (SGD) arises as an alternative that estimates dynamic error surfaces with respect to a single example once.

Figure 8 – Convergence of the minibatch gradient descend



Source: Adapted from Buduma and Locascio (2017)

The minibatch gradient descent improves the algorithm by computing the error surface from multiple subsets of the complete dataset (minibatch), which is more efficient than a single sample evaluation.

$$\Delta w_{ij} = - \sum_{k \, \epsilon \, minibatch} \epsilon y_i^{(k)} y_j^{(k)} \left(1 - y_j^{(k)}\right) \frac{\partial E^{(k)]}}{\partial y_j^{(k)}} \tag{3.8}$$

3.2.7 Local minima and flat regions

An issue when dealing with error surfaces is that the local minimum is not necessarily the global minimum. Usually, this local information can extrapolate the global response. This extrapolation relates to the concept of model identifiability (BUDUMA; LOCASCIO, 2017).

Error surfaces in deep models have several local minima due to two influential reasons: Rearranging neurons in a layer results in equivalent configurations, and the possibility of infinite configurations result in equivalent networks after the ReLU activation. The behavior of nonidentifiable configurations is independent of the inputs, achieving the same error for training, validation, and testing data sets.

However, error rates and generalization characteristics are equivalent for local and global minima. The exceptions are the spurious local minima. This statement corroborates by the evidence that the distance of a randomly initialized error surface and its stochastic gradient descend solution is not affected in the local minima region (GOODFELLOW; VINYALS, 2015). The challenge is not the existence of a local minimum but to find the appropriate direction to reach it. Other difficulties for optimization algorithms are flat regions that slow down the learning process. It can be critical or saddle points.

In many circumstances it is also desirable to measure gradient changes between two positions of the calculation step, computing the second derivatives of the error (BUDUMA; LOCASCIO, 2017). The Hessian matrix (H) allows us to determine the second derivative when moving in a specific direction and limits the gradient descend optimization process. The second derivative is $d^H d$ in the direction of the unit vector d. Using second-order Taylor series approximation and stating that the optimizer is moving ϵ units in the gradient direction Buduma and Locascio (2017) write three terms for an expression that updates the error (equation 3.9): the error at the previous parameter, the error improvement due to the magnitude of the gradient and a correction term that incorporates the Hessian surface curvature. The computing of the Hessian matrix exactly is not a trivial task (BUDUMA; LOCASCIO, 2017).

$$E(x^{(t)} - \epsilon g) = E(x^{(t)}) - \epsilon g^T g + \frac{1}{2} \epsilon^2 g^T H g \tag{3.9}$$

3.2.8 Momentum-based optimization and second-order methods

It is desirable to achieve a velocity-driven motion in gradient descend to prevent large effects of wide fluctuations. Velocity smooths the trajectory, working as a memory that accumulates the movement in the minimum direction and cancels orthogonal accelerations (SUTSKEVER et al., 2013).

Introducing the momentum hyperparameter m in equation 3.10, the momentum term increases the step size and reduces the volatility of the model. It may require a reduction of the learning rate when compared to vanilla SGD ($m = 0$).

$$\begin{aligned} v_i &= m v_{i-1} - \epsilon g_i \\ \theta_i &= \theta_{i-1} + v_i \end{aligned} \tag{3.10}$$

Second-order methods come to directly approximate the Hessian. One of these methods is the conjugate gradient descend that successively computes the gradient direction and finds the minimum along that direction (MØLLER, 1993). Instead of moving to the steepest descend, a step in a conjugate direction can prevent oscillation. The conjugate direction is selected by operating a linear combination of the gradient and the prior direction (BUDUMA; LOCASCIO, 2017). Another second-order method is the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, introduced by Broyden (1969). It optimizes the parameter vector with the inverse Hessian matrix, computed iteratively (BUDUMA; LOCASCIO, 2017).

3.2.9 Learning Rate Adaption

The selection of the learning rate is both important and challenging. There are some different algorithms, where "mini-batch gradient descent, mini-batch gradient with momentum, RMSProp, RMSProp with momentum, Adam, AdaGrad, and AdaDelta" are the most popular (BUDUMA; LOCASCIO, 2017). Adagrad/Adadelta have the advantage of not depending too much on learning rates settings, but well-tuned SGD+Momentum almost always converges faster to better final values (ZEILER, 2012).

AdaGrad adapts the global learning rate over time inversely to the square root of the sum of all the squared gradients. Because of that, parameters with large gradients cause a fast reduction in their learning rate, while small gradients experience a small decrease in learning rates. The algorithm presents practical difficulties in model training.

$$\begin{aligned} r_i &= r_{i-1} + g_i \cdot g_i \\ \theta_i &= \theta_{i-1} - \frac{\epsilon}{10^{-7} + \sqrt{r_i}} \cdot g \end{aligned} \tag{3.11}$$

The exponentially weighted moving average of gradients (RMSProp) enables a more recent representation of the gradient accumulation once it imposes exponential weights to historical gradients. The decay factor ρ determines the influence time of previous gradients.

$$r_i = \rho r_{i-1} + (1-\rho) g_i \cdot g_i \tag{3.12}$$

The Adam algorithm combines the core concepts behind RMSProp and momentum: The first moment of the gradient keeps the convergence momentum and the second moment maintains the RMSProp of the historical gradients (BUDUMA; LOCASCIO, 2017). The second moment is recurrently represented as a function of past gradients and some simplifications are made using the algebraic identity. The final Adam update is given by equation 3.13.

$$\theta_i = \theta_{i-1} - \frac{\epsilon}{10^{-7} + \sqrt{\overline{v_i}}} \cdot \overline{m_i} \tag{3.13}$$

Where $\overline{v_i} = \frac{v_i}{1-\beta_2^i}$ and $\overline{m_i} = \frac{m_i}{1-\beta_1^i}$.

3.2.10 Input Data - Preprocessing

As we move for more complex engineering problems, data become high dimensional. Then, the relations we want to represent become increasingly nonlinear. These data must be vectorized or standardized, once neural networks do not process raw data, like text, image, or CSV files.

It is desirable to preprocess the input data inside the model. Using an external pipeline makes the models less portable, once it is not so simple to export an end-to-end model for production uses without including preprocessing in it. The input is much close to raw data, once the consumer of the exported model is not necessarily advised about the preprocessing pipeline (CHOLLET et al., 2015).

The preprocessing includes normalization and rescaling for numerical data. Input values should be in the [0, 1] range or zero-mean and unit-variance type. This implies that all features must range similarly. If there are missing values in the dataset it is usually safe to input missing values as 0.

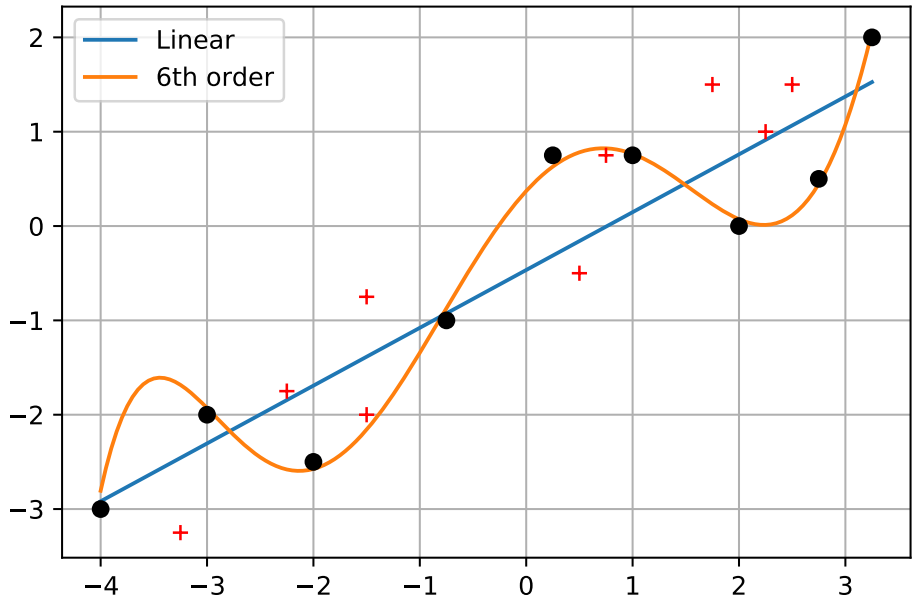
3.2.11 Model generalization - Overfitting, Validation and Test

Modern neural networks can extract features from raw data to ease a problem by expressing it simpler. This process is known as feature engineering. Specifically, deep learning is suitable for highly nondimensional problems, once the great numbers of neurons and layers can represent this nonlinearity by features that are not intelligible for the data

engineer. By providing a dataset for the neural network training, it is possible to reduce errors in the network model if it has enough degrees of freedom (number of layers and neurons). In opposition, the model can perform poorly to different input data, when it is excessively large compared to the problem order. It may suggest that the model is overfitting.

Overfitting is a phenomenon related to a loss of generalization in the model. It means that our modeling is excessively nonlinear and represents only the specific input data. According to Buduma and Locascio (2017), there is a "direct trade-off between overfitting and model complexity". Figure 9 shows two examples of trained models for the same dataset, one is well representing the data and the other one is overfitting. If more points of the same dataset (for example a training set) were added to the plot, the linear model could represent it better. To assure the model generalization, all available data is divided into a training and a test set.

Figure 9 – Example of overfitting model



Source: Author

The training process is divided into epochs, that is an iteration over the entire training set. Considering the mini-batch gradient descend, the number of epochs will be the size of the training set over the mini-batch size. To measure the model generalization at the end of each epoch, a validation step is performed. The model has a good generalization if both the accuracy of training, and validation sets are increasing or in a steady state. If one of them is decreasing, the model is overfitting.

The trained model is subjected to unseen data, i.e., the test set to measure the model's performance. If the performance is adequate, the model is finished and can be deployed and used for production. However, if the model is not performing well during the training, validation, or test steps, it may be necessary to reconsider the architecture of the network once it is not been capable of capturing important features on the data.

3.2.12 Techniques for overfitting prevention

Some different methods can be used to prevent overfitting during the neural network training step. The most trivial is to reduce the model size to achieve a better relationship between the number of coefficients and the number of training samples. Alternatively, create more training samples from the existing one through geometrical transformations (translation, scaling, rotation...) or filters (blur) can tilt the ratio between coefficients and training samples. During the validation step, it is observable a performance drop at some point in the graph. Another possible approach is to "early stop" the training when such inflection is detected.

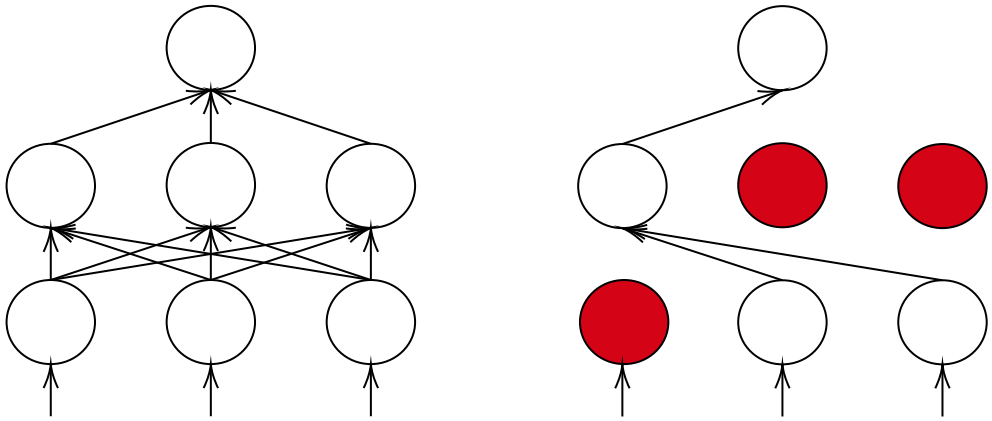
Regularization or penalization sums additional terms to objective function, penalizing large errors, as expressed in equation 3.14. The hyperparameter regularization strength λ is now defined and $f(\theta)$ is proportional to weights θ.

$$Error_{reg} = Error + \lambda f(\theta) \tag{3.14}$$

The most common machine learning regularization is the $L2$ regularization or "weight decay", that is a sum of $\frac{1}{2}\lambda w^2$ to the error, forcing every weight to decay linearly to zero (TIKHONOV; GLASKO, 1965). The $L1$ regularization adds $\lambda|w|$ to each weight w. It turns weight vectors sparse during optimization and benefits the understand of each feature contribution.

Finally, a random portion of the outputs can be dropped out in each batch, to avoid strong dependencies between portions of adjacent layers. To perform a dropout (figure 10) it is necessary to define the hyperparameter p that is the probability from that a neuron is kept active during the network training (usually set to 0.5 for training, and 1 for model evaluation). Otherwise, it is set to zero. The method prevents excessive dependency on small combinations of neurons. It is preferable to scale any active neuron before passing to the next layer, dividing by p during training time. It is the so-called inverted dropout that contributes to test-time performance (BUDUMA; LOCASCIO, 2017).

Figure 10 – Dropout



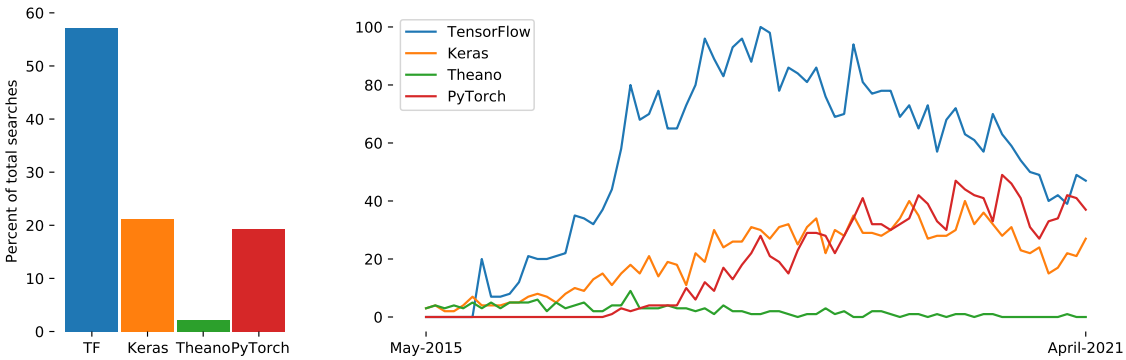
Source: Adapted from Buduma and Locascio (2017)

3.3 TensorFlow

Tensorflow is an open-source software used to develop deep learning models (ABADI et al., 2016). It works with data flows as tensors that are operated via graph structures. Constants are represented with a 0D tensor, vectors with a 1D tensor, matrices with a 2D tensor and the higher dimensional tensor are also available. The tensor representation allows the speed-up of computations, using parallel operations in GPUs and TPUs, for example.

Tensorflow has been released by Google since 2015. Other software options for building neural networks are available, including Theano, Torch, Caffe, Neon and Keras. The last one is an useful complement for TensorFlow and will be presented in the end of this section. Figure 11 shows the evolution of the use of these software in recent years. The criteria to select TensorFlow and Keras were the production oriented characteristics and ease of use of both software.

Figure 11 – Searchs of the main deep learning engines in Google Trends



Source: Author

3.3.1 Variables

TensorFlow variables are tensors hold by in-memory buffers that are explicitly initialized before the graph use. The performance of a deep neural network is highly impacted by the quality of parameters initialization. After each iteration, gradient methods modify the variables and store their values, searching for optimal parameters. Variables that are jointly instantiate in one place can be reused and shared to build more complex models (ABADI et al., 2016).

3.3.2 Operations

Operations are abstract transformations in the computational graph, applied to tensors. They are composed of one or more kernels, allowing the separation in CPU and GPU tasks. Tensorflow allows multiple computing devices to build and train deep models. Mathematical operations include element-wise, array and matrix options. Other classes are related to stateful, neural network building, checkpointing, queue, syncronization and control flow operations (ABADI et al., 2016).

3.3.3 Placeholders

Every time the computational graph runs, it populates a placeholder that is the way to input data into the deep learning model both during training time and test time.

3.3.4 Session

The Tensorflow session builds the initial graph, initialize all variables and run the computational graph. It is the interaction between the program and the graph model. During the session Tensorflow allows to monitor the minibatch cost, validation error, and the distribution of parameters, as much as save the model parameters.

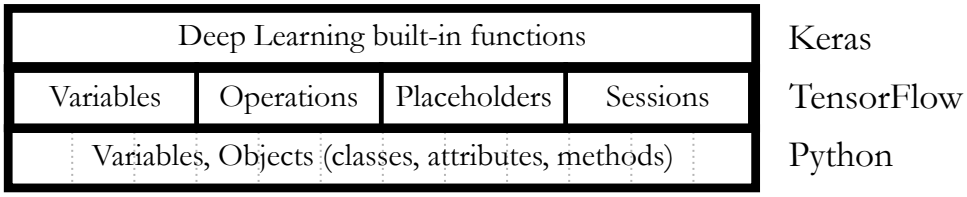
3.3.5 Initialization

As the number of layers in a deep neural network increases, the complexity of error surfaces increases together and it makes model optimization using vanilla stochastic gradient descent more difficult. As commented before, the initialization procedure is a manner to smooth this problem. For ReLU units, the variance of weights should be $\frac{2}{n_{input}}$, where n_{input} is the number of inputs reaching the neuron. The most common initialization strategies are the random normal and random uniform algorithms, although it is still possible to initialize all variables to zeros, ones, or other constants (HE et al., 2015).

3.3.6 Keras

Keras is a deep learning API written in Python (CHOLLET et al., 2015), running on top of the machine learning platform TensorFlow. Figure 12 illustrates the relation between Python, TensorFlow and Keras. The main strengths of Keras are the fast experimentation, scalability and cross-platform capabilities (It is possible to run the model in a browser, mobile device, GPU or TPU). Keras models accept three types of inputs: NumPy arrays that are a good option if the data is not too extensive, TensorFlow Dataset objects that are recommended for large datasets (with memory issues), and Python generators that groups data in batches (CHOLLET et al., 2015).

Figure 12 – TensorFlow and Keras: A high level deep learning environment



Source: Author

The Functional API is the main tool in Keras to build models. A model is a directed acyclic graph of layers. Each layer in that is a simple input-output transformation (CHOLLET et al., 2015). In practice, dense or fully connected layers are operations in that all inputs are densely connected to all outputs with a linear projection. Multiple layers can also be grouped together in a sequential structure. A sequential model is defined with:

```
model = tf..keras.models.Sequential()
```

Each dense layer is instantiated as follows, where the arguments provide the input shape, activation function, kernel initializer, kernel regularizer and bias regularizer. For varying input dimensions the shape must be specified as None.

```
model.add(
    Dense(
        units=1, input_shape=(n_samples,), activation = 'relu',
        kernel_initializer = tf.keras.initializers.RandomUniform(seed=1),
        kernel_regularizer='l2', bias_regularizer='l2'
    )
)
```

A good practice is to check the sizes of each layer's inputs/output with the summary of the model. The Functional API is still capable to build multiple inputs or multiple output models.

The Model class allows the definition of the optimizer and the loss function via the built-in function *compile*(). Both of them can have their arguments defined as shown below. The main argument in the optimizer definition is the learning rate. Otherwise, if the functions were specified via their string identifiers, the default argument values will be used.

```
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),
    loss=keras.losses.MeanSquaredError()
)
```

Another built-in function of the Model class is the $fit()$ method. After compile the model, Dataset objects or NumPy arrays are used as inputs to train the model. To do that, it is necessary to specify the batch size and the number of epochs (CHOLLET et al., 2015), if it is not previously defined. The batch size can be defined with an int value or set to None, which indicates the possibility of process batches of any size.

```
model.fit(
    train_X, train_y,
    batch_size=Nbatch,epochs=Nepochs
)
```

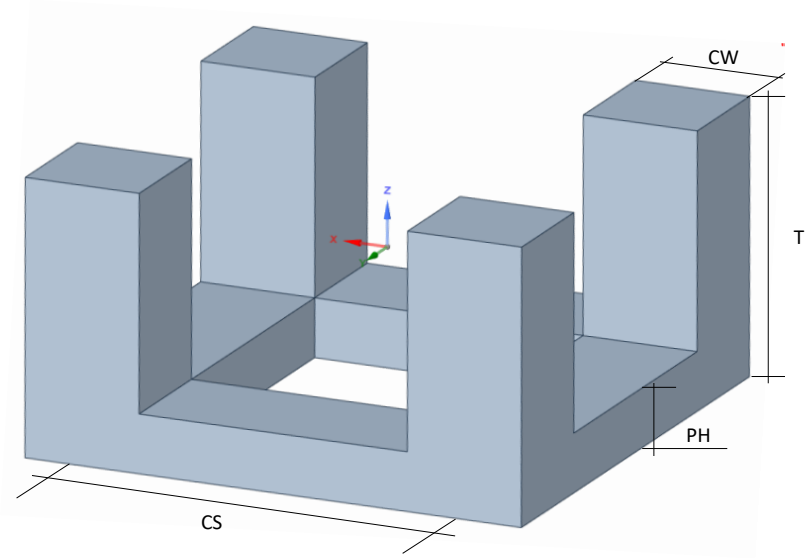
During the training/validation step, the dictionary history.history stores per-epoch timeseries of the training/validation metrics values such as classification accuracy, precision, recall, AUC (CHOLLET et al., 2015).

4 Semi-submersible FPU Design Model

The semi-submersible unit is traditionally a concept of floating O&G platforms, suitable for deepwater production and drilling (HALKYARD, 2005). Its applicability for deep-water developments inspired extensive research on the physical modeling, heave motion reduction, and new concept designs of this type of platforms. Zhang et al. (2017) proposed a traditional design of a semi-submersible floating production unit (SS-FPU), which geometry is shown in figure 13 and main dimensions reproduced in table 2.

This chapter presents the implementation and validation of the hydrodynamic model for heave response of a semi-submersible floating production unit under wave excitation, by means of reduced order formulations. The base geometry is presented by Zhang et al. (2017), that estimate the airgap and platform motions employing a panel method software for the computation of diffraction and Morison's theory equations. The space of solutions presented at the end of this chapter represents the systematic variation of the total draft, column spacing, column width, and pontoon height, following the design of Experiments presented by Qiu et al. (2019). The obtained space of solutions was used to train, validate and test a deep neural network in chapter 5.

Figure 13 – Schematics of the Semi-submersible FPU



Source: Author's simplification of the geometry presented by Zhang et al. (2017)

In this case study, analytical formulations replaced the more sophisticated models adopted by Zhang et al. (2017). The computational cost of the trained ANN present a

considerable reduction, when compared to the direct computation from the reduced-order hydrodynamic model. However, the savings on computational cost would be higher if the DNN was replacing panel method or finite volume software.

Table 2 – SS-FPU Platform Dimensions

Description	Unit	Value
Length	m	94.72
Beam	m	94.72
Draft	m	31
Freeboard	m	20
Number of columns	-	4
Column width	m	21.76
Column length	m	21.76
Column spacing (center to center)	m	72.96
Column corner radius	m	5.045
Pontoon width	m	21.76
Pontoon height	m	10.88
Pontoon corner radius	m	1.28
Width of cakepiece	m	5.12
Displacement	metric ton	108,031
Radius of gyration about x-axis	m	34.68
Radius of gyration about y-axis	m	34.68
Radius of gyration about z-axis	m	39.23
Transverse metacentric height	m	1
Longitudinal metacentric height	m	1

Source: Zhang et al. (2017)

4.1 Preliminary Design

The preliminary design of an offshore system requires the use of simplified functions that compute the platform stability criteria, the natural frequencies, and the response in waves. The ocean engineer can include simplified models to design an initial mooring system, the structural arrangement, estimate weights, provide fatigue analysis, select the positioning of risers and the field architecture. The light weight of the platform, for example, is often calculated by multiplying the surface area by a fixed thickness and the steel density, or given by a function of the structure volume. Additionally, the risers, marine equipment and topside weights can be fixed. The ballast weight is the difference between buoyancy and weight. It means that the total weight of the platform is equal to mass displacement for given geometric characteristics, i. e., considering the hydrostatic equilibrium of the platform.

For stability criteria, the metacentric radius is defined from the classical formulation of naval architecture and must be greater than zero to ensure a stable behaviour in calm

waters. In this study was kept in 1 m, as described by Zhang et al. (2017).

$$BM = KB + \frac{I}{\nabla} - KG \tag{4.1}$$

Where KB is the vertical position of the buoyancy center from the keel, I is the moment of inertia of the water-plane area, ∇ the displaced volume and KG the vertical position of the center of mass from the keel. The moments of inertia in roll and pitch are equal, and were described by Zhang et al. (2017) as function of the gyradius $R_g = 34.68$ and the displaced mass Δ:

$$I = \Delta \times R_g^2 \tag{4.2}$$

The added masses were calculated following the DNV (2017) reccomendations, in which the individual added masses of the structure elements are summed. For the pontoons it was calculated considering the aspect ratio between pontoon height and width $\frac{H_p}{W_p} = \frac{2b}{2a} = 0.5$. The references Kaneko, Nakamura and Inada (2008) and Newman (1977) present tabulated values for it, in this case the added mass coefficient is $C_{M,p} = 1.36$. Considering the reference area A_p for the added mass model, the number of pontoons equal to four, and each pontoon length L_p:

$$m_{a,pontoons} = 4\rho C_{M,p} L_p A_p \tag{4.3}$$

To compute the added mass of the columns, the coefficient $C_{M,c} = 0.47$ is related to a square plate of side equal to the column width ($\frac{W_c}{W_c} = \frac{2d}{2c} = 1$) oscillating in its normal direction (KANEKO; NAKAMURA; INADA, 2008). Once the columns cross the free surface, the added mass is compared to the volume of the hemisphere originated from the reference disk (ZHU; LIM, 2017). These components showed to be less relevant than the added mass on the pontoons.

$$m_{a,columns} = 4\rho C_{M,c} \pi \left(\frac{W_c}{2}\right)^3 \tag{4.4}$$

Where W_c is the column width. The total added mass m_a is the sum of $m_{a,pontoons}$ and $m_{a,columns}$.

The natural frequency in heave become:

$$\omega_{n,heave} = \sqrt{\frac{\rho g A_{WL}}{M + m_a}} \tag{4.5}$$

Where A_{WL} is the waterline area.The natural period is:

$$Tn = \frac{2\pi}{\omega_n} \tag{4.6}$$

4.2 Hydrodynamic Modeling

Modeling the dynamics of offshore structures is a complex issue yet for simple geometries due to the random nature of the ocean waves. By considering the random seas as a superposition of sinusoidal waves, the linear model introduces the Response Amplitude Operator (RAO), the transfer function of a floating system that represents its response in each degree of freedom (DoF). The product of the squared RAO to the wave energy spectrum provides the response spectrum of the system in waves. The offshore design requires the modeling of multiple phenomena to consider the loads on the structure and its respective responses, given the wave condition. The choice of the hydrodynamic model is essentially related to the dimensions of the platform and to the sea state (PATEL, 1989).

During the preliminary/conceptual design of semi-submersible platforms, the Morison, Johnson and Schaaf (1950) equation returns suitable approximations for the inertia and drag loads. Depending on the dimensions of the structural elements that compose the semi-submersible platforms, it may be necessary to consider the diffraction forces using boundary integral, or boundary element (BANERJEE, 1994) numerical methods, once an analytical solution is only available for vertical cylinders (PATEL, 1989). However, diffraction and viscous terms were neglected in this work.

Two simplified dynamic models represented the offshore structure at different levels of accuracy. Firstly, the expression of the 1 DoF mass-spring-damper system in the non-dimensional form reduced the input parameters to two. After, a hydrodynamic model represented the problem, considering wave excitation and waves radiation terms for heave loads. For this case, the offshore system is defined by its geometric characteristics.

All the equations presented in this chapter were implemented in Python language, using a Google Colab notebook in the web browser. The tool is fully compatible to TensorFlow and Keras, allowing the continuity of the deep learning training in the same pipeline. See chapter 5.

4.2.1 1 DoF Dynamic Model

An offshore structure is essentially a forced mass-spring-damper (MSD) system (RAO, 2011), with the harmonic variation of the external force, due to wave excitation (PATEL, 1989). The equations for each component derived from the physics of the problem:

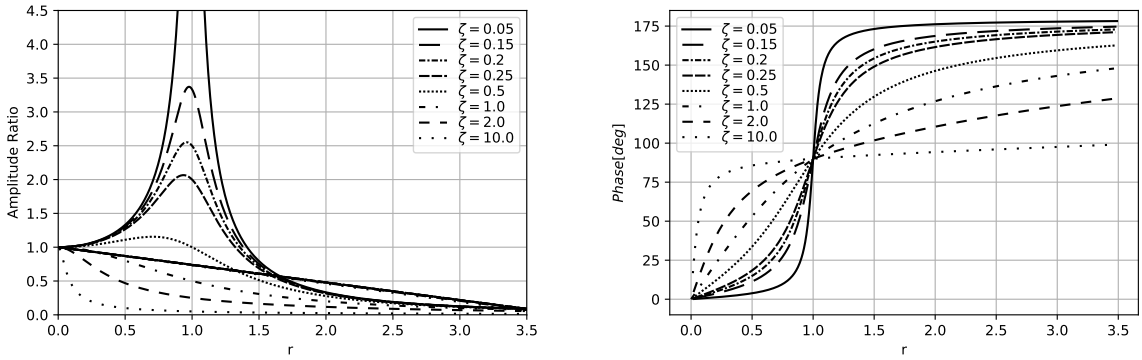
the mass term includes the inertia and added mass (PATEL, 1989) (FALTINSEN, 1993) of the platform components (pontoons and columns), the spring term arises from the hydrostatics, and the damping from both potential and viscous effects. This section introduces a general formulation for forced MSD dynamic systems, varying the frequency ratio r and the damping factor ζ, as a more comprehensive model than the complex hydrodynamic model of the platform.

The solution of the damped system under harmonic forces is a function of the amplitude of the external force F_0, the spring constant k, the mass of the system m, the damping coefficient c, and the oscillation frequency ω (RAO, 2011). Considering the natural frequency $\omega_n = \sqrt{k/m}$, the damping factor $\zeta = c/c_c = c/2\sqrt{km}$, the static deflection $\delta_{st} = F_0/k$ and the frequency ratio $r = \omega/\omega_n$, we can write the non-dimensional response amplitude and phase of a generic system:

$$\frac{X}{\delta_{st}} = \frac{1}{\sqrt{(1-r^2)^2 + (2\zeta r)^2}} \quad , \quad \Phi = tg^{-1}\left(\frac{2\zeta r}{1-r^2}\right). \tag{4.7}$$

Where X is the response's amplitude, and Φ is its phase.

Figure 14 – Response magnification and phase of a forced MSD system



Source: Author

4.2.1.1 Space of Solutions

For generalization, the space of solutions has examples of critically damped, overdamped, and underdamped systems, obtained by varying the damping factor ζ between 0.05 and 10. Offshore systems are usually underdamped. The frequency ratio is in the $[0, 3.5]$ interval, assuming 701 different values for each damping factor to provide a suitable resolution for resonant cases. Within this interval, solutions are around the resonance

($r = 1$), and it is possible to note the different effects of damping factors (figure 14). All the system's amplitudes series tend to zero as r grows infinitely large and to one when r approaches zero. The number of solutions to compose the dataset is 5608 dynamic systems, combining ζ and r variations.

4.2.2 Heave response of a semi-submersible Platform

Hooft (1972) described the following dimensionless approach to compute the heave response of a semi-submersible platform, this method shows good adherence to several RAO results. The procedure consists of dividing the offshore platform into its columns and pontoons and is described in detail by Assi et al. (2016), Newman (1977), and Kaneko, Nakamura and Inada (2008). These forces are a composition of the Froude-Krilov wave excitation force and the radiated wave force. The Froude-Krilov component (PATEL, 1989) considers no perturbation in waves due to the presence of the platform. It is analog to the buoyancy calculation for a two-dimensional section of the pontoon but using the wave acceleration in a vertical plane instead of gravity (PATEL, 1989). The diffraction wave and viscous forces were neglected for simplicity. The Froude-Krilov force component is:

$$f_{\zeta,2Dpontoon}^{Froude-Krilov} = \rho a_s \dot{w}(x, \chi_s, \zeta_s, \alpha) \tag{4.8}$$

Where ρ is the salt-water density, a_s is the two-dimensional pontoon section area, $\dot{w}$ the vertical acceleration, in the pontoon center of area (χ_s , ζ_s). A similar account is taken for the correction imposed to the reactive force: the platform moving without waves. This invokes the added mass m_a concept, which is related to forces of inertial nature of the fluid around a moving body, to consider the effect of the platform movement.

$$f_{\zeta,2Dpontoon}^{Added\ mass} = m_a \dot{w}(x, \chi_s, \zeta_s, \alpha) \tag{4.9}$$

Where m_a is the added mass. The integration of the sectional forces results in the total force acting on pontoons. The wave incidence angle α must be considered in the integration. For columns, both the Froude-Krilov and the radiated wave forces are punctual, in the center of the buoyancy plane. The heave excitation forces of the pontoons $F_{pontoon}$ and columns F_{column} are parcels of the equation 4.12, which is the total force F_ζ acting on the platform. The restoring forces depend on the heave added mass coefficient $C_a = \frac{m_a}{\rho a_S}$ of all the structures that compose the platform. The derivation of this equation, considering the incidence angle, results in a dependence on factors Q (equation 4.13) and P (equation 4.14).

By defining the wave incidence angle α, we can write the vertical acceleration as $\dot{w}_A(\chi, \psi, Z_S, \alpha, t) = IRe(-\omega^2 A e^{ik\,Zs} e^{k\,(\chi cos\alpha + \psi sin\alpha)} e^{-i\omega t})$, since $\chi cos\alpha + \psi sin\alpha$ is the wave propagation. The formulation also considers the exponential decaying of the wave

orbitals with the water depth Z. Fujarra (2009) proposes the integration of the heave excitation forces on the pontoons $F_{pontoon}$ and columns F_{column}. Here, we kept separated the Froud-Krilov and radiation terms using the reference area of Kaneko, Nakamura and Inada (2008) for the calculation of the added mass, as can be seen in equations 4.10 and 4.11.

$$F_{pontoon} = -(\rho\nabla + 4\rho C_{a,p} L_p A_p)\omega^2 A e^{-k\left(T-\frac{H_p}{2}\right)} e^{-i\omega t} \times \frac{1}{2}\left(\frac{sin(kL_p cos\alpha)cos(kL_p sin\alpha)}{kL_p cos\alpha} + \frac{sin(kL_p sin\alpha)cos(kL_p cos\alpha)}{kL_p sin\alpha}\right) \tag{4.10}$$

$$F_{column} = A_{WL} A\rho g e^{-kH} e^{-i\omega t} \times cos\left(k\frac{L_p+W_c}{2} sin\alpha\right) cos\left(k\frac{L_p+W_c}{2} cos\alpha\right) \tag{4.11}$$

Here, L_p is the pontoon length, W_C is the column width, T is the total draft, H_P is the pontoon height, A_{WL} is the waterline area, ∇ the displaced volume of the platform, the pontoon added mass coefficient $C_{a,p}$ was obtained for a section of aspect ratio $\frac{a}{b} = 2$ (DNV, 2010), ρ is the water density, g the gravitational acceleration, t the time, ω the wave frequency, A the wave amplitude, and k is the dispersion relation. Thus, the total excitation force in heave is a composition of the terms referring to pontoons and columns. The final expression remains:

$$F_\zeta = |F_{pontoon} + F_{column}| = -(1 + C_{a,p})\rho\nabla\omega^2 A e^{-k\left(H-\frac{H_p}{2}\right)} Q e^{-i\omega t} + A_{WL} A\rho g e^{-kH} P e^{-i\omega t} \tag{4.12}$$

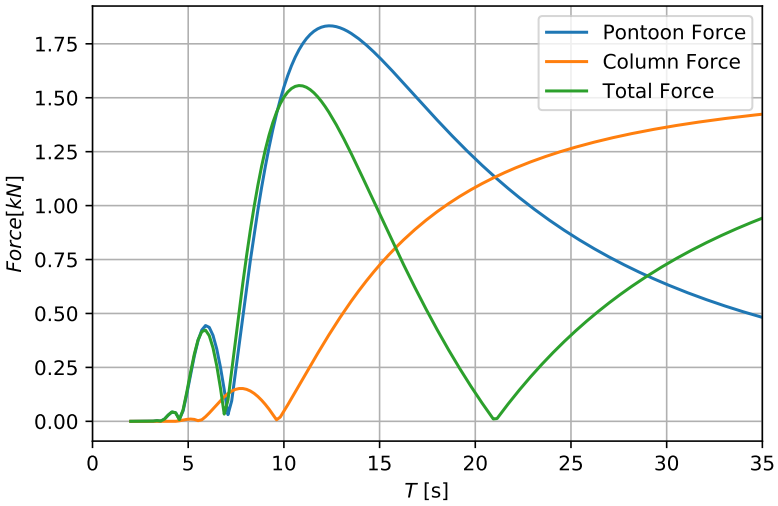
Where:

$$Q = \frac{1}{2}\left(\frac{sin(kL_p cos\alpha)cos(kL_p sin\alpha)}{kL_p cos\alpha} + \frac{sin(kL_p sin\alpha)cos(kL_p cos\alpha)}{kL_p sin\alpha}\right) \tag{4.13}$$

$$P = cos\left(k\frac{L_p+W_c}{2} sin\alpha\right) cos\left(k\frac{L_p+W_c}{2} cos\alpha\right) \tag{4.14}$$

The figure 15 shows the magnitude of heave forces. In some points, the column force equals the pontoon force in magnitude, but in opposite directions. These zero points can be seen in total force series.

Figure 15 – Total force in heave, incidence $\alpha = 0\,degrees$



Source: Author

According to Conti (2017), the dynamic equilibrium of vertical forces on the platform is composed of the inertial force, incident wave forces (Froude-Krilov and diffraction), restoring forces (gravity) and irradiation.

$$\underbrace{m(-\ddot{z}_0(t))}_{\text{Inertia}} = \underbrace{F_\zeta(t)}_{\text{Waves}} - \underbrace{\rho g A_{WL}(-z_0(t))}_{\text{Buoyancy - Weight}} - \underbrace{m_{azz}|_{pontoons}(-\ddot{z}_0(t))}_{\text{Irradiation}} \tag{4.15}$$

Considering the heave motion as an harmonic motion, we obtain the complex amplitude z_{0A}:

$$-z_0(t) = IRe\left(-z_{0A}e^{-i\omega t}\right) \tag{4.16}$$

What allow us to replace the amplitude in the vertical dynamic equilibrium:

$$\begin{aligned} &(m + m_{azz}|_{pontoons})(i\omega)^2(-z_{0A}) + \rho g A_{WL}(-z_{0A}) = F_{\zeta A} \\ \implies\ &-z_{0A} = \frac{1}{-\omega^2(m + m_{azz}|_{pontoons}) + \rho g A_{WL}} \cdot F_{\zeta A} \end{aligned} \tag{4.17}$$

Considering the dimensionless forms:

$$-z'_{0A} = \frac{-z_{0A}}{\zeta_{wA}} \quad , \quad F'_{\zeta A} = \frac{F_{\zeta A}}{\rho g A_{WL}\zeta_{wA}} \quad , \quad \omega' = \frac{\omega}{w_{n,heave}} \tag{4.18}$$

Where ζ_{wA} is the wave elevation. ω is a frequency and $\omega_{n,heave}$ is the heave natural frequency. The dimensionless response amplitude operator (RAO) remains as product of

the dimensionless excitation force and a magnification/attenuation factor Λ:

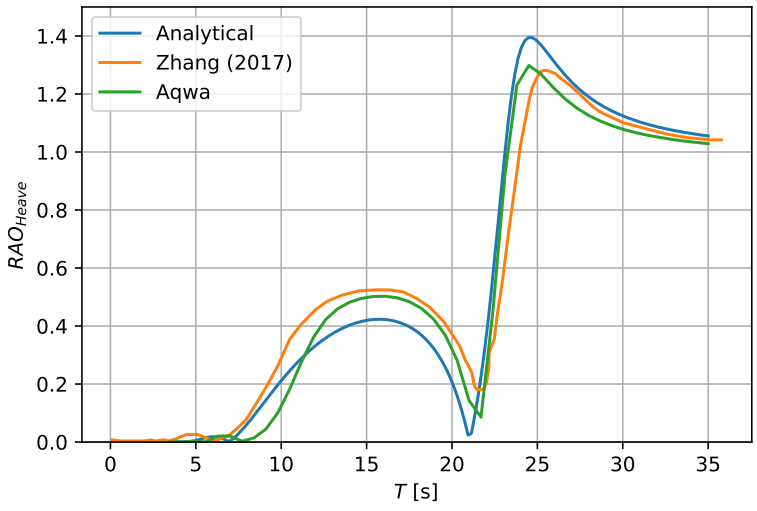
$$-z'_{0A} = \underbrace{\frac{1}{-\omega'^2 + 1}}_{\Lambda(\omega')} \cdot F'_{\zeta A} \tag{4.19}$$

The linearized viscous damping must be considered in these computations to avoid infinite heave response near the natural frequency. In this study case, the viscous damping was considered equal to 7% of the critical damping, i. e., an underdamped case as mentioned before. The $-z_{0A}$ equation remains:

$$z'_{0A} = \frac{F_{\zeta A}}{\rho g A_{WL} \sqrt{(-\omega'^2 + 1)^2 + (2\zeta_v \omega')^2}} \tag{4.20}$$

The graph of the heave RAO as a function of the heave period is shown in figure 16. Note that the natural period is about 23.37 s and the RAO is close to those presented by Zhang et al. (2017). Once the analytical method did not consider the cakepiece geometry, the appendix A presents a numerical experiment using Ansys Aqwa, with the simplified geometry considered in this chapter. The result of this experiment is also plotted in figure 16, further discussions on this result are in appendix A.

Figure 16 – Response Amplitude Operator (RAO) with damping



Source: Author

The Response Amplitude Operators (RAOs) are transfer functions to the unit wave amplitude. It is convenient to obtain the energy spectra of movements (also called response spectra) for any sea spectrum. The JONSWAP (HASSELMANN et al., 1973) is one of the most common sea spectra, requiring as inputs the significant wave height and

period (obtained as the average height and period of the highest third waves in a certain record). This spectrum is expressed as a function of the frequency:

$$A = e^{-\left(\frac{\frac{\omega}{\omega_p}-1}{\tau\sqrt{2}}\right)^2} \tag{4.21}$$

$$S = \frac{320H_s^2}{T_p^4}\omega^{-5}\, e^{\frac{-1950}{T_p^4}\omega^{-4}}\, \gamma^A \tag{4.22}$$

Where $\gamma = -3.3$ and $\tau = 0.07$ for $\omega \leq \omega_p$ or $\tau = 0.09$ for $\omega > \omega_p$. H_s and T_p are the significant wave-height and the significant wave-period, respectively. For validation purposes, these values were set as $H_s = 15.8\,m$ and $T_p = 15.4\,s$, equivalent to the survival condition of the validation platform in the Gulf of Mexico, reported by Zhang et al. (2017).

$$H = S|z'_{0A}|^2 \tag{4.23}$$

The heave response spectrum of the validation platform was obtained in the time domain (figure 17). The upper graphs represent the platform RAO (left) and the wave spectrum for the survival conditions mentioned above (right). The energy spectrum of heave movement, computed by equation 4.23, is plotted in the lower graph.

It can be observed that the heave natural period is out of the high energy peak of the JONSWAP spectrum, for the survival condition. Although system resonance occurs around 23.37 s, only the tail end of the wave spectrum is located in this region. There is a small peak near the heave natural period on the energy density spectrum, however the most critical region is the one with the highest wave energy, around the 15.4 s peak.

The response statistics are calculated from the zeroth moment of the spectrum, given by the integration 4.24 that is the area under the energy density spectrum:

$$m_0 = \int_0^\infty H(\omega)d\omega \tag{4.24}$$

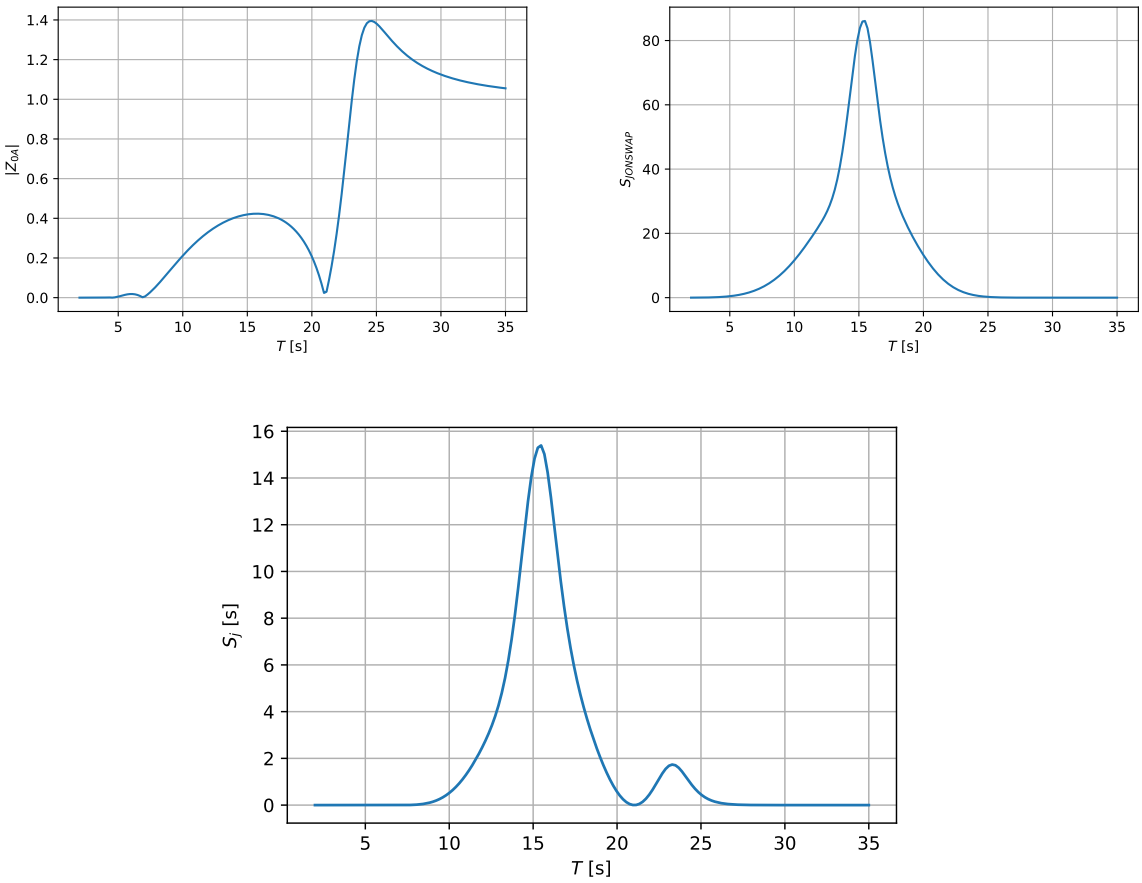
In short-term statistics, the significant height (of the response, in this case) is:

$$H_s = 4\sqrt{m_0} \tag{4.25}$$

By considering multiple incidence angles, we can obtain the variance σ^2 integrating the equation 4.24 in the incidence angle α. The variance allows the comparison of the response in different wave headings (CHAKRABARTI, 1987). No spreading function was employed at this part of the work.

$$\sigma^2 = \int_0^{2\pi}\int_0^\infty H(\omega)d\omega d\alpha \tag{4.26}$$

Figure 17 – Spectral crossing: Damped RAO, JONSWAP spectra and the heave response, as function of the period T_{Heave}



Source: Author

The most probable largest response, based on the assumption that the response follows a Rayleigh distribution is:

$$A_{max} = \sqrt{2\sigma^2 log\left(\frac{D_s}{T_A}\right)} \tag{4.27}$$

Where σ^2 is the variance of the responses, D_s is the duration of short term sea state (set to 3 hours by Qiu et al. (2019)), and T_A is the mean-zero-upcrossing response period (assumed equal to the significant period).

4.2.2.1 Space of Solutions

The validated hydrodynamic model was used to produce a space of solutions with 225 samples, i.e., different platform geometries: ranging the draft between 27.90 m and 34.10 m, the column spacing between 65.66 m and 80.26 m, the column width between 19.58 m and 23.94 m, and pontoon height between 9.79 m and 11.97 m. The Design of

Experiments (DOE) procedure proposed by Qiu et al. (2019) provided the sample values to each geometric parameter. We neglected the cakepiece width, whose effect was not considered in the dynamic model. The four input variables were linearly varied between the maximum and minimum limits shown in the table 3.

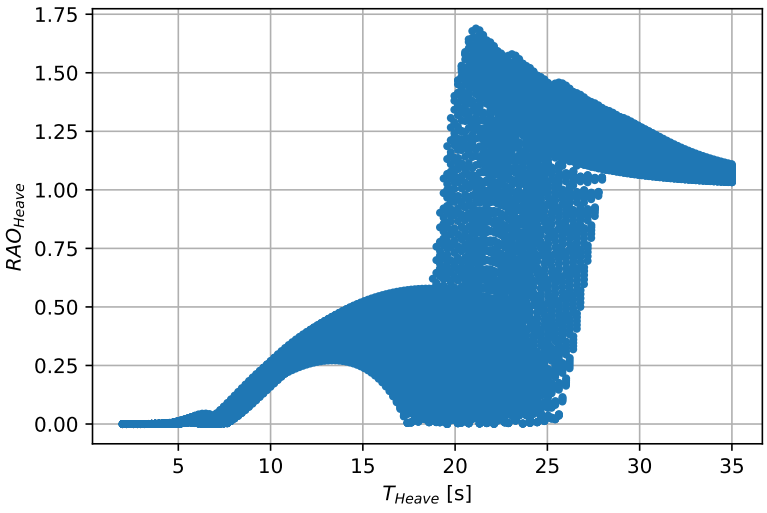
Table 3 – Bounds for each variable in the space of solutions

Variable	Lower bound	Upper bound	N Samples
Draft	27.9 m	34.1 m	5
Column spacing	65.664 m	80.256 m	5
Column width	19.584 m	23.936 m	3
Pontoon height	9.792 m	11.968 m	3

Source: Author

The dataset also included a period discretization with 170 elements. The total number of samples provided to the neural network was 38250 dataset elements, each one containing the five inputs and one output: the heave response. Figure 18 presents all heave responses as a function of the excitation period, considering all the geometric combinations.

Figure 18 – Heave response dispersion as function of the excitation period, for all geometric combinations



Source: Author

With the proposed hydrodynamic model, it was possible to obtain the MPM of the heave response (A_{max}), which is one of the objectives of the multi-objective optimization performed by Qiu et al. (2019). The total mass is the second objective and was obtained with complementary functions in our code. The GM and the airgap are constraints in the optimization algorithm, respectively obtained with the complementary functions and hydrodynamic model.

5 Response Surface of the SS-FPU Dynamic Model with Deep Learning

The reduction of the computation time during the early design is the central motivation of this chapter. To enhance delivery pace in the conceptual design phase, it is necessary to tackle its most time demanding activity: the dynamic model computation. The idea is to use a surface response that can be a deep neural network (DNN), among other techniques, instead of direct calculations. Faster results are valuable to performing optimization algorithms, for example. Following the main applications of deep learning in engineering design, the method consists of an assessment to reach acceptable representation levels of the hydrodynamic model using a deep neural network. The investigation aims to ensure that the engineer can replace the model with a neural network, not compromising the quality of the response.

To understand the use of two DNN parameters: the number of layers and the number of neurons per layer, the assessment is a sensitive analysis of the DNN system in TensorFlow (ABADI et al., 2016) using the sequential API in Keras (CHOLLET et al., 2015). Seventy percent of the space of solutions was employed for the neural network (NN) training, with ten percent internally dedicated to validation, while the remaining thirty percent was used for testing.

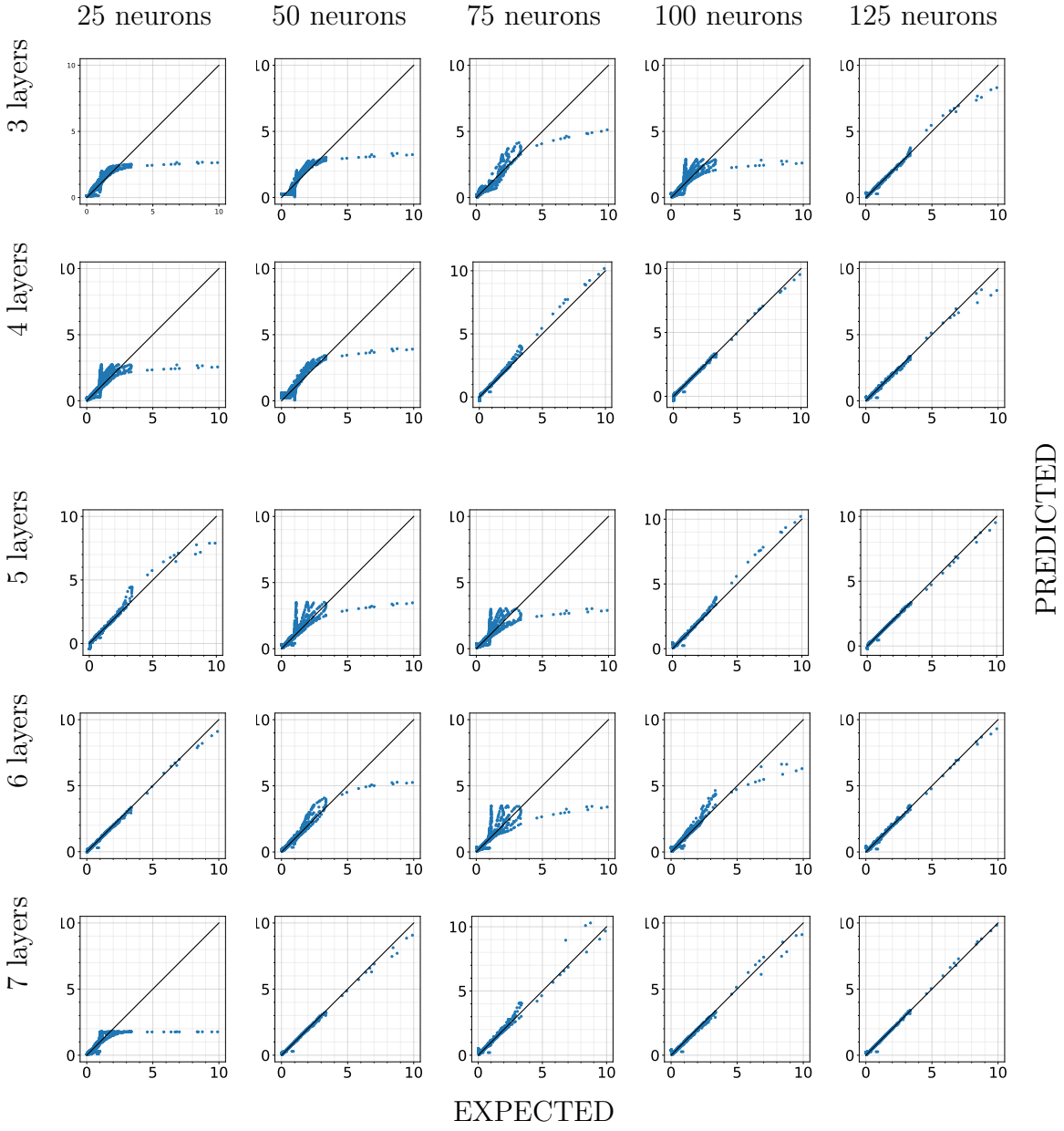
5.1 1 DoF Dynamic Model

For this study, we considered two inputs - damping factor ζ and frequency ratio r - and one output - Amplitude ratio $\frac{X}{\delta_{st}}$. The initialization of the weights followed a random uniform distribution, the regularization employed the L2 term, and the activation function is the Rectified Linear Unit (ReLU). The training process employed the Adam optimizer for 250 epochs and the mean absolute error as a loss function. The learning rate was 0.00005, and the batch size equal to 50.

Systematically varying the number of layers and the number of neurons per layer, it was possible to study how sensitive is neural network training regarding these two parameters. This study can be seen in figure 19, in which the x-axis of each graph is the expected value of the transfer function provided by the dataset. The values in the y-axis were predicted by the neural network, varying from 3 to 7 hidden layers (rows) and from 25 to 125 neurons per layer (columns).

During the DNN testing phase, we computed the mean squared error (MSE) of

Figure 19 – MSD analytical model: Sensitivity analysis of the neural network to the number of layers and neurons per layer



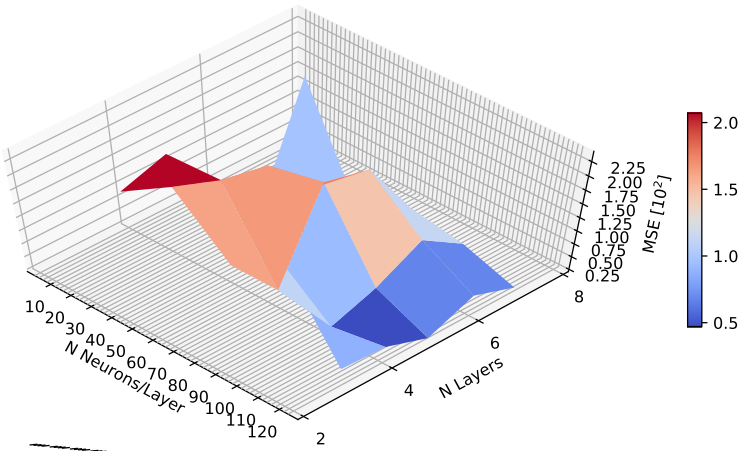
Source: Author

all the testing cases for each trained neural network. Figure 20 shows that the best performance, i.e., the minimum error, was obtained with five hidden layers of 125 neurons each. The MSE for this case is approximately 2.47 E-3.

$$MSE = \frac{1}{n}\sum (x_{expected} - x_{predicted})^2. \tag{5.1}$$

It is important to highlight that the DNN quality is directly associated with its convergence level that can be influenced by the random weights and bias initialization.

Figure 20 – MSD analytical model: MSE during testing phase of the NN, as a function of the number of hidden layers and number of neurons in each hidden layer



Source: Author

5.2 Heave Response of a semi-submersible Platform

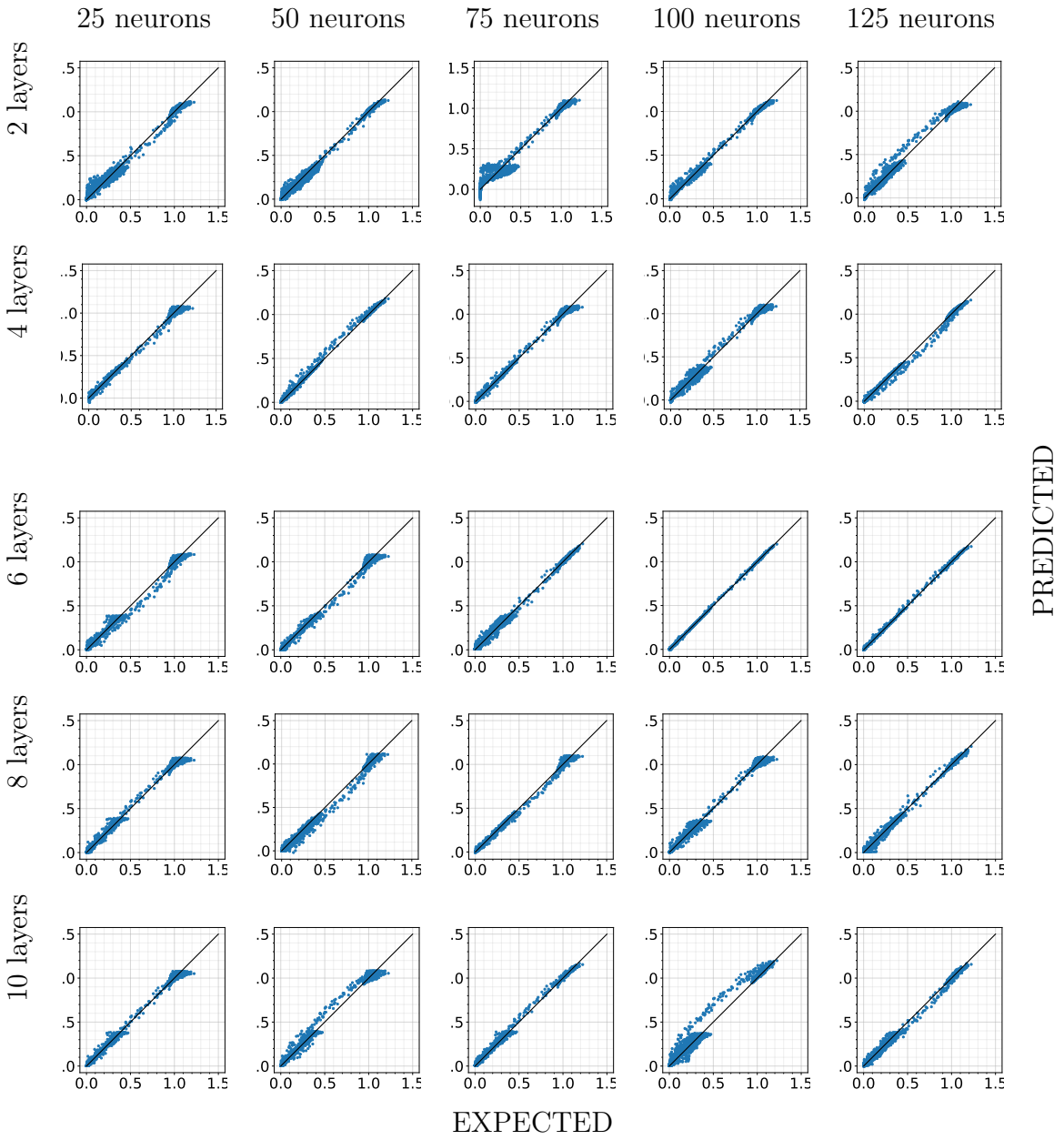
The analytical model for SS platform response allowed the generation of 225 cases by varying the draft, the column spacing, the column width, and the pontoon height. For each case, we computed the RAO considering 170 wave periods to produce a 38250 elements dataset, as described in chapter 4. As mentioned before, the complete dataset consists of five inputs (column spacing, draft, pontoon width, column width, and the excitation frequency $\omega = 2\pi/T$) and one output (heave RAO). The data pre-processing resulted in two sets, one for training with 70% of the samples and another for testing with 30%. In the training dataset, 10% was destined internally for validation. Data normalization ensured that each output value was within the range of zero to one.

Again, the author assessed five different numbers of layers (two to ten) and five numbers of neurons per layer (25 to 125). The study employed a ReLU activation function and random uniform weights initialization. Both kernel and bias regularizers were L2 types. During compilation, the Adam optimizer had a learning rate equal to 0.00005 and the mean absolute error as the loss function (CHOLLET et al., 2015). Convergent training was achieved with 250 epochs when using a batch size of 500. The reduction of the computed loss along the epochs expresses a convergence of the model training.

For test purposes, the author evaluated the predictions capabilities of the trained model with the 30% of the dataset selected to test. The trained model received the five inputs for each test sample and computed the output. The predicted outputs were compared to the known responses, as shown in figure 21. The measure of the mean squared error for each DNN model, i.e., each combination of the number of layers and number of neurons per layer, showed that a six-layered DNN, with 100 neurons in each layer, performed better (see figure 22). The MSE for this case is about 4.78 E-3, and the expected

vs. predicted plot suits better to the diagonal (figure 21) than in the other cases.

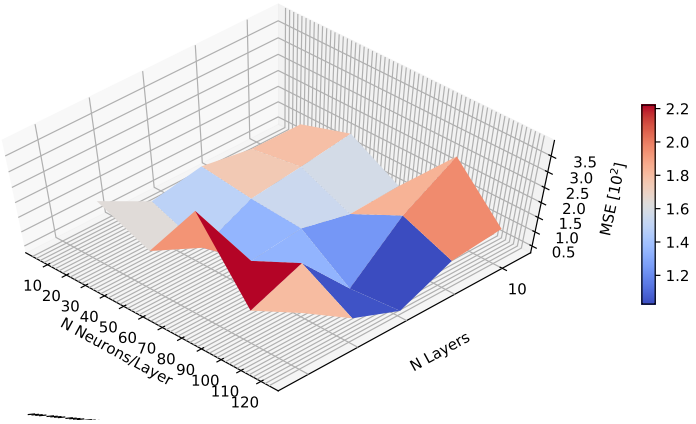
Figure 21 – SS analytical model: Sensitivity analysis of the neural network to the number of layers and neurons per layer



Source: Author

This exercise reveals the implementation challenges of the Artificial Neural Network (ANN) in the dynamic modeling of an offshore platform. The proposed method uses the evaluation of some hyperparameters during the traditional training, validation, and testing steps. The literature shows that the response surface obtained from a trained ANN is often a faster way to compute physical models in engineering problems that require repetitive tasks to compose a large sample (YONEKURA; HATTORI, 2019). The response surface represented by the DNN can easily be coupled to an optimization algorithm that evaluates hundreds of viable solutions and finds the optimal design. If the number of platforms

Figure 22 – SS analytical model: MSE during testing phase of the NN, as function of the number of hidden layers and number of neurons in each hidden layer



Source: Author

necessary for the optimization algorithm evaluation is higher than the samples provided to train a neural network, it is less expensive to employ the response surface. Moreover, if the offshore design changes requisites throughout the conceptual design phase, the number of cases evaluated is more substantial than the number of samples needed to train the network.

6 Conclusion

Artificial neural networks are robust response surfaces, representing the dynamics of floating systems. In this work, two different analysis methods produced distinct datasets: a one-degree-of-freedom mass-spring-damper system and an analytical model of the hydrodynamic response of a semi-submersible platform. They were individually trained, validated, and tested. The minimum mean square error for the mass-spring-damper system is relative to a DNN with five hidden layers of 125 neurons each. For the semi-submersible analytical model, a six-layer neural network with 100 neurons in each layer achieved the best result. When assembling the same dataset used in the analytical model for the numerical experiment carried out in appendix A, the same network configuration (6 hidden layers with 100 neurons each) gave the best result. The best-trained DNN accurately reproduced the RAO in the validation case.

Although these ANN usually represent relatively more extensive and complex problems than those presented in this work, the degree of generalization of the method is satisfactory, allowing it to handle data from multiple inputs in a vast range of applications. The same framework would process and reduce larger datasets like CFD simulations and experimental results in which neural networks could be a method for reduced-order modeling. Specifically for the simulations, it is known that they generally compute continuous solutions. The continuity of numerical solutions represents a benefit for the engineer allowing to use of a response surface with great assertiveness in the output, especially in the early design phases.

As the main objective of this application was to learn the data pattern using the neural network and not make a data reduction, successive layers of an equal number of neurons are enough. The great challenge was to define the number of layers, the number of neurons per layer, the learning rate, the batch size, and the number of epochs to train the DNN. A logical procedure can pursue acceptable levels of convergence and quality of the solutions. The result of a properly trained neural network reproduces the validation data effectively. Although the rational procedure consumes some training and testing time for the different combinations of parameters to compose a neural network, the single training time is negligible compared to the processing time of the analysis methods.

It is important to emphasize that an adequate volume of data is necessary to perform the correct training process of the neural network. Thus, the analysis methods require extensive computer processing before response surface preparation. Intelligent algorithms can define the appropriate sampling of experiments. In some applications, generating the samples can be more expensive than the process itself. Given this drawback,

neural networks are advantageous when the optimization task requires a larger sample space than the DNN training. The same consideration may be valid when running the optimization function more than once. For example, when changing design requirements along the process.

Future works include the extension of dynamic models, mainly considering the inclusion of the 6 DoF in the analysis. However, a mooring model is necessary to add surge, sway, and yaw movements to the model. Another consideration is that the complete wave statistics simulation will return a result considering the downtime of the platform and not the single analysis of a critical condition. Adding multiple ocean data and wave directions improves the comprehension of the different operational scenarios. Both analytical and panel-method are possible approaches for this attempt.

As mentioned in chapter 2, this work focuses on the more straightforward applications in offshore design. Vanguard applications combine deep learning with reinforcement learning to explore better design policies. The newest trend, Physics Informed Neural Networks, can model highly complex physical phenomena such as turbulence and separation points that take a long time to calculate in CFD solvers and have no analytical solution. The method applies to physical phenomena whose governing equations are PDEs (Partial Differential Equations). In these cases, deep learning helps to approximate high-dimensional functions.

Bibliography

ABADI, M. et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *ArXiv*, abs/1603.04467, 2016. Cited 3 times on pages 44, 45, and 61.

America's Cup Emirates NZ Team. 2021. <https://www.mckinsey.com/business-functions/mckinsey-digital/how-we-help-clients/flying-across-the-sea-propelled-by-ai>. Accessed: 2021-05-03. Cited 2 times on pages 23 and 30.

ANDREWS, D. J. Simulation and the design building block approach in the design of ships and other complex systems. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, v. 462, p. 3407 – 3433, 2006. Cited on page 27.

ANSYS, I. *Ansys Aqwa Reference Manual*. 2022. Cited on page 77.

API, A. P. I. Recommended practice for planning, designing and constructing tension leg platforms. In: . [S.l.: s.n.], 2007. Cited on page 24.

ASSI, G. R. d. S. et al. *Applied topics in marine hydrodynamics*. [S.l.]: EPUSP, 2016. Cited on page 54.

BAIGES, J. et al. A finite element reduced-order model based on adaptive mesh refinement and artificial neural networks. *International Journal for Numerical Methods in Engineering*, v. 121, p. 588–601, 2020. Cited on page 24.

BANERJEE, P. K. The boundary element methods in engineering. In: . [S.l.: s.n.], 1994. Cited on page 52.

BIRK, L. Application of Constrained Multi-Objective Optimization to the Design of Offshore Structure Hulls. *Journal of Offshore Mechanics and Arctic Engineering*, v. 131, n. 1, 12 2008. ISSN 0892-7219. 011301. Available from: <https://doi.org/10.1115/1.2957919>. Cited on page 29.

Automated Hull Optimisation of Offshore Structures Based On Rational Seakeeping Criteria, All Days de *International Ocean and Polar Engineering Conference*, (International Ocean and Polar Engineering Conference, All Days). ISOPE-I-01-058. Cited on page 29.

Boston Dynamics robots. 2021. <https://www.bostondynamics.com/>. Accessed: 2021-05-03. Cited on page 23.

BROYDEN, C. A new method of solving nonlinear simultaneous equations. *Comput. J.*, v. 12, p. 94–99, 1969. Cited on page 40.

BRUNTON, S.; NOACK, B. R.; KOUMOUTSAKOS, P. Machine learning for fluid mechanics. *ArXiv*, abs/1905.11075, 2019. Cited 3 times on pages 23, 24, and 29.

BUDUMA, N.; LOCASCIO, N. Fundamentals of deep learning: Designing next-generation machine intelligence algorithms. In: . [S.l.: s.n.], 2017. Cited 10 times on pages 31, 35, 37, 38, 39, 40, 41, 42, 43, and 44.

CHAKRABARTI, S. Hydrodynamics of offshore structures. In: . [S.l.: s.n.], 1987. Cited on page 58.

CHAKRABARTI, S. Handbook of offshore engineering. In: . [S.l.: s.n.], 2005. Cited on page 28.

CHAVES, V. T.; TANCREDI, T. P.; ANDRADE, B. D. Optimal structural design of small ships with response surface. In: . [S.l.: s.n.], 2013. Cited on page 24.

CHERNOFF, H.; MOSES, L. Elementary decision theory. In: . [S.l.]: Dover Publications, 1986. Cited on page 27.

CHOLLET, F. Deep learning with python. In: . [S.l.: s.n.], 2017. Cited 3 times on pages 31, 32, and 34.

CHOLLET, F. et al. *Keras*. 2015. <https://keras.io>. Cited 5 times on pages 41, 46, 47, 61, and 63.

CHRISTIANSEN, N. et al. Efficient mooring line fatigue analysis using a hybrid method time domain simulation scheme. In: . [S.l.: s.n.], 2013. Cited on page 25.

CLAUSS, G.; BIRK, L. Hydrodynamic shape optimization of large offshore structures. *Applied Ocean Research*, v. 18, n. 4, p. 157–171, 1996. ISSN 0141-1187. Available from: <https://www.sciencedirect.com/science/article/pii/S0141118796000284>. Cited on page 29.

CLAUSS, G.; LEHMANN, E.; ÖSTERGAARD, C. Strength and safety for structural design. In: . [S.l.: s.n.], 1988. Cited on page 28.

CLAUSS, G.; LEHMANN, E.; ÖSTERGAARD, C. Conceptual design and hydromechanics. In: . [S.l.: s.n.], 1992. Cited on page 28.

CONTI, M. B. de; ANDRADE, B. L. R. de; BIRK, L. Differentiation between the lower and upper parts of columns of semi-submersibles for heave response improvement. *Marine Systems & Ocean Technology*, v. 4, p. 63–72, 2008. Cited on page 29.

CONTI, M. de. *Notas de Aula*. 2017. Cited on page 56.

DHANAK, M.; XIROS, N. Springer handbook of ocean engineering. In: . [S.l.: s.n.], 2016. Cited on page 27.

DNV. *DNV-RP-C205 - Environmental Conditions and Environmental Loads*. [S.l.], 2010. Cited on page 55.

DNV. *DNVGL-RP-N103 - Modelling and analysis of marine operations*. [S.l.], 2017. Cited on page 51.

EL-REEDY, M. Offshore structures: Design, construction and maintenance. In: . [S.l.: s.n.], 2012. Cited on page 28.

ESTEVES, F. R. L.; ANDRADE, B. L. R.; NISHIMOTO, K. Enhancing the offshore conceptual design delivery pace using deep neural networks. In: *Rio Oil Gas 2021*. [S.l.: s.n.], 2022. Cited on page 78.

FALTINSEN, O. Sea loads on ships and offshore structures. In: . [S.l.: s.n.], 1993. Cited on page 53.

FUJARRA, A. L. C. *Lecture notes in Systems Dynamics II*. [S.l.]: University of Sao Paulo, 2009. Cited on page 55.

GASPAR, H. M. Handling aspects of complexity in conceptual ship design. In: . [S.l.: s.n.], 2013. Cited on page 28.

GOODFELLOW, I. J.; VINYALS, O. Qualitatively characterizing neural network optimization problems. *CoRR*, abs/1412.6544, 2015. Cited on page 39.

HAGHIGHAT, E. et al. A deep learning framework for solution and discovery in solid mechanics: linear elasticity. *ArXiv*, abs/2003.02751, 2020. Cited on page 24.

HALKYARD, J. Chapter 7 – floating offshore platform design. In: . [S.l.: s.n.], 2005. Cited 2 times on pages 28 and 49.

HASSELMANN, K. et al. Measurements of wind-wave growth and swell decay during the joint north sea wave project (jonswap). In: . [S.l.: s.n.], 1973. Cited on page 57.

HE, K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, p. 1026–1034, 2015. Cited on page 45.

HOLMAGER, A. Offshorebook: Oil & gas. In: . [S.l.: s.n.], 2014. Cited on page 29.

HOOFT, J. P. Hydrodynamic aspects of semi-submersible platforms. In: . [S.l.: s.n.], 1972. Cited on page 54.

KANEKO, S.; NAKAMURA, T.; INADA, F. Flow induced vibrations : Classifications and lessons from practical experiences. In: . [S.l.: s.n.], 2008. Cited 3 times on pages 51, 54, and 55.

KOBER, J.; BAGNELL, J.; PETERS, J. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, v. 32, p. 1238 – 1274, 2013. Cited on page 24.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, v. 60, p. 84 – 90, 2012. Cited on page 23.

LEITE, A. J. P.; NISHIMOTO, K. Minimisation of vertical wave exciting force and heave motion of a production semisubmersible with rigid risers. In: . [S.l.: s.n.], 1992. Cited on page 29.

LEITE, A. J. P.; VASCONCELLOS, J. M.; NISHIMOTO, K. A method for generation and parametric comparison of production semisubmersibles hull forms. In: . [S.l.: s.n.], 1992. Cited on page 29.

LØNNUM, O. J. J. Deep learning metocean simulation and its applications in marine simulation-based design. In: . [S.l.: s.n.], 2018. Cited on page 24.

MCCORMICK, M. Ocean engineering mechanics: With applications. In: . [S.l.: s.n.], 2009. Cited on page 28.

MCCULLOCH, W.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, v. 52, p. 99–115, 1990. Cited on page 35.

MIKOLOV, T. et al. Recurrent neural network based language model. In: *INTERSPEECH*. [S.l.: s.n.], 2010. Cited on page 24.

MISTREE, F.; MUSTER, D. Decision-based design - a contemporary paradigm for ship design. *Transactions of the Society of Naval Architects and Marine Engineers*, v. 98, p. 565–597, 1990. Cited on page 27.

MØLLER, M. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, v. 6, p. 525–533, 1993. Cited on page 40.

MORISON, J.; JOHNSON, J.; SCHAAF, S. The Force Exerted by Surface Waves on Piles. *Journal of Petroleum Technology*, v. 2, n. 05, p. 149–154, 05 1950. ISSN 0149-2136. Available from: <https://doi.org/10.2118/950149-G>. Cited on page 52.

NEWMAN, J. N. Book. *Marine hydrodynamics / J. N. Newman*. [S.l.]: MIT Press Cambridge, Mass, 1977. xiii, 402 p. : p. ISBN 0262140268. Cited 2 times on pages 51 and 54.

NISHIMOTO, K.; FUCATU, C. H.; MASETTI, I. Q. Dynasim : A time domain simulator of anchored fpso. In: . [S.l.: s.n.], 2001. Cited on page 29.

Effect of Lateral Keel And Blisters On Semisubmersibles For the Minimization of Heave Motion, All Days de *International Ocean and Polar Engineering Conference*, (International Ocean and Polar Engineering Conference, All Days). ISOPE-I-93-303. Cited on page 29.

NISHIMOTO, K. et al. Development of minimum heave motion semisubmersible. In: . [S.l.: s.n.], 1992. Cited on page 29.

NOWACKI, H. On the history of ship design for the life cycle. In: . [S.l.: s.n.], 2019. Cited on page 27.

NVIDIA. 2021. Cited on page 23.

PAHL, G.; BEITZ, W. Engineering design: A systematic approach. In: . [S.l.: s.n.], 1984. Cited on page 27.

PAPANIKOLAOU, A. *Risk-based ship design: Methods, tools and applications*. [S.l.: s.n.], 2009. Cited on page 27.

PATEL, M. Dynamics of offshore structures. In: . [S.l.: s.n.], 1989. Cited 4 times on pages 28, 52, 53, and 54.

QIU, W. et al. Multi-objective optimization of semi-submersible platforms using particle swam optimization algorithm based on surrogate models. In: . [S.l.: s.n.], 2019. Cited 3 times on pages 49, 59, and 60.

RAO, S. S. Mechanical vibrations fifth edition in si units. In: . [S.l.: s.n.], 2011. Cited 2 times on pages 52 and 53.

RUMELHART, D.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, p. 533–536, 1986. Cited on page 38.

RUSSELL, S. J.; NORVIG, P. Artificial intelligence - a modern approach, third international edition. In: . [S.l.: s.n.], 2010. Cited 2 times on pages 31 and 34.

SAEID, M. The usage of artificial neural networks in hydrodynamic analysis of floating offshore platforms. In: . [S.l.: s.n.], 2006. Cited on page 24.

SUH, N. P. Axiomatic design theory for systems. *Research in Engineering Design*, v. 10, p. 189–209, 1998. Cited on page 27.

SUTSKEVER, I. et al. On the importance of initialization and momentum in deep learning. In: *ICML*. [S.l.: s.n.], 2013. Cited on page 40.

TANCREDI, T. P. Otimização multidisciplinar distribuída aplicada a projetos de engenharia. In: . [S.l.: s.n.], 2009. Cited on page 29.

Tesla Autopilot AI. 2021. <https://www.tesla.com/autopilotAI>. Accessed: 2021-05-03. Cited on page 23.

TIKHONOV, A. N.; GLASKO, V. B. Use of the regularization method in non-linear problems. *Ussr Computational Mathematics and Mathematical Physics*, v. 5, p. 93–107, 1965. Cited on page 43.

VENZON, R. Z.; TANCREDI, T. P.; ANDRADE, B. L. R. d. Hull optimization of semisubmersible with multidirectional seakeeping criteria evaluated with neural network response surface. In: *International Congress of the International Maritime Association of the Mediterranean*. [S.l.]: CRC Press/Balkema, 2014. Cited 2 times on pages 25 and 29.

WANG, B.; WANG, J. Application of artificial intelligence in computational fluid dynamics. *Industrial & Engineering Chemistry Research*, v. 60, n. 7, p. 2772–2790, 2021. Available from: <https://doi.org/10.1021/acs.iecr.0c05045>. Cited on page 24.

WATSON, D. G. *Practical Ship Design*. Elsevier, 1998. v. 1. (Elsevier Ocean Engineering Series, v. 1). ISSN 1571-9952. Available from: <https://www.sciencedirect.com/science/article/pii/S1571995298800020>. Cited on page 29.

YONEKURA, K.; HATTORI, H. Framework for design optimization using deep reinforcement learning. *Structural and Multidisciplinary Optimization*, p. 1–5, 2019. Cited 2 times on pages 24 and 64.

ZEILER, M. D. Adadelta: An adaptive learning rate method. *ArXiv*, abs/1212.5701, 2012. Cited on page 40.

ZHANG, X. et al. Global motion and airgap computations for semi-submersible floating production unit in waves. *Journal of Ocean Engineering*, n. 141, p. 176–204, 2017. Available from: <http://dx.doi.org/10.1016/j.oceaneng.2017.06.004>. Cited 7 times on pages 24, 49, 50, 51, 57, 58, and 77.

ZHU, L.; LIM, H. Hydrodynamic characteristics of a separated heave plate mounted at a vertical circular cylinder. *Ocean Engineering*, v. 131, p. 213–223, 2017. Cited on page 51.

Appendix

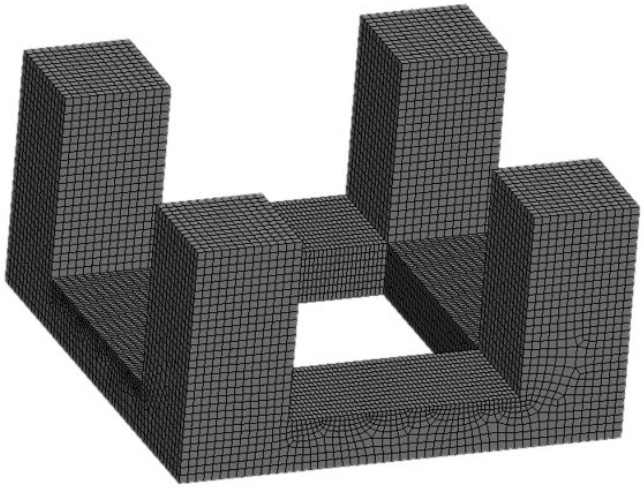
APPENDIX A – Numerical experiment using Ansys Aqwa

A suitable method for platform analysis is the frequency domain model for 6 DoF motions, based on the panel method and potential theory. This appendix presents the numerical experiment that validates the reduced-order model using Ansys Aqwa. The dynamic model of the platform introduced in chapter 4 was entirely built in the Ansys Workbench toolbox. The parameterized geometry was prepared with the aid of DesignModeler, and the model setup using the Hydrodynamic Diffraction and Hydrodynamic Response utilities in Ansys Aqwa.

> Aqwa can simulate linearized hydrodynamic fluid wave loading on floating or fixed rigid bodies. This is accomplished by employing three-dimensional radiation/diffraction theory and/or Morison's equation in regular waves in the frequency domain. Unidirectional or multiple directional second-order drift forces are evaluated by the far-field, or near-field solution, or full quadratic transfer function (QTF) matrix. Free floating hydrostatic and hydrodynamic analyses in the frequency domain can also be performed. (ANSYS, 2022)

The water surface size was set to 500 m, and the mass properties were defined according to Zhang et al. (2017). The surface mesh counted 8868 elements with maximum element size equal to 4.16 m (figure 23). The wave period ranges from 3.5 s to 35 s, with 46 wave period values. The results obtained for a base case compared well to those presented by Zhang et al. (2017) and to the analytical method presented in chapter 4.

Figure 23 – Surface mesh in Ansys Aqwa model

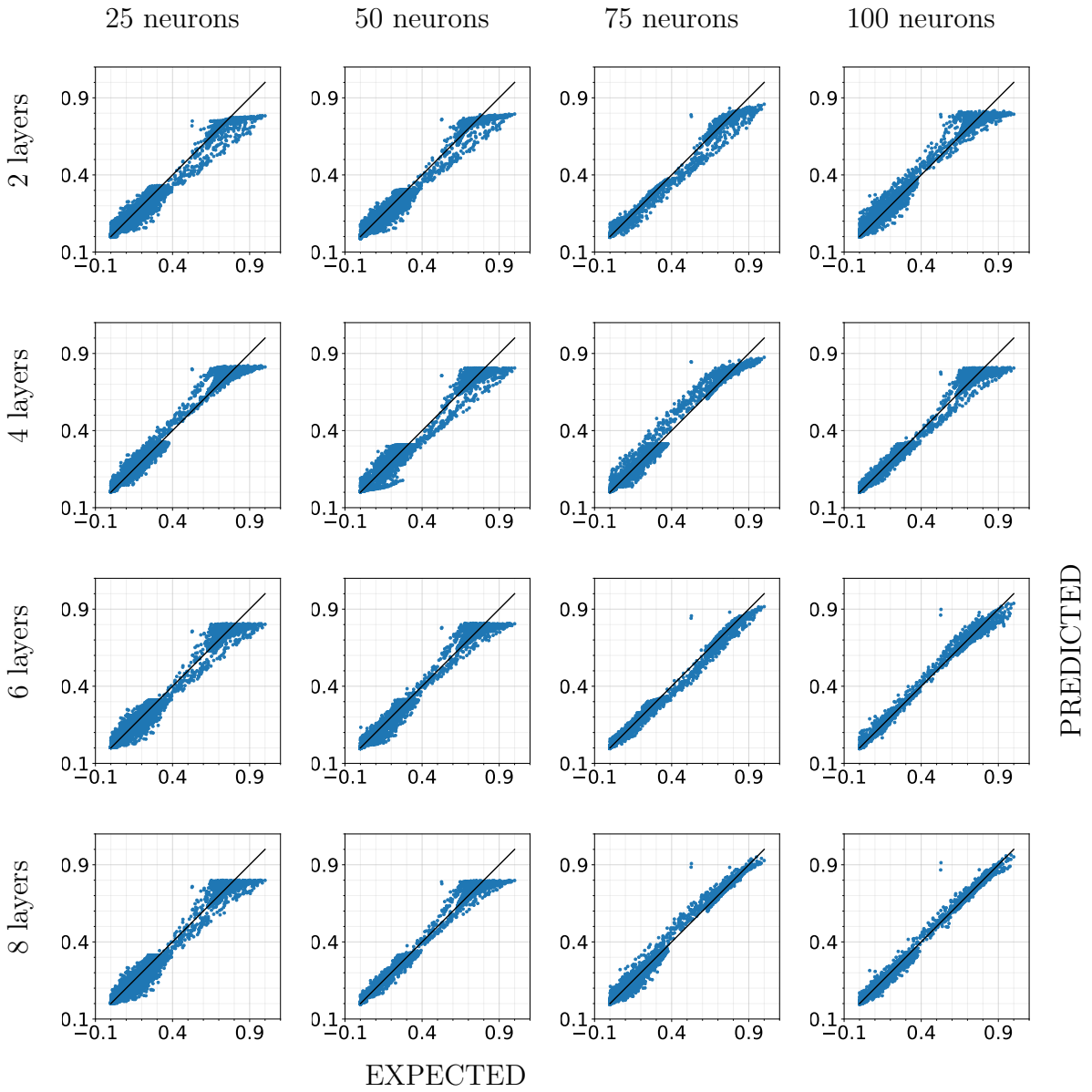


Source: Author

The graph of the heave RAO as a function of the heave period is shown in figure 16. Both the numerical experiment using Ansys Aqwa and the analytical method did not consider the cakepiece geometry.

A.1 DNN with the numerical experiment using Ansys Aqwa

Figure 24 – SS Aqwa model: Sensitivity analysis of the neural network to the number of layers and neurons per layer

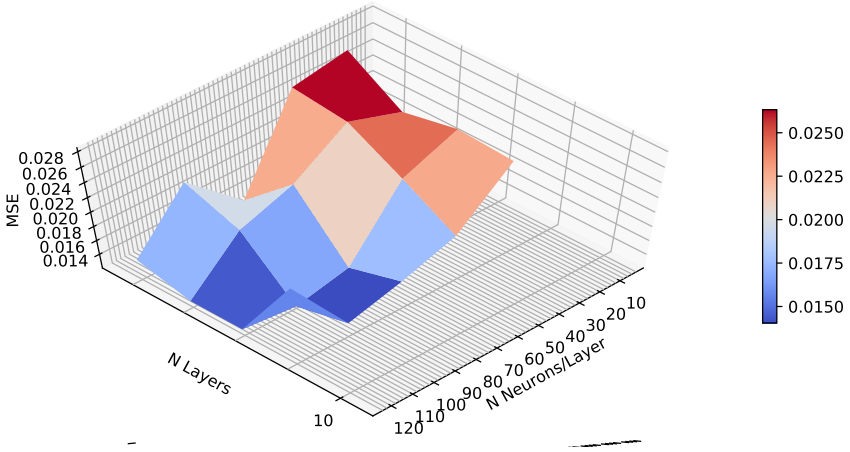


Source: Author

The assessment of DNNs containing from 2 to 8 layers and from 25 to 100 neurons per layer replicated the chapter 5 study. Esteves, Andrade and Nishimoto (2022) evaluated the predictions capabilities of the trained model with 30% of the dataset selected to test.

The trained model received the six inputs for each test sample and computed the output. The predicted outputs were compared to the known responses, as shown in figure 24. The measure of the mean squared error for each DNN model, i.e., each combination of the number of layers and number of neurons per layer, showed that a six-layered DNN, with 100 neurons in each layer, performed better. The MSE for this case is about 1.23 E-2 (figure 25), and the expected vs. predicted plot suits better to the diagonal (figure 24) than in the other cases. It is worth highlighting that the DNN quality is directly associated with its convergence level. The convergence can be influenced by the random initialization of weights and bias.

Figure 25 – SS Aqwa model: MSE during testing phase of the NN, as function of the number of hidden layers and number of neurons in each hidden layer



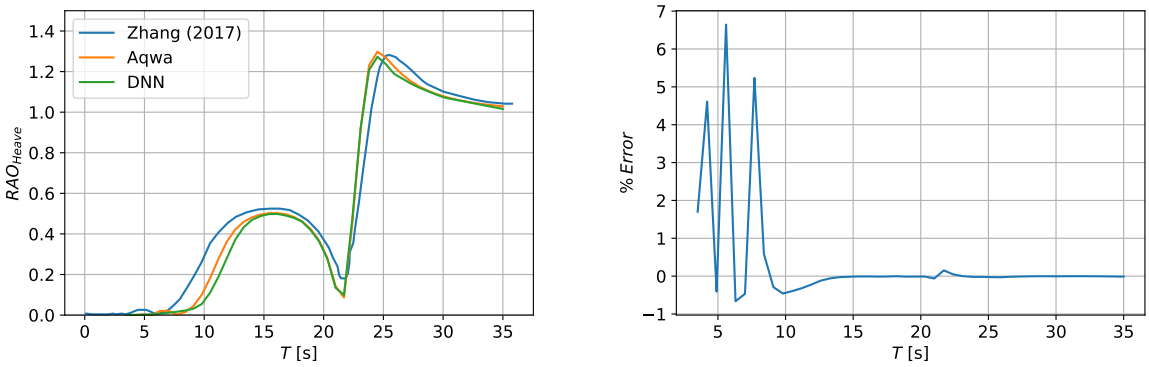
Source: Author

The study shows that the computation cost saving is more significant when the model is evaluated in Ansys Aqwa since each case lasts from 30 to 50 minutes with two Intel Xeon E5-2687W v3 @ 3.10 GHz processors (total: 20 cores). The computation time of the 225 cases presented in chapter 4 using the panel method software is in the order of 150 hours. It is needed less than 2 minutes using the free cloud service of Google Colaboratory to train the neural network with this number of parameters and design points. The complete response of an arbitrary platform can be generated instantaneously, with geometric parameters inside the training interval. However, the use of the DNN requires some input platforms that need to be previously simulated in Ansys Aqwa. The use of the neural network as a response surface has excellent cost-benefit in preliminary design dynamic modeling, in cases where the available time before the optimization tasks is long enough to prepare a training dataset, and in cases subjected to requisites updates throughout the conceptual design phase.

Although the base case was not considered in the training dataset, the selected DNN could reproduce its results proficiently. The exclusion of the base case from the dataset prevented overfitting. Figure 26 left shows that the RAO predicted by de DNN

fits well with the RAO computed by Ansys Aqwa for the same platform. The differences between them are below 7 percent, as shown in figure 26, right.

Figure 26 – Response magnification and phase of a forced MSD system



Source: Author