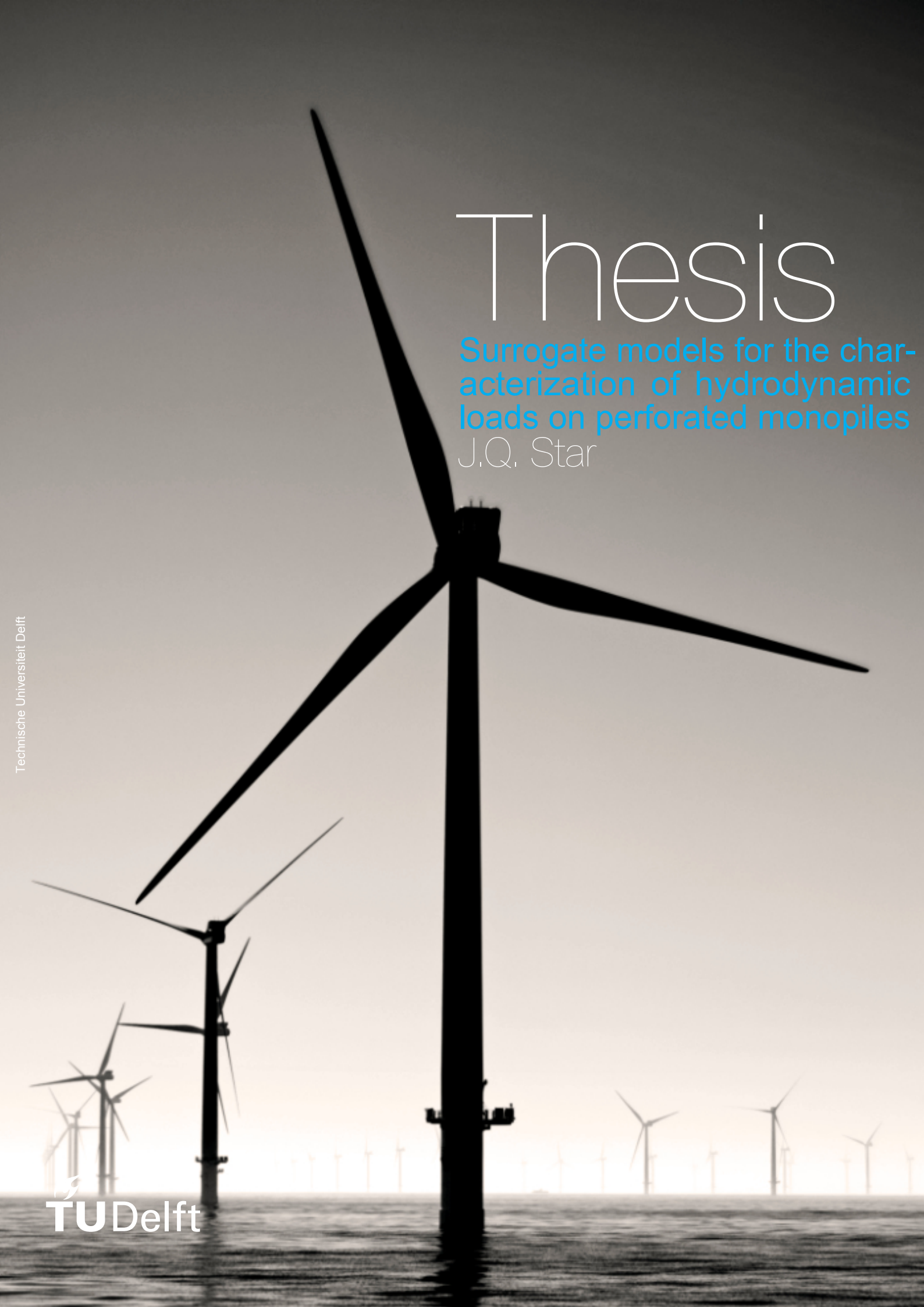


Thesis

Surrogate models for the characterization of hydrodynamic loads on perforated monopiles

J.Q. Star

Technische Universiteit Delft

TU Delft

Thesis

Surrogate models for the characterization of hydrodynamic loads on perforated monopiles

by

J.Q. Star

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday August 29, 2022 at 10:00 AM.

Student number:	4235061	
Project duration:	September 1, 2021 – August 29, 2022	
Thesis committee:	Dr. O.J. Colomés Gené,	TU Delft, chairman
	Dr. A. Heinlein,	TU Delft, supervisor
	Dr. A. Cicirello,	TU Delft, quality control

TU Delft

Preface

This thesis will conclude my master's degree in Offshore Engineering at the Delft University of Technology and my time as a student. Deciding on my bachelor study choice was a difficult choice between Computer Science and Aerospace Engineering. Even after successfully completing my bachelor's it was again a difficult choice between staying with Aerospace Engineering, pursuing computer science or doing something else all together. Offshore Engineering became the new and challenging fresh start I was looking for. A master thesis topic that combines offshore engineering, fluid flows and machine learning is therefore an excellent final project of my studies, that combines different interest of my.

Having never had any formal courses in Machine Learning, or experience in tackling such a large project from scratch I definitely gained a lot of experience and new knowledge during the course of the project. This will definitely prove to be valuable in my career going forward.

Firstly, I would like to thank both my supervisors for all their support over the course of the project. The regular bi-weekly meetings proved to be of great value in always keep moving forward. And for discussing results and any problems along the way. Especially, a great deal appreciation for Dr. Colomés for taking time to personally help fix a few issues with Gridap.

Furthermore, a lot of appreciation for Dr. Cicirello for being available near the end, even during the holidays, to provide the necessary quality assurance.

Finally, I would like to conclude that I hope you readers will enjoy reading this thesis!

J.Q. Star
Delft, August 22, 2022

Abstract

Perforated monopiles show promise in providing a better alternative to the commonly used jacket-like substructures used in intermediate water depths in the range of $30\,\mathrm{m}$ to $120\,\mathrm{m}$. By introducing perforations near the vicinity of the splash zone the wave loads on the monopile can be mitigated and the fatigue damage reduced, which is the main advantage jackets have over monopiles. Furthermore, perforated monopiles would be easier to install and manufacture compared to jacket structures, which has the potential to reduce the Levelized Cost of Electricty (LCoE) considerably. Perforated monopiles also have the benefit of reduced (local) scour, reduced corrosion damage and providing a habitat for marine life.

To optimize the design of perforated monopiles fast novel surrogate models are needed to determine the (wave) load reduction by the introduction of perforations on the monopile. As traditional (full) order Computational Fluid Dynamics (CFD) models are slow and expensive to evaluate. Using a surrogate model a wider range of geometries can be used in the early design optimization steps. Promising designs can then be further evaluated using full models.

This thesis develops a surrogate model for the characterization of the (hydrodynamic) loads on perforated monopiles using Convolution Neural Networks (CNN). A dataset is created of $15\,561$ samples, which considers geometries of varying number of perforations, porosity and 'angle of attacks'. A Finite Element Method (FEM) CFD model is made in Gridap for a 'creeping flow' to determine the flow fields. And determines a load Reduction Factor (RF) by the introduction of perforations. Furthermore, using a Signed Distance Function (SDF) a representation of the geometry is made for use in the CNN model.

Using PyTorch a CNN model is used to predict the RF for a certain input. The model combines multiple convolutional layers with an optional regression head (with linear layers). Five different models are created. Three for representing the geometry by using the SDF, decomposing the SDF in its x and y distance components and the binary representation. Furthermore, two models combine the SDF or distance components with the flow fields from the CFD model.

The results show that a speedup of 386 times is achieved by using the surrogate model, showing the main benefit of using a surrogate model. This is likely to improve considerably in further research when turbulence is taken into account.

Furthermore, each of the five models show a good accuracy in predicting the RF. A general trend is observed that the more information is provided in the input to the CNN model the better the accuracy tends to be. The combination of implicit representation of the geometry by using distance components with the CFD flow fields shows the best accuracy with an expected maximum relative error rate of $0.94\,\%$ within the parametric space of the dataset.

A 'two step' model is proposed for this model. The first step should use an autoencoder network architecture to reconstruct a flow field from a representation of the geometry. In the second part the flow fields and representation of the geometry is combined to be used with the created model in this research to predict the (load) Reduction Factor.

The approach taken in this thesis could be relatively easily be adapted to other kind of specific geometries. And be used to create surrogate models for optimizing designs in other industries.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

Binary	Binary representation function to the boundary of the monopile
Dist	Distance function to the boundary of the monopile
SDF	Signed Distance Function
API	Application Programming Interface
CFD	Computational Fluid Dynamics
CSG	Constructive Solid Geometry
DL	Deep Learning
FE	Finite Element
FEM	Finite Element Method
FRC	Flow Region Channel
FSI	Fluid Structure Interaction
HPC	Hyper Performance Computing
JIT	Just In Time
KNN	K nearest Neighbors
LCoE	Levelized Cost of Energy
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MSE	Mean Squared Error
PDEs	Partial Differential Equations
POD	Proper Orthogonal Decomposition
RANS	Reynolds-Averaged Navier-Stokes
RB	Reduced Basis
ReLU	Rectified Linear Unit (layer)
ROM	Reduced Order model
ROM	Support Vector Machines
TPE	Tree-structured Parzen Estimator

Symbols

α	Angle of rotation of the monopile	rad
β	Porosity of the monopile	–
γ	Learning rate	–
λ	Weight decay	–
ϕ	Angle between two consecutive perforations	rad
Re	Reynolds number	–
RF	Hydrodynamic load reduction factor	–
θ	Perforation width in degrees	rad
a_{mesh}	Mesh size decay factor	–
A_{perf}	Wetted area of the perforated monopile	m^2
A_{tot}	Wetted area of the non-perforated monopile	m^2
b_{mesh}	Mesh size decay exponent	–
c	Chord length	m
D	(Outer) diameter of the monopile	m
f	Filter size	–
$F_{D_{\text{perf}}}$	Drag force on perforated monopile	N
$F_{D_{\text{ref}}}$	Drag force on non-perforated monopile	N
H	Height of the computational domain	m
h	Mesh size	m
$h_{\min}$	Minimum mesh element size	m
k	Kernel size	–
L	Tensor of Euclidean distances to the origin of the domain	–
N_x	Domain resolution in x-direction	px
N_y	Domain resolution in y-direction	px
N_{perf}	Number of perforations	–
q	Intermediary value for the SDF of a (rotated) rectangle	m
R	Outer radius of the monopile	m
r	Minimum distance from the monopile boundary	m
s	Stride	–
t	Thickness of the monopile	m

U_{in}	Channel inflow velocity	$\mathrm{m\,s^{-1}}$
W	Width of the computational domain	m
X	Tensor of x-coordinates within the domain	m
x_c	x-coordinate of the center of the monopile within the computational domain	m
x_0	SDF rectangle offset	m
Y	Tensor of y-coordinates within the domain	m
y_c	y-coordinate of the center of the monopile within the computational domain	m

1

Introduction

This introductory chapter starts with some industry background for this project in Section 1.1. Then the concept of the perforated monopile as a promising alternative to the jacket substructure for intermediate water depths is introduced in Section 1.2. The need for surrogate modelling for the design of perforated monopile structures is explored in Section 1.3. Section 1.4 covers the main research objective and questions for this project. And finally, Section 1.5 provides answering overview of the structure of this thesis.

Contents

1.1. Industry developments

In order to reach the goals set out in the Paris climate accord to reduce the overall carbon emissions and limit the global warming a lot more renawable energy capacity is required. The offshore industry is part of the solution by providing offshore wind energy capacity. However, the growth of installed offshore wind energy capacity has to quadruple by the end of the decade to reach the goals in 2030. And almost nine folds at the end of 2050. [1, 2]

In 2021 Europe installed 3.2 GW of new offshore wind energy capacity. [3] Most of the Europe's offshore wind capacity (around 79 %) is located in the North Sea. With a lot of potential for further developments of offshore wind resources in the North sea. Offshore wind energy could provide reliable and renewable energy and has the ability to power the North Sea region. [4]

It's clear that the North Sea will remain an important area for new offshore developments in the coming decades. However, most offshore wind energy developments, that are either already built or planned, are mostly located in typically shallow water depths (and near the shore), see Figures 1.4a and 1.4c. Furthermore, not all locations are suitable for offshore wind energy developments due to various restrictions, see Figure 1.4b. For example due to shipping, fishing, cables & pipelines, oil & gas or military activities or marine protected areas. Or simply due to sight lines. Therefore, suitable locations for new offshore wind developments, especially in shallow waters and near shore, become increasingly

more difficult to still exploit. [5]

Therefore, there is a shift to develop new wind turbines in deeper waters. This also allows taking advantage of the higher (average) wind speeds at these locations, see Figure 1.4d. Higher (average) wind speeds also mean higher wind energy resources (more potential power available per area) that can be utilized. This has the potential to reduce the cost of wind energy.

For intermediary water depths in the range of about $30\,\mathrm{m}$ to $120\,\mathrm{m}$ typically jacket-like substructures are used as one of the few viable options, see Figure 1.1. Higher water depths increase the demands on the structure significantly. Jackets provide the benefit of a large lateral stiffness, lowerer scour, less material use and reduced frontal area, which reduces the wave and current load contributions to the fatigue lifetime of the structure, in comparison to the monopile.

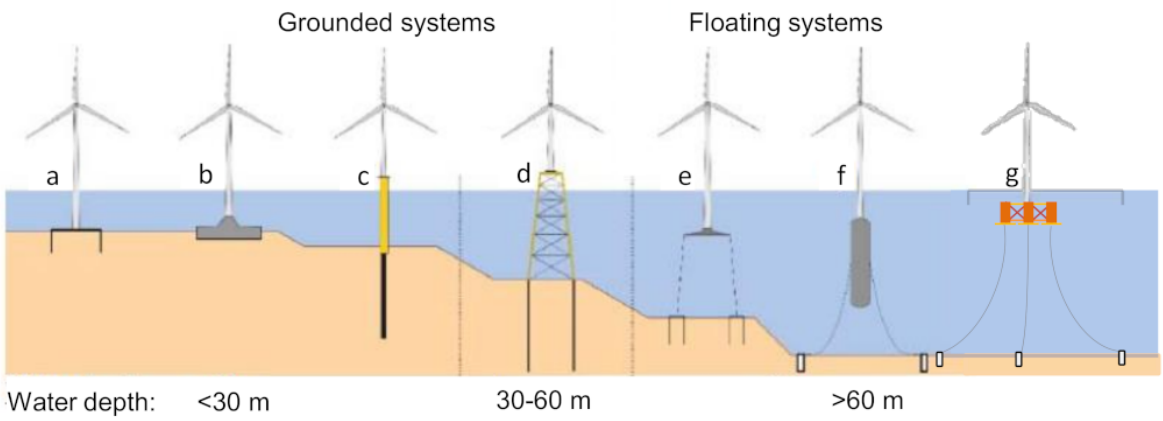


Figure 1.1: Commonly used wind turbine substructures at various water depths. [6]

However, typically jackets structures are more difficult and expensive to manufacture and install, in large series required for wind energy farms. Especially, due to the many welded connections and joints that are required. As these require a lot more laborious production.

One of the main drivers for the offshore wind energy is reducing the Levelized Cost of Energy (LCoE). The global weighted-average LCoE has dropped by 48% between 2010 and 2020, from 0.162 to 0.084 USD/kWh. With a further yearly reduction of 9% each year. [7] To continue this trend, remain competitive relative to other energy sources (especially fossil fuels) and be less dependent on government subsidies, new developments are needed to further reduce costs.

Currently, jackets have a lower LCoE at intermediate water depths, see Figure 1.2. With the share of the substructure and foundations for bottom-founded structures typically contributing around 12.6%, see Figure 1.3, to the LCoE. However, large diameter monopiles have the potential to be less costly to the jacket structure, at these intermediary water depths.

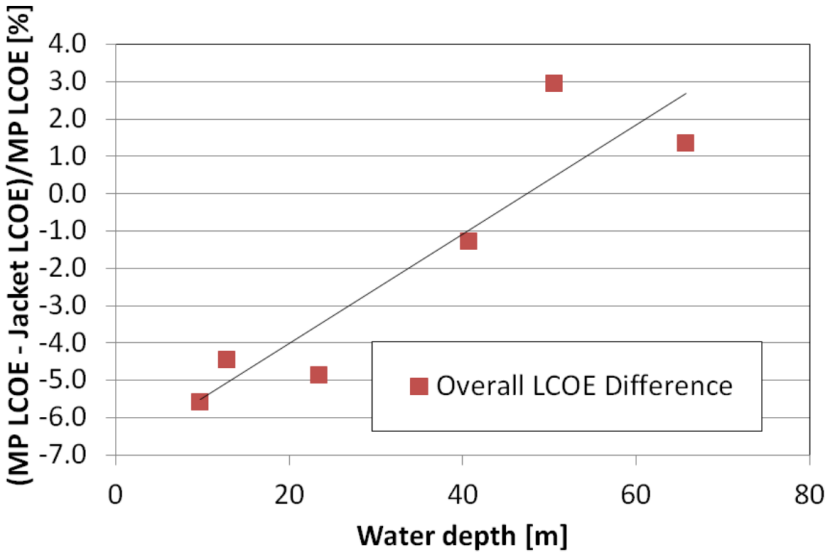


Figure 1.2: The influence of the choice between jacket and monopile substructures on the LCoE as a function of water depth. [8]

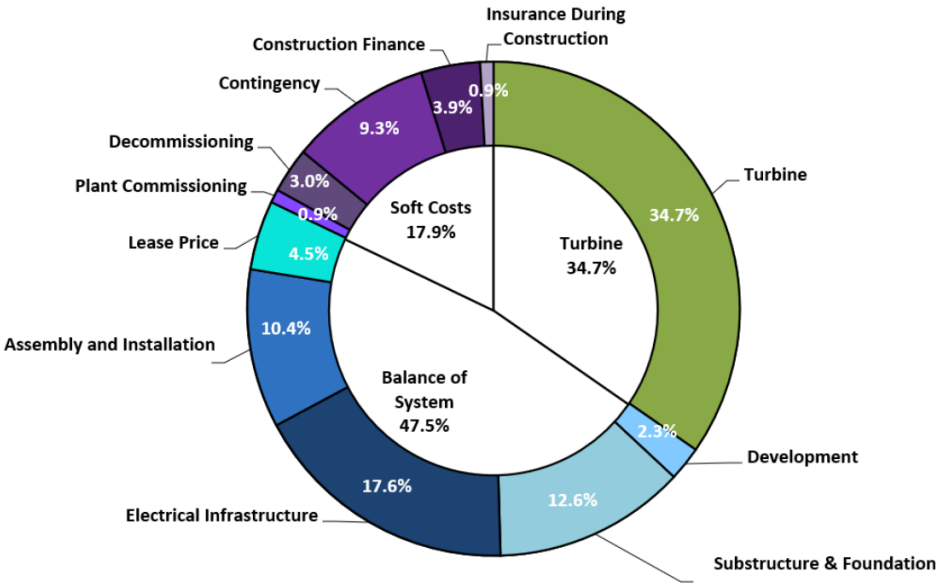
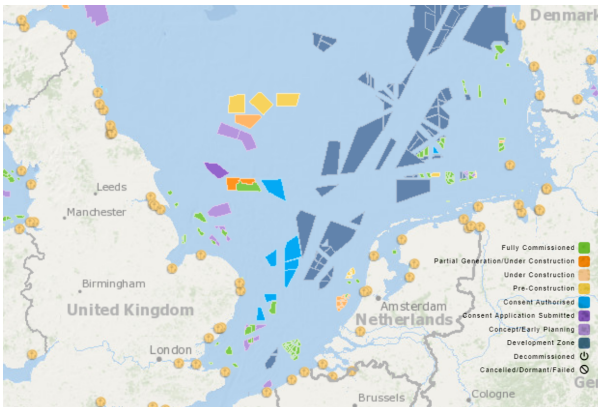
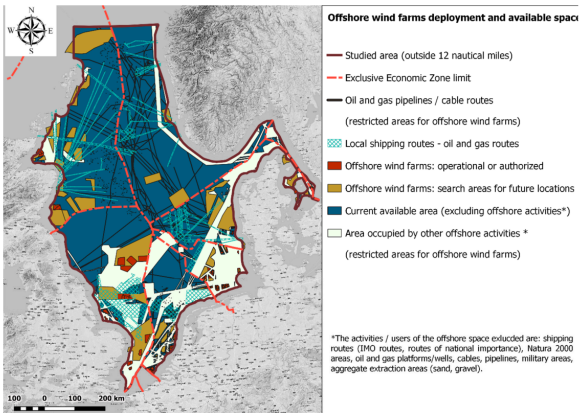


Figure 1.3: Typical CAPEX breakdown for a monopile at around 34 m. [9]

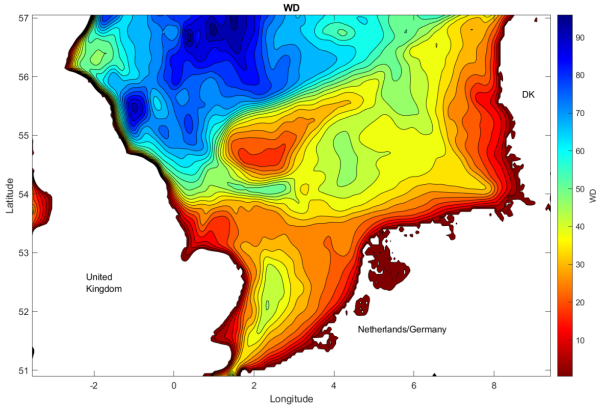
Monopiles, which are commonly used already at shallow water depth up to 30 m (see Figure 1.1), have several advantages over the jacket structure. Monopiles will maintain a relative easy installation and manufacturing process. Moreover, there is a lot of experience with monopiles already in the industry, which can be further utilized. However, the design of these monopiles will be dominated by the fatigue wave loads. And will be constrained by manufacturability constraints.



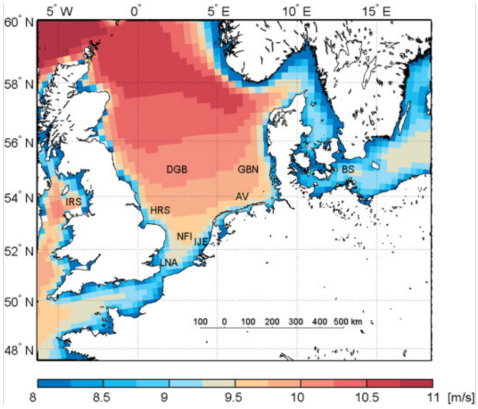
(a) Plannend and realized developments. [10]



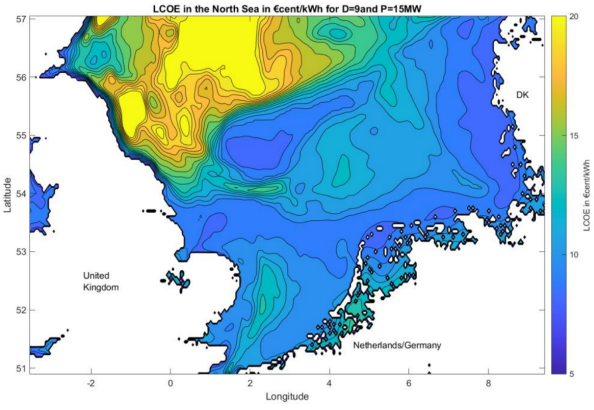
(b) LCoE [5]



(c) Water depth [11]



(d) Mean wind speed at 100 m. [12]



(e) LCoE [11]

Figure 1.4: Maps of among others the current (planned) wind turbine developments, water depth and LCoE for the North Sea.

1.2. Perforated monopiles

To overcome these challenges hybrid designs are considered as an alternative, which would combine the benefits of monopile and jacket substructures.

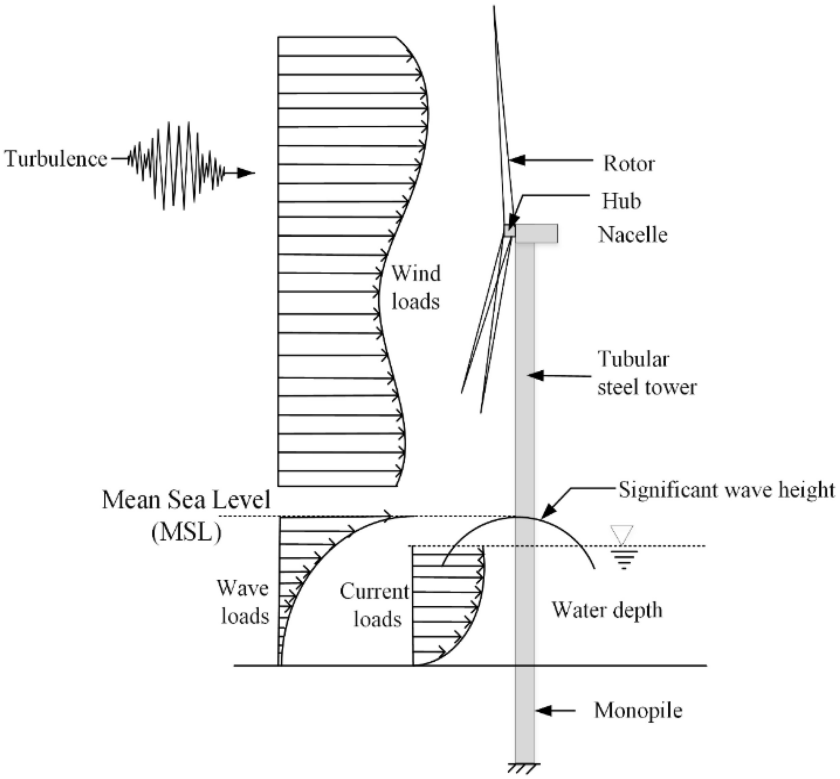


Figure 1.5: Loading cases on a wind turbine and the substructure. [13]

On the wind turbine (predominately) three important loading cases are presented, see Figure 1.5. The (irregular) wind load on the wind turbine (rotors and tower structure), the current and wave loads on the monopile. Especially, the wave loading is high in the splash zone (at the water surface level).

The exact magnitude of these three loading times will be a function of the time of the year, sea state, exact location in the North Sea and many other factors. But some time-averaged mean values for the mean and maximum wind speed, significant wave height and current speed for the North Sea can bee seen in Figure 1.6.

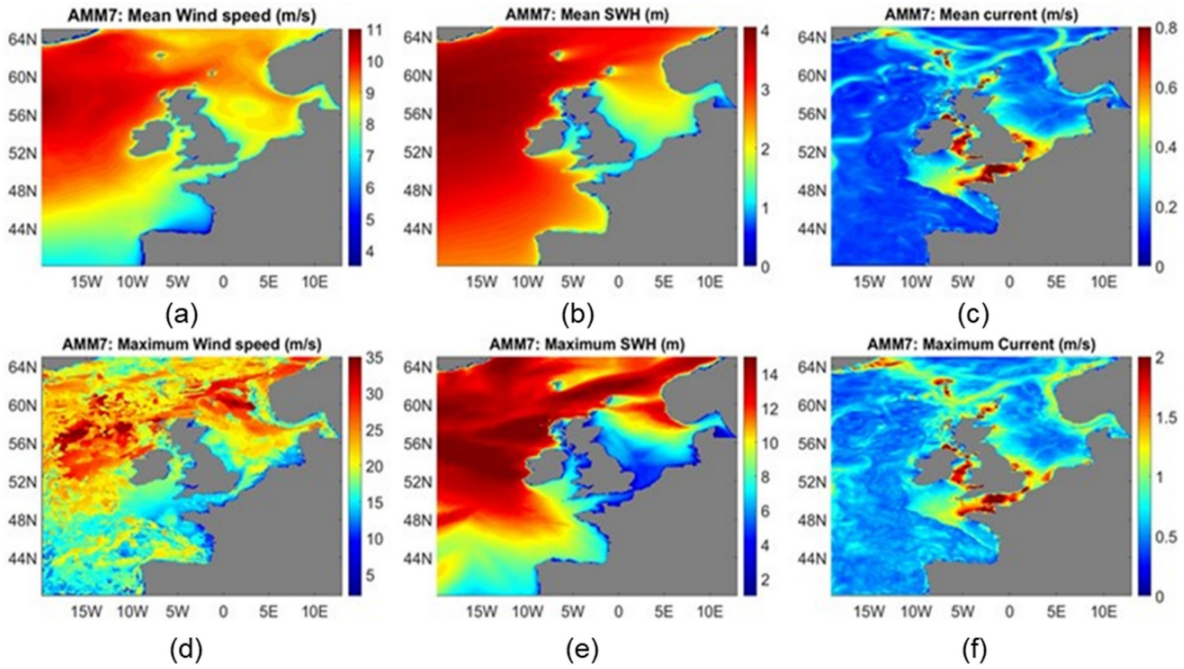


Figure 1.6: Temporal mean values for the mean and maximum wind speed, significant wave height and current speed at the North Sea. [14]

Furthermore, the time-dependent wind and wave loading are the primary drivers behind the dynamic response of the monopile. Especially, the deflection of the tip of the (wind turbine and) monopile interacts with the fluid. This Fluid Structure Interaction (FSI) can have great influence on the fatigue life of the structure. [15]

The main limiting factor for the (continued) use of monopiles in intermediate water depths is the fatigue limit state due to the wave loading. Therefore, these concepts try to mitigate the wave loading on the structure, which are the highest near the splash zone. This can be done either by shielding the monopile or by reducing the frontal (projected) area around the splash zone.

For the latter, one such concept is the Hybrid Monopile, see Figure 1.7a, which transitions from a monopile to a jacket-like structure near the splash zone. [16] Another concept is the perforated monopile, illustrated in Figure 1.7b. The perforated monopile introduces a few holes near the splash zone to reduce the frontal area and allow some discharge through the monopile. [17]

(a) Example of a hybrid monopile. [16]

(b) Example of a perforated monopile. [17]

Figure 1.7: Alternatives to jacket-like structures at water depths up to 120 meters.

Both concepts provide similar benefits, but it's clear that the perforated monopile would be much easier to manufacture in contrast to the hybrid monopile. As only, a relative simple, extra manufacturing process has to be added to the normal production of a monopile. The perforated monopile design will likely be overall cheaper and a better design (lower LCoE). Therefore, this research will focus on the perforated monopile.

The perforated monopile provides several benefits over the traditional non-perforated monopile.

As stated above, the main benefit of the perforated monopile is that it improves the fatigue life by reducing the wave loading near the splash zone. This is accomplished by reducing the (projected) frontal area and altering the flow by allowing some discharge through the monopile.

The perforations will also alter the aforementioned FSI effects. By reducing the wave loading near the tip of the monopile. And the tip deflection will also affect the flow around and through the perforated monopile. With the dynamic response due to FSI having an influence on the fatigue life of the monopile. Depending on the exact dynamic properties of the monopile structure it can be expected that by reducing the wave load magnitude due to perforations the dynamic response and FSI will be reduced.

Furthermore, allowing some discharge through the monopile, due to the perforations, at the location where the incoming flow velocities are the highest, it is expected that this could reduce the amount of local scour, see Figure 1.8. [18, 17].

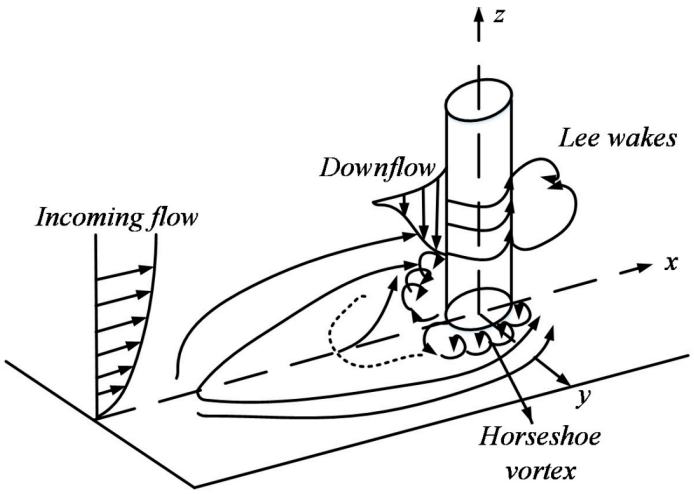


Figure 1.8: Schematic showing flow structures around a monopile, which causes scour around the seabed. [18]

While the perforations will cause a reduction in the increased velocities around the monopile and a change in the wake vortices, which causes scour. However, the perforations will (likely) be too far from the seabed to have enough effect on the development of scour, but the perforations could have significant effect on the downflow.

The downflow is developed by the large vertical pressure gradient, due to the stagnation pressure being higher at higher flow velocities. Reducing this pressure gradient by allowing discharge, at the location where the stagnation pressure is the highest, the downflow can be reduced in magnitude. This also reduces the associated horseshoe vortices. And therefore can reduce the amount of expected scour. Reducing the development of the downflow as a scour mitigation measure could also be achieved by for example installing a 'collar' around the monopile, which obstructs the downflow [19].

Additionally, the allowed discharge mitigates water acidification and hydrogen sulphide formation, which normally would occur in non-perforated (sealed) monopile, by allowing free circulation of ambient seawater. This reduces damage from corrosion, extends the service life of the structure and reduces the need for active corrosion protection on the interiors. Furthermore, as a similar environment is kept for both interior and exterior surfaces the corrosion is more predictable. And similar corrosion protection using sacrificial cathodic protection can be used. [20]

Finally, perforated monopiles can provide a favourable environment for marine life. This would be a net gain to enhance the regional ecosystem. And has the potential to economically benefit the seafood industries. [20]

The present research on perforated monopiles is still sparse. [17] conducted several (scaled) wave flume experiments to determine the wave load reduction potential of perforated monopiles. [21] continued by assessing the effect on the limit states and the fatigue damage reduction. Finally, [20] conducted field experiments to study the potential benefits for the corrosion mitigation and benefits for marine life.

However, no research has yet been conducted to try to optimize the design of a perforated monopile, which is a crucial step for the practical application of this concept.

1.3. Surrogate models

Figure 1.9 considers a simplified overview of the steps required for design optimization.

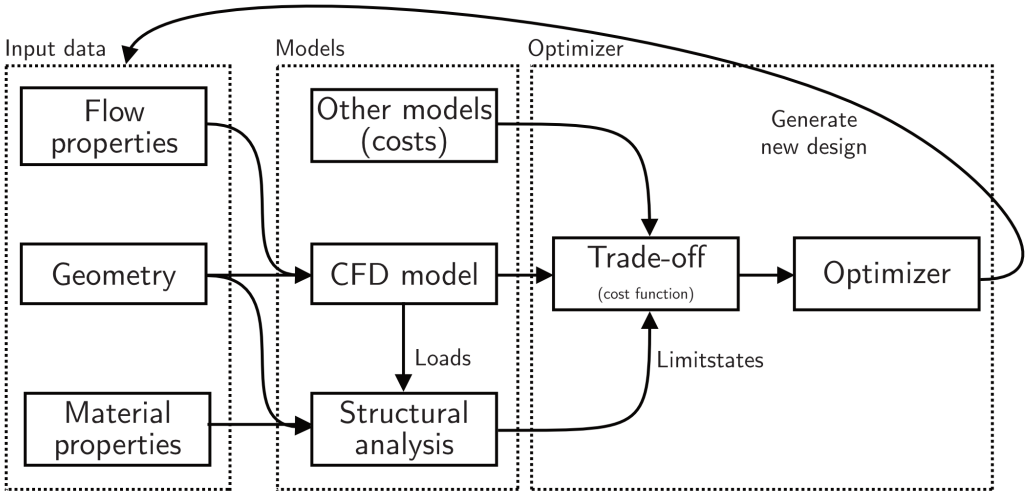


Figure 1.9: Simplified design optimization steps.

1.3.1. Computational Fluid Dynamics

Note, that to determine the loads on the structure instead of CFD simulations also either physical simulations (experiments) or analytical models could be used. However, physical simulations (in the quantity) needed for the design optimization will require more time and be more expensive than CFD simulations. And an analytical (simple surrogate) model for the loads on a perforated monopile doesn't exist yet in literature to be used. Therefore, CFD simulations are used in practice for this purpose.

For the (numerical) CFD simulations different methods could be used. Such as the Finite Element Method (FEM), Finite Difference Method (FDM) or the Finite Volume Method (FVM). The main challenge is that when you want to incorporate the flow fields from the CFD simulation the result flow fields should be directly mappable to the input (geometry) used in the surrogate model.

As commonly structured (grid like) input data is used in Deep Learning (DL) and considering the perforated monopile is round, the CFD method should reconcile the problem of (possible) cut cells. For example, by using techniques like the Shifted Boundary Method [22] or using a conformal mesh and interpolating the results to a structured grid.

Furthermore, the design optimization steps assume solving the CFD simulations and structural analysis are independent and one-directional from each other. However, when FSI (Fluid-Structure Interaction) is of importance this assumption can't be made. In this project, the effect of FSI (tip deflection of the monopile) is neglected and the focus is on the creation of a surrogate model for the CFD part. As FSI structure would greatly complexity the problem and will require a lot more computation resources. The

structural analysis of the perforated monopile, fatigue life assessment, design optimization and trade-off is out of the scope of this project. However, some research already has been done on these topics in [21].

1.3.2. Advantages and applications of surrogate models

Keeping the above two considerations in mind and returning to Figure 1.9, there is a major bottleneck to evaluate a lot of design fast. The evaluation of high fidelity CFD models, that simulate turbulent flows around complex geometries, can be very expensive in terms of computing resources required. Especially, the demands it places on the CPU and disk storage space. These CFD models could take hours, or even days to evaluate just one design.

For design optimization problems, where many (parametric) geometries and possible varying input conditions need to be considered, there is a clear demand for faster CFD models. Surrogate modelling can replace the slow CFD model with a much faster model at the cost of some (limited) reduction in accuracy. Especially, in the (early) design phase this loss of accuracy is acceptable. After, (a few) promising designs have been found they can be validated using the full (high fidelity) CFD model.

Outside the use of surrogate models for design optimization, applications includes (among others):

- Design optimization
- Design space exploration
- (active) flow control
- Real time applications
- Inverse problems
- Uncertainty propagation (sensitivity analysis)

1.3.3. Surrogate model types

Different types of surrogate models can broadly be divided in two categories.

Physics-based surrogate models, which for example include the Reduced Basis (RB), Proper Orthogonal Decomposition (POD) technique and Reduced Order Models (ROM) techniques, these aim to simplify the physics involved. [23]

And data-driven surrogate models, that uses response surface modelling to fit an approximated function to the data. These models involve creating an expensive dataset (up front) of input and desired output data (in the ‘offline phase’).

However, data-driven surrogate model don’t require physics to be solved in the model once the surrogate model is trained. Therefore, these models will (generally) be much faster than physics-based surrogate models. For applications where a model is evaluated many times this up front cost is overall a worthwhile investment in the long run.

1.3.4. Machine learning

The basis of (supervised) data-driven surrogate modelling is that an approximate function is searched using a certain machine learning method. Wherein, the approximate function is a mapping from some input to some output. For example to label some output based on the input (classification) or the predict a value based on some input (regression). To construct this function samples, examples of input and corresponding (desired) output is used to (‘train’) fit the approximate function to the data.

The main challenges in construction the data-driven surrogate model is two folds. Getting a large enough volume and good quality of samples to train the model with. And to use the right machine learning technique and model to create the model using the data with.

In this particular problem it means finding some good representation of the geometry (and potentially flow properties) to be used input to the surrogate model. And similarly a good desired output from the (surrogate) CFD model. The surrogate model technique should be able to handle the complex geometry and underlying (non-linear) physics of the problem.

Different machine learning (ML) techniques exists to create data-driven (regression) surrogate models. These include linear regression models, Support Vector Machines (SVM), Random forest or K nearest Neighbors (KNN).

However, this research will focus on the application of deep learning (DL) using a Convolutional Neural Network (CNN) to create the surrogate model. Research has shown that the benefit of using deep learning for highly non-linear problems that learns from representations of (input) geometry. And its application to create surrogate CFD models. [24]

Using DL has the advantage of it being able to (potentially) learn (slightly) more complex geometries, than is given in the dataset (more ability to generalize). Furthermore, using the spatial input (representation of the geometry) data, more complex features can be learned. While the more traditional machine learning methods will be constrained to the features (geometry parameters) that where used to train the model.

1.4. Research questions

The Offshore wind energy sector is increasingly moving towards deeper waters. At intermediate water depths from in the range of 30 m to 120 m often jacket-like structures have been used as the only realistic option. However, novel concepts like the perforated monopile are currently researched to replace the jacket structure.

Perforated monopiles show promise for the continued use of monopile structures at these water depths. As currently manufacturability and the fatigue limit state due to the high contribution of wave loads on the fatigue lifetime on monopiles. There is potential to lower the Levelized Cost of Electricity (LCoE) by reducing manufacturing and installations costs compared to jacket structures.

For the design (optimization) of perforated monopiles many expensive CFD model evaluations need to be conducted. There is a clear need to create novel new surrogate models to replace the CFD calculations during the (early) design phase. This research will focus on creating a surrogate model to predict the hydrodynamic load reduction by introducing perforations in the monopile based on the geometry of the monopile.

Therefore, the main research question of this thesis is formulated to the following:

> 'How to best develop a machine learning-based surrogate model for the characterization of hydrodynamic loads on perforated monopiles?'

In order give an answer to the main research question it is subdivided into the following sub-questions:

1. Which (geometric) inputs are of importance for the design of a perforated monopile?
2. What type of input should be used in the CNN model to achieve the best performance?
3. What CNN architecture should be used for this surrogate model?
4. How accurate is the surrogate model for determining the load reduction of the perforated monopile?
5. How much faster is the surrogate model to evaluate one design compared to the full CFD model?

1.5. Thesis outline

In order to reach the research objective of developing a surrogate model for the characterization of hydrondynamic loads on perforated monopiles, several steps need to be taken. This thesis is subdivided in two major parts. The first part considers the creation of a dataset in order to train the surrogate model. The second part will go into process of creating the surrogate model using deep learning.

In Chapter 2, a model is setup up for the flow over a perforated monopile. In this chapter a simplified (2D) geometry of a perforated monopile is considered in Section 2.1. Using the geometry of a perforated monopile a representation is created that can be used as input to CNN models in Section 2.2. In addition, a CFD model for the flow around a perforated model using the Finite Element Method (FEM) is developed in Section 2.3.

Using the model a dataset is created in Chapter 3, which is needed to train a surrogate model. Furthermore, a description of the processes used in creating the dataset is given in Section 3.1. Finally, in Section 3.2 the dataset that is used during this research is discussed.

In Chapter 4, a brief introduction to deep learning using Convolutional Neural Networks (CNNs) is given. Then a model architecture is proposed for the surrogate model to predict the hydrodynamic load reduction factor for perforated monopiles. Among others, the model training process, dataset preparation and choice of hyperparameters for the network is disccused. For five different model input options a CNN model is created.

In Chapter 5, the result for the hyperparameter optimization of the five models are given. And the performance of the five models is evaluated and discussed. The performance of the models are compared to the full CFD model based on two metrics. The accuracy of the surrogate model to predict the correct reduction factor within a certain error margin. And the expected speed-up by using a surrogate model in comparison to the full CFD model.

Chapter 6 presents the conclusions of this research. The relevance of this research and applicability for use in practice is discussed. Finally, recommendations are made in Chapter 7 for further research and possible avenues to explore to improve the surrogate model further.

2

Modelling

In order to create a surrogate model for the characterization of the hydrodynamic load using machine learning first a dataset has to be created. The dataset should consist of many samples providing examples of the load on the perforated monopile for different configurations.

The dataset will consist of three distinct parts:

1. A geometric representation of the perforated monopile configuration, which will be used as input to the machine learning model.
2. Results from a CFD model giving the numerical result for the hydrodynamic load for the perforated monopile configuration, which will be used to train the output of the machine learning model.
3. Metadata containing the geometric parameters of each sample. And any dataset parameters, such as the resolution and name of the dataset. The metadata will be mostly used for reporting purposes, such as the plotting of results.

Each of these parts are derived from a common parametric description of the perforated monopile geometry. An overview of the dataset creation process can be seen in Figure 2.1.

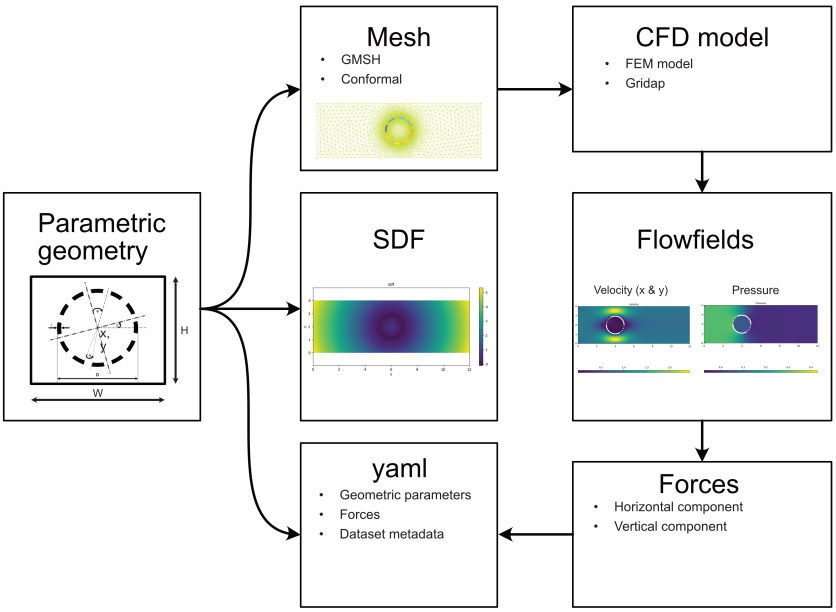


Figure 2.1: Overview of the dataset creation process.

This chapter deals with the setup of the (CFD) model for the flow over and through a perforated model. It starts with parametric description of the perforated monopile in Section 2.1. In Section 2.1 the geometric representation for the model input is given. In Section 2.3.4 the mesh generation using GMSH is explained, which is used as input for the CFD model created using Gridap in Section 2.3. Finally, the model is then used in Chapter 3 in order to create the dataset used for the surrogate model.

Contents

2.1. Geometric description of a simplified perforated monopile

While the flow around and through the perforated monopile is fundamentally 3D in nature the perforated monopile geometry has been idealized in 2D. The reason for this is practical in nature, as 3D would require a lot more computer resources, than what was available to work with.

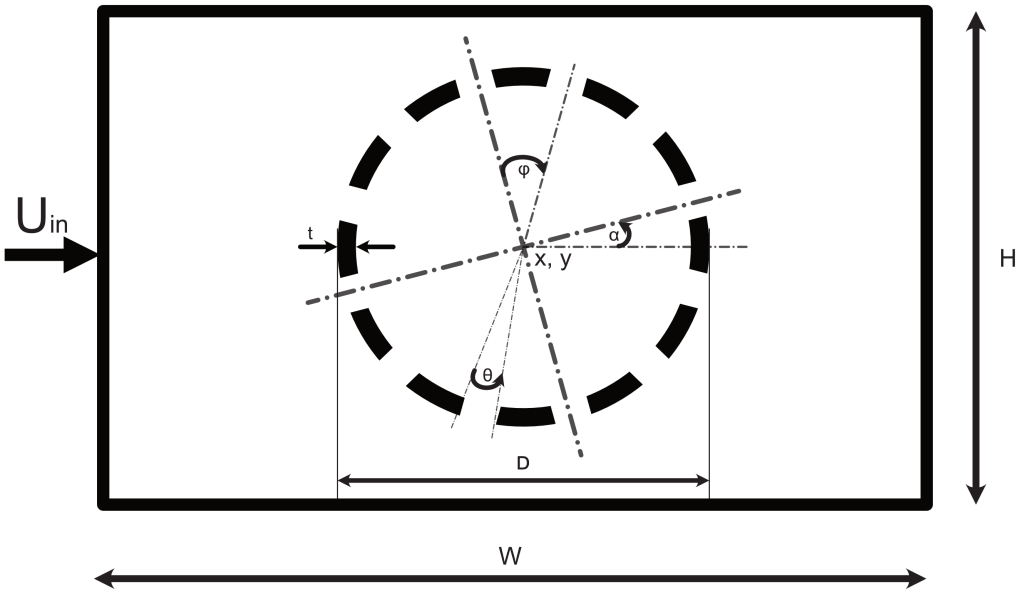


Figure 2.2: Simplified geometric description of a perforated monopile in a channel flow.

The geometry of the perforated monopile can be seen in Figure 2.2. The geometry consists of several parts.

First, the computational domain is a rectangular channel flow with width W and height H and the origin located at the left bottom corner of the domain. To ensure the pixel resolution $\mathrm{px\,m^{-1}}$ will be equal in both direction to avoid any squeezing of the input geometry in the CNN model the width of the domain is assumed to be twice the height ($W = 2H$).

Second, a circular disk (monopile) is placed at location (x_c, y_c) in the computational domain, with (outer) diameter D and thickness t.

And finally the perforations are assumed to be uniformly distributed with an angle ϕ between each perforation. The perforations are assumed to have a rectangular shape and drilled through the center of the monopile. The width of the perforation is given by the angle θ. The whole monopile is rotated by an angle α (the 'angle of attack') with respect to the inflow velocity.

$$\theta = \beta \cdot \phi \tag{2.1}$$

The perforation width angle θ is made independent of the other geometry parameters by defining it as a fraction β of the 'available' angle ϕ between two consecutive perforations, see Equation (2.1). This is done to easily create a dataset and completely tile the parametric space with unique samples. An additional benefit is, that as a uniform perforation distribution is assumed, the perforation width fraction β equals the porosity of the perforated monopile. With the porosity defined as the factor of frontal area reduction due to perforations with respect to the non-perforated monopile, see Equation (2.2). [25, 21]

$$\beta = 1 - \frac{A_{\mathrm{perf}}}{A_{\mathrm{tot}}} \tag{2.2}$$

As can be seen in Figure 2.3, the geometry can easily be constructed using Constructive Solid Geometry (CSG) by combining basic shapes with boolean operators. [26] This basic recipe will be used in both Section 2.2 to create the SDF and in Section 2.3.4 to construct the computational mesh.

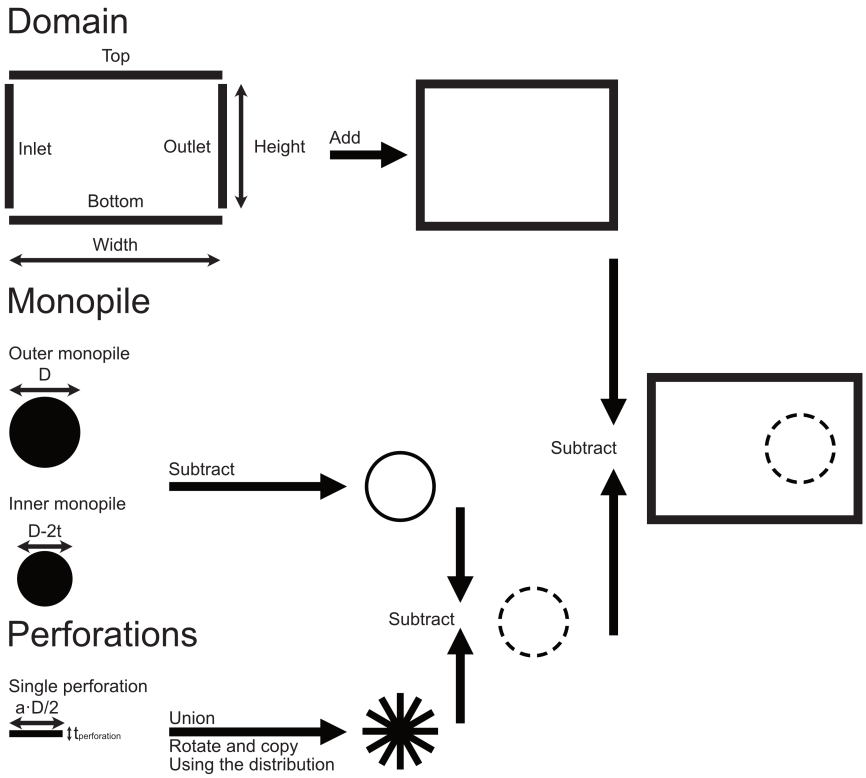


Figure 2.3: Boolean operations to construct the geometry.

2.2. Implicit representation of geometry

Structured image-like data is often used as input and output to CNN layers. CNN models have been successfully in learning geometry representations in order to make predictions. [27] Several input options are discussed in this section.

The geometry of the perforated monopile is implicitly represented using one of three options: Using a Signed Distance Function (SDF) (see Section 2.2.1), the SDF decomposed in the x and y-distance components (see Section 2.2.2) and a binary encoding of the geometry (see Section 2.2.3). Additionally, the input to the CNN model can be padded by directly inputting the flow fields along the geometry representation, see Section 2.2.4.

2.2.1. Signed Distance Function

The Signed Distance Function (SDF) has often been used in literature as a CNN model input, especially for CFD flow prediction models. [24, 28, 29, 30, 31]

The SDF is a level-set method to implicitly describe the geometry. The SDF gives the minimum Euclidean distance for each point to the boundary of the geometry. With negative values of the SDF in the interior region, zero exactly on the boundary and positive values in the exterior region, see Equation (2.3). [32]

$$SDF(x,y) = \begin{cases} < 0 & \text{for } (x,y) \in \Omega^c \\ = 0 & \text{for } (x,y) \in \partial\Omega \\ > 0 & \text{for } (x,y) \in \Omega \end{cases} \tag{2.3}$$

The SDF has some good properties to use as the main choice in representing implicitly the geometry of the perforated monopile as the model input. Firstly, the SDF describes the geometry implicitly at each point in the domain. Each point in the domain carries information about the geometry. Secondly, the SDF is also differentiable, outside the local minimum where the $\mathrm{SDF} = 0$, and the length of the gradient has the nice property of $\|\nabla \mathrm{SDF}\| = 1$. Furthermore, more complex SDF's can be constructed from simple shapes and boolean operations (see Figure 2.3). Finally, the alternative geometry representations that are used in this thesis can be easily created from the SDF by applying a simple function on the SDF.

The SDF for the perforated monopile geometry, see Section 2.1, is created as follows. [33]

First tensors are constructed with a resolution N_x by N_y with the x-coordinates ($X(x,y)$) and y-coordinates ($Y(x,y)$) for each point in the computational domain. The origin of the mesh points is translated to the center point of the monopile $(x_c y_c)$, see 2.4.

$$X(x,y) = X(x,y) - x_c Y(x,y) = Y(x,y) - y_c \tag{2.4}$$

Then the euclidean distance to the center of the monopile (L) is calculated using Equation (2.5).

$$L(x,y) = \sqrt{\left(X(x,y)^2 + Y(x,y)^2\right)} \tag{2.5}$$

The SDF to the monopile with (outer) radius R is calculated using Equation (2.6). With R equal to half the diameter $\frac{D}{2}$ for the outer monopile and $\frac{D}{2} - t$ for the inner monopile.

$$\mathrm{SDF}_{\text{circle}}(x,y) = L(x,y) - R \tag{2.6}$$

Next, the SDF for the perforations has to be calculated. The coordinate system with the origin centred at the center of the monopile is first rotated by the 'angle of attack' α for each perforation, see Equation (2.9).

$$X(x,y) = X(x,y)\cos(\alpha) + Y(x,y)\sin(\alpha) \tag{2.7}$$
$$Y(x,y) = X(x,y)\sin(\alpha) - Y(x,y)\cos(\alpha) \tag{2.8}$$
$$\tag{2.9}$$

The perforation width for a perforation angle θ and (outer) monopile diameter D using the formula for the chord length of a circle, see Equation (2.10).

$$c = D\sin\left(\frac{\theta}{2}\right) \tag{2.10}$$

The SDF for a rectangle is calculated using Equation (2.12). With the formula for intermediary variable q given in Equation (2.11) with offset to the 'lower bottom point' of the rectangle x_0 and the length/width l.

$$q_x(x,y) = \left|X(x,y) - x_0\right| - \frac{l}{2} \tag{2.11}$$

$$\mathrm{SDF}_{\text{rectangle}}(x,y) = \left\|\max\{q_x,0\}, \max\{q_y,0\}\right\| + \min\left\{\max\{q_x,q_y\},0\right\} \tag{2.12}$$

Finally, the SDF for the monopile and rectangular perforations can be combined using boolean operations, following the recipe given in Figure 2.3. With the SDF union operator given in Equation (2.14).

$$\mathrm{SDF}\left(\mathrm{SDF}_1, \mathrm{SDF}_2\right) = \max\left\{-\mathrm{SDF}_1, \mathrm{SDF}_2\right\} \tag{2.13}$$

And the SDF difference operator given in Equation (2.13).

$$\mathrm{SDF}\left(\mathrm{SDF}_1, \mathrm{SDF}_2\right) = \min\left\{\mathrm{SDF}_1, \mathrm{SDF}_2\right\} \tag{2.14}$$

An example of the SDF output for a perforated monopile can be seen in Figure 2.4.

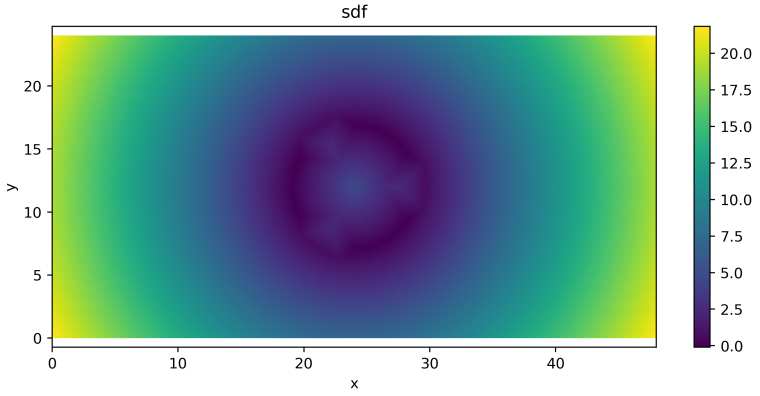


Figure 2.4: Example of the SDF output for a perforated monopile ($D = 10, N_{\text{perf}} = 3, \alpha = 0, \beta = 0.5$)

2.2.2. Distance Function

The first alternative input geometry representation is decomposing the SDF in the x- and y-distance components.

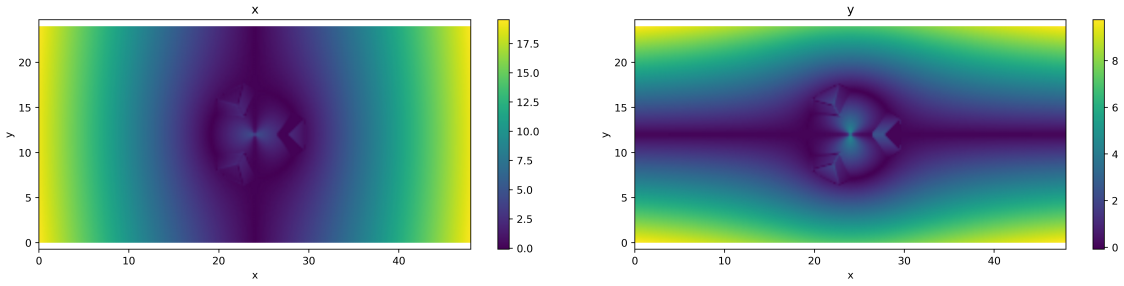
These can be easily calculated by applying a transformation function on the SDF, see Equation (2.15). The gradient for each direction is divided by the length of the normal vector and multiplied by the SDF to get the distance component for each direction. The absolute value of the gradient in each direction is used, as the correct sign of the Dist component is determined by the sign of the SDF.

The Dist function retains most properties of the SDF as described in Section 2.2.1. However, the Dist function will be non-smooth along one the domain center axis. However, this is likely not really a problem. As the input goes through many convolutional layers, which the result can be either smooth or non-smooth afterwards, depending on the kernel weights used. Furthermore, the SDF can rather easily be obtained from the Dist and vice versa. Of importance is mostly to have a smooth output to train a model with high accuracy.

The transformation function is implemented as a Torch module to efficiently and fast perform this transformation during training. [34]

$$\mathrm{Dist}_x(x,y) = \mathrm{SDF}(x,y)\frac{\left|\frac{\partial\,\mathrm{SDF}(x,y)}{\partial x}\right|}{\left\|\nabla\,\mathrm{SDF}(x,y)\right\|_2} \tag{2.15}$$

An example of the Dist components output for a perforated monopile can be seen in Figure 2.5.



(a) Dist x-component

(b) Dist y-component

Figure 2.5: Example of the Dist output for a perforated monopile ($D = 10, N_{\text{perf}} = 3, \alpha = 0, \beta = 0.5$)

2.2.3. Binary

The binary representation of the geometry is equal to zero if the center of the pixel is within the bounds of the perforated monopile geometry ($\text{SDF} < 0$) and one everywhere else ($\text{SDF} \geq 0$), see Equation (2.16).

The binary representation has been successfully used in [30, 24]. Though, in contrast to the SDF and Dist inputs the binary representation only provides information about the geometry locally and no distance information to the boundary. Especially, for thin-walled structures that are considered in this project there will be relatively few pixels provide information about the geometry. As the SDF for the center of the pixel has to be negative (fully inside the monopile geometry). Such sparsity might make the binary representation unable to accuracy describe the geometry in this application.

$$\text{Binary}(x, y) = \begin{cases} 1 & \text{if } \text{sign}\left(\text{SDF}(x, y)\right) \geq 0 \\ 0 & \text{if } \text{sign}\left(\text{SDF}(x, y)\right) < 0 \end{cases} \tag{2.16}$$

An example of the Binary output for a perforated monopile can be seen in Figure 2.6.

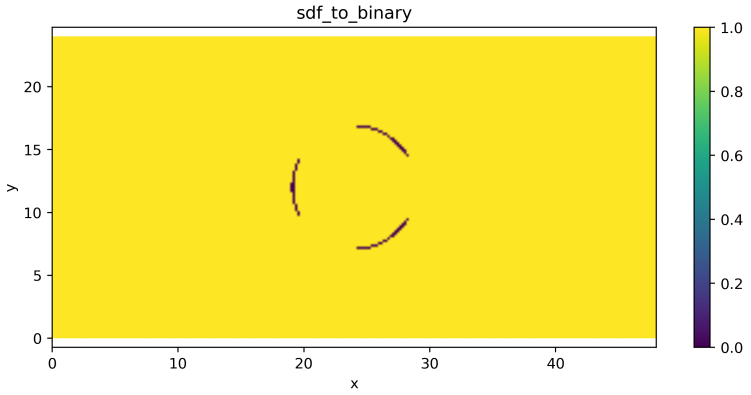


Figure 2.6: Example of the Binary output for a perforated monopile ($D = 10, N_{\text{perf}} = 3, \alpha = 0, \beta = 0.5$)

2.2.4. Other representations

Additionally, the represention of the geometry of the input is aided by stacking the flow fields on top of it for some models considered.

Furthermore, more information could be provided to the model. For example, a Flow Region Channel (FRC), which analogous to the binary representation provides information about the obstacle, but also encodes information about the boundary conditions (Fluid inflow speed, (no)-slip walls and outlet boundary condition). [31, 24, 35] An example of the FRC (geometry) input representation can be seen

in Figure 2.7. As the boundary conditions are assumed to be constant for each sample in this project such an additional channel provides no extra information to the model to learn from.

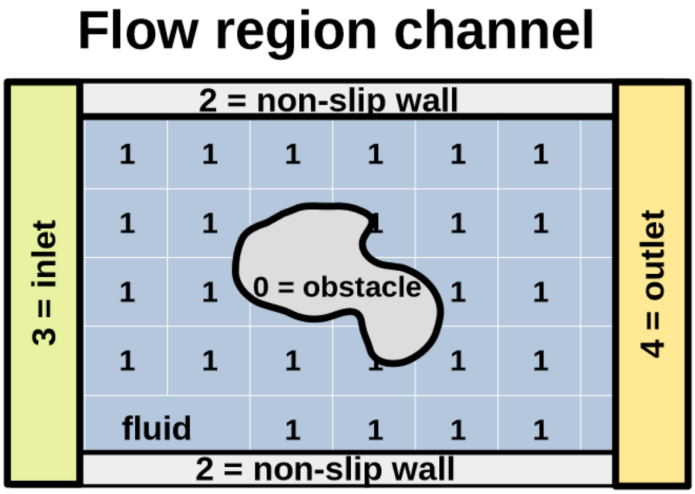


Figure 2.7: Example of a Flow Region Channel (FRC) input representation. [31]

Similarly, also the SDF to the walls could be determined and included as additional channels. [35]

2.3. Computational Fluid Dynamics

In order to determine the hydrodynamic load on the perforated monopile samples a lot of CFD models have to be evaluated to get the flow fields.

The method of choice is to implement a CFD model using the Finite Element Method (FEM) in Gridap. Gridap is a Finite Element (FE) solver written in the Julia programming language. Gridap is a fast framework for solving PDEs utilizing Julia's JIT compiler. Using Gridap's expressive API the weak form of PDEs can be written in a form very close to the mathematical notation. [36, 37]

The CFD model consists of several steps. First, the choice of flow model is discussed in Section 2.3.1. Then, the strong form of the governing equations and boundary conditions are given in Section 2.3.2. In Section 2.3.3 the weak form of the governing equations are given. And in Section 2.3.4 the (conformal) FEM mesh generation process is given. The output of the (forces) of the CFD model is discussed in Section 2.3.5. Finally, the interpolation of the flow fields from a conformal mesh to a structured grid is discussed in Section 2.3.6.

2.3.1. Choice of flow model

For a monopile with a diameter of $10\,\mathrm{m}$ and a flow speed around $1\,\mathrm{m\,s^{-1}}$ the expected Reynolds number (see Equation (2.17)) for the flow around the monopile is in the order of 1×10^7. Therefore, the flow is fully turbulent, and the inertial forces dominate the viscous forces (see Figure 2.8b). A turbulent CFD model, such as the (steady) Reynolds-Averaged Navier-Stokes, should be used. [38]

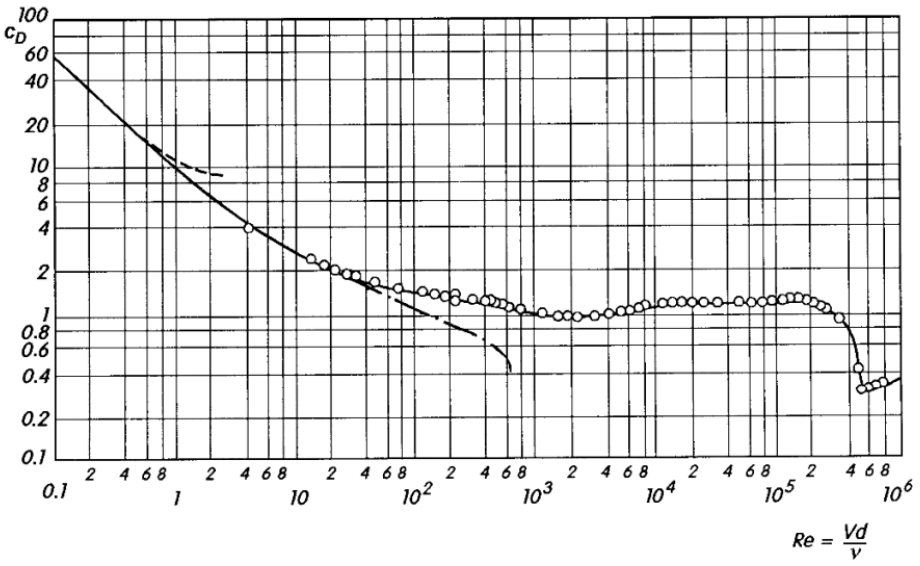
$$Re = \frac{\rho U D}{\mu} \tag{2.17}$$

However, such models are slow to evaluate as they require non-linear solvers. This requires a lot of computing resources to create a dataset of thousands of samples. As computing resources were limited for the CFD calculations a simpler model is used. Therefore, the Stokes 'creeping flow' equations are used, which is only valid for very low Reynolds numbers. A flow regime (see Figure 2.8b) where the inertial forces are negligible compared to the viscous forces and no turbulence is taken into account. [38]

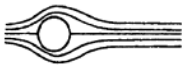
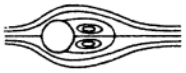
With more computing resources available the flow model can easily be upgraded. Though, some results might change the general approach for the constructing of a surrogate model using CNN's remains the

same.

Furthermore, the flow fields from the Stokes flow model could potentially be used as an initial condition for a more accurate CFD model. Since the resulting flow fields will be close to each other, the convergence of the non-linear solver will be faster. The number of iterations will be greatly reduced in order to converge to the steady state solution. [24]



(a) Drag coefficient of a cylinder

Reynolds number regime	Flow regime	Flow form	Flow characteristic
$Re \to 0$	Creeping flow		Steady, no wake
$3-4 < Re < 30-40$	Vortex pairs in wake		Steady, symmetric separation
$\frac{30}{40} < Re < \frac{80}{90}$	Onset of Karman vortex street		Laminar, unstable wake
$\frac{80}{90} < Re < \frac{150}{300}$	Pure Karman vortex street		Karman vortex street
$\frac{150}{300} < Re < \frac{10^5}{1.3 \cdot 10^5}$	Subcritical regime		Laminar, with vortex street instabilities
$\frac{10^5}{1.3 \cdot 10^5} < Re < 3.5 \cdot 10^6$	Critical regime		Laminar separation Turbulent reattachment Turbulent separation Turbulent wake
$3.5 \cdot 10^6 < Re$	Supercritical regime (transcritical)		Turbulent separation

(b) Flow regimes for different Reynold numbers.

Figure 2.8: Reynold flow regimes around a circular cylinder. [39]

2.3.2. Governing equations

The computational domain is illustrated in Figure 2.9. It features a rectangular channel domain with fluid domain (Ω_{fluid}) no-slip boundary conditions on the sides walls ($\Gamma_{\text{top}}, \Gamma_{\text{bottom}}$) and monopile boundary (Γ_{monopile}), a parabolic input velocity at the inlet (Γ_{inlet}) and a traction free condition at the outlet (Γ_{outlet}).

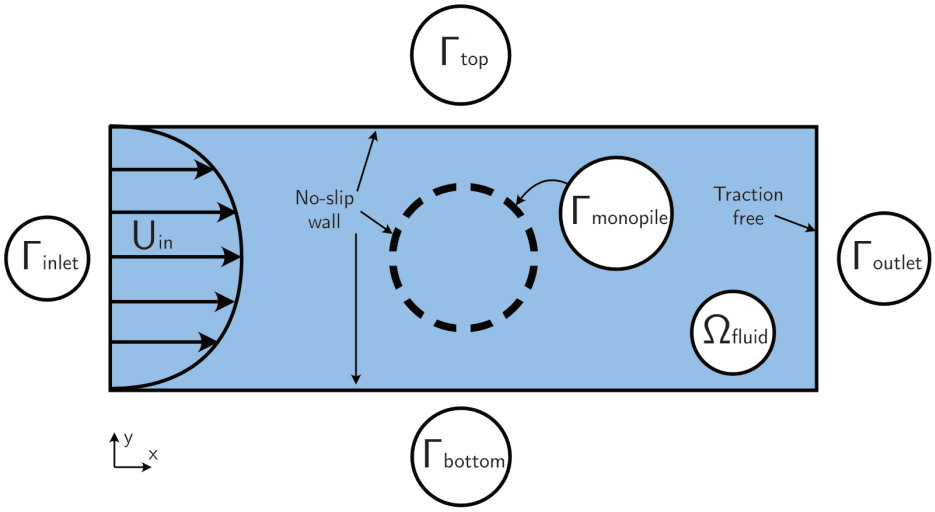


Figure 2.9: Overview of the model domains and boundaries.

The strong form of the Stokes Equations (viscous, steady and incompressible flow) is given by the momentum and continuity equations. [38] The momentum equation is given in Equation (2.18) with σ the Cauchy stress and f the applied body forces (assumed to be zero).

$$\nabla \cdot \sigma + f = 0 \tag{2.18}$$

And the continuity equation in Equation (2.19) with u the fluid velocity.

$$\nabla \cdot u = 0 \tag{2.19}$$

For the no-slip walls (on this side walls and monopile boundary) the Dirichlet boundary condition is given by Equation (2.20).

$$u = 0 \qquad \text{on } \Gamma_{\text{top}} \cup \Gamma_{\text{bottom}} \cup \Gamma_{\text{monopile}} \tag{2.20}$$

At the inlet a parabolic input velocity is enforced, see Equation (2.21), with domain height H, and U_{in} the input velocity.

$$u(x,y) = 1.5U_{\text{in}} \frac{y(H-y)}{\left(\frac{H}{2}\right)^2} \qquad \text{on } \Gamma_{\text{inlet}} \tag{2.21}$$

And the free traction condition at the outlet boundary with n the outer normal to the boundary, see Equation (2.22).

$$n \cdot \sigma = 0 \qquad \text{on } \Gamma_{\text{outlet}} \tag{2.22}$$

And the Cauchy stress is given by Equation (2.23) with σ^{dev} the deviatoric part of the stress and p the static pressure.

$$\sigma = \sigma^{\text{dev}} - pI \tag{2.23}$$

The pressure will only be solved for the deviatoric part of the stress. The stress law for the fluid is given by Equation (2.24), with μ the dynamic viscosity of the fluid and $\epsilon(u)$ is the symmetric gradient of the velocity.

$$\sigma^{\text{dev}} = 2\mu\epsilon \tag{2.24}$$

2.3.3. Weak form

The governing equations, see Section 2.3.2, that are given in the strong form. For the FEM the strong form is multiplied with an arbitrary 'test functions' ($\mathbf{v}$ and q) and integrated over the domain. After integration by parts and discarding higher order derivatives and ensuring continuity is ensured on get the weak form of the problem. Solutions to the weak form will also be a solution to the strong form of the problem.

The bilinear form for the Stokes Equations is given by Equation (2.25).

$$a = ((u,p),(\mathbf{v},q)) = \int \left(\varepsilon(\mathbf{v}) : \sigma^{\text{dev}}(u) - (\nabla\cdot\mathbf{v})\, p + q\,(\nabla\cdot u)\right) d\Omega_{\text{fluid}} \tag{2.25}$$

With the linear form given in Equation (2.26).

$$l_{\text{fluid}}(\mathbf{v},q) = \int \mathbf{v}\cdot f\, d\Omega_{\text{fluid}} \tag{2.26}$$

And the linear form for the outlet boundary condition in Equation (2.27).

$$l_{\Gamma_{\text{outlet}}}(\mathbf{v}) = \int \mathbf{v}\cdot h\, d\Gamma_{\text{outlet}} \tag{2.27}$$

The weak form is solved using Gridap on a conformal mesh, see Section 2.3.4. The velocity field is solved using Lagrangian Finite Element space with a second order interpolation and H1-conformity. And the pressure field using Lagrangian Finite element space with a first order and C0-conformity. Furthermore, the boundary conditions associated with the inlet and no-slip walls are implemented directly in the velocity trial space. While the boundary condition associated with the free traction outlet is included in the weak form. As the Stokes Equations are linear in nature the solution can be obtained by solving a linear system of equations.

2.3.4. Mesh generation

In order to solve the weak form using the FEM, as described in the sections above, a conformal has to be created. The domain as described in Figure 2.9 is subdivided into smaller discrete elements.

GMSH, which is a three-dimensional finite element mesh generator, is the tool of choice to create the meshes. There are several reasons why GMSH is the clear choice. With GMSH meshes can be created using Constructive Solid Geometry (CSG) methods using the Open CASCADE library. More complex meshes can be created using a series of boolean operations. And the recipe as described in Figure 2.3 can be used. Furthermore, GMSH has Python (and Julia) API's, which allow for an easy integration in the meshing pipeline, see Section 3.1.1. Finally, through the GridapGmsh library GMSH meshes can be directly used with Gridap and without requiring any extra work. [40, 37]

With GMSH each geometric entity needs to be assigned a 'physical tag', which specifies what part of the computational domain it represents (inlet, outlet, side walls, monopile or fluid domain). As the number of perforations is variable, the number of geometric entities created for each mesh is variable as well.

Keeping track of each geometric entity after boolean operations and assigning the proper physical tag is not trivial. To solve this problem the PyGMSH (a python package) is used, which is a higher level API on top of the GMSH Python API. PyGMSH provides further useful abstractions to make it easier to create more complex meshes in a more Pytonic way. Furthermore, it automatically keeps track of all entities, which solves the aforementioned problem. [40, 41]

To ensure a consistent and good quality of the generated meshes a custom mesh size function is used. As the flow directly around the perforated monopile geometry is of most interest a very fine mesh is required here. The mesh size through the interior of the monopile is also controlled to coarsen not too much. Far away from the monopile the mesh can quickly coarsen. A balance is struck between a fine mesh to get an accurate result in regions of interest and a coarser mesh in regions further away in order to save computation time and mesh file sizes.

The chosen mesh size function is given in Equation (2.28). In this equation h is the mesh size as a function of the radial distance r from the monopile boundary, $h_{\min}$ the minimum mesh size length (directly at the monopile boundary), a_{mesh} and b_{mesh} the mesh size function factor and exponent.

$$h(r) = h_{\min}\left(1 + a_{\text{mesh}}\left|r^{b_{\text{mesh}}}\right|\right) \tag{2.28}$$

The radial distance r from the monopile boundary is calculated using Equation (2.29). With x and y the coordinate from the bottom left origin, x_c and y_c the coordinates of the monopile center and R the monopile radius.

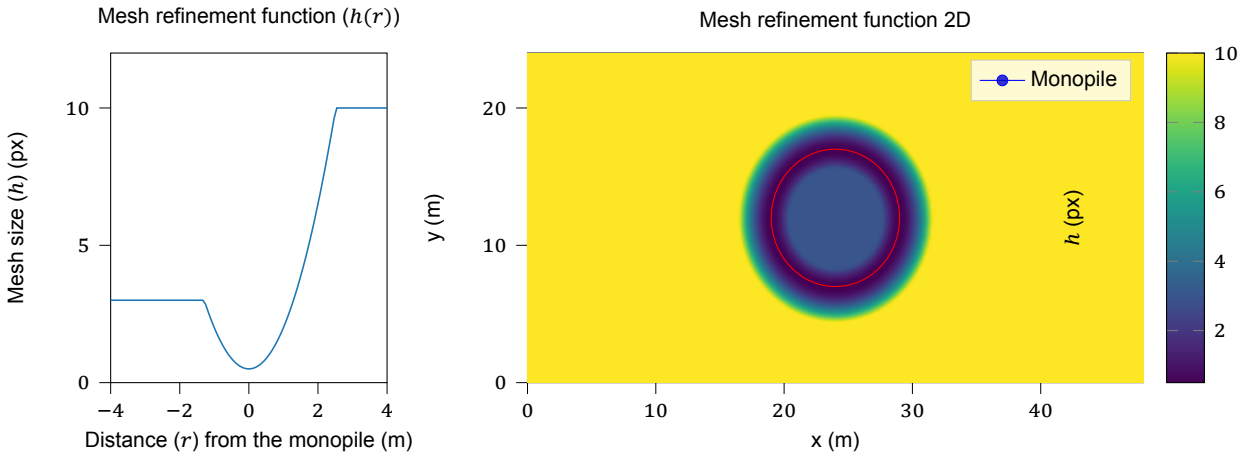
$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2} - R \tag{2.29}$$

For the generated meshes in the dataset the values in Table 2.1 are used. The mesh size parameters are given in pixels with respect to be relative to the resolution of the input geometry and flow fields in the dataset.

Table 2.1: Chosen mesh size function parameters.

Parameter	Value	Unit
Minimum mesh size ($h_{\min}$)	0.5	px
Inside monopile circumference	3	px
Maximum mesh size	10	px
Decay factor (a_{mesh})	3	–
Decay exponent (b_{mesh})	2	–

Using the values from Table 2.1 the mesh size function is plotted in Figure 2.10a and Figure 2.10b.



(a) Mesh refinement function. (b) Mesh refinement function 2D.

Figure 2.10: Mesh refinement function with parameters from Table 2.1.

Finally, an example of a generated mesh for a perforated monopile is given in Figure 2.11.

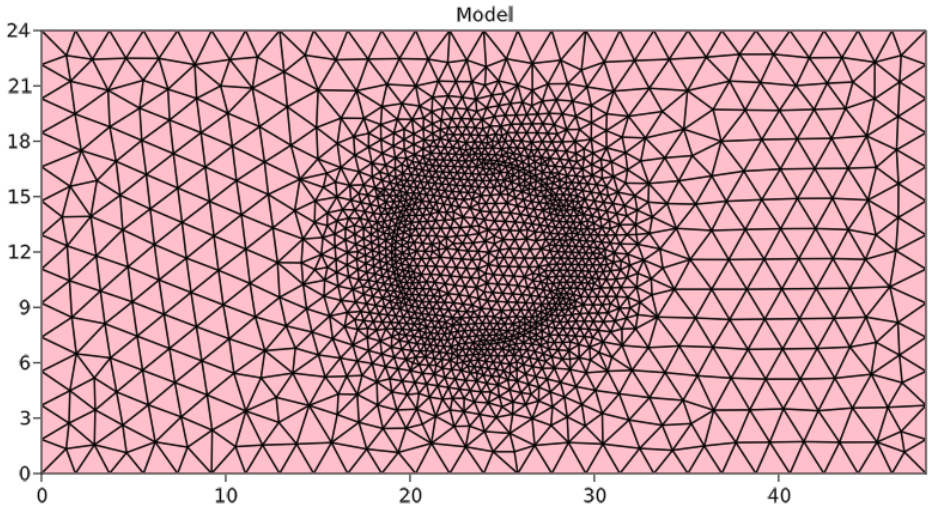


Figure 2.11: Example conformal mesh output ($D = 10, N_{\text{perf}} = 3, \alpha = 0, \beta = 0.5$).

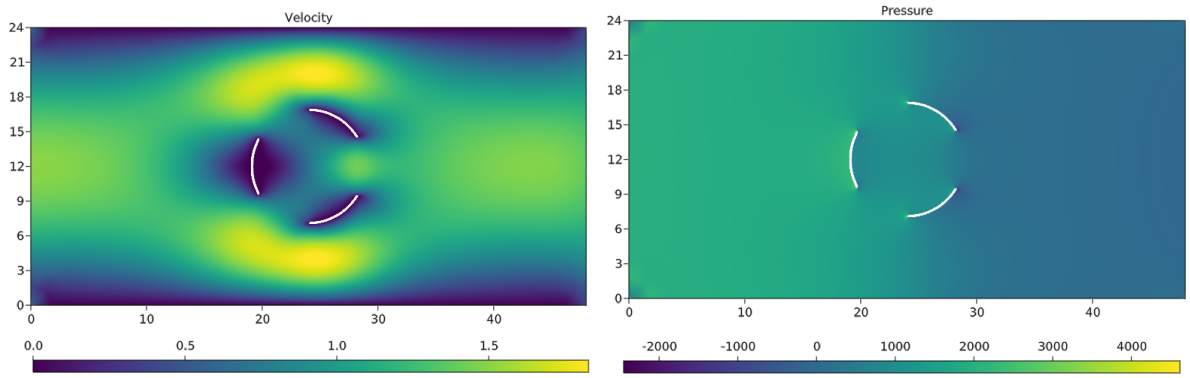
2.3.5. Finite Element Model output

Using the FEM model described above a solution for the flow fields can be found. See Figure 2.12a for an example of the solution for the velocity field and Figure 2.12b for an example With the flow fields known the force on the monopile can be calculated by integrating the stresses along the boundary, see Equation (2.30).

$$F = \int n_{\Gamma_{\text{monopile}}} \cdot \sigma^{\text{dev}}(\epsilon(u)) - p n_{\Gamma_{\text{monopile}}} d\Gamma_{\text{monopile}} \tag{2.30}$$

The result force consists of a drag component in the horizontal direction and a ‘lift’ component in the vertical direction.

As the lift force will be several orders of magnitude smaller than the drag force, have a mean of zero across all samples and will be more noisy and of less interest, the lift force is discarded.



(a) Example of the velocity field.

(b) Example of the pressure field

Figure 2.12: Example flow fields for ($D = 10, N_{\text{perf}} = 3, \alpha = 0, \beta = 0.5$).

The drag force of the perforated monopile $F_{D_{\text{perf}}}$ is divided through the drag force of the equivalent diameter non-perforated monopile $F_{D_{\text{ref}}}$ to get a non-dimensional reduction factor RF, see Equation (2.31). [25]

$$\mathrm{RF} = \frac{F_{D_{\text{perf}}}}{F_{D_{\text{ref}}}} \tag{2.31}$$

The RF gives a value between zero and one and is introduced to normalize the output of the model to improve the surrogate model prediction accuracy. The RF, for this CFD model, is a function of the porosity β, 'angle of attack' α and the number of perforations N_{perf}.

The RF could become higher than one for non-uniform distributed perforations (with low porosity). As likely the flow for the rear part of the monopile becomes relevant and starts to resemble a concave plate. [25] Another case were the RF could become higher than one is for side-perforatations ($N_{\text{perf}} = 2, \alpha = 90°$). This is related to changes in the wake structure and introduction of synchronized vortex shedding. [25] Both cases will not occur in this CFD model as only uniformly distributed perforations and no turbulence is taken into account.

2.3.6. Interpolating flow fields

The flow fields on the conformal FEM mesh are interpolated to a structured Cartesian mesh with the same resolution as the input geometry. [42] The purpose is to allow the use of the flow fields as an input to the surrogate model along with the representation of the geometry input, see Section 2.2.4.

3

Dataset

Using the model setup from Chapter 2 a dataset is created in this chapter, which will be used to create the surrogate model.

Section 3.1 gives a description of the workings of the pipeline used to generate the dataset. Two separate pipelines are used for the dataset generation. A pipeline for the creation of the (CFD) meshes and geometric representation of the perforated monopile in Section 3.1.1. And a pipeline for the CFD simulations in Section 3.1.2.

Finally, Section 3.2 gives the parameters used to create the dataset in Section 3.2.1 CFD results inSection 3.2.2 and dataset creation speeds in Section 3.2.3.

Contents

3.1. Dataset pipelines

In this section some additional information is given about the dataset creation pipelines. Starting from Figure 2.1 the dataset creation process is split in two independent pipelines. The first pipeline, see Section 3.1.1, responsibility is to create the SDF and mesh files based on geometric parameters for each sample. The second pipeline, see Section 3.1.2, consumes the mesh files and using the FEM model in Gridap determine the forces and flow fields.

Both pipelines are an example of an embarrassingly parallel job. Therefore, sample can be processed completely independently. This allows the use of parallel programming to increase the efficiency of both pipelines. [43] Both pipelines run on an AMD Ryzen 5 2600 (six cores) CPU.

3.1.1. Meshing pipeline

The meshing pipeline starts from a set of geometric parameters for each sample. Then three independent steps are performed: the SDF creation, mesh generation and saving meta data. And overview of the steps can be seen in Figure 3.1

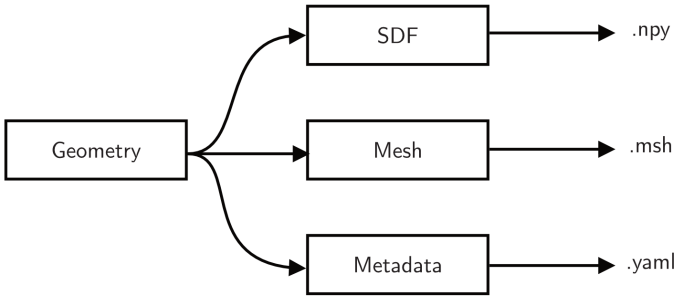


Figure 3.1: Overview of the meshing pipeline steps in order to create the SDF and mesh files.

The SDF is calculated using the process described in Section 2.2.1 using the NumPy package. The SDF is exported to the filesystem in the ".npy" format with 32 bit float precision to save disk space. The ".npy" file format is a NumPy array file format and can be directly loaded in PyTorch. [44, 34]

The mesh is generated using GMSH through the PyGMSH API as explained in Section 2.3.4. The mesh file is saved to the filesystem in the ".msh" fileformat, which is a GMSH fileformat for saving meshes. [40, 41]

As a final step metadata is saved with the rest of the dataset through Pandas in "yaml" file format. [45, 46] Metadata includes the geometric parameters for each sample, the dataset name, resolution, file structure, etc. The metadata aids in plotting of results and reproducibility of the generated dataset.

The meshing pipeline is completely implemented in Python. And is made parallel using the MPIRE parallel Python package. [47, 48]

3.1.2. CFD pipeline

The CFD pipeline takes as input the mesh files generated in the meshing pipeline, see Section 3.1.1. The mesh files are loaded with the GridapGMSH package. And sing the FEM CFD model as described in Section 2.3 the flow fields are solved on the conformal mesh with Gridap.

The interpolated flow fields to a structured Cartesian mesh are saved to the filesystem to the ".npy" format, the same that is used for the SDF input, with the use of the NPZ Julia package.

The forces on the monopile are saved to a simple ".csv" file using a combination of the CSV and DataFrames Julia packages. An overview of the CFD pipeline can be seen in Figure 3.2.

The CFD pipeline is implemented in the Julia language and made parallel using the Julia Distributed package. [36]

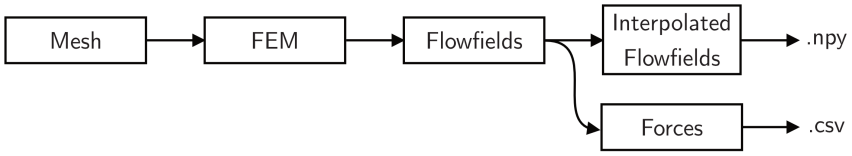


Figure 3.2: Overview of the CFD pipeline steps in order to calculate the forces and get the interpolated flowfields.

3.2. Generated dataset

With the dataset creation process setup in previous sections explained a dataset can be created, which will be used in the CNN model. In Section 3.2.1 the chosen geometric input parameter are given. And in Section 3.2.2 the CFD results for this dataset are given. Finally, in Section 3.2.3 the dataset creation times are given, which will be used to compare the CFD model with the results for the surrogate model.

3.2.1. Parametric model space

To generate the dataset there are several parameters that will remain constant, which are given in Table 3.1. And certain parameters that will make up the parametric space of the surrogate model, see Table 3.2.

To avoid tile quantization the dimensions of the resolution of the dataset is set to a power of two. [49] Furthermore, as a rectangular domain is needed the resolution is set to a 256 times 128 pixels size. The same resolution as used in [30, 24]. The resolution size is a trade-off between accuracy in capturing accurately the flow characteristics in the interpolated flow fields and the geometric features in the representation of the geometry input and the required dataset disk space, model size and required training times.

In order to avoid unequal spatial resolution sizes ($\mathrm{m\,px^{-1}}$) the width W and height H of the domain has the same ratio as the resolution.

Furthermore, the monopile is placed vertically in the center of the domain, which allows for the data augmentation as explained in Section 4.3.3. And is set to a constant diameter size D of $10\,\mathrm{m}$, which would be reasonable size for wind turbines of the size around the (lower) intermediate water depths that can be expected [50, 51]. The reason for this is to keep a consistent mesh quality. Furthermore, as only a creeping flow is considered the RF will not be a function of the Reynolds number.

The wall thickness is typically between $\frac{1}{80} \leq \frac{t}{D} \leq \frac{1}{120}$.However, such a thickness would require a much higher dataset resolution and mesh size. Therefore, the monopile thickness t is set slightly larger to be at least one pixel wide. This also ensure that the binary representation of the geometry, see Section 2.2.3, will be better able to represent the monopile geometry.

Table 3.1: Constant dataset parameters.

Symbol	Parameter	Value	Unit
N_x	x-resolution	256	px
N_y	y-resolution	128	px
W	Domain width	48	m
H	Domain height	24	m
D	Monopile (outer) diameter	10	m
t	Monopile thickness	0.2	m
y_c	Monopile y-position	12	m

The parametric design space of the surrogate model is given in Table 3.2. The load reduction factor RF is a function of the 'angle of attack' α, the porosity β and the number of perforations N_{perf}. The horizontal

position of the monopile within the domain x_c is also changed in order to generate more unique samples. As there is a limit with respect to the mesh sizing and dataset resolution in the ranges possible for the combination of porosity and the number of perforations.

Each of the ranges are chosen such that each sample parameters will a round number (with just a few significant numbers).

The x_c is chosen such that the monopile has some variation around the center of the domain, but doesn't get too close to the edges of the domain, to effect the CFD results (too much).

The number of perforations N_{perf} are chosen between 2 and 20. Two perforations are at minimum needed to allow discharge through the monopile. And 20 is chosen in accordance with the maximum perforation width (porosity) to ensure that in the worst case scenario (minimum perforation width and maximum number of perforations) the perforation will be big enough still. To have at least 2 CFD elements and pixels describing the perforation for both the CFD calculations and the representation of the geometry for the ML model respectively. Going any further would negatively affect the results and these ranges are thus predominately constraint by the resolution chosen for the ML model input array resolution and the mesh refinement resolution in the mesh generation. Furthermore, a perforation width too large is not really of interest above a certain percentage of porosity. As this will severely negatively affect the structural integrity of the monopile. But is chosen such that it at least cover the same range as used in [25].

For the 'angle of attack' α a range between 0 and 90 degrees is chosen. Due to the uniform perforation distribution and symmetry around the horizontal axis (monopile center vertically in the domain), this range covers all possible configurations (with the random vertical flip data augmentation, see Section 4.3.3) with minimal overlap for the different number of perforations.

Table 3.2: Dataset parametric space

			Value			
Symbol	Parameter	Unit	Min.	Max.	Step	Amount
x_c	Monopile x-position	m	18	30	2	7
N_{perf}	Number of perforations	–	2	20	1	19
β	Porosity	–	0.1	0.7	0.5	13
α	'Angle of attack'	°	0	90	11.25	9
			Total number of samples:			15 561

In total $15\,561$ samples are generated. Note, that as each geometric parameter is independent and a uniform distribution of the perforations is assumed, see Section 2.1, there is the possibility of having geometric equivalent samples. Figure 3.3 shows the occurrence rate of the percentage of samples that are the geometrically equivalent as a function of the number of perforations.

As can be seen this is happens for multitudes of four number of perforations. Especially, for sixteen number of perforations it could be the case that the accuracy of the CNN model might suffer due to the lack of unique samples.

Due to how GMSH generates the meshes these samples will have slightly different meshes. And therefore the RF will be slightly different.

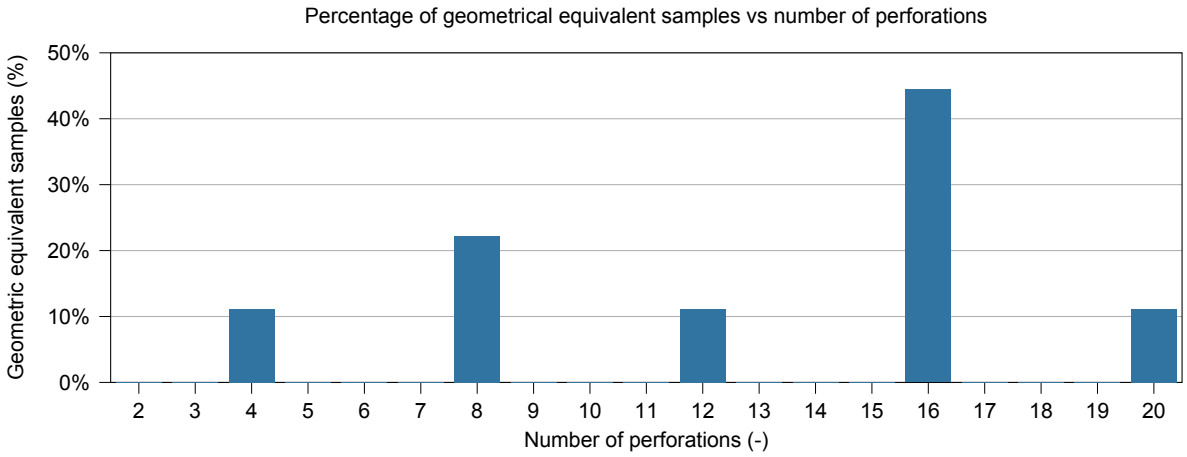


Figure 3.3: Percentage of double samples due to the interplay between the 'angle of attack' (α) and the number of perforations (N_{perf}).

3.2.2. CFD results

The load Reduction Factor RF for the dataset generated in Section 3.2.1 are plotted against the number of perforations (Figure 3.4), the porosity (Figure 3.5) and the 'angle of attack' (Figure 3.6).

A clear non-linear relationship can be seen between the RF and the number of perforations in Figure 3.4, which becomes more linear (and RF approaches a value of one) as the porosity goes to zero (and approaches a non-perforated monopile).

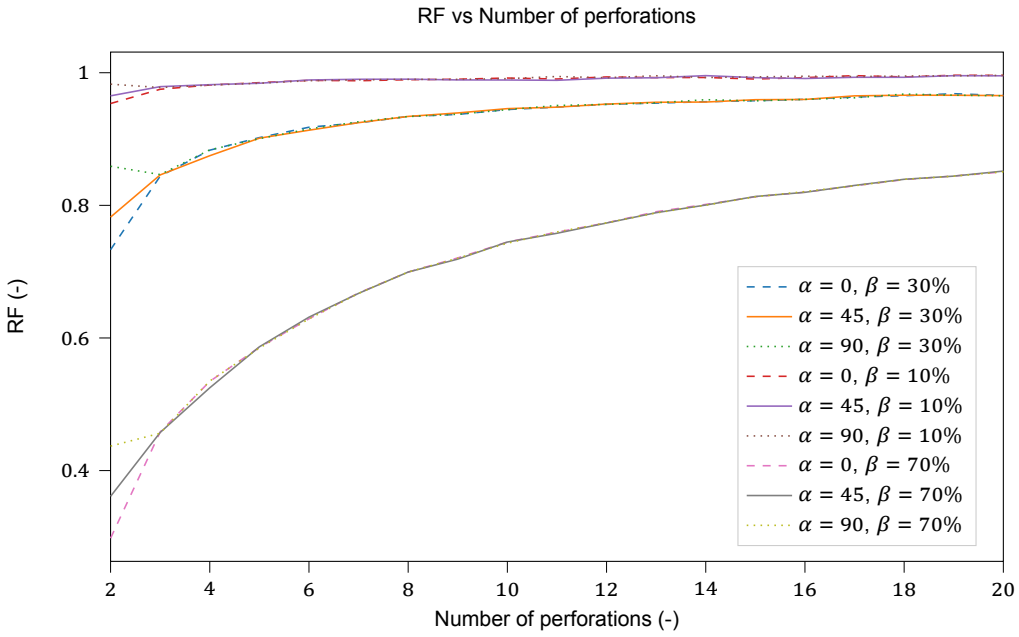


Figure 3.4: CFD results for the generated dataset showing the relationship between the RF and the number of perforations.

This same relationship can be seen in Figure 3.5 from a different perspective.

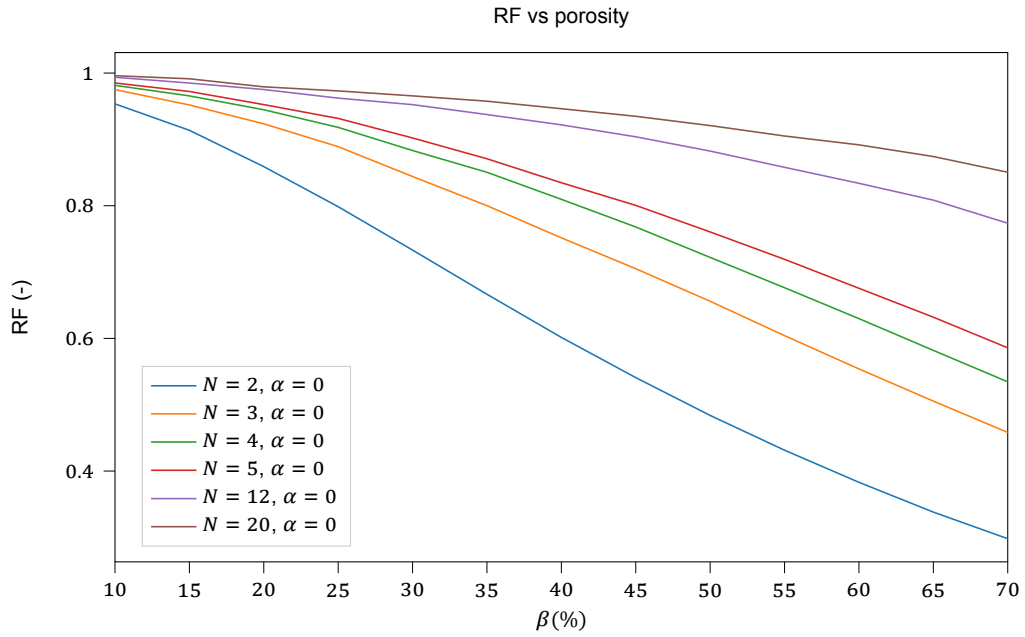


Figure 3.5: CFD results for the generated dataset showing the relationship between the RF and porosity.

Figure 3.6 shows the effect of the 'angle of attack' on the RF. It can be seen that a clear non-linear relationship exist for just two perforations. Though, for any other number of perforations the RF becomes almost constant.

For the inclusion of turbulence in the CFD model this plot should be especially different. As the 'angle of attack' will influence the wake structure, as explained in Section 2.3.5. Then it's expected that for low number of perforations there will be a strong relation between the 'angle of attack' and the Reduction Factor. With this correlation diminishing with increasing number of perforations. As due to the uniform distribution of perforations the wave number of this relationship (for alpha between $-90°$ to $90°$) should be equal to $N_{perforations}$. And the amplitude decreasing, as the change in frontal projectal area reduction (as a function of the angle of attack) will be less for higher number of perforations.

Excluding any additional non-linear effects due to the inclusion of turbulence, that could have an influence on this relationship (such as change in wake structure).

Why this behaviour is very minimally observed without turbulence is unknown and quite an unexpected result. However, this different behaviour for only two perforations observed may cause the surrogate CNN model to perform poorly for two perforations. As the model will have relatively few samples to learn from.

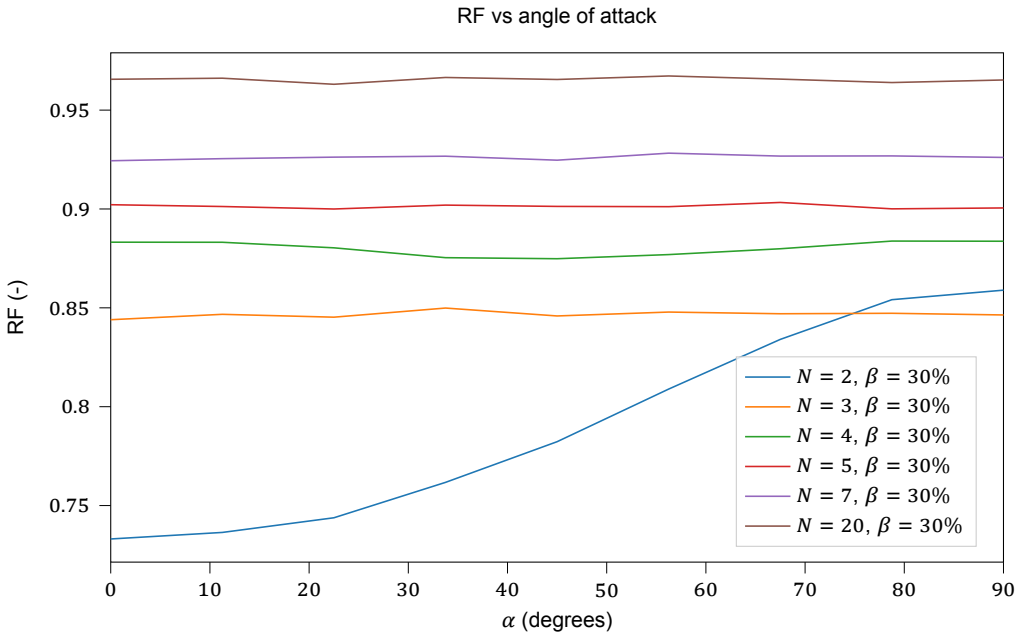


Figure 3.6: CFD results for the generated dataset showing the relationship between the RF and the 'angle of attack'.

Increasing the porosity and decreasing the number of perforations has the most influence on reducing the load reduction factor RF. As reducing the load on the monopile is mainly driven by reducing the projected frontal area of the monopile. And the frontal projected area is proportional to the drag force acting on the structure. Especially, when the effect that perforations have on the flow pattern with the inclusion of turbulence is not taken into account.

3.2.3. Dataset pipeline times

In Table 3.3 the average time per sample required for each pipeline is given. As can be seen generating a conformal mesh already takes significantly longer than generating a SDF for the same geometry.

Furthermore, the average file size for each sample is given as well. With a total of $15\,561$ the total disk space for the dataset is about 13.8 GB.

Table 3.3: Average results per sample for each of the pipeline processes.

	Average per sample	
Pipeline	Time (s)	Size (kB)
SDF	0.003	131.2
Mesh	0.132	432.0
CFD (full order)	1.411	393.6

4

CNN model

In this chapter, the Convolutional Neural Network (CNN) architecture used to create the surrogate model is discussed. First, a brief introduction to deep learning using covolutational neural networks is given in Section 4.1. The overall CNN model architecture that is used is proposed in Section 4.2. Section 4.3 explains the dataset preparation used during model training. And Section 4.4 explains the choice of loss function. The five models that will be trained are summarized in Section 4.5. And finally, Section 4.6 explains the choice of hyperparameters and the method of hyperparameter optimization.

Contents

4.1. Introduction to Convolutional Neural Networks

Deep learning is a machine learning method that uses artificial neural networks consisting of many layers (deep) stacked on top of each other. Many of these layers have trainable 'weights', which are tuned during model training to fit the model to the provided (output) data, in the case of supervised learning (where both input and corrosponding output data examples are used). By stacking a lot of layers together a highly complex non-linear mapping between input and output data can be found, that is able to accuracy predict the output (and generalize) a model based on the input data.

4.1.1. Advantages and disadvantages of deep learning

There are several advantages that deep learning has over traditional machine learning methods (such as for example linear regression or support vector machines). The first is that feature extraction from the data is done in the model layers. The right feature extraction is solved during the model training. The advantage of this is that it requires less input from the engineer to create the surrogate model, but the disadvantage is that it will require a lot more data in comparison. [52]

Another advantage is that deep learning networks are well suited to take advantage of the Graphical Processing Unit (GPU). Leveraging the GPU and neural network can allow for huge performance gains in comparison to traditional (full scale (single) CPU) (CFD) models. [52]

To the train a deep learning model and find the right (hyper)parameters used to tune the model can be computationally and time expensive. This means that to create the surrogate model it has a high cost upfront cost in the 'offline' phase. Furthermore, tuning the model (hyper)parameters and getting an understanding on how to find the right network architecture (combination of stacked and inter-connected network layers) can be poorly understood and also be a lot of trail and error to get right. [52]

4.1.2. Convolutional Neural Network

The most common type of deep neural network is the Convolutional Neural Network (CNN). CNN's use multiple convolution layers. These consist of kernel with trainable weights that performs a convolution operation many times along the input image. An example of the process of convolution can be seen in Figure 4.1. The convolution layer can learn this way a feature representation based on the input.

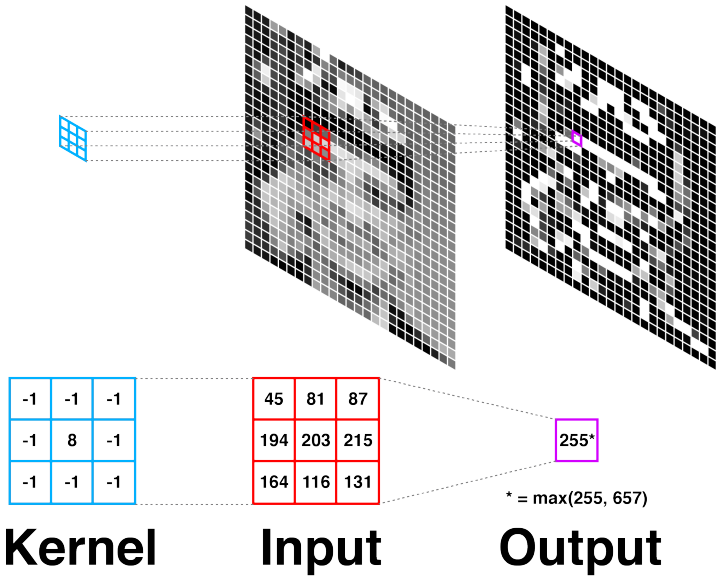


Figure 4.1: Example of a convolution layer with a 3x3 kernel. [53]

These layers are often followed by an activation function (layer), that multiplies the output. Other type of layers that are used are different type of pooling and upsampling layers, that increase or decrease the resolution of the feature maps. These layers also apply a kernel, but don't have trainable weights.

CNN's are especially well suited to work with image-like data. Data that has a structured grid like 2D structure with multiple (feature) channels. Furthermore, CNN's have sparse weights. As the kernel weights are applied across the image and have much smaller dimensions than the input layer. This in contrast to dense (linear) layers. Where each layers node is connected to every node in the next layer with a separate weight. This allows CNN's to use far fewer weights that need to be trained to accomplish the same task, with far less training time and computer memory requirements in comparison to dense (linear) networks. [52]

4.1.3. Model training and optimization

To train the deep learning network input data is fed in batches through the surrogate model. The output of the model (in supervised machine learning) is compared to the ground truth output in the dataset. A loss function determines the degree of the error between the desired output and the output from the model. Examples of loss function often used are for example the Mean Squared Error (MSE) or the Mean Absolute Error (MAE).

By back propagating the loss through the model, the weights of the model can be updated to minimize

the loss. The learning rate and optimizer used determine by how much the weights are updated at each iteration. To prevent and evaluate the degree of overfitting and ability to generalize based on (new) data two separate losses are used. A training loss that is used to update the weights. And a validation loss to verify the performance of the mode during training and to stop the training if the validation loss does not improve any more (and start to diverge from the training loss, indicating overfitting), see Figure 4.2. For this purpose the dataset set is split into a training set and a validation set (with no overlap). [52]

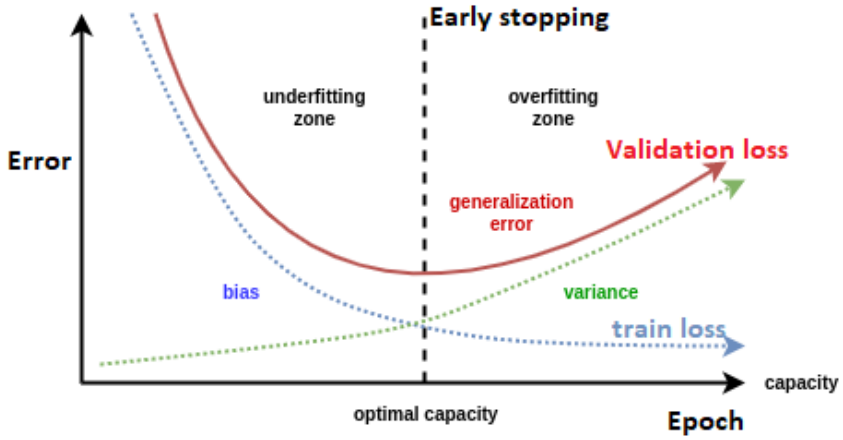


Figure 4.2: Figure showing the concept of overfitting.

Finally, after training the model, the model performance can be evaluated on the test data set by determining the test loss (or for example a model accuracy metric). Preferably, this would be a data set that is not used for training the model. This allows for an unbiased evaluation of the model performance in order to compare different CNN model iterations with each other. As the final step to create the surrogate model, is to find an optimal set of model (hyper)parameters to optimize the model performance (the lowest test loss). [52]

4.2. Model architecture

A neural network architecture for the surrogate model is proposed in Figure 4.3, that would be used in this thesis. It's roughly based on the based on the LeNet-5 and similar CNN architectures. [54] Similar CNN networks have been used in comparable regression problems such as in [28, 55, 56].

The model architecture takes an image-like data as input. Then a series of convolutional layers are used to extract and learn features in the input data. Each convolution layer starts with a 2D convolution layer with a certain filter size f, kernel size k of three and stride s of one. Only for the binary representation of the geometry a 2D Batch Normalization layer is inserted. Then it goes through a Rectified Linear Unit (ReLU) layer. And finally the feature maps are reduced in half by a 2D Max Pooling layer, with kernel size of 2 and stride of 2.

In total the network uses N_{conv} number of these convolution layers in series. The first convolution layer has a filter size f of $f_{0,\text{conv}}$. Each consecutive convolutional layer doubles the number of filters used.

After these layers the remaining feature maps are flattened and goes through a linear regression head. The linear regression head has N_{lin} number of (optional) extra hidden linear layers. Each hidden layer is also followed by a ReLU layer. And the first hidden layer has a size of $f_{0,\text{lin}}$ and each consecutive hidden layer reduces it size by a factor of a_{lin}.

At the end of the network the output is a linear layer of size one (and using only a linear activation layer). And outputs the prediction of the load reduction factor RF.

Five different input types are considered, which uses this basic network architecture. These five model

inputs are:

- Signed Distance function (SDF)
- Signed Distance components of the SDF (Dist)
- Binary geometry representation (Binary)
- SDF + CFD (flow fields)
- Dist + CFD (flow fields)

To tune and optimize the hyperparameters of the model described above, with a few others, are part of the hyperparameter optimization, see Section 4.6.

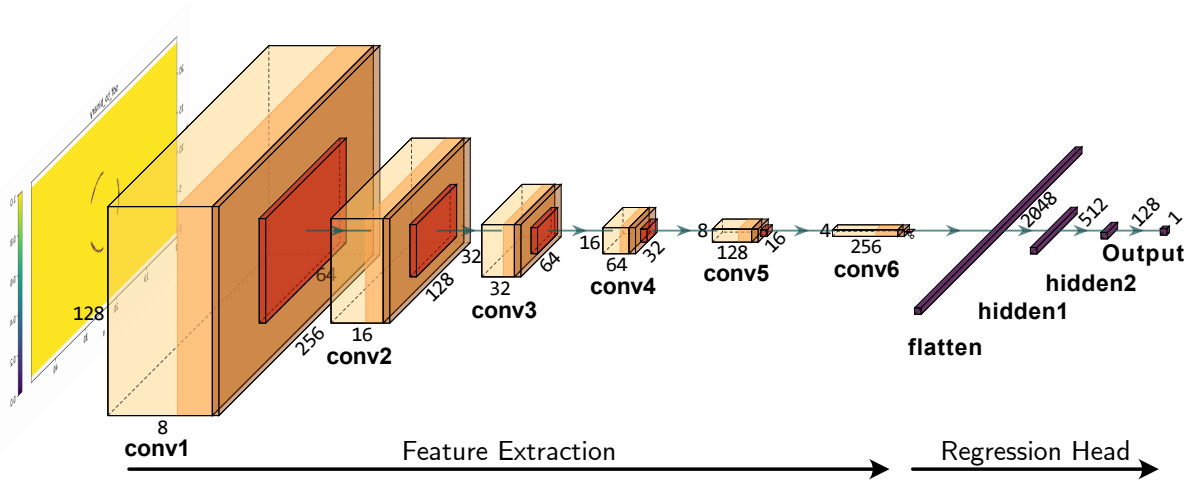


Figure 4.3: Model architecture used in this thesis. Yellow blocks show a 2D convolution layer with ReLU activation function, dark yellow a BatchNorm2d layer, red a Pool2D layer and finally purple show Linear layers. Created using [57].

4.3. Dataset preparation

In this section, the splits and transformations that are used on the dataset during model training are given.

4.3.1. Dataset split

The dataset created in Section 3.2 is randomly split in a train ($80\,\%$, $12\,449$ samples) and validation dataset ($20\,\%$, 3112 samples). For the test dataset the whole dataset is used. This is done to ensure the same data is used across different cluster nodes during the hyperparameter optimization. And get a complete overview of the model performance within the defined parameter space in the dataset.

4.3.2. Data normalization

The input data is normalized to eliminate the different scales that each input channel has. Furthermore, normalization is applied to avoid the 'vanishing gradient problem' and could reduce the required training time. [58]

Following, the data normalization used in [30], the input channels (except for the binary represention) are normalized using a Z-normalization. [59] The mean and standard deviation for these input channels that are used to normalize the data is given in Table 4.1.

Table 4.1: Mean and standard deviation of each input channel to normalize the input data.

	SDF	Dist-x	Dist-y	Pressure	Velocity-x	Velocity-y
Mean	9.84	8.56	3.81	1.20×10^{3}	9.43×10^{-1}	1.2×10^{-4}
Std.	6.33	6.50	2.63	1.04×10^{3}	4.98×10^{-1}	2.22×10^{-1}

The binary representation of the input uses batch normalization. Batch normalization allow for higher learning rates (and increase training speed). And it may eliminate the need for other regularization techniques. [60] The batch normalization is applied by including a batch normalization layer in the CNN, see Section 4.2.

4.3.3. Data augmentation

The accuracy of the trained model is directly proportional to the number of (qualitative) samples available in the dataset for training. And the networks are reliant on a lot of data to avoid overfitting problems.

However, creating additional new samples is computational very expensive. As such there is a limit using this approach for what is feasible given a certain time frame (and disk space available).

Data augmentation circumvents this problem by (effectively) creating new samples (on the fly during training) from the data that is already available in the dataset. Commonly used techniques include applying a combination of translations, rotations or flips to (image-like) samples to create new samples. [61]

Furthermore, more sophisticated techniques have been developed specifically for CFD deep learning models. These use the similarity principle of fluid dynamics to create flow fields with different input velocities and/or diameters, but keeping the Reynolds number constant. [62]

However, in this research due to the choice of CFD model considered and a model that predicts the RF values this will not provide any useful additional samples. Therefore, a simpler data augmentation technique is applied.

A random vertical flip with a probability of $50\,\%$ is applied to the input data, using Torchvision. [59] This corresponds to a sample with the negative 'angle of attack' α. As the perforated monopile is vertically centred in the computational domain, the RF stays the same. This simple data augmentation has the benefit of almost doubling the effective number of samples in the dataset.

4.4. Choice of loss function

Three different options for the loss functions are considered to train the model: The Mean Squared Error (MSE, see Equation (4.1)), the Mean Absolute Percentage Error (MAPE, see Equation (4.2)) and the sum of the MSE and MAPE. [63] As the choice of loss function, along with the network architecture, have a lot of effect on the performance (accuracy) of the model. [30]

$$\mathrm{MSE} = \frac{1}{N}\sum_{i}^{N}(y_i - \hat{y}_i)^2 \tag{4.1}$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{\max(\epsilon, |y_i|)} \tag{4.2}$$

To decide on the loss function to choice a simple test is conducted. Using the binary input three models are trained with each option for the loss function and the results are compared. Figure 4.4 plots the distribution of the relative error (see Section 5.3) for this test. As can be seen from this figure the MSE is the clear choice for the loss function to use.

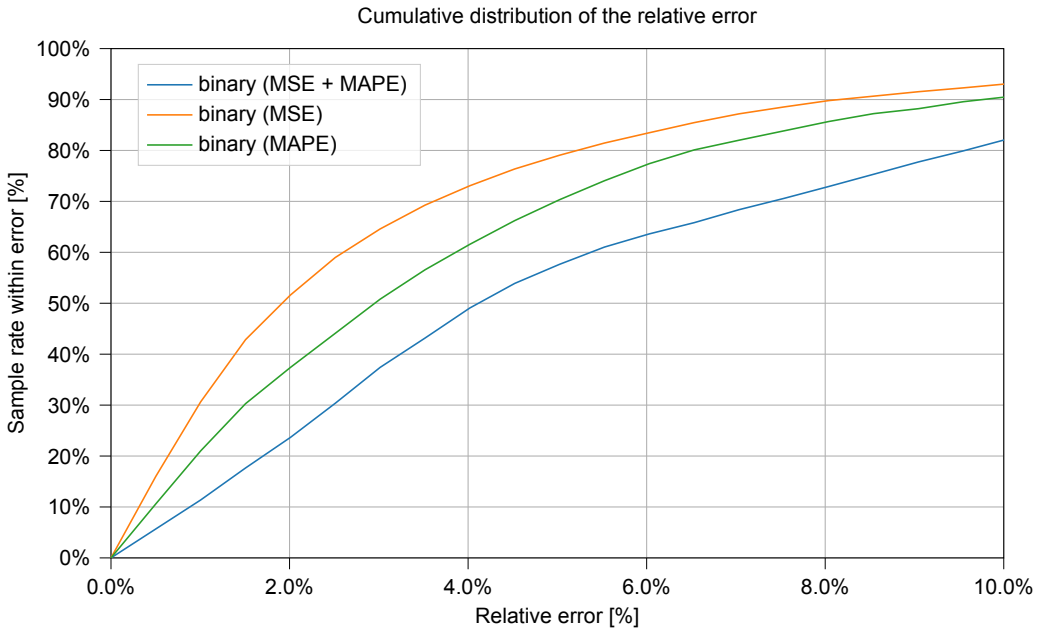


Figure 4.4: Effect of the choice of loss function on the accuracy of the model.

4.5. Model summary

Below in Table 4.2 a summary of the most important fixed model training parameters is given.

For model training the AdamW optimizer is used. Furthermore, the model is trained for a maximum of 120 epochs, with early stopping if the validation loss doesn't improve for more than 5 epochs to prevent overfitting, see Figure 4.2. One GPU and four (CPU) dataloaders with mixed precision is used.

Mixed precision combines both 32 and 16 bit floating points, which reduces the memory requirements during model training and reduces training time, at minimum cost in model accuracy. [64]

Table 4.2: Fixed model train parameters.

Parameter	Value
Optimizer	AdamW [65]
Loss function	MSE
Data augmentation	Random vertical flip ($p = 0.5$)
Max. Epochs	120
Early stopping (Patience)	5 epochs
Validation / train fraction	0.2
Model precision	16 bit
Number of GPU's	1
Number of (CPU) dataloaders	4
Max. model size	3 GB

Table 4.3 gives a short overview of the five different models that are trained, see Section 4.3.2.

Table 4.3: Summary of the different models.

	Number of input channels	Normalize	BatchNorm2D
SDF	1	✓	
SDF + CFD	4	✓	
Dist	2	✓	
Dist + CFD	5	✓	
Binary	1		✓

4.6. Hyperparameters choice and optimization

This section discusses the choices for the hyperparameters using in training the different network architecture for the five different model inputs. Using an optimization process these hyperparameters are optimized based on minimizing the chosen test metric. By minimizing the test metric the accuracy and performance of the model is improved. A total of eight different hyperparameters are used.

First are the hyperparameters related to the network architecture. This includes the number of convolution layers N_{conv} and the number of filters in the first convolution layer $f_{0,\text{conv}}$. For the regression head, the number of extra hidden linear is used N_{lin}. Furthermore, if there are extra hidden linear layers the size of the first hidden layer $f_{0,\text{lin}}$ is controlled. This is done to constrain the number of parameters allowed for the regression head. Furthermore, a factor a_{lin} reduces the size of each subsequent hidden layer.

Additionally, the batch size N_{batch} is used as a hyperparameter. As the whole dataset doesn't fit in the GPU memory the training is perforated on (mini)batches of (training) data. Training with a large batch size is allows for the better utilization of the power of the GPU. This has the benefit of reduced training times. However, in general a larger batch size leads to a more poor generalization. Typically, the batch size is set to a value with a power of two to utilize the GPU efficiently, as explained in Section 3.2.1.

Another hyperparameter that is optimized is the learning rate γ of the optimizer. The learning rate affects how much the model weights are adjusted after each training step. A lower learning rate leads to slower convergence of the model and thus increased training times. However, a too large learning rate the network might not (properly) converge.

Finally, the weight decay N_{conv} is used. Weight decay is a regularization technique, which mitigates overfitting the models and improves the generalization of the model. Weight decay adds a penalty term to the loss function of the model, which reduces the weights of the model during back propagation. [65]

The hyperparameter optimization search space is given in Table 4.4.

Table 4.4: Hyperparameter optimization search space.

Symbol	Hyperparameter	Options	Number of options
N_{batch}	Batch size	32, 64 & 128	3
γ	Learning rate	3×10^{-4} & 1×10^{-5}	2
λ	Weight decay	0, 1×10^{-2} & 1×10^{-3}	3
N_{conv}	Number of convolution layers	3, 4, 5, 6 & 7	5
$f_{0,\text{conv}}$	Number of features first convolution layer	4, 8, 16, 32, 64, 128 & 256	7
N_{lin}	Number of hidden linear layers	0, 1 & 2	3
$f_{0,\text{lin}}$	First hidden linear layer size	128, 256, 512, 1024, 2048 & 4096	6
a_{lin}	Factor of hidden layer size	1, 2 & 4	3
		Total number of combinations:	34 020

The hyperparameter optimization uses the Optuna optimization framework [66] with the Hydra config management framework [67] (both in Python) to create 'sweeps' to run the optimization pipeline in parallel.

The hyperparameter optimization is run on the TU Delft Hyper Performance Computing (HPC) cluster. This allows running the optimization in parallel on multiple nodes. Using the cluster a lot more hyperparameter trails could be conducted.

A total of 400 trails were conducted for each model. Each optimization step considered 50 trails. The first 100 used a random search. With the remaining 300 further optimized using the Tree-structured Parzen Estimator (TPE). The TPE sampler uses the results from the previous optimization step to choice a new set of hyperparameters, which are most likely to be good. [68, 69]

There are $34\,020$ unique model configurations considered in the search space, see Table 4.4. The chosen sampler strategy doesn't guarantee the global optimum within the search space is found. However, with some luck, a set of hyperparameters is found that has good (enough) model performance.

To evaluate the model performance the Mean Squared Error on the test dataset is used, see Section 4.4. This test metric is minimized in the hyperparameter optimization. Minimizing the MSE, minimizes both the bias and variance of the (relative) error in the test set, see Section 5.3.

Table 4.5: Chosen hyperparameter optimization parameters.

Parameter	Value
Sweeper	Hydra [67]
Optimizer	Optuna [66]
Sampler	Tree-structured Parzen Estimator [68, 69]
Test metric	MSE
Number of parallel jobs	50
Number of (random) startup trials	100
Total number of trials	400

During the course of the hyperparameter optimization for the five models no proper fault tolerance had yet been implemented in the Hydra (Optuna) sweeper. [70] This causes an issue of the optimization process crashing due to an out of memory issue on a single iteration. To mitigate this issue any model with an expected model size bigger than 3 GB is immediately rejected before training has started.

5

Results

Based on the work in Chapter 4 five surrogate models using a CNN have been created. Each model considers different inputs to the CNN and the model architecture has been decided upon by the process of hyperparameter optimization. The results of the hyperparameter optimization are given in Section 5.1 and the learning curves during training for these models in Section 5.2. In Section 5.3 the accuracy of the model predictions is evaluated with respect to the full CFD model. Finally, Section 5.4 considers the model evaluation speed. Comparing it to the CFD model the expected time savings by using a surrogate model is determined.

Contents

5.1. Model Hyperparameters

Result from the hyperparameter optimization from Section 4.6 are summarized in Table 5.1. A more detailed description of the model architecture for each model can be found in Appendix B.

As can be observed only the Dist and Binary models have extra hidden layers in the regression head. However, the Dist model requires by far the most number of model parameters to be trained, see Appendix B.

Table 5.1: Found Hyperparameters

	Optimizer			Convolution layers		Linear layers		
Model	N_{batch}	γ	λ	N_{conv}	$f_{0,\text{conv}}$	N_{lin}	$f_{0,\text{lin}}$	a_{lin}
SDF	32	3×10^{-4}	1×10^{-2}	6	16	0	-	-
SDF + CFD	32	1×10^{-5}	0	6	32	0	-	-
Dist	32	3×10^{-4}	0	5	32	1	1024	4
Dist + CFD	32	1×10^{-5}	1×10^{-2}	6	32	0	-	-
Binary	32	1×10^{-5}	1×10^{-3}	6	8	2	512	4

The performance of the models is evaluated and compared based on two metrics. The ability of the model to make accurate predictions compared to the CFD model, which is reviewed in Section 5.3.

And the main driver for using a surrogate is the faster model evaluation speed that can be achieved, see Section 5.4.

5.2. Loss curves

The learning curve for the (optimized) Dist + CFD model can be seen in Figure 5.1 and the learning curves for the other models are given in Appendix A.

The learning curve in Figure 5.1 plots the training and validation loss of the (optimized) model training as function of the epoch. These curves give some insight in the model training behaviour. The general trend for each of these loss curves is that both the training and validation loss decrease over time. In other words the model accuracy will improve.

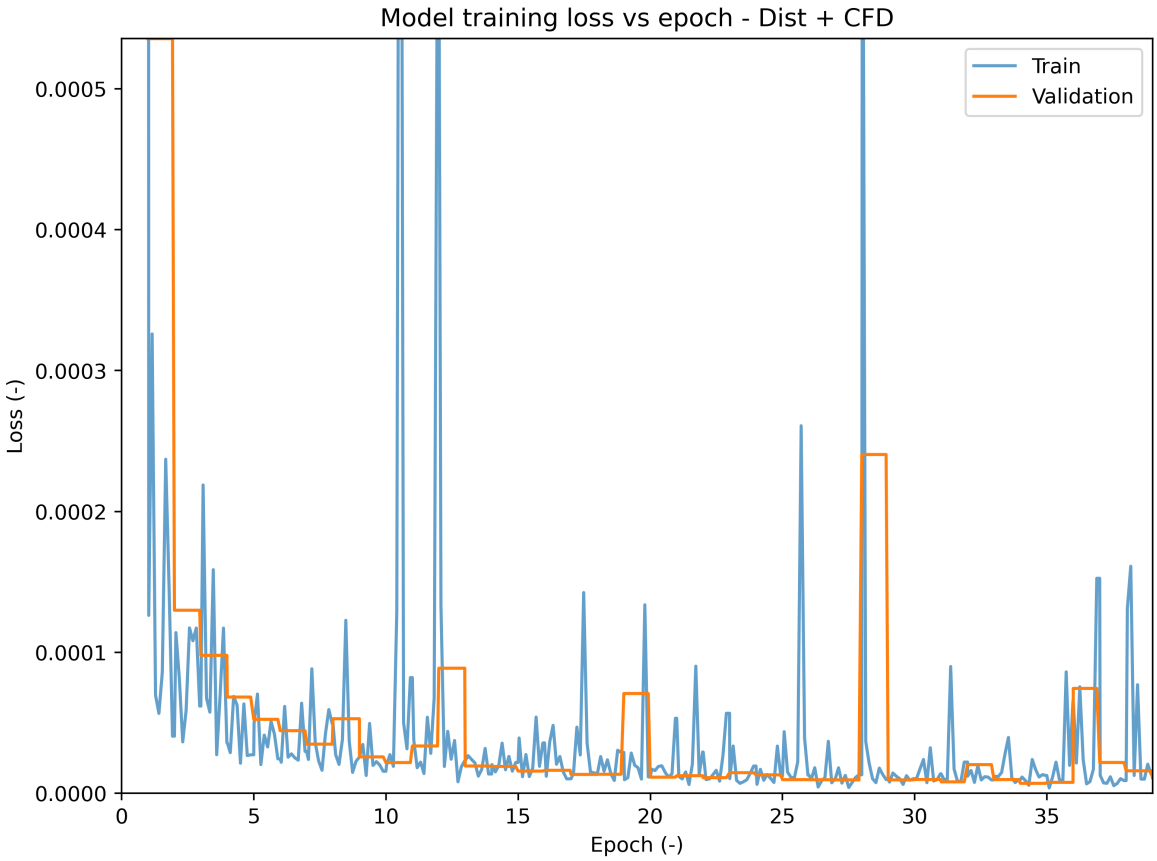


Figure 5.1: Loss curve for the Dist + CFD model.

When the validation loss starts to increase (for a patience of 5 epochs) while the training loss continue to decrease (see Figure 4.2) the model is likely to start overfit on the data, and training is stopped.

While some spikes in the loss curves is to be expected, as long as the magnitude decreases, the spikes observed in the (training) loss curves for the SDF + CFD and Dist + CFD models are unexpected. A possible explanation for this could be that it could be caused by some bad (input) data or some outlier in the dataset. As this behaviour is predominately observed in the models that include the CFD flowfields in the input it's likely that the problem is in (some) of the flowfields. But this will require a much more in depth evaluation of the CFD flow fields.

5.3. Model accuracy

To determine the quality of the predictions of the surrogate model it's compared to the "true" values from the full order CFD model. A relative error metric is introduced in Equation (5.1), which calculates

what fraction the prediction of the RF using the surrogate model is off from the CFD model.

$$\text{Error} = \frac{\text{RF}_{\text{pred}} - \text{RF}_{\text{CFD}}}{\text{RF}_{\text{CFD}}} \tag{5.1}$$

The relative error is computed for each sample within the complete dataset (all 15 561). As these samples are readily available for this purpose and gives a complete overview of the parametric space for which the surrogate model is valid. Of interest is to known the expected error rate for an average sample, the range of possible errors and how the error behaves as a function of the parametric space parameters.

To gain further insight Figure 5.2 plots the cumulative distribution of the relative error metric. And Table 5.2 tabulates the most important results that are derived from Figure 5.2.

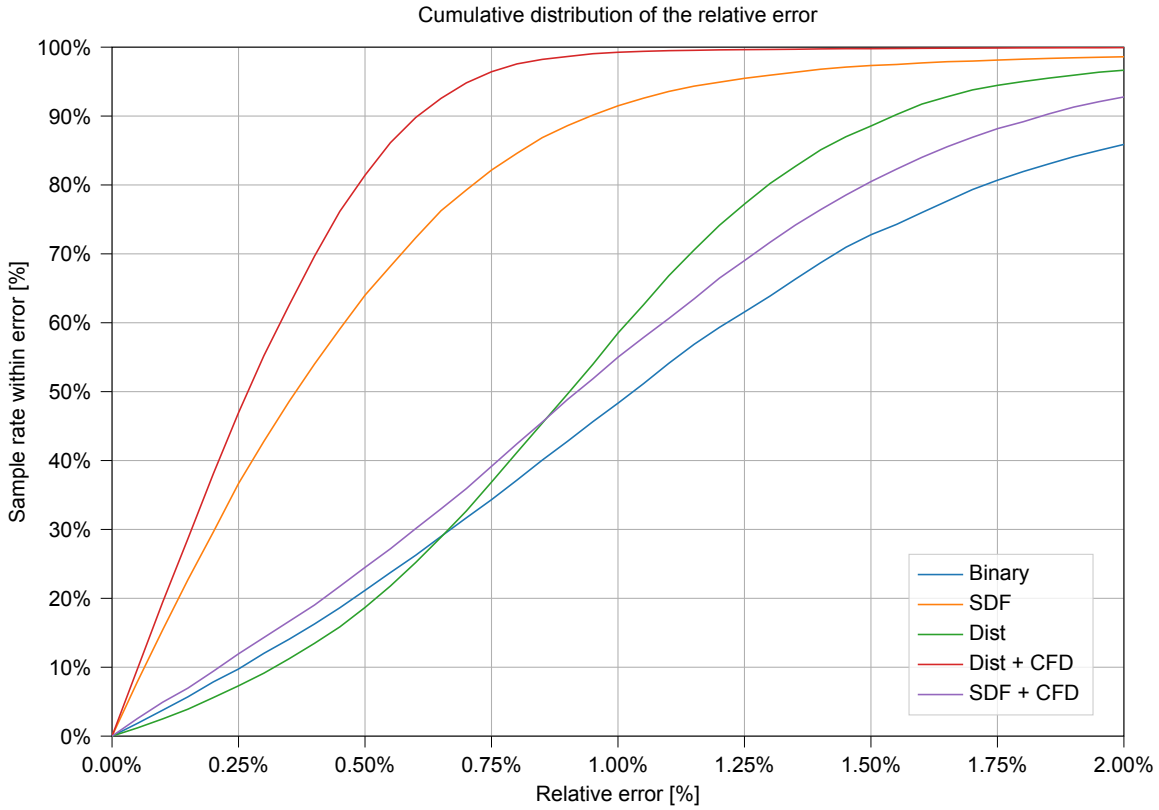


Figure 5.2: Plot showing the distribution of the relative error of the different models.

Model	Median (%)	99th (%)	Max. (%)
SDF	0.363	2.279	7.654
SDF + CFD	0.919	3.397	19.410
Dist	0.904	3.156	14.007
Dist + CFD	0.269	0.944	5.453
Binary	1.030	4.512	16.258

Table 5.2: Summary of the relative error distribution results.

It can be seen that each of the models have a relative error rate well below 5 % for the 99th quantile. From an engineering perspective the surrogate model serves as preliminary design tool, as explained in Section 1.3, to optimize the geometry of a perforated monopile. Promising designs after the optimization using the surrogate model can then be validated using a full and higher fidelity CFD model further in the design process. A known maximum relative error rate of 5 % will likely be sufficient in those cases. As such while some of the models perform better than others (accuracy wise), they could potentially be

used interchangeably in practice. And the easier to use and train model (not required the flow fields) might be more suitable from a practical standpoint.

Additionally, quite a discrepancy can be observed between the 99th-quantile and the maximum relative error rate for the dataset. It's likely that the relative error rate is distributed inhomogeneously over the parametric space. In other words there may be regions which the model performs worse in terms of prediction accuracy.

To investigate this further the relative error is plotted as a function of the parametric space parameters in Figure 5.3. For this figure the Dist + CFD (the model with the highest accuracy) is chosen. As the other models show a very similar result the figures for the remaining models are provided in Appendix C.

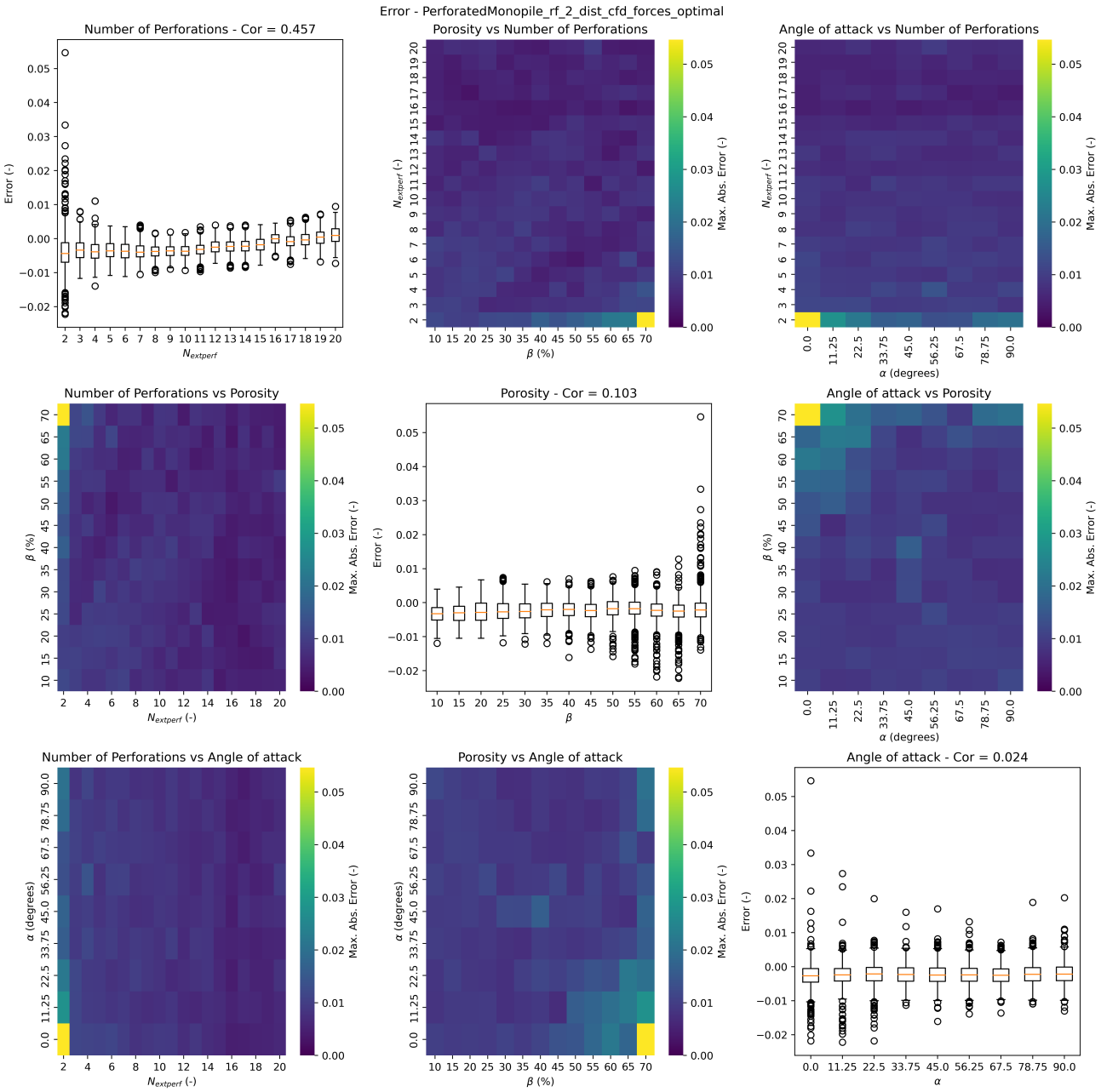


Figure 5.3: Dist + CFD input model relative error plot. On the diagonal the relative error is plotted for each of the independent geometry variables. On the off-diagonal the maximum absolute relative error is plotted for a combination of two geometry variables.

From Figure 5.2 it can be observed that the relative error rate is the highest for cases with just 2 perforations N_{perf}, a high porosity β and a low 'angle of attack' α. A likely reason for the poor ability of the surrogate model to accuracy predict the RF in these regions is related to Figure 3.6 in Section 3.2.2.

As the RF is only strongly a function of the 'angle of attack' α for just two perforations. It was hypothesized that for these cases the model might struggle to learn this relationship, as relatively only a very few samples are available to learn this non-linear relationship.

Furthermore, Figure 3.4 also shows some exceptional behaviour only seen at two number of perforations. Showing again the behaviour that the (RF) is only (strongly) a function of the 'angle of attack' (α) for just two perforations.

Restricting the validity of the model to exclude these poor regions in the parametric space the surrogate model is expected to have a maximum relative error rate (almost) equal to the reported 99th-quantile error rate from Table 5.2.

Comparing the accuracy of the different models, by using Figure 5.2, several things can be observed. Ranking the models based on the prediction accuracy the order is: Dist + CFD, SDF, Dist, SDF + CFD and Binary.

A general can be observed in that the more information is provided in the input to the CNN, the better the prediction accuracy tends to get. Especially, the binary representation of the geometry, which only gives locally information about the perforated monopile geometry and no distance information performs the worst. This is in line with what is expected.

Furthermore, the direct input of the flow fields along the representation of the geometry improves the accuracy of the Dist input model significantly. In contrast, worse accuracy is observed for the SDF model when the flow fields are included. Even though the performance of the SDF model is 2nd to the Dist + CFD model. A possible explanation could be a bad hyperparameter optimization result. Or that this model would require even more layers (seven or more convolution layers), which wasn't considered in the hyperparameter optimization in Section 4.6.

5.4. Model evaluation speed

One of the research questions is to quantify the performance of the surrogate model in terms of model evaluation speed. In this section the speed-up of using a surrogate model with respect to the CFD model is determined.

Table 5.3 combines the results from Table 3.3 with the average time per sample required for the surrogate model to make a prediction of the RF value. As there is no significant difference in the prediction times between the different models (and input types) only one average value for all models is reported in this table.

These results have been obtained by running the model pipelines on an AMD Ryzen 5 2600 (six cores) CPU and a Nvidia GTX 1060. The CNN model is run on 1 GPU with 4 CPU dataloaders. To get a fairer comparison the times for the other pipelines are also reported when maximizing the hardware available. Therefore, the SDF, mesh and (full order) CFD models reflect the average time per sample when (trying to) utilize all six CPU available.

Table 5.3: Comparison in time (in seconds) per sample between the surrogate CNN model and the full order CFD model to evaluate the geometry.

	Pipeline				
Model	SDF	Mesh	CFD (full order)	CNN (prediction)	Total
CFD	-	0.132	1.411	-	1.543
CNN	0.003	-	-	0.001	0.004

A ‘speed-up’ factor can be determined by dividing the average time required per sample for the CFD model through the average time required per sample for the CNN model. For the CFD model this includes generating a conformal mesh using GMSH and solving the FEM solution using Gridap. And for the CNN models it includes creating the SDF and make a model prediction using the CNN model. Therefore, it can be concluded that the CNN surrogate model is about 386 times faster than the traditional (full order) CFD model.

It should be noted that the result for the speed-up factor is highly depends on a few important factors. Most importantly the hardware used to obtain these results and the choice of full order CFD model. When turbulence is taken into account the speed-up factor is expected to improve significantly. As the time required to evaluate the CFD model will increase substantially, but the complexity of the CNN model (and the required prediction time) is likely to not differ much.

6

Conclusions

Perforated monopile designs show good promise to provide an alternative to jacket substructures for intermediate water depths of around $30\,\text{m}$ to $120\,\text{m}$. As the jacket substructure is in general more expensive and difficult to manufacture and install there are costs savings to be gained by the continued use of monopiles at these water depths. Lowering the Levelized Cost of Electricity (LCoE) is one of the main drivers for the industry. However, at these water depths the required monopile dimensions would reach manufacturability limits. These dimensions are mainly driven by the fatigue limit state, which is mainly affected by the wave loading acting on the structure.

By creating perforations in the vicinity of the splash zone, were the wave loads are the highest, the (wave) load on the monopile can be reduced by allowing discharge through the monopile. The main mechanism by which the load is decreased, due to introduction of perforations, is by reducing the frontal (projected) area of the monopile to the flow. Reducing the load improves the fatigue lifetime of the structure. Additionally, perforated monopiles also show reduced (local) scour, reduced corrosion damage and provides a habitat for marine life, as additional benefits over conventional monopiles.

As determining the expected load reduction for a certain perforated monopile geometry would require the evaluation of expensive CFD models, which could take hours or even days, showing there is a clear need for faster surrogate models. Replacing the full order by a surrogate model a lot more designs can be considered, which is (for example) needed in (early) design optimization studies.

The presented research has a goal to develop such a surrogate model. And the main research question is formulated to the following:

> 'How to best develop a machine learning-based surrogate model for the characterization of hydrodynamic loads on perforated monopiles?'

To answer this main research questions several subquestions are formulated:

1. Which (geometric) inputs are of importance for the design of a perforated monopile?
2. What type of input should be used in the CNN model to achieve the best performance?
3. What CNN architecture should be used for this surrogate model?
4. How accurate is the surrogate model for determining the load reduction of the perforated monopile?
5. How much faster is the surrogate model to evaluate one design compared to the full CFD model?

A surrogate model was developed by leveraging a Convolutional Neural Network (CNN) to predict a load Reduction Factor RF for a certain perforated monopile geometry. The neural network considers a few convolutional layers with an optional linear regression head. The model is based on models used in related similar surrogate models and shows good promise for these kind of problems (answering subquestion 2).

Five different models were created and optimized, which uses different options for the input to the CNN. Three different representations of the geometry, which consist of using a Signed Distance Function (SDF), decomposing the SDF in its x and y distance components and a binary representation were used. Furthermore, two more models are considered, which uses the SDF or the distance components along with the flow fields as input to the CNN model.

A dataset was created to train these models with a total of $15\,561$ samples. These mainly considered monopiles with different number of perforations, porosities and 'angle of attacks'. As the perforated monopile was simplified and idealized to a 2D geometry. With rectangular equally and uniformed spaced perforations. While the flow around and through perforated monopile is essentially 3D this simplification is chosen to considerably reduce the required computational resources and parameters needed for this project.

The non-linear behaviour between these input parameters and the RF has been shown. It's shown that decreasing the number of perforations and increasing the porosity has the most influence on decreasing the reduction factor (answering subquestion 1). As the projected frontal area of the perforated monopile determines predominately the value of the RF, in the absence of turbulence.

All five models showed promising result in its ability to predict the RF within a reasonable (relative) error rate for its intended purpose in the early design phase for geometry optimization.

A general trend is observed that the more information is given to the neural network the better the accuracy will be. With the sole exception of the SDF input model, but it's hypothesized that its hyperparameters were badly optimized. Or that even more convolutional layers are needed, then was accounted for in the hyperparameter optimization search space.

Each of the models showed particularly difficulty in accurately predict the RF for cases where the number of perforations is equal to two and a high porosity (at least 50% or higher). It's likely related that there is only a strong non-linear behaviour between the RF and the angle of attack for only two perforations. When accounting for these few edge cases and ensuring the model is not used with these input parameters the expected maximum relative error rate goes down considerably.

The model that uses the (implicit) representation of the geometry using the distance components with the flow fields has the best accuracy among the five models (answering subquestion 2). With a median relative error rate of $0.27\,\%$ and maximum relative error rate of only $0.94\,\%$, based on the (test) dataset (answering subquestion 4). However, to iterate the point made above, each of the models performs adequately for the intended application. And therefore there is also the case to be made to rather use the SDF model (without additional flow field input). For a minimal reduction in accuracy the model can be kept more simple. Though, the inclusion of the flow fields might have a bigger effect on the accuracy when turbulence is included.

Furthermore, the results show that a speed-up of 386 is achieved by using our surrogate model over the full CFD model (answering subquestion 5). This is likely to improve considerably more when using a more sophisticated CFD model for creating the dataset and using it as a benchmark for this metric.

As the relationship between the geometric input parameters and the RF (in the absence of turbulence) is quite simple in nature. More traditional machine learning techniques could likely perform just as well, or even better, than the CNN model used. And at the same time being even faster (to train and make predictions) and more memory efficient. However, this is not expected to be the case when turbulence is included. Furthermore, by using the CNN model the model can be easier generalized to (slight) alterations in the geometry from the basic input geometric parameters that were used to create the dataset.

Finally, the presented research is a first step (and first attempt) in the creation of a surrogate model for the characterization of hydrodynamic loads on perforated monopiles. Along with other (follow-up) research it is the next step in the design (and ultimately the use in industry) of the perforated monopile concept.

Furthermore, this thesis applies deep learning to create a surrogate model for a very specific geometry, which isn't used much yet. Previous works mostly focussed efforts on more generalized CFD surrogate models. With datasets generated from mostly convex blob geometries. One of the main challenge of this

project is the thin-walled nature of the perforated monopile and small size of the remaining structure with respect to the pixel resolution. This required extra care in the limitations on the geometry parameters, pixel resolution size and mesh refinement. In contrast to (some) previous work the SDF (and distance components) outperformed the binary representation of the geometry.

The use of the distance components of the Signed Distance Function and the flow fields as (additional) input to the CNN model is a novel attempt at improving deep learning performance for CFD regression problems. The overall pipelines used in this thesis can also easily be adopted for other domain specific CFD regression problems.

6.1. Proposed model

Using the combination of an implicit representation of the geometry, using the distance components of the SDF, and the flow fields as input for the surrogate model results in the best accuracy. Obviously, when considering new configurations these flow fields are not readily available. And need to be derived from the input geometry.

For this the model in Figure 6.1 is proposed. It uses two different CNN models that need to be trained separately.

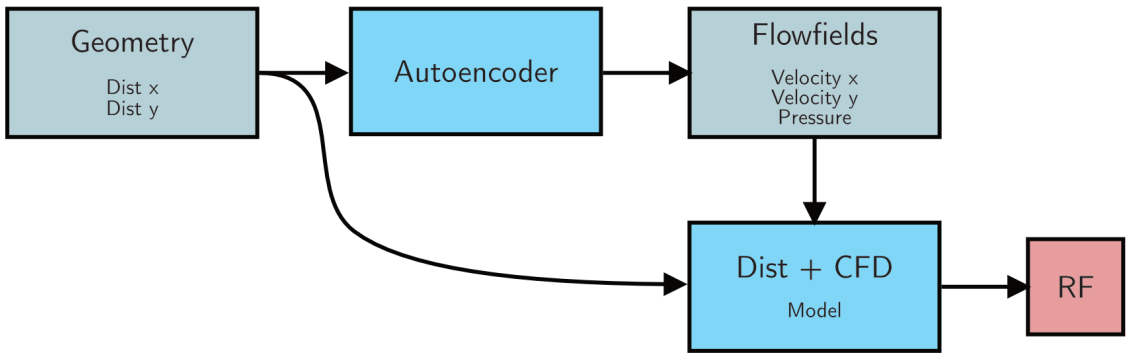


Figure 6.1: Proposed CNN surrogate model for the characterization of the reduction in hydrodynamic load on perforated monopiles.

The first model is an autoencoder network, similarly to reconstruct the flow fields from an implicit representation of the geometry (using signed distance fields). Such an autoencoder already has been shown to work in previous research, such as [30, 24, 31]. This part of the model has not been the aim of this research, due to time constraints on the project.

The second model has been the focus of this project. The only change required would be to use the output of the autoencoder network instead of using the flow fields directly from the full order CFD model.

7

Recommendations

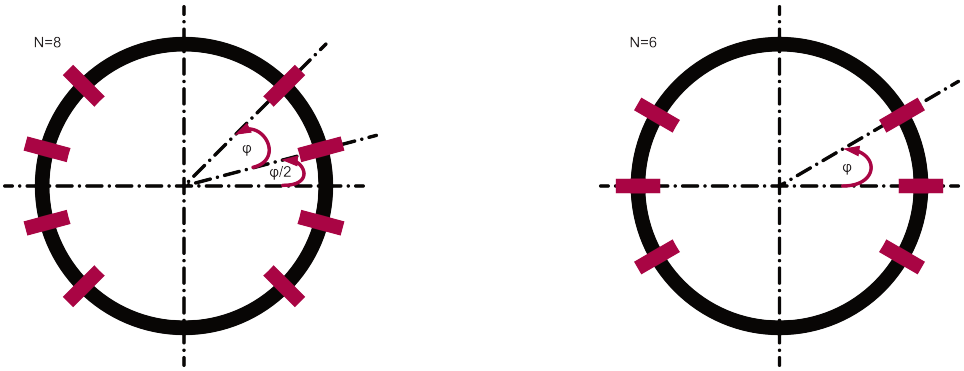
There are several directions that could be explored for further research opportunities. Due to limits in the available computing resources and time available for the thesis several simplifications have been made that could be changed. Furthermore, some recommendations are given to further improve the accuracy of the surrogate model. And to improve the applicability of the model for engineering purposes.

7.1. Geometry

Currently, the model considers a simplified 2D geometry description of a perforated monopile. However, the flow through a perforated monopile is fundamentally 3D in nature. A 3D perforated monopile could consider more design parameters. For example, the (elliptical) shape of the perforations and the column and row spacing of the perforations.

Luckily, adopting 3D geometries would require relatively little extra work. As the SDF that is used to translate the geometry description to something that could be used as input to the CNN model could be easily extended to more dimensions. Furthermore, (Py)GMSH allows for 3D (conformal) mesh generation. And very little change would be required for the setup of the CNN model. The main drawback would be the significant increase in computing power to both create the dataset and train the models.

In this thesis and previous research on perforated monopiles a uniform and linear spacing of the perforations is assumed. Often at the wind turbine park site location there might be a dominant wave direction. Perhaps, only a few perforations have to be made in along the axis of the dominant wave direction. An example of such a perforation distribution is illustrated in Figure 7.1.



(a) Even number of perforations per side.

(b) Uneven number of perforations per side.

Figure 7.1: Illustrative example for a perforation distribution for a dominant wave direction.

However, there is a potential danger in using such a perforation distribution. Research has shown that for such perforation distribution the hydrodynamic load on the perforated could be higher compared to the non-perforated monopile. This could happen for large frontal perforations, which can make the rear part of the monopile relevant for the drag. Or for cases when the wave direction is perpendicular to the dominant wave direction (with a low number of perforations), which can cause increased coherence of shed vortices. [25]

Even when only uniform perforation distribution will be produced there is a clear need to be able to study these effects in the design phase. As there is a potential risk of perforations that could get (partially) obstructed. This would increase the RF, which could become even higher than one, and affect the fatigue life of the monopile. Example of possible obstructions of the perforations could be caused due to marine growth, ice accumulation or marine debris.

7.2. Dataset resolution

The dataset was created on the relatively low resolution of 256 times 128 pixel resolution. As was mentioned this proved to be a bit too low to fully and properly represent the thin-walled monopiles, which is the unique challenge for these geometries. An effective $\frac{D}{t}$ of 50 was used. While in practices for monopiles a $\frac{D}{t}$ between 80 and 120 is used. Therefore, a resolution of at least four times is required to account for the the lower ranges of possible $\frac{D}{t}$ values. Furthermore, this would mean the conformal mesh will become finer too. Obviously, a higher resolution would require a lot more computing resources, but the accuracy of the surrogate model should improve.

Another method to solve this problem is to move away from Euclidean structured grid (image-like) inputs. Instead of using the SDF to represent the geometry, the geometry could alternatively be represented by geometric signals. One major is the advantage of much lower representation loss (alliasing) of the geometry for the same resolution. [71]

7.3. Choice of CFD model

As is mentioned before, from a practical consideration due to limited computing resources, only a ‘creeping flow’ is used for the CFD model. However, the flow around a monopile is highly turbulent in nature. Therefore, a better and more computational expensive CFD model should be used to generate the dataset, which account for the presence of turbulence.

Furthermore, to properly account for the wave loading on the (perforated) monopile and characterize the expected fatigue damage a wave-induced oscillatory flow should be considered instead of a constant inflow velocity. This would require solving the flow in the time domain. [17, 21] It would also be desirable if the surrogate model could account for a range of different inflow parameters (sea states).

Also stated above research has shown that the loads for a perforated monopile could increase compared to the non-perforated monopile. Turbulence needs to be taken into account to include these scenarios.

7.4. Structural analysis model

The focus of this research is to develop a surrogate model for the CFD model in the design of a perforated monopile. However, the CFD model evaluation for the loads on the monopile is just part of the overall design steps. To optimize the design for a perforated monopile other models are still required, see Figure 1.9.

For example, a structural analysis model that determines the stresses on the structure and quantifies the fatigue limit state for a certain geometry, flow and materials used. Some research for this is done in [21]. Or a cost model that takes (for example) into account the material cost that is saved by introducing perforations, the increased fatigue life and increased cost due to any additional manufacturing steps.

Especially, for the structural analysis model a surrogate model may be needed to be able to evaluate a lot of geometries within a certain timeframe for an optimization study. Finally, in this research no Fluid Structure Interaction (FSI) is taken into account. But potentially the effect of the tip deflection of the monopile under wave loading and its effect on the flow needs to be taken into account.

7.5. Other surrogate models

Though not part of this research, it would be interesting the compare the results of this research with results of other types of surrogate models for the characterization of (hydrodynamic) load reduction of perforated monopiles. To verify the benefits of the deep learning approach over more traditional machine learning techniques. Or physical based surrogate modelling.

Especially, as the use of deep learning in this particular case (stokes flow, no turbulence), is a bit overkill. And perhaps a simpler, faster and more accurate surrogate model could be obtained by creating a surrogate model that directly maps the geometric parameters (number of perforations, porosity and ‘angle of attack’) directly to the Reduction Factor (RF).

7.6. Application in other engineering problems

While previous research focussed predominantly on general applicable models, which are trained often on a dataset for geometries of convex blobs. Likely such models would perform worse for geometries, which are quite removed from the geometry of interest, in comparison to a surrogate model specifically trained for these geometries (as is done in this research). This hypothesis should be confirmed in some future research. Sadly such a comparison couldn’t yet be made in this thesis due to the readily unavailability of these models to be used.

In this thesis a surrogate model is made to learn a (hydrodynamic) characteristic based on some input geometry. It should be noted that the approach taken in this thesis could be easily adapted to completely different geometries and engineering problems. The code to generate the geometries for both the creation of the SDF and the mesh files in GMSH consist of just a few tens of lines of code. With more abstraction this could even be further reduced in half. The dataset creation pipelines wouldn’t really have to be changed. Furthermore, most of the code regarding the deep learning part could be reused or would need to be changed little. Finally, CFD model could easily be swapped out for something more sophisticated, which accounts for the presence of turbulence.

Bibliography

[1] Global Wind Energy Council. “Global Wind Report 2022”. In: *Global Wind Energy Council: Brussels, Belgium* (2022).

[2] IRENA. *FUTURE OF WIND: Deployment, Investment, Technology, Grid Integration and Socio-Economic Aspects*. 2019. ISBN: 978-92-9260-155-3.

[3] Ivan Komusanac et al. *Wind Energy in Europe. 2021 Statistics and the Outlook for 2022-2026*. 2021.

[4] Windeurope. *Offshore Wind Energy in the North Sea*. Windeurope, Nov. 2017.

[5] Laura Florentina Gusatu et al. “A Spatial Analysis of the Potentials for Offshore Wind Farm Locations in the North Sea Region: Challenges and Opportunities”. In: *ISPRS International Journal of Geo-Information* 9.2 (2 Feb. 2020), p. 96. ISSN: 2220-9964. DOI: 10.3390/ijgi9020096. URL: https://www.mdpi.com/2220-9964/9/2/96.

[6] Ramon Varghese, Vikram Pakrashi, and Subhamoy Bhattacharya. “A Compendium of Formulae for Natural Frequencies of Offshore Wind Turbine Structures”. In: *Energies* 15.8 (8 Jan. 2022), p. 2967. ISSN: 1996-1073. DOI: 10.3390/en15082967. URL: https://www.mdpi.com/1996-1073/15/8/2967.

[7] Mehmet Bilgili and Hakan Alphan. “Global Growth in Offshore Wind Turbine Technology”. In: *Clean Technologies and Environmental Policy* (Apr. 15, 2022). ISSN: 1618-954X, 1618-9558. DOI: 10.1007/s10098-022-02314-0. URL: https://link.springer.com/10.1007/s10098-022-02314-0.

[8] R Damiani, K Dykes, and G Scott. “A Comparison Study of Offshore Wind Support Structures with Monopiles and Jackets for U.S. Waters”. In: *Journal of Physics: Conference Series* 753 (Sept. 2016), p. 092003. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/753/9/092003. URL: https://iopscience.iop.org/article/10.1088/1742-6596/753/9/092003 (visited on 06/27/2022).

[9] Tyler Stehly and Patrick Duffy. *2020 Cost of Wind Energy Review*. National Renewable Energy Laboratory, 2022, p. 77.

[10] *Global Offshore Renewable Map | 4C Offshore*. URL: https://www.4coffshore.com/offshorewind/.

[11] J. N. Sørensen, G. C. Larsen, and A. Cazin-Bourguignon. “Production and Cost Assessment of Offshore Wind Power in the North Sea”. In: *Journal of Physics: Conference Series* 1934.1 (May 2021), p. 012019. ISSN: 1742-6596. DOI: 10.1088/1742-6596/1934/1/012019. URL: https://doi.org/10.1088/1742-6596/1934/1/012019.

[12] Beate Geyer et al. “Climatology of North Sea Wind Energy Derived from a Model Hindcast for 1958–2012”. In: *Journal of Wind Engineering and Industrial Aerodynamics* 147 (Dec. 1, 2015), pp. 18–29. ISSN: 0167-6105. DOI: 10.1016/j.jweia.2015.09.005. URL: https://www.sciencedirect.com/science/article/pii/S0167610515002214.

[13] Jharna Pokhrel and Junwon Seo. “Statistical Model for Fragility Estimates of Offshore Wind Turbines Subjected to Aero-Hydro Dynamic Loads”. In: *Renewable Energy* 163 (Jan. 2021), pp. 1495–1507. ISSN: 09601481. DOI: 10.1016/j.renene.2020.10.015. URL: https://linkinghub.elsevier.com/retrieve/pii/S0960148120315810.

[14] Nikolaos Skliris et al. “Assessing Extreme Environmental Loads on Offshore Structures in the North Sea from High-Resolution Ocean Currents, Waves and Wind Forecasting”. In: *Journal of Marine Science and Engineering* 9.10 (Sept. 24, 2021), p. 1052. ISSN: 2077-1312. DOI: 10.3390/jmse9101052. URL: https://www.mdpi.com/2077-1312/9/10/1052.

[15] Pim van der Male, Marco Vergassola, and Karel van Dalen. “Decoupled Modelling Approaches for Environmental Interactions with Monopile-Based Offshore Wind Support Structures”. In: *Energies* 13.19 (Oct. 5, 2020), p. 5195. ISSN: 1996-1073. DOI: 10.3390/en13195195. URL: https://www.mdpi.com/1996-1073/13/19/5195 (visited on 08/21/2022).

[16] M. C. Anderson. “The Hybrid Monopile: Design of a Novel Foundation Structure for Large Offshore Wind Turbines in Intermediate Water Depths”. In: (2017). URL: https://repository.tudelft.nl/islandora/object/uuid%3Ab67deab1-d5fd-46ea-9a8f-765d3cdf2485.

[17] Jacob Andersen et al. “Wave Load Mitigation by Perforation of Monopiles”. In: *Journal of Marine Science and Engineering* 8.5 (May 16, 2020), p. 352. ISSN: 2077-1312. DOI: 10.3390/jmse8050352. URL: https://www.mdpi.com/2077-1312/8/5/352.

[18] Shaohua Wang et al. “Effect of Inclination Angles on the Local Scour around a Submerged Cylinder”. In: *Water* 12.10 (Sept. 25, 2020), p. 2687. ISSN: 2073-4441. DOI: 10.3390/w12102687. URL: https://www.mdpi.com/2073-4441/12/10/2687.

[19] Manish Pandey et al. “Reduction of Scour around Circular Piers Using Collars”. In: *Journal of Flood Risk Management* 15.3 (Sept. 2022). ISSN: 1753-318X, 1753-318X. DOI: 10.1111/jfr3.12812. URL: https://onlinelibrary.wiley.com/doi/10.1111/jfr3.12812.

[20] Monica M. Maher. “The Corrosion and Biofouling Characteristics of Sealed vs. Perforated Offshore Monopile Interiors: Experiment Design Comparing Corrosion and Environment Inside Steel Pipe”. Thesis. Dec. 2018. URL: https://repository.lib.fit.edu/handle/11141/2703.

[21] Jorne van der Ploeg. *Perforation of Monopiles to Reduce Hydrodynamic Loads and Enable Use in Deep Waters*. 2021. URL: https://repository.tudelft.nl/islandora/object/uuid%3A91eada6f-4f2b-4ae6-be59-2b5ff0590c6f.

[22] Oriol Colomés et al. “A Weighted Shifted Boundary Method for Free Surface Flow Problems”. In: *Journal of Computational Physics* 424 (Jan. 1, 2021), p. 109837. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2020.109837. URL: https://www.sciencedirect.com/science/article/pii/S0021999120306112.

[23] Sabrina Kelbij Star et al. “Reduced Order Models for the Incompressible Navier□Stokes Equations on Collocated Grids Using a ‘Discretize□then□project’ Approach”. In: *International Journal for Numerical Methods in Fluids* 93.8 (Aug. 2021), pp. 2694–2722. ISSN: 0271-2091, 1097-0363. DOI: 10.1002/fld.4994. URL: https://onlinelibrary.wiley.com/doi/10.1002/fld.4994.

[24] Xiaoxiao Guo, Wei Li, and Francesco Iorio. “Convolutional Neural Networks for Steady Flow Approximation”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco California USA: ACM, Aug. 13, 2016, pp. 481–490. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939738. URL: https://dl.acm.org/doi/10.1145/2939672.2939738.

[25] K. Steiros et al. “The Effect of Porosity on the Drag of Cylinders”. In: *Journal of Fluid Mechanics* 901 (Oct. 25, 2020). URL: https://doi.org/10.1017/jfm.2020.606.

[26] Aristides AG Requicha and Herbert B Voelcker. “Constructive Solid Geometry”. In: (1977).

[27] Saurabh Gupta et al. *Learning Rich Features from RGB-D Images for Object Detection and Segmentation*. July 22, 2014. DOI: 10.48550/arXiv.1407.5736. arXiv: 1407.5736 [cs]. URL: http://arxiv.org/abs/1407.5736.

[28] Longyan Wang et al. “A Deep Learning-Based Optimization Framework of Two-Dimensional Hydrofoils for Tidal Turbine Rotor Design”. In: *Energy* 253 (Aug. 15, 2022), p. 124130. ISSN: 0360-5442. DOI: 10.1016/j.energy.2022.124130. URL: https://www.sciencedirect.com/science/article/pii/S0360544222010337.

[29] Jiang-Zhou Peng et al. “Data-Driven Modeling of Geometry-Adaptive Steady Heat Convection Based on Convolutional Neural Networks”. In: *Fluids* 6.12 (12 Dec. 2021), p. 436. ISSN: 2311-5521. DOI: 10.3390/fluids6120436. URL: https://www.mdpi.com/2311-5521/6/12/436.

[30] Matthias Eichinger, Alexander Heinlein, and Axel Klawonn. “Stationary Flow Predictions Using Convolutional Neural Networks”. In: *Numerical Mathematics and Advanced Applications ENUMATH 2019*. Ed. by Fred J. Vermolen and Cornelis Vuik. Lecture Notes in Computational Science and Engineering. Cham: Springer International Publishing, 2021, pp. 541–549. ISBN: 978-3-030-55874-1. DOI: 10.1007/978-3-030-55874-1_53.

[31] Mateus Dias Ribeiro et al. *DeepCFD: Efficient Steady-State Laminar Flow Approximation with Deep Convolutional Neural Networks*. Nov. 26, 2021. DOI: 10.48550/arXiv.2004.08826. arXiv: 2004.08826 [physics]. URL: http://arxiv.org/abs/2004.08826.

[32] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Applied Mathematical Sciences 153. New York Berlin Heidelberg: Springer, 2003. 273 pp. ISBN: 978-0-387-95482-0.

[33] Inigo Quilez. *Inigo Quilez*. URL: https://iquilezles.org.

[34] Adam Paszke et al. “Automatic Differentiation in PyTorch”. In: *NIPS Autodiff Workshop*. 2017.

[35] Koldo Portal-Porras et al. “Alternative Artificial Neural Network Structures for Turbulent Flow Velocity Field Prediction”. In: *Mathematics* 9.16 (16 Jan. 2021), p. 1939. ISSN: 2227-7390. DOI: 10.3390/math9161939. URL: https://www.mdpi.com/2227-7390/9/16/1939.

[36] Jeff Bezanson et al. “Julia: A Fresh Approach to Numerical Computing”. In: *SIAM review* 59.1 (2017), pp. 65–98. URL: https://doi.org/10.1137/141000671.

[37] Santiago Badia and Francesc Verdugo. “Gridap: An Extensible Finite Element Toolbox in Julia”. In: *Journal of Open Source Software* 5.52 (2020), p. 2520. DOI: 10.21105/joss.02520. URL: https://doi.org/10.21105/joss.02520.

[38] Joel H. Ferziger, Milovan Perić, and Robert L. Street. *Computational Methods for Fluid Dynamics*. 4th ed. 2020 edition. Cham: Springer, Aug. 28, 2019. 614 pp. ISBN: 978-3-319-99691-2.

[39] Hermann Schlichting and Klaus Gersten. *Boundary-Layer Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017. ISBN: 978-3-662-52917-1. DOI: 10.1007/978-3-662-52919-5.

[40] Christophe Geuzaine and Jean-François Remacle. “Gmsh: A 3-D Finite Element Mesh Generator with Built-in Pre- and Post-Processing Facilities”. In: *International Journal for Numerical Methods in Engineering* 79.11 (2009), pp. 1309–1331. ISSN: 1097-0207. DOI: 10.1002/nme.2579. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2579.

[41] Nico Schlömer. *Pygmsh: A Python Frontend for Gmsh*. DOI: 10.5281/zenodo.1173105. URL: https://github.com/nschloe/pygmsh.

[42] Balaje Kalyanaraman. *Wrapping up GSoC*. Balaje Kalyanaraman. URL: https://balaje.github.io/2021/08/20/Wrapping-up.html.

[43] Maurice Herlihy and Nir Shavit. *The Art of Multiprocessor Programming*. Revised first edition. Amsterdam: Morgan Kaufmann, 2012. 508 pp. ISBN: 978-0-12-397337-5.

[44] Charles R Harris et al. “Array Programming with NumPy”. In: *Nature* 585.7825 (2020), pp. 357–362. ISSN: 1476-4687. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[45] Wes McKinney. “Data Structures for Statistical Computing in Python”. In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 51–56.

[46] Oren Ben-Kiki, Clark Evans, and Brian Ingerson. “YAML Ain’t Markup Language (YAML)(Tm) Version 1.2”. In: *YAML. org, Tech. Rep* 359 (2009).

[47] Python Core Team. *Python: A Dynamic, Open Source Programming Language*. manual. Python Software Foundation. 2019. URL: https://www.python.org/.

[48] *MPIRE*. Slimmer AI. URL: https://slimmer-ai.github.io/mpire/.

[49] Nvidia. *Deep Learning Performance Documentation*. Nvidia accelerated computing. URL: http://docs.nvidia.com/deeplearning/frameworks/dl-performance-matrix-multiplication/index.html.

[50] Dan Kallehave et al. “Optimization of Monopiles for Offshore Wind Turbines”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 373.2035 (Feb. 28, 2015), p. 20140100. ISSN: 1364-503X, 1471-2962. DOI: 10.1098/rsta.2014.0100. URL: https://royalsocietypublishing.org/doi/10.1098/rsta.2014.0100.

[51] Michael K. McWilliam et al. “Conceptual Monopile and Tower Sizing for the IEA Wind Task 37 Borssele Reference Wind Farm”. In: *Journal of Physics: Conference Series* 2018.1 (Sept. 1, 2021), p. 012025. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/2018/1/012025. URL: https://iopscience.iop.org/article/10.1088/1742-6596/2018/1/012025.

[52] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[53] Gregory Gundersen. *From Convolution to Neural Network*. URL: https://gregorygundersen.com/blog/2017/02/24/cnns/ (visited on 08/15/2022).

[54] Yann LeCun et al. “LeNet-5, Convolutional Neural Networks”. In: *URL: http://yann. lecun. com/exdb/lenet* 20.5 (2015), p. 14.

[55] Hongyang Dong et al. “A Deep Convolutional Neural Network for Real-Time Full Profile Analysis of Big Powder Diffraction Data”. In: *npj Computational Materials* 7.1 (1 May 21, 2021), pp. 1–9. ISSN: 2057-3960. DOI: 10.1038/s41524-021-00542-4. URL: https://www.nature.com/articles/s41524-021-00542-4.

[56] Yao Zhang, Woong Je Sung, and Dimitri N. Mavris. “Application of Convolutional Neural Network to Predict Airfoil Lift Coefficient”. In: *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference. Kissimmee, Florida: American Institute of Aeronautics and Astronautics, Jan. 8, 2018. ISBN: 978-1-62410-532-6. DOI: 10.2514/6.2018-1903. URL: https://arc.aiaa.org/doi/10.2514/6.2018-1903.

[57] Haris Iqbal. *HarisIqbal88/PlotNeuralNet v1.0.0*. Version v1.0.0. Zenodo, Dec. 25, 2018. DOI: 10.5281/ZENODO.2526395. URL: https://zenodo.org/record/2526395.

[58] Zhenwei Dai and Reinhard Heckel. *Channel Normalization in Convolutional Neural Network Avoids Vanishing Gradients*. July 22, 2019. DOI: 10.48550/arXiv.1907.09539. arXiv: 1907.09539 [cs, stat]. URL: http://arxiv.org/abs/1907.09539.

[59] Sébastien Marcel and Yann Rodriguez. “Torchvision the Machine-Vision Package of Torch”. In: *Proceedings of the International Conference on Multimedia - MM ’10*. The International Conference. Firenze, Italy: ACM Press, 2010, p. 1485. ISBN: 978-1-60558-933-6. DOI: 10.1145/1873951.1874254. URL: http://dl.acm.org/citation.cfm?doid=1873951.1874254.

[60] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: 2015, pp. 448–456. URL: http://jmlr.org/proceedings/papers/v37/ioffe15.pdf.

[61] Connor Shorten and Taghi M. Khoshgoftaar. “A Survey on Image Data Augmentation for Deep Learning”. In: *Journal of Big Data* 6.1 (July 6, 2019), p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0.

[62] Alvaro Abucide-Armas et al. “A Data Augmentation-Based Technique for Deep Learning Applied to CFD Simulations”. In: *Mathematics* 9.16 (16 Jan. 2021), p. 1843. DOI: 10.3390/math9161843. URL: https://www.mdpi.com/2227-7390/9/16/1843.

[63] Nicki Skafte Detlefsen et al. *TorchMetrics - Measuring Reproducibility in PyTorch*. Feb. 2022. DOI: 10.21105/joss.04101. URL: https://github.com/PyTorchLightning/metrics.

[64] Paulius Micikevicius et al. *Mixed Precision Training*. Feb. 15, 2018. DOI: 10.48550/arXiv.1710.03740. arXiv: 1710.03740 [cs, stat]. URL: http://arxiv.org/abs/1710.03740.

[65] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. Jan. 4, 2019. DOI: 10.48550/arXiv.1711.05101. arXiv: 1711.05101 [cs, math]. URL: http://arxiv.org/abs/1711.05101.

[66] Takuya Akiba et al. *Optuna: A Next-generation Hyperparameter Optimization Framework*. July 25, 2019. DOI: 10.48550/arXiv.1907.10902. arXiv: 1907.10902 [cs, stat]. URL: http://arxiv.org/abs/1907.10902.

[67] Omry Yadan. *Hydra - A Framework for Elegantly Configuring Complex Applications*. Github. 2019. URL: https://github.com/facebookresearch/hydra.

[68] James Bergstra et al. “Algorithms for Hyper-Parameter Optimization”. In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[69] James Bergstra, Daniel Yamins, and David Cox. “Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures”. In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 1. Atlanta, Georgia, USA: PMLR, June 17–19, 2013, pp. 115–123. URL: https://proceedings.mlr.press/v28/bergstra13.html.

[70] Mattias De Charleroy. *Hydra Optuna Sweeper Max Failure Rate Support 1513*. GitHub. URL: https://github.com/facebookresearch/hydra/pull/2215.

[71] Chiyu Max Jiang et al. “Convolutional Neural Networks on Non-uniform Geometrical Signals Using Euclidean Spectral Transformation”. In: International Conference on Learning Representations. Jan. 10, 2019. URL: https://openreview.net/forum?id=B1G5ViAqFm.

A

Learning curves

In this appendix the learning curves during training of the (hyperparameter optimized) models are given. The details of the model architecture can be seen in Appendix B.

A.1. Binary

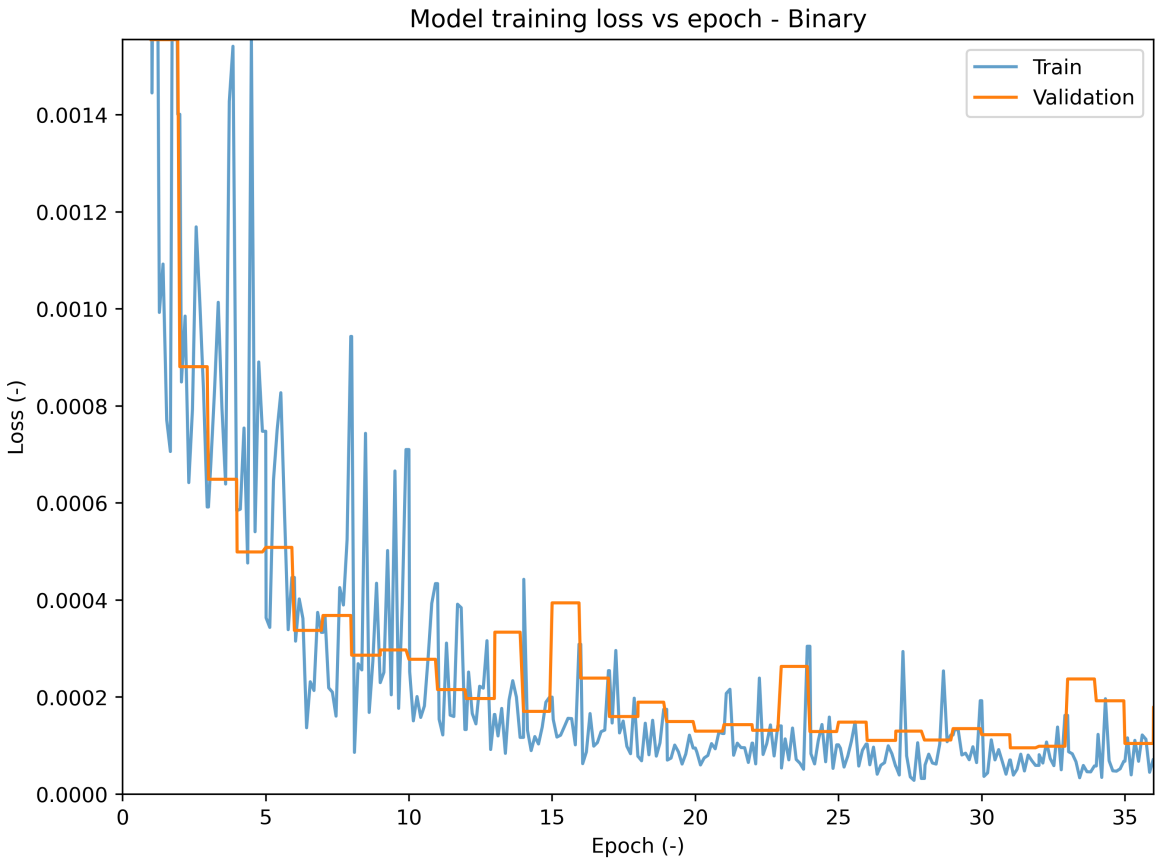


Figure A.1: Loss curve for the Binary model.

A.2. SDF

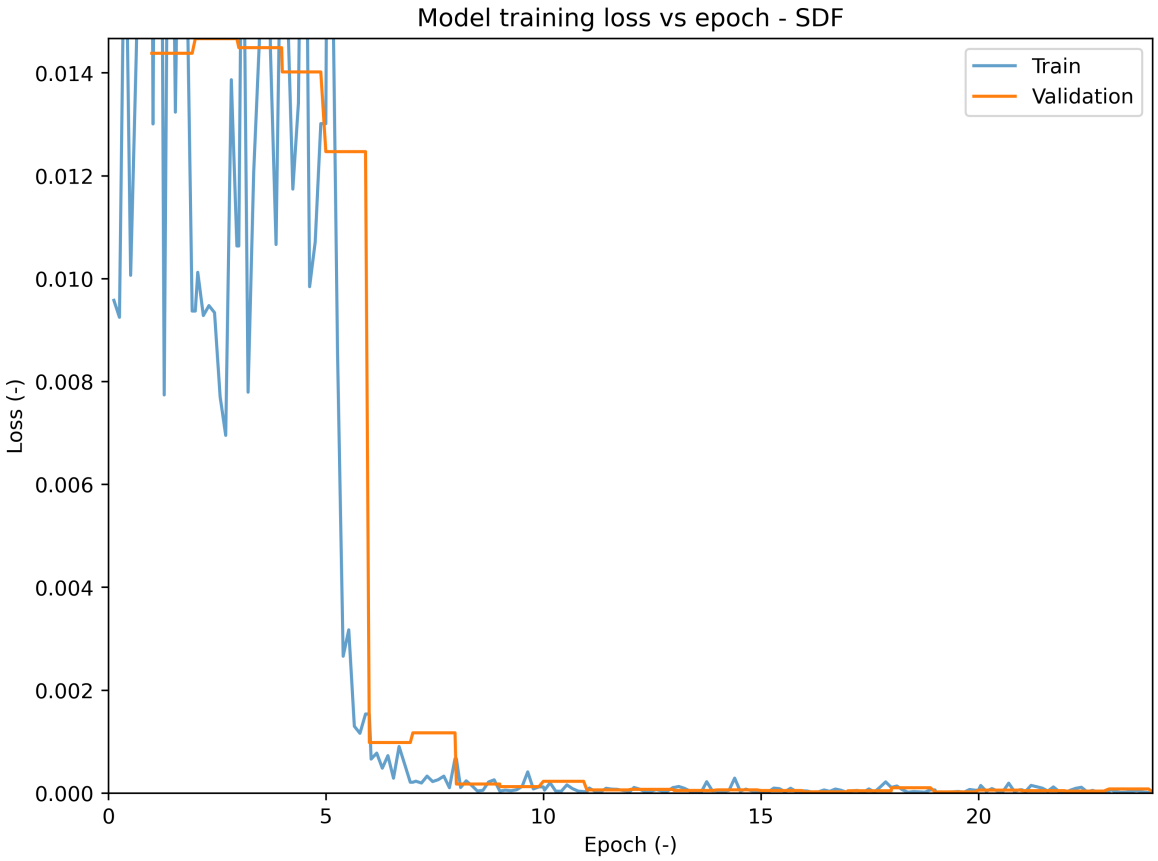


Figure A.2: Loss curve for the SDF model.

A.3. SDF + CFD

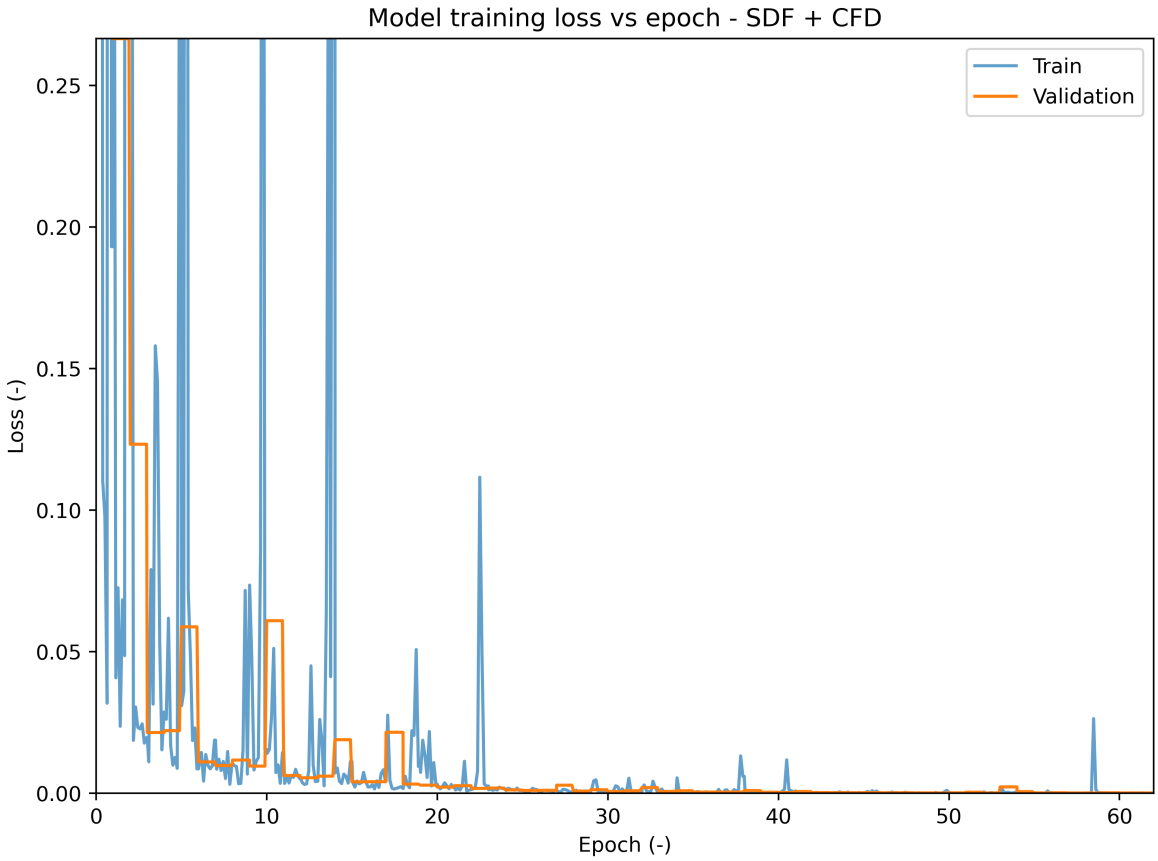


Figure A.3: Loss curve for the SDF + CFD model.

A.4. Dist

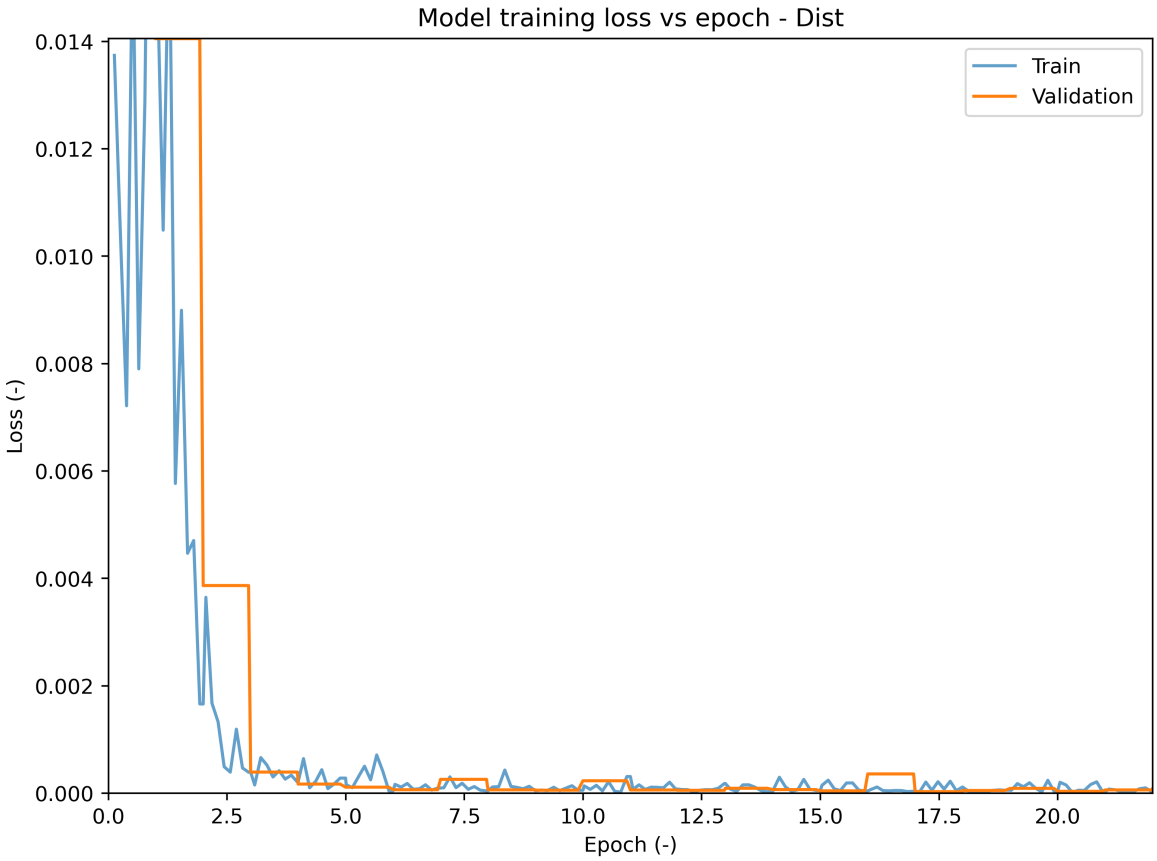


Figure A.4: Loss curve for the Dist model.

A.5. Dist + CFD

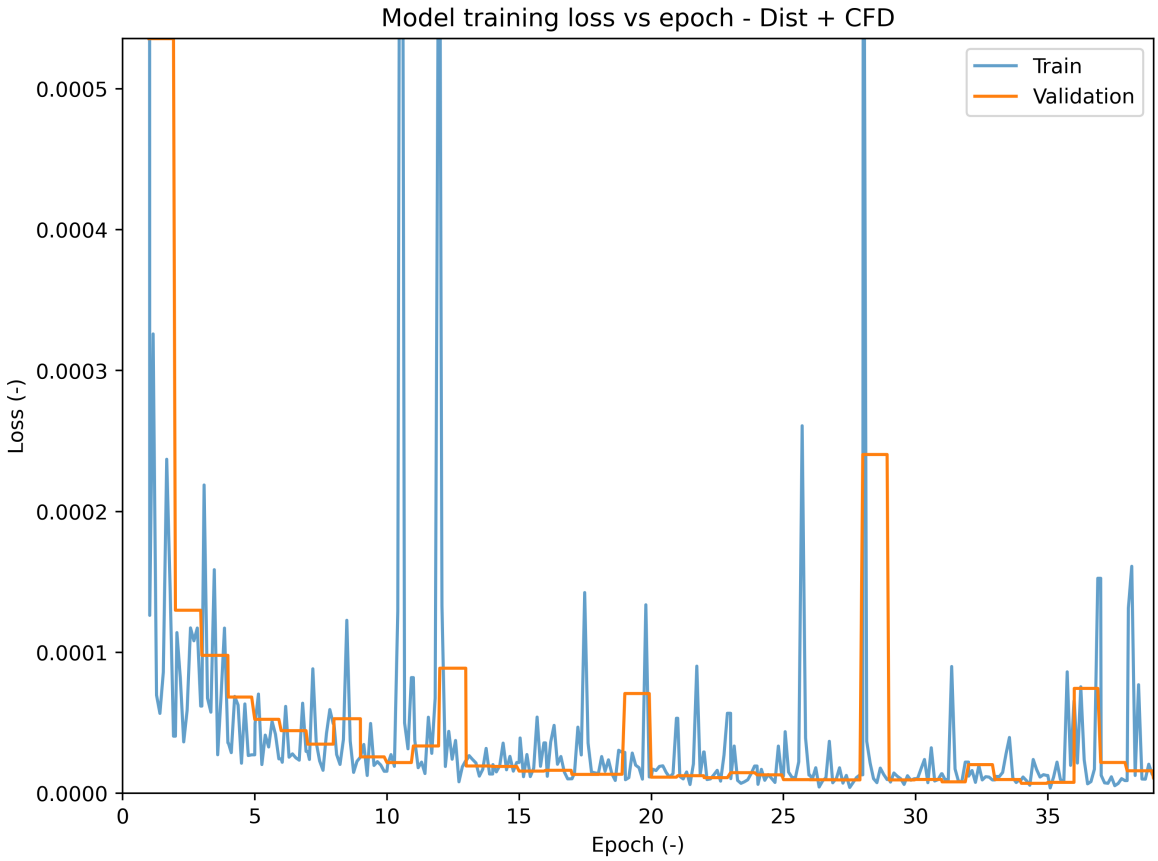


Figure A.5: Loss curve for the Dist + CFD model.

B

Model architecture details

In this appendix the details of each of the different CNN models trained and optimized in this thesis are given.

B.1. Binary

Table B.1: Binary input model details.

Parameter	Value
Number of input channels	1
Batch size	32
Optimizer	AdamW
Learning rate γ	1×10^{-5}
Weight decay λ	1×10^{-3}
Loss function	MSE
Test metric	MSE
Validation / train fraction	0.2
Model disk size	17.72 MB

Table B.2: Binary input model architecture details.

Layers				Parameters			Info	
Layer	Conv layer	#	Layer type	f	k	s	Output shape	Param #
Convolution	Conv layer 1	1	Conv2d	8	3	1	(8, 128, 256)	72
		2	BatchNorm2d				(8, 128, 256)	16
		3	ReLU				(8, 128, 256)	
		4	MaxPool2d		2	2	(8, 64, 128)	
	Conv layer 2	5	Conv2d	16	3	1	(16, 64, 128)	1152
		6	BatchNorm2d				(16, 64, 128)	32
		7	ReLU				(16, 64, 128)	
		8	MaxPool2d		2	2	(16, 32, 64)	
	Conv layer 3	9	Conv2d	32	3	1	(32, 32, 64)	4608
		10	BatchNorm2d				(32, 32, 64)	64
		11	ReLU				(32, 32, 64)	
		12	MaxPool2d		2	2	(32, 16, 32)	
	Conv layer 4	13	Conv2d	64	3	1	(64, 16, 32)	18 432
		14	BatchNorm2d				(64, 16, 32)	128
		15	ReLU				(64, 16, 32)	
		16	MaxPool2d		2	2	(64, 8, 16)	
	Conv layer 5	17	Conv2d	128	3	1	(128, 8, 16)	73 728
		18	BatchNorm2d				(128, 8, 16)	256
		19	ReLU				(128, 8, 16)	
		20	MaxPool2d		2	2	(128, 4, 8)	
	Conv layer 6	21	Conv2d	256	3	1	(256, 4, 8)	294 912
		22	BatchNorm2d				(256, 4, 8)	512
		23	ReLU				(256, 4, 8)	
		24	MaxPool2d		2	2	(256, 2, 4)	
	Flatten	25	Flatten				(2048)	
Regression head	Hidden layer 1	26	Linear	512			(512)	1 049 088
		27	ReLU				(512)	
	Hidden layer 2	28	Linear	128			(128)	65 664
		29	ReLU				(128)	
	Output layer	30	Linear	1			(1)	129
							Total number of parameters:	1 508 793

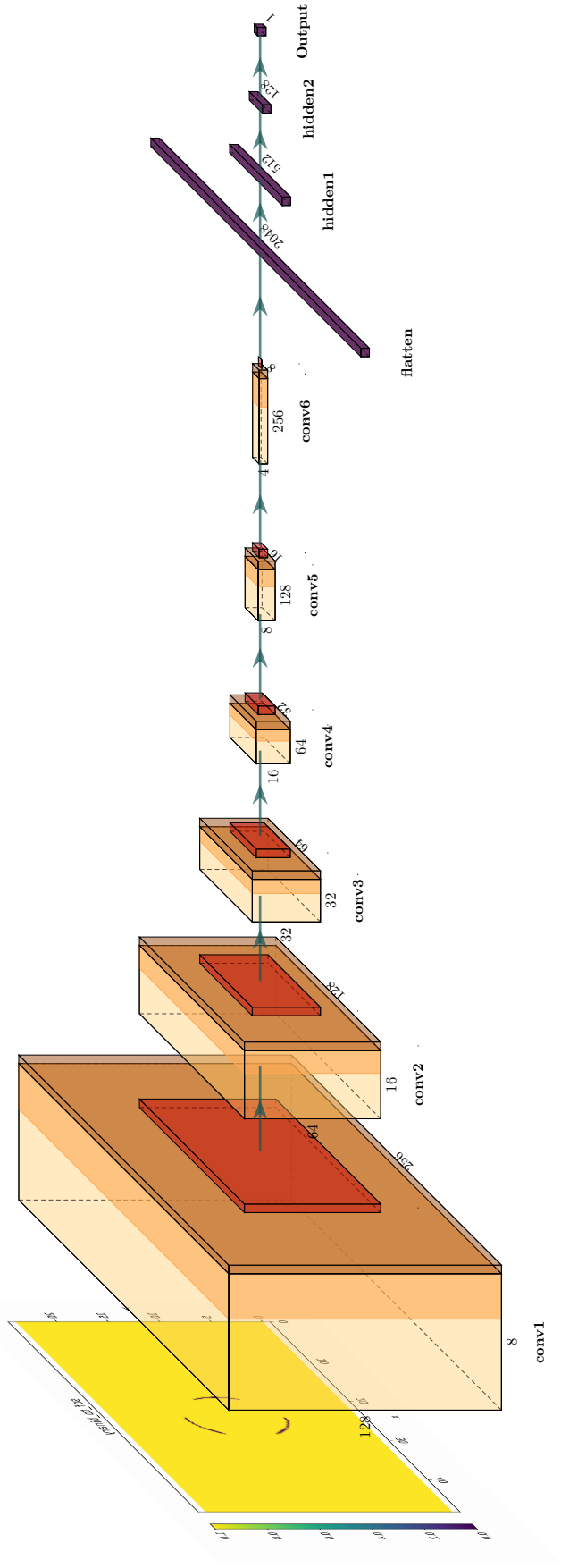


Figure B.1: Binary model architecture schematic. Yellow blocks show a 2D convolution layer with ReLU activation function, dark yellow a BatchNorm2d layer, red a Pool2D layer and finally purple show Linear layers.

B.2. SDF

Table B.3: SDF input model details.

Parameter	Value
Number of input channels	1
Batch size	32
Optimizer	AdamW
Learning rate γ	3×10^{-4}
Weight decay λ	1×10^{-2}
Loss function	MSE
Test metric	MSE
Validation / train fraction	0.2
Model disk size	18.50 MB

Table B.4: SDF input model architecture details.

Layers				Parameters			Info	
Layer	Conv layer	#	Layer type	f	k	s	Output shape	Param #
Convolution	Conv layer 1	1	Conv2d	16	3	1	(16, 128, 256)	160
		2	ReLU				(16, 128, 256)	
		3	MaxPool2d		2	2	(16, 64, 128)	
	Conv layer 2	4	Conv2d	32	3	1	(32, 64, 128)	4640
		5	ReLU				(32, 64, 128)	
		6	MaxPool2d		2	2	(32, 32, 64)	
	Conv layer 3	7	Conv2d	64	3	1	(64, 32, 64)	18 496
		8	ReLU				(64, 32, 64)	
		9	MaxPool2d		2	2	(64, 16, 32)	
	Conv layer 4	10	Conv2d	128	3	1	(128, 16, 32)	73 856
		11	ReLU				(128, 16, 32)	
		12	MaxPool2d		2	2	(128, 8, 16)	
	Conv layer 5	13	Conv2d	256	3	1	(256, 8, 16)	295 168
		14	ReLU				(256, 8, 16)	
		15	MaxPool2d		2	2	(256, 4, 8)	
	Conv layer 6	16	Conv2d	512	3	1	(512, 4, 8)	1 180 160
		17	ReLU				(512, 4, 8)	
		18	MaxPool2d		2	2	(512, 2, 4)	
	Flatten	19	Flatten				(4096)	
Regression head	Output layer	20	Linear	1			(1)	4097
Total number of parameters:								1 576 577

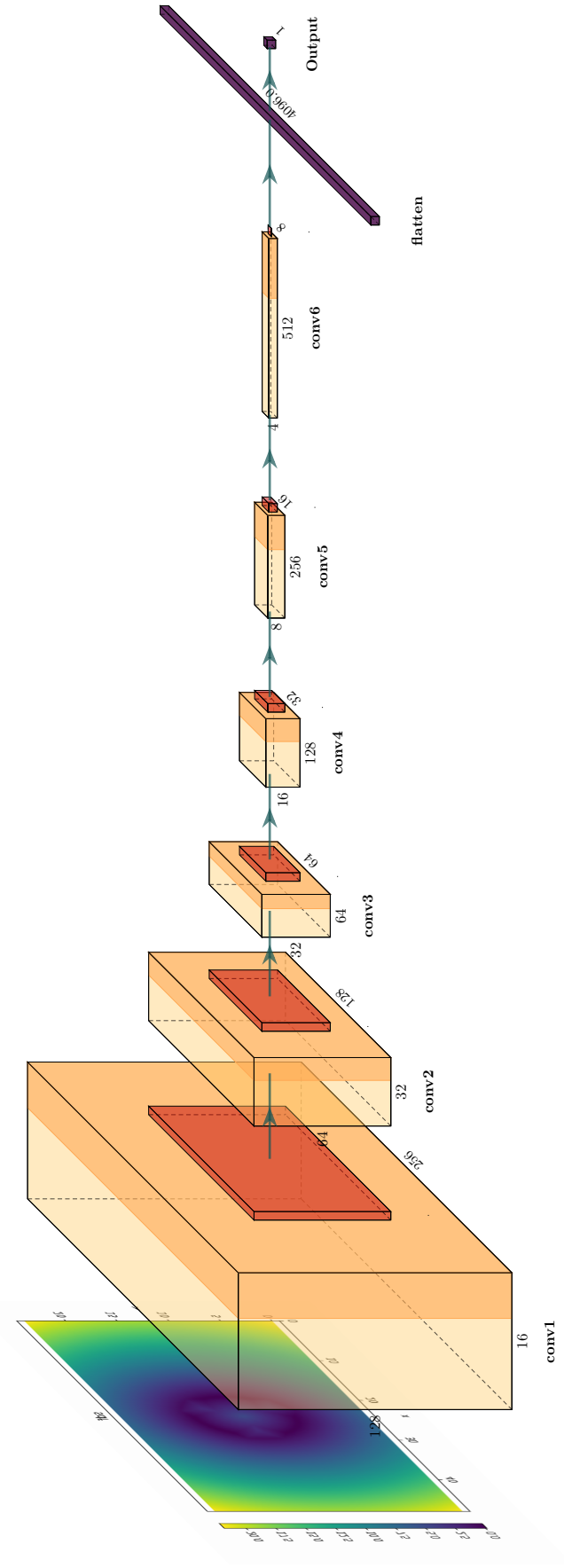


Figure B.2: SDF model architecture schematic. Yellow blocks show a 2D convolution layer with ReLU activation function, dark yellow a BatchNorm2d layer, red a Pool2D layer and finally purple show Linear layers.

B.3. SDF + CFD

Table B.5: SDF + CFD input model details.

Parameter	Value
Number of input channels	4
Batch size	32
Optimizer	AdamW
Learning rate γ	1×10^{-5}
Weight decay λ	0
Loss function	MSE
Test metric	MSE
Validation / train fraction	0.2
Model disk size	73.813 MB

Table B.6: SDF + CFD input model architecture details.

Layers				Parameters			Info	
Layer	Conv layer	#	Layer type	f	k	s	Output shape	Param #
Convolution	Conv layer 1	1	Conv2d	32	3	1	(32, 128, 256)	1184
		2	ReLU				(32, 128, 256)	
		3	MaxPool2d		2	2	(32, 64, 128)	
	Conv layer 2	4	Conv2d	64	3	1	(64, 64, 128)	18 496
		5	ReLU				(64, 64, 128)	
		6	MaxPool2d		2	2	(64, 32, 64)	
	Conv layer 3	7	Conv2d	128	3	1	(128, 32, 64)	73 856
		8	ReLU				(128, 32, 64)	
		9	MaxPool2d		2	2	(128, 16, 32)	
	Conv layer 4	10	Conv2d	256	3	1	(256, 16, 32)	295 168
		11	ReLU				(256, 16, 32)	
		12	MaxPool2d		2	2	(256, 8, 16)	
	Conv layer 5	13	Conv2d	512	3	1	(512, 8, 16)	1 180 160
		14	ReLU				(512, 8, 16)	
		15	MaxPool2d		2	2	(512, 4, 8)	
	Conv layer 6	16	Conv2d	1024	3	1	(1024, 4, 8)	4 719 616
		17	ReLU				(1024, 4, 8)	
		18	MaxPool2d		2	2	(1024, 2, 4)	
	Flatten	19	Flatten				(8192)	
Regression head	Output layer	20	Linear	1			(1)	8193
Total number of parameters:								6 296 673

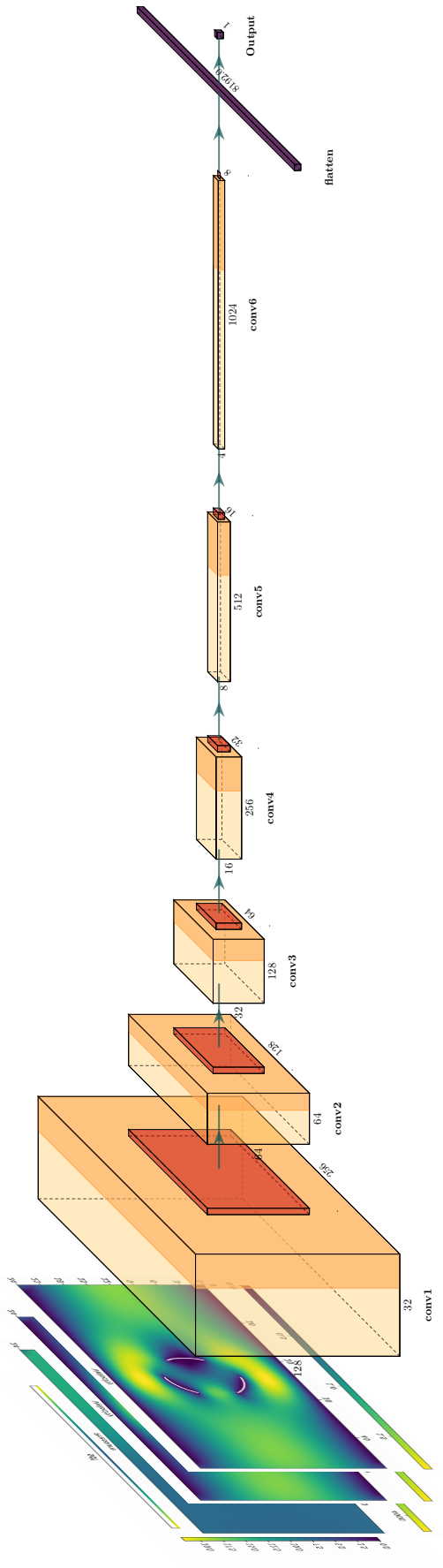


Figure B.3: SDF + CFD model architecture schematic. Yellow blocks show a 2D convolution layer with ReLU activation function, dark yellow a BatchNorm2d layer, red a Pool2D layer and finally purple show Linear layers.

B.4. Dist

Table B.7: Dist input model details.

Parameter	Value
Number of input channels	2
Batch size	32
Optimizer	AdamW
Learning rate γ	3×10^{-4}
Weight decay λ	0
Loss function	MSE
Test metric	MSE
Validation / train fraction	0.2
Model disk size	213.035 MB

Table B.8: Dist input model architecture details.

Layers				Parameters			Info	
Layer	Conv layer	#	Layer type	f	k	s	Output shape	Param #
Convolution	Conv layer 1	1	Conv2d	32	3	1	(32, 128, 256)	608
		2	ReLU				(32, 128, 256)	
		3	MaxPool2d		2	2	(32, 64, 128)	
	Conv layer 2	4	Conv2d	64	3	1	(64, 64, 128)	18 496
		5	ReLU				(64, 64, 128)	
		6	MaxPool2d		2	2	(64, 32, 64)	
	Conv layer 3	7	Conv2d	128	3	1	(128, 32, 64)	73 856
		8	ReLU				(128, 32, 64)	
		9	MaxPool2d		2	2	(128, 16, 32)	
	Conv layer 4	10	Conv2d	256	3	1	(256, 16, 32)	295 168
		11	ReLU				(256, 16, 32)	
		12	MaxPool2d		2	2	(256, 8, 16)	
	Conv layer 5	13	Conv2d	512	3	1	(512, 8, 16)	1 180 160
		14	ReLU				(512, 8, 16)	
		15	MaxPool2d		2	2	(512, 4, 8)	
	Flatten	16	Flatten				(16384)	
Regression head	Hidden layer 1	17	Linear	1024			(1024)	16 778 240
		18	ReLU				(1024)	
	Output layer	19	Linear	1			(1)	1025
Total number of parameters:								18 347 553

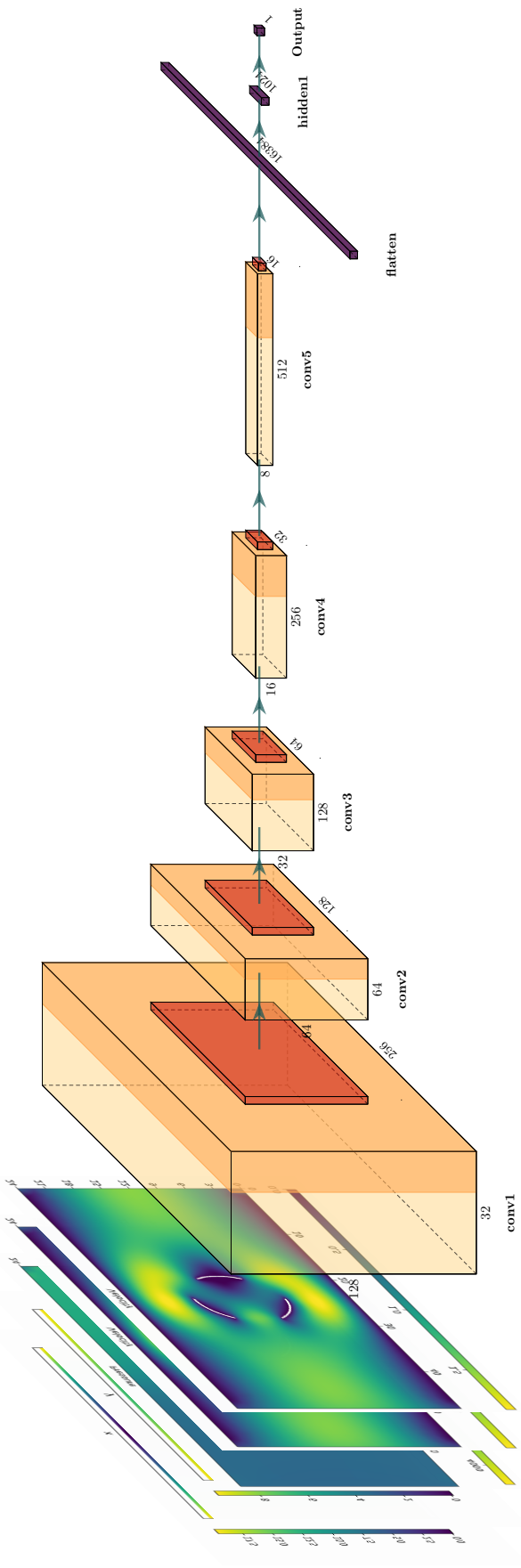


Figure B.4: Dist model architecture schematic. Yellow blocks show a 2D convolution layer with ReLU activation function, dark yellow a BatchNorm2d layer, red a Pool2D layer and finally purple show Linear layers.

B.5. Dist + CFD

Table B.9: Dist + CFD input model details.

Parameter	Value
Number of input channels	5
Batch size	32
Optimizer	AdamW
Learning rate γ	1×10^{-5}
Weight decay λ	1×10^{-2}
Loss function	MSE
Test metric	MSE
Validation / train fraction	0.2
Model disk size	73.817 MB

Table B.10: Dist + CFD input model architecture details.

Layers				Parameters			Info	
Layer	Conv layer	#	Layer type	f	k	s	Output shape	Param #
Convolution	Conv layer 1	1	Conv2d	32	3	1	(32, 128, 256)	1472
		2	ReLU				(32, 128, 256)	
		3	MaxPool2d		2	2	(32, 64, 128)	
	Conv layer 2	4	Conv2d	64	3	1	(64, 64, 128)	18 496
		5	ReLU				(64, 64, 128)	
		6	MaxPool2d		2	2	(64, 32, 64)	
	Conv layer 3	7	Conv2d	128	3	1	(128, 32, 64)	73 856
		8	ReLU				(128, 32, 64)	
		9	MaxPool2d		2	2	(128, 16, 32)	
	Conv layer 4	10	Conv2d	256	3	1	(256, 16, 32)	295 168
		11	ReLU				(256, 16, 32)	
		12	MaxPool2d		2	2	(256, 8, 16)	
	Conv layer 5	13	Conv2d	512	3	1	(512, 8, 16)	1 180 160
		14	ReLU				(512, 8, 16)	
		15	MaxPool2d		2	2	(512, 4, 8)	
	Conv layer 6	16	Conv2d	1024	3	1	(1024, 4, 8)	4 719 616
		17	ReLU				(1024, 4, 8)	
		18	MaxPool2d		2	2	(1024, 2, 4)	
	Flatten	19	Flatten				(8192)	
Regression head	Output layer	20	Linear	1			(1)	8193
Total number of parameters:								6 296 961

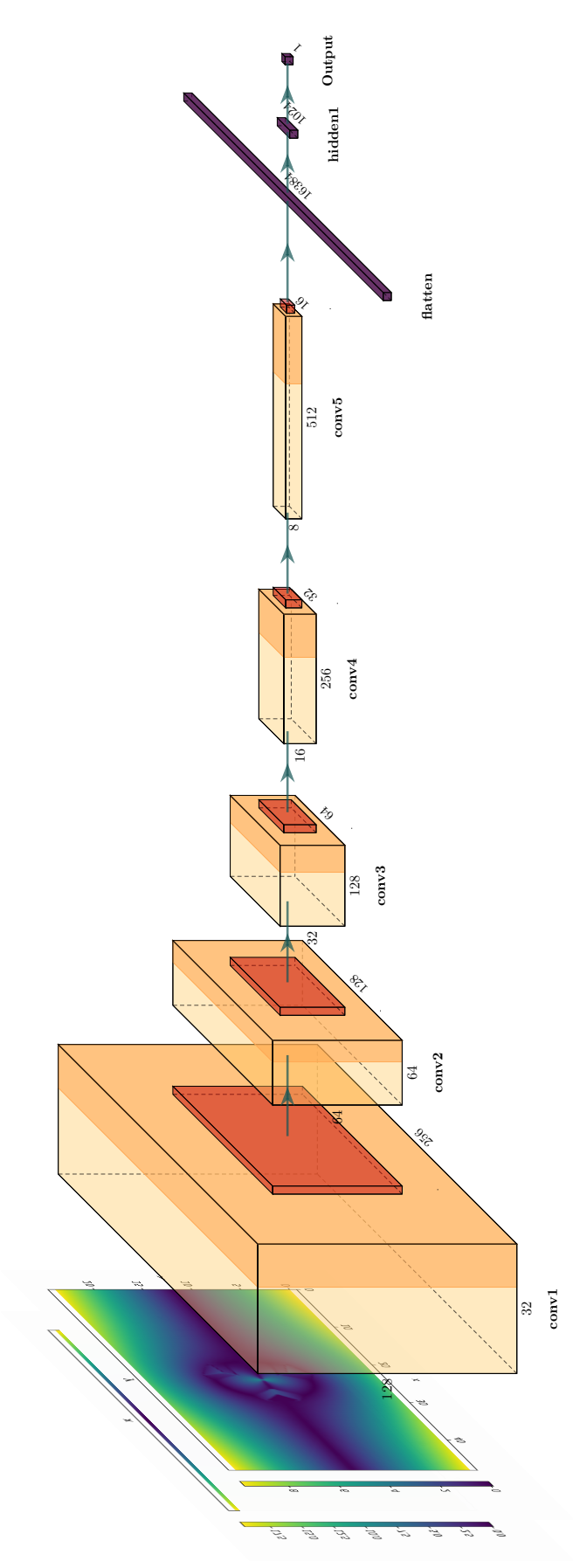


Figure B.5: Dist + CFD model architecture schematic. Yellow blocks show a 2D convolution layer with ReLU activation function, dark yellow a BatchNorm2d layer, red a Pool2D layer and finally purple show Linear layers.

C

Relative error plots

C.1. Binary

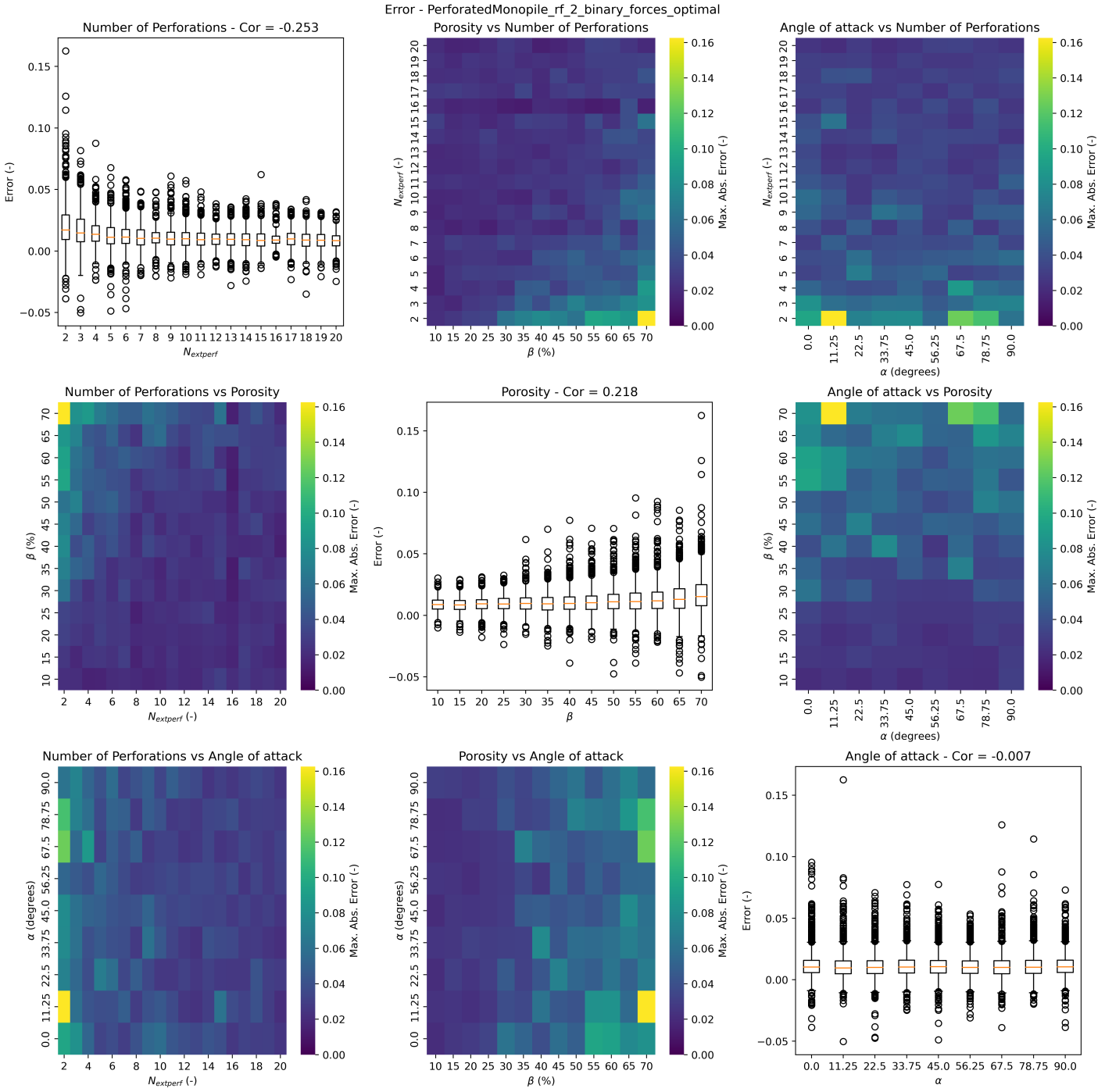


Figure C.1: Binary input model relative error plot. On the diagonal the relative error is plotted for each of the independent geometry variables. On the off-diagonal the maximum absolute relative error is plotted for a combination of two geometry variables.

C.2. SDF

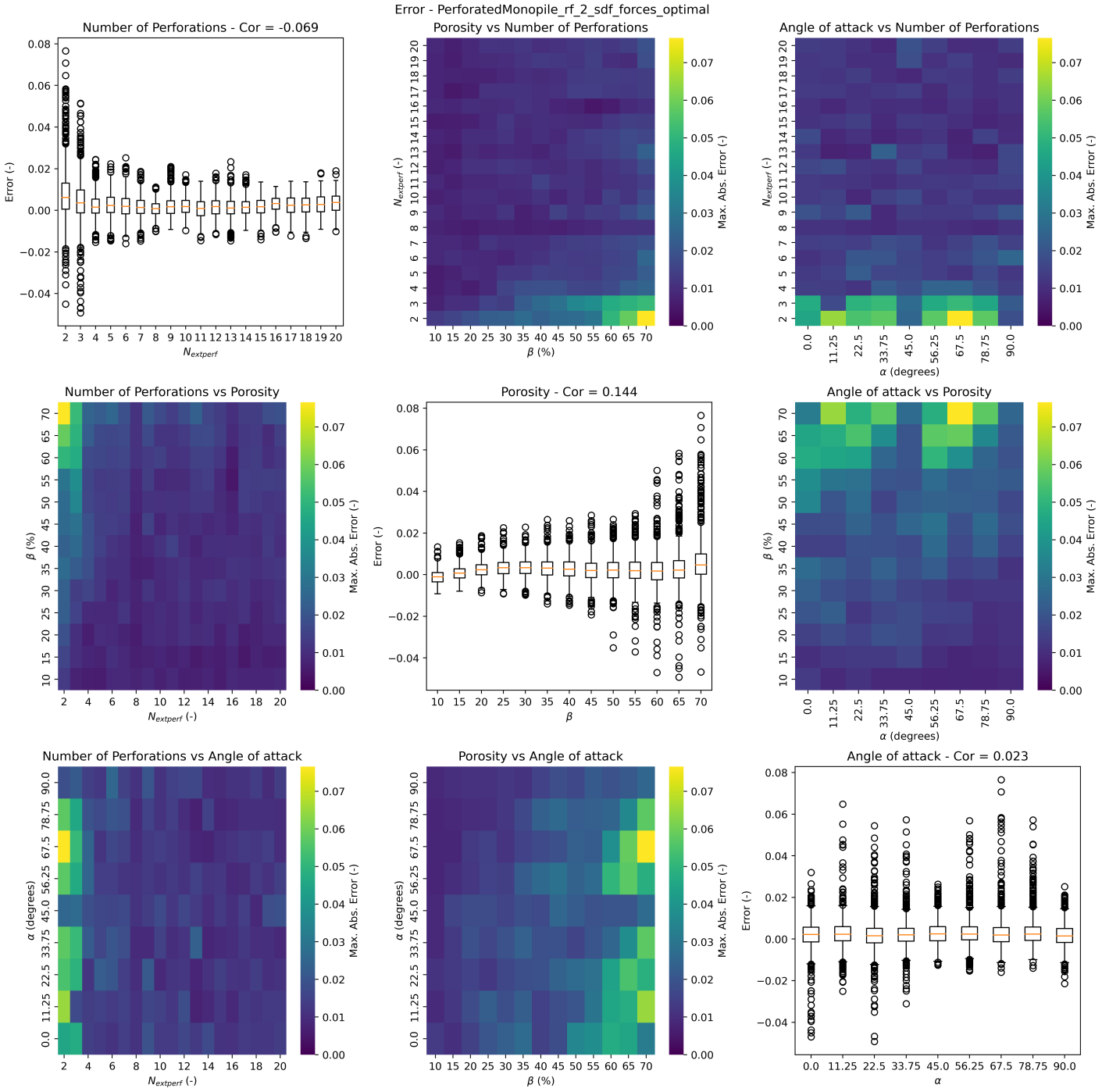


Figure C.2: SDF input model relative error plot. On the diagonal the relative error is plotted for each of the independent geometry variables. On the off-diagonal the maximum absolute relative error is plotted for a combination of two geometry variables.

C.3. SDF + CFD

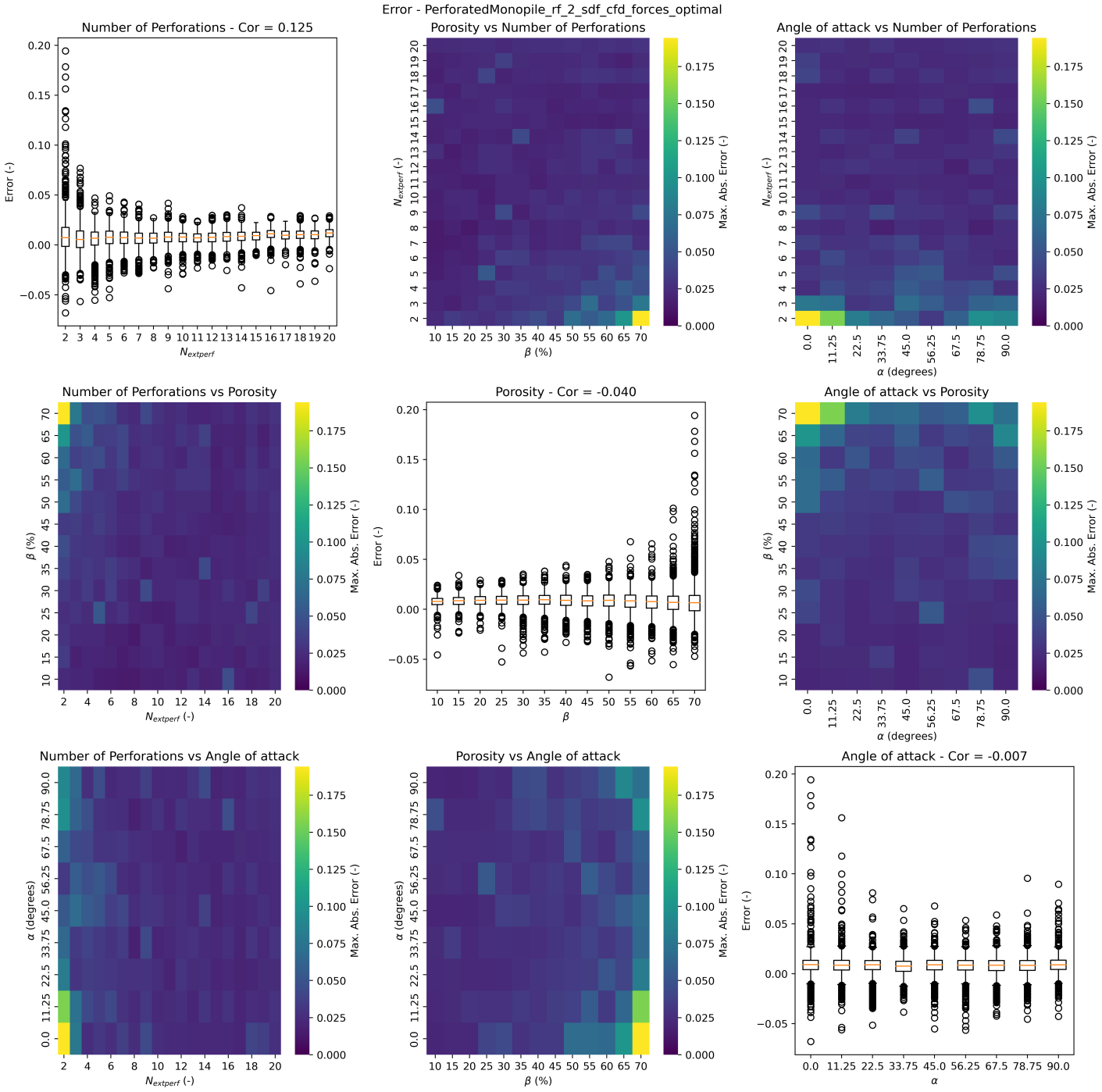


Figure C.3: SDF + CFD input model relative error plot. On the diagonal the relative error is plotted for each of the independent geometry variables. On the off-diagonal the maximum absolute relative error is plotted for a combination of two geometry variables.

C.4. Dist

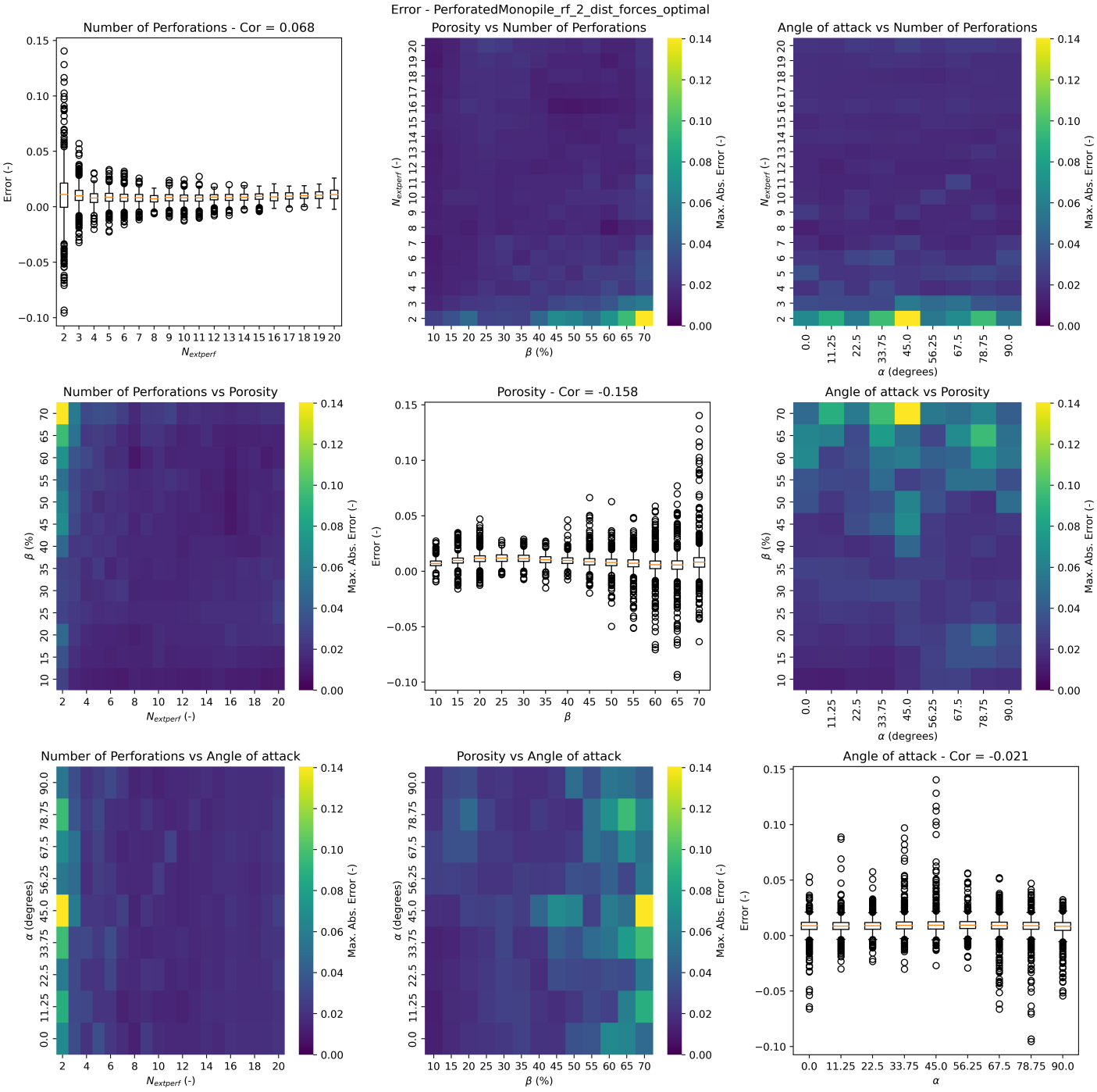


Figure C.4: Dist input model relative error plot. On the diagonal the relative error is plotted for each of the independent geometry variables. On the off-diagonal the maximum absolute relative error is plotted for a combination of two geometry variables.

C.5. Dist + CFD

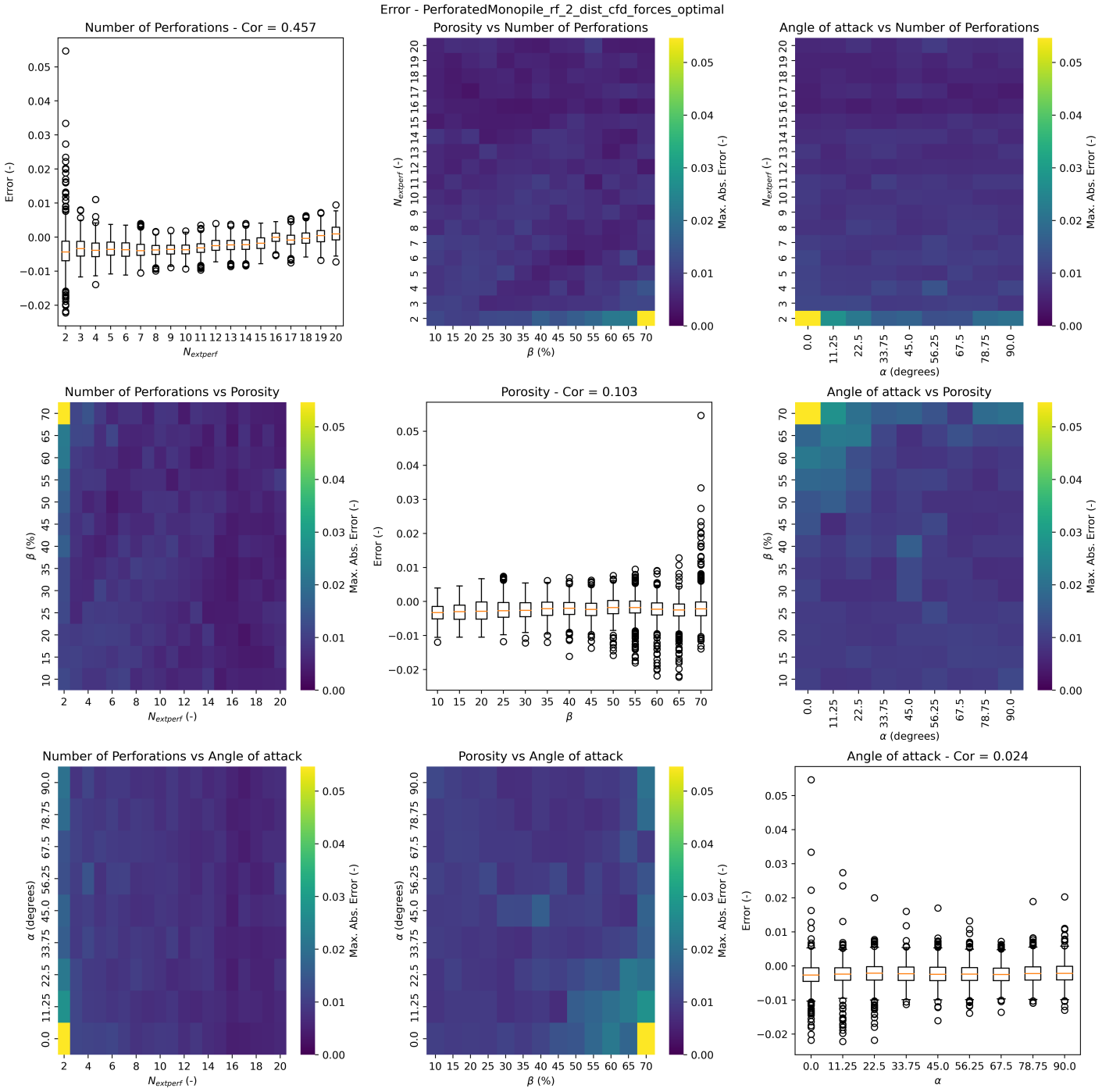


Figure C.5: Dist + CFD input model relative error plot. On the diagonal the relative error is plotted for each of the independent geometry variables. On the off-diagonal the maximum absolute relative error is plotted for a combination of two geometry variables.