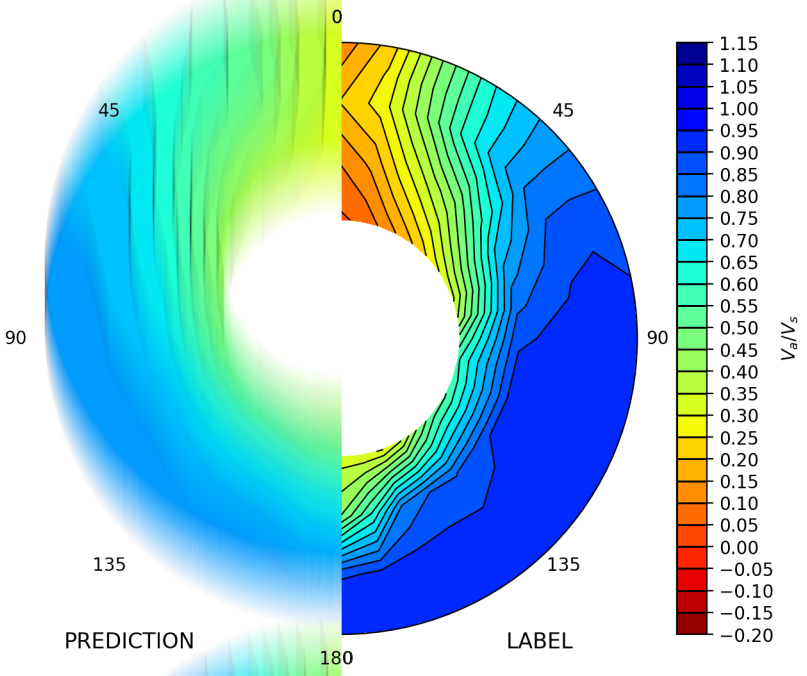
Predicting a Nominal Wake Field

A Data-Driven Approach



MSc Thesis
Stijn Vervoordeldonk

WÄRTSILÄ

TU Delft

Thesis for the degree of MSc in Marine Technology in the specialization of Ship Hydromechanics

Predicting a Nominal Wake Field

A Data-Driven Approach

by

Stijn Vervoordeldonk

performed in collaboration with

Wärtsilä Netherlands B.V.

March 12, 2025
Faculty of Mechanical Engineering, Delft University of Technology

Company Supervisors

Responsible supervisor:	Ir. M. Cleijsen
Daily supervisor:	J. Gjerdevik, MSc.
Supervisor:	I. Hubbard, MSc.

Thesis Exam Committee

Chair:	Prof. dr. ir. T.J.C. van Terwisga
Staff member:	Dr. A. Coraddu
Staff member:	Dr. D. Fiscaletti
Company member:	J. Gjerdevik, MSc.

Document Information

Thesis number:
Student number:
Author contact e-mail:

Cover: Comparison between labelled and predicted wake field graphs
An electronic version of this thesis is available at https://repository.tudelft.nl

TU Delft

WÄRTSILÄ

Preface

Dear reader,

The completion of this report marks one of the final steps before obtaining my Master's degree in Maritime Technology. Over the course of my studies, and especially during my Master's, it were the hydrodynamics courses that most sparked my interest. Not only do hydrodynamic engineering problems often require creativity in addition to field knowledge, but they are also deeply connected to the goal of reducing the amount of energy needed while underway, contributing to a more sustainable and future-proof maritime sector.

For this thesis project on nominal wake field prediction using data-driven methods, I had to delve into the world of data science and machine learning. Having no prior experience in that field, it was fascinating to discover the potential, and maybe more importantly, the shortcomings of applying data-driven methods in ship hydrodynamics. I strongly believe that knowledge on the application of such methods should be in the toolbox of the "engineer of the future".

This project was performed in collaboration with the hydrodynamic R&D department of Wärtsilä Netherlands B.V. This opportunity was given to me by Marcel Cleijsen, for which I would like to thank him. I would like to thank Jannicke Gjerdevik of Wärtsilä Norway for being my daily supervisor. Jannicke, thank you for our weekly meetings, your tips and advice, and most of all, your kindness and support. Additionally, I owe a thank you to Ian Hubbard, who enthusiastically shared his practical knowledge on machine learning engineering and thereby helped steer the project in this direction. I also want to express my gratitude to the whole team at Wärtsilä for all the tips and advice everyone provided me with, and for being such supportive and nice colleagues during this project.

In addition to everyone at Wärtsilä, I would like to thank Tom van Terwisga, my main TU Delft supervisor and committee chairman. Tom, thank you for your supervision and your enthusiastic, inspiring approach to hydrodynamics. Furthermore, I received a lot of support from Andrea Coraddu (until the very last moment before handing in this report!), especially on the topic of machine learning. For this, I am very grateful. I also need to express my gratitude to Evert-Jan Foeth from MARIN, for providing me with useful insights during the update meetings.

Furthermore, I would like to thank my family and friends for their support. A very special thank you goes to Daan, Roan, and Mo for reading my work and providing feedback, as well as offering the necessary distractions from time to time. I also want to thank my girlfriend, Roisin, for her support and for keeping me sane throughout the project.

I hope that reading this report will contribute to your understanding of the wake field, hydrodynamic interaction, and data-driven methods, and that it will persuade you to believe that I have rightfully earned the title of *ingenieur*!

S. Vervoordeldonk
Delft, March 2025

Summary

The objective of this thesis is to explore the possibilities of utilising machine learning to predict the axial velocities in a nominal wake field of a single-propeller vessel, given a database with only basic parameters describing hull geometries, such as length, beam, and block coefficient. The nominal wake field is of primary importance for the propeller design process, as it represents the influence of the hull on the flow field that a propeller encounters. Furthermore, propeller-hull interactions can be derived from it.

A machine learning model, after it has been trained, requires less computational resources for making predictions than a physics-driven model, such as those used in computational fluid dynamics (CFD). As the basic ship parameters that serve as input to the machine learning model are typically known early in the design process, being able to predict the nominal wake field using machine learning allows for a more iterative and integral aft ship and propeller blade design approach compared to the linear approach currently in use.

For this project, five different machine learning models were developed and compared to each other: three different feed-forward neural networks (FNN), an ensemble model and a long short-term memory (LSTM) model. The three feed-forward neural networks differ in how their labels are expressed: one network directly uses the axial velocities in the wake field, non-dimensionalised with the ship speed. The second network employs the Fourier-like discrete cosine transform (DCT) to express wake field velocities at individual radii using the first n DCT coefficients. The third and final network also uses the discrete cosine transform but utilises standardised coefficients.

The steps taken to develop these five models, consisting of data collection and preparation, feature engineering, model training and model evaluation, are described in this thesis. All five models were evaluated in different ways. Visual inspection of predictions made by all five models revealed that none of the models has successfully captured the underlying physical phenomena that drive the wake field: all predictions show highly generalised wake fields. This finding was supported by a feature importance study, which showed that the features that supposedly contribute the most to understanding the hydrodynamic phenomena that drive the wake field do not have the highest relative importance.

A relative comparison of model performance was conducted using repeated random subsampling validation, where all models were re-trained on different data splits. From this validation and the subsequent comparison, it was found that the LSTM model performs best in terms of average error, while the first FNN has the lowest standard deviation in performance across the different splits. The relatively high standard deviation in performance across the different splits further shows the limited understanding of the underlying physical phenomena of the model.

A case study was performed, in which the cavitation performance of eight propellers was assessed, using both predicted and “true” nominal wake fields and comparing between the results. Despite the limited performance of all developed models, the difference in calculated cavitation performance between the predicted and true nominal wake fields was relatively small. It was confirmed by Wärtsilä propeller designers that the predicted wake fields serve as a better substitute for a missing nominal wake field than performing a manual regression.

The limited performance of the developed models is primarily attributed to two factors: the limited dataset size and a lack of feature informativeness, meaning that the used features contain insufficient information about the problem to let a model effectively capture the underlying physical phenomena. To a lesser extent, computational limitations also play a role.

These two factors serve as the basis for the main recommendation: for further research, it is recommended to not only obtain a larger dataset but also to incorporate more features that are descriptive of the aft ship geometry to allow for better generalisation of a machine learning model.

Contents

List of Figures

List of Tables

Nomenclature

List of Abbreviations

Abbreviation	Definition
AI	Artificial intelligence
ANN	Artificial neural network
BSRA	British Ship Research Association
CII	Carbon intensity index
COB	Centre of buoyancy
COG	Centre of gravity
CPP	Controllable pitch propeller
CRS	Conventional rudder system
DCT	Discrete cosine transform
DNN	Deep neural network
EEDI	Energy efficiency design index
EEXI	Energy efficiency existing ship index
ENN	Edited nearest neighbours
ESD	Energy saving device
FNN	Feed-forward neural network
FPP	Fixed pitch propeller
GRS	Gate rudder system
GUI	Graphical user interface
iDCT	Inverse discrete cosine transform
IMO	International Maritime Organization
ITTC	International Towing Tank Conference
JSON	JavaScript object notation
LSTM	Long short-term memory
MAPE	Mean absolute percentage error
ML	Machine learning
MSE	Mean squared error
NACA	National Advisory Committee for Aeronautics
NN	Neural network
PPMCC	Pearson product-moment correlation coefficient
RBF	Radial basis function
ReLU	Rectified linear unit
REP	Relative error percentage
RNN	Recurrent neural network
SMOTE	Synthetic minority oversampling technique
SVM	Support vector machine
SVR	Support vector regression
VLM	Vortex lattice method

List of Symbols

Symbol	Definition	Unit
A	Area	m^2
A_0	Propeller plane area	m^2
A_E/A_0	Expanded blade area ratio	-
B	Beam	m
c	Chord length	m
$\bar{c}$	Average chord length	m
c_R	Root chord length	m
c_T	Tip chord length	m
C	Regularisation parameter	-
C_B	Block coefficient	-
C_t	Cell state	-
d_h	Hub diameter	m
d_h/D	Hub ratio	-
D_{prop}	Propeller diameter	m
F	Force	N
Fr	Froude number	-
g	Gravitational constant	Nm^2/kg^2
H	Hinge loss function	-
J	Advance coefficient	-
k_p	Number of propellers	-
k_λ	Factor between geometric and effective aspect ratio	-
K_Q	Torque coefficient	-
K_T	Thrust coefficient	-
L	Loss function	-
L_{pp}	Length between perpendiculars	m
ℓ	Loss function	-
n_p	Propeller speed	1/s
P	Pitch	m
P/D	Pitch ratio	-
Q	Torque	Nm
R	Radius	m
R^2	Coefficient of determination	
Re	Reynolds number	-
s	Shell appendage factor	-
s	Span	m
t	Blade thickness	m
t/c	Propeller thickness ratio	-
T	Draught	m
T	Thrust	N
v	Velocity	m/s
v_A	Advance velocity	m/s
w	Wake fraction	-
y	Label / target	-
$\hat{y}$	Prediction	-
α	Angle of attack	rad
α	Learning rate	-
β	Advance angle	-
γ	Kernel coefficient	-
δ_R	Rudder deflection	rad
Δ	Displacement mass	kg
η	Efficiency	-
η_0	Open water efficiency	-

Symbol	Definition	Unit
θ	Wake field angle	rad
λ_E	Effective aspect ratio	-
λ_G	Geometric aspect ratio	-
μ	Mean	-
ν	Kinematic viscosity	m^2/s
ρ	Density	kg/m^3
σ	Sigmoid activation function	-
σ	Standard deviation	-
∇	Displacement volume	m^3
Ω	Rudder sweep angle	rad

1

Introduction

According to the latest IPCC report, climate change is already causing extreme weather events worldwide, negatively impacting people and nature. There exists a broad consensus attributing climate change to human-caused greenhouse gas emissions [1].

Up until today, the maritime sector still accounts for a large part of these emissions. In 2018, the maritime sector alone emitted 1056 million tonnes of CO_2, accounting for 2.89% of worldwide emissions [2]. This excludes other pollutants like sulphur oxides (SO_x), nitrogen oxides (NO_x) and particulate matter (PM). Overall, maritime emissions have not decreased between 2013 and 2023 but have instead increased by 20%, with 98.8% of the world fleet still operating on fossil fuels [3].

In order to reduce maritime emissions, the International Maritime Organization (IMO) has introduced regulations and guidelines to decrease the ecological footprint of the world fleet, such as the EEDI (Energy Efficiency Design Index) in 2013 and the EEXI (Energy Efficiency Existing Ship Index) and CII (Carbon Intensity Index) in 2023 [4, 5]. The EEDI dictates a maximum CO_2-emission per ton-mile for new-built ships, the EEXI is a similar measure for existing ships. The CII is more rigorous and forces ships to improve their efficiency to keep complying as the rules become more strict over time. The goal is to reduce CO_2 emissions by at least 40% in 2030 as compared to 2008. Furthermore, total GHG emissions need to be reduced by 20-30% by 2030 and 70-80% by 2040 compared to 2008, as checkpoints for net-zero shipping around 2050 [6]. Lastly, the aim is to have 5-10% of the energy used by shipping coming from (almost) net-zero energy sources by 2030.

To meet these goals, increasing the energy efficiency of ships is needed. For fossil fuel powered ships, increased efficiency means less fuel usage at equal sailing speeds, hence mitigating part of the emissions. For alternative fuels, despite their potential for net-zero emissions, energy density (either volumetric or gravimetric) is often a limiting factor for the large-scale application of a technology [7]. Therefore, increasing efficiency and reducing the amount of fuel storage needed is still essential. One way to increase ship efficiency is to improve the hydrodynamic performance of the vessel. The way in which a vessel interacts with the water surrounding it is complex, as there are many components involved that influence the flow around the vessel's hull. Furthermore, these components influence each other's behaviour as well, through hydrodynamic interaction. Examples of such components are the hull itself, the propeller(s), the rudder(s) and potential appendages such as energy saving devices (ESDs).

A relatively new type of ESD is the gate rudder (GR), invented by Sasaki and Kuribayashi in 2012 and first mentioned by Sasaki in 2013 [8, 9]. Consisting of two foils that wrap around the top and sides of the propeller and are able to pivot individually, it is a cross-over between a duct-like energy saving device and a new type of rudder. In real-life conditions, savings up to 27% have been reported [10]. From these reported savings only, the gate rudder seems a very promising innovation. However, research into its hydrodynamic behaviour is still ongoing. Among investigation by the research and development teams within Wärtsilä the gate rudder is currently being investigated by the European Union-funded project GATERS [11].

Because of the interest in improving the hydrodynamic performance of a vessel, it is essential to enhance not only the hydrodynamic interaction between conventional components (hull, propeller, and rudder) but also to actively influence the hydrodynamic interaction with appendages (ESDs) or through component substitution (GR). A thorough understanding of the flow field in which these components operate is required. This understanding can be derived from model tests (Experimental Fluid Dynamics, EFD). However, following advances in computer science and an increase in available computational power over the last decades, Computational Fluid Dynamics (CFD) has been rapidly developing since its initial development in the 1960s.

A benefit of CFD over EFD is the much higher resolution following from the calculations than could be obtained experimentally using sensors, providing insight in local flow phenomena [12]. Despite this advantage, the further optimisation of propellers and rudders and the development of energy saving devices and the gate rudder is an iterative design process, necessitating a high level of understanding of the flow field in an early design stage. This leads to a paradoxical situation: there is a need for knowledge on detailed flow characteristics while not all design parameters are known. This reveals a downside of CFD: the still significant computation time makes it less suitable for performing numerous small experiments, such as those needed to determine initial parameters like dimensions of aft ship components.

While CFD is well-suited for a linear design process, machine learning (ML) techniques might offer a solution to the information paradox in an iterative design process. This thesis provides a pragmatic approach to data-driven flow field (nominal wake field) prediction by training and evaluating various machine learning models that predict nominal wake fields for single-propeller vessels. The pragmatic element involves the use of a readily-available dataset of ships with only limited parameters describing the hull geometries, while this geometry has a significant influence on the nominal wake field. This report presents the findings of the conducted project. One main and three sub-research questions, each covering a different part of the scope of this project, will be answered in this thesis report. The main research question is as follows:

To what extent is it possible to train and validate a machine learning model to accurately predict the nominal wake field of a vessel, given a database of single-propeller vessels with only basic parameters describing hull geometries?

The three sub-questions that have been formulated are as follows:

1. What is the importance of understanding the nominal wake field in propeller design, and at which points in the aft ship design process can predicted nominal wake fields be applied?
2. Which machine learning algorithms and structures are most suitable for predicting the nominal wake field, and what are the steps that need to be taken to develop such models?
3. What criteria determine the quality of a predicted nominal wake field, and what are the requirements for accurate nominal wake field prediction?

This report is structured in the following way. Chapters 2 and 3 will delve into hydrodynamics and machine learning theory, respectively. These chapters will provide theoretical backgrounds relevant for the rest of the thesis and are partially adopted from the literature review that was written as part of this thesis project [13]. The first and third sub-questions are answered in chapter 2, while the second sub-question is partially answered in chapter 3 and partially throughout the rest of the thesis report.

Chapters 4 to 7 describe the process of building, training, and evaluating the machine learning model. Firstly, chapter 4 will elaborate on the collection and preparation of the data used to train the model. Afterwards, chapter 5 will explain the process of selecting, combining and manipulating variables from the dataset to use as inputs (“features”) to the machine learning models. Chapter 6 will present the tuning and training setup of the various machine learning models. As a final step in the machine learning engineering procedure, in chapter 7 all models are evaluated and tested.

After presenting the relevant theory and describing the building of the machine learning model, chapter 8 contains a discussion of project limitations and this project in the light of the Gate Rudder project, as well as some recommendations for further research. Finally, conclusions will be drawn in chapter 9.

2

Hydrodynamics Theory

In this chapter, the hydrodynamic theory relevant to this thesis will be discussed. Parts of this discussion are adopted from the literature review that was written as part of this thesis project [13]. The first and third sub-questions mentioned in the introduction are as follows:

What is the importance of understanding the nominal wake field in propeller design, and at which points in the aft ship design process can predicted nominal wake fields be applied?

What criteria determine the quality of a predicted nominal wake field, and what are the requirements for accurate nominal wake field prediction?

This chapter aims to answer these sub-questions. First, relevant knowledge on propellers and rudders will be provided in sections 2.1 and 2.2, respectively. Thereafter, different kinds of hydrodynamic interaction are touched upon in section 2.3. The wake field will be introduced in section 2.4, followed by cavitation theory in section 2.5. Sections 2.6 and 2.7 will elaborate on energy saving devices (ESDs) and the gate rudder (GR), and finally, in section 2.8 the first and third sub-questions will be answered by drawing conclusions based on the content of this chapter.

2.1. Propellers

Propellers are the most common propulsion system found on board modern ships. The main purpose of a propeller is to convert rotational kinetic energy delivered by the propeller shaft into usable thrust by accelerating the surrounding water. The efficiency of a propeller measures how effectively it performs this task. This section will present a brief introduction to propellers by first introducing commonly-used terminology and propeller types, and afterwards showing some propeller performance indicators.

2.1.1. Terminology and Types

In this report, terminology and definitions as defined by the International Towing Tank Conference (ITTC) [14] will be followed. First, a propeller reference line is defined. This line originates at the center of the hub and extends straight upward through the reference point (center) of the root section of one of the blades. Propeller rotation will be given with respect to this generator line, so a zero-degree rotation will always be an orientation where one blade faces directly upward.

Propeller pitch P is defined as the distance that the propeller would travel in one revolution without slip. Pitch is often given with respect to propeller diameter as the pitch ratio P/D. As pitch usually varies along the propeller radius R, it is common practice to provide a P/D ratio at 70% radius, as that area is most representative of propeller behaviour and generally bears most thrust. There are more important ratios, like the expanded area ratio A_E/A_0, denoting the percentage of surface area of the propeller plane that is covered by the propeller. It can be larger than one if the propeller blades overlap. Futhermore, the thickness ratio of the propeller at a certain radius is given by t/c where t and c denote blade thickness and local chord length, respectively. Lastly, the hub ratio d_h/D is a measure for the hub size with respect to the propeller diameter.

Skew and rake angles are defined by the tangential and axial angles between the propeller reference line and the blade reference line, which connects all average chord points on the propeller with each other. Skew and rake are usually applied to modify propeller-induced pressure pulses, therefore changing cavitation behaviour.

The most commonly used propellers are fixed pitch propellers (FPP) and controllable pitch propellers (CPP). The most important difference is that using a CPP allows the operator to rotate the propeller blades and set a desired pitch angle, leading to better manoeuvrability and adaptability to different operational profiles. However, this comes at the cost of larger hub losses and higher maintenance cost due to the additional systems required. FPPs are less expensive but have an optimal working point, so in case of multiple operational profiles a trade-off has to be made.

2.1.2. Performance Indicators

To be able to compare the performance of different propellers, advance velocity v_A, thrust T and torque Q are often expressed non-dimensionally [15], see equation 2.1:

$$J = \frac{v_A}{n_p D} \qquad K_T = \frac{T}{\rho n_p^2 D^4} \qquad K_Q = \frac{Q}{\rho n_p^2 D^5} \tag{2.1}$$

Where J is the advance coefficient, K_T the thrust coefficient and K_Q the torque coefficient. Furthermore, v_A is the advance velocity, or the relative velocity of the blade with respect to the water. Lastly, n_p is the propeller speed in rotations per second.

From these three dimensionless parameters, the open water efficiency η_0 can be determined as shown in equation 2.2.

$$\eta_0 = \frac{1}{2\pi}\frac{T v_A}{Q n_p} = \frac{K_T J}{2\pi K_Q} \tag{2.2}$$

The nominator in equation 2.2 is a measure for propulsive energy, where the denominator is a measure for the energy used to turn the propeller. The open water efficiency includes axial, rotational and viscous losses but does not include losses due to interaction effects.

Plotting K_T, K_Q and η_0 against J results in an open water propeller plot, as shown in figure 2.1a. The position at J =0, corresponding with maximum values for K_T and K_Q, is called bollard pull.

Figure 2.1a can be used for so-called first quadrant operation. There are four quadrants, defined by an advance angle β, which are explained in table 2.1:

$$\beta = \tan^{-1}\left(\frac{v_a}{0.7\pi n_p D}\right) \tag{2.3}$$

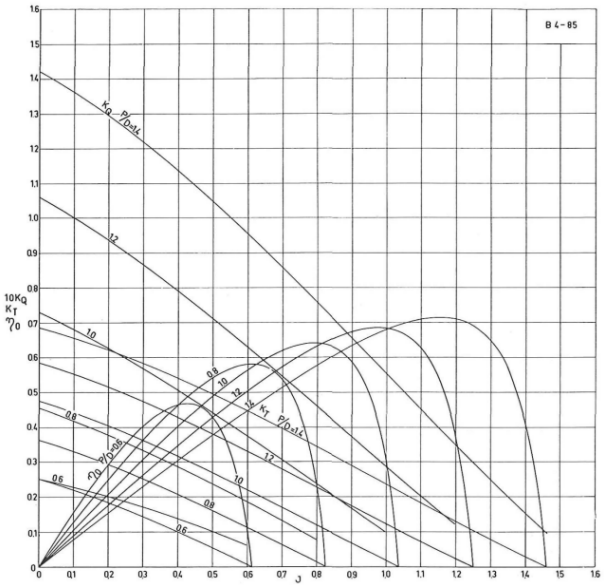
Table 2.1: The four quadrants of propeller operation, including ahead and astern bollard pull.

Quadrant	Advance angle	v_A	n_p	Interpretation
	0	0	+	Ahead bollard pull
1	0 < $\beta \leq \pi$/2	+	+	Ahead operation
2	π/2 < β < π	+	-	Breaking while ahead
	β = π	0	-	Astern bollard pull
3	π < $\beta \leq$ 3π/2	-	-	Astern operation
4	3π/2 < β < 2π	-	+	Breaking while astern

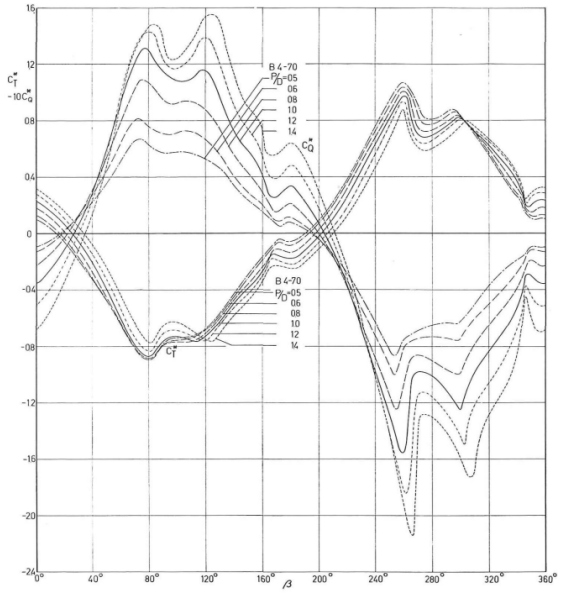
The use of an advance angle β rather than advance velocity J prevents singularities where $J \to \infty$. The thrust and torque coefficients are also rewritten:

$$C_T^* = \frac{T}{\frac{1}{2}\rho v_R^2 A_0} \qquad C_Q^* = \frac{Q}{\frac{1}{2}\rho v_R^2 A_0 D} \tag{2.4}$$

Where v_R equals the advance velocity at a radius $0.7R$ and A_0 the area of the propeller plane. C_T^* and C_Q^* can now be plotted against β to find the four-quadrant open water diagram of a propeller, as shown in figure 2.1b.



(a) First-quadrant open water diagram for different P/D ratios of the Wageningen B4-85 propeller.



(b) Four-quadrant open water diagram for different P/D ratios of the Wageningen B4-70 propeller.

Figure 2.1: First- and four-quadrant open water diagrams. From [16].

2.2. Rudders

A rudder is a control surface used primarily for course-keeping and initiating turns, and sometimes for roll damping [17]. Rudders have a long history of use, dating back to ancient Egypt, where oars were specifically positioned to function as rudders for steering [18]. Although common, rudders are not the only way for a vessel to maintain course or initiate turns. For instance, water jets can deflect flow by moving a bucket to steer the vessel [19] and the Voith-Schneider propeller allows for thrust direction control by influencing the vane-pitch angles [20]. Lastly, an azimuthing or podded propulsor also eliminates the need for a separate rudder.

In this section, the working principles behind and terminology of a conventional rudder system (CRS) will be discussed, along with an overview of different types of conventional rudders. The gate rudder system (GRS), which exhibits characteristics of CRS's and energy saving devices (ESDs), will be discussed in section 2.7.

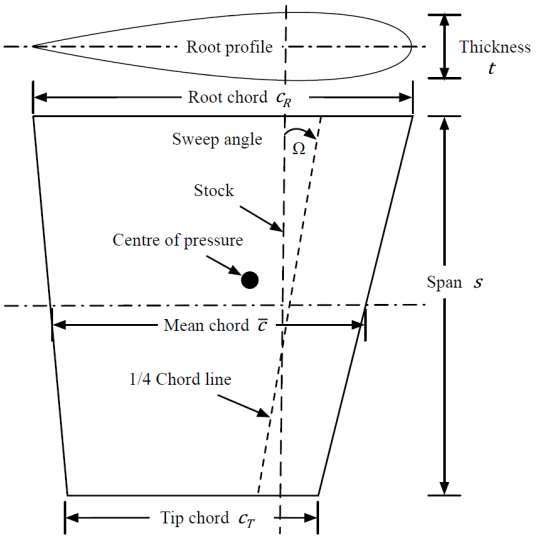


Figure 2.2: Rudder terminology. Adapted from [17].

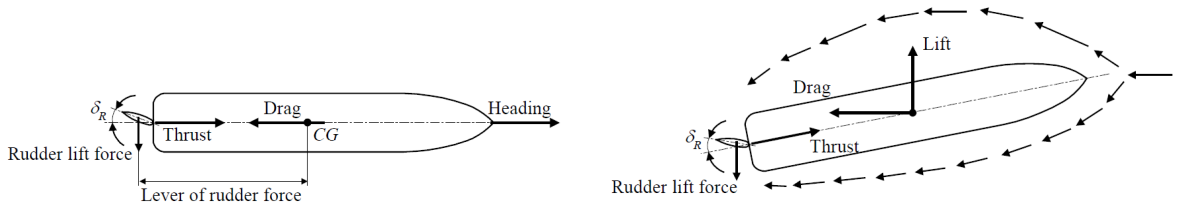
2.2.1. Terminology

Rudders function as hydrofoils and are almost always positioned at the aft of the ship on a vertical or near-vertical axis known as the stock [21]. Each water plane slice of the rudder exhibits a foil geometry. Various foil geometries are possible, for instance a certain NACA profile or having a fish tail. The

simplest geometry is nothing more than a flat plate [17, 22]. Figure 2.2 illustrates some key terminology related to the rudder from the centerline plane perspective [17]. In figure 2.2, c_R, c_T and $\bar{c}$ denote the root, tip and average chord length, respectively. The sweep angle Ω denotes the angle between the 1/4 chord line and the vertical mean stock line. Span s denotes the vertical distance between the root and the tip, and from span and root chord the geometric aspect ratio is defined as $\lambda_G = s/c_R$. The effective aspect ratio λ_E takes into account the finite span of the rudder as well as asymmetry in the rudder and is expressed as a factor k_λ of λ_G: $\lambda_E = k_\lambda \lambda_G$. Not shown in figure 2.2, rudder deflection in radians with respect to the hull it is attached to is denoted with δ_R. Lastly, also not shown, an important parameter is the rudder gap, denoting the distance between the hull and the rudder root. As this region is dominated by separation due to sharp corners, the gap width has influence on rudder efficiency.

2.2.2. Working Principle

A rudder serves as the initiator of a vessel turn [23]. The combination of rudder lift force and the lever from this force to the center of gravity leads to a yaw moment on the hull. When the turn has been initiated, most of the force required to turn is accounted for by the hull lift force, because the hull under an angle of attack starts to act like an extremely low aspect ratio foil, where span s equals draught T and chord c the distance between the forward perpendicular and the stern frame. This was confirmed by a mathematical study performed by Fuss [24]. An overview can be seen in figure 2.3 [23]. When manoeuvring at nearly zero speed, the rudder accounts for a larger part of the force required for turning, as there is less flow around the ship hull. Rudders of big ships account for 2 to 3 percent of ship resistance [25]. During the design phase, rudder design is a trade-off between manoeuvrability and efficiency which typically depends on the owners' requirements.



(a) Rudder lift force initiating turn.

(b) Ship hull acting as a lifting surface.

Figure 2.3: Hydrodynamic forces on a hull when turning. From [23].

2.2.3. Rudder Types

There are several types of rudders, differing in profile type and attachment methods to the hull. Regarding profile type, various options exist and a certain profile is selected or designed to optimise for low drag or high manoeuvrability at a certain ship speed. One notable type is the flap rudder, which features a flap at the trailing edge to be able to increase the camber of the rudder and enhance manoeuvrability. For attachment methods, Tupper [26] categorises rudders based on two criteria. The first criterion divides between balanced, semi-balanced, and unbalanced rudders, based on the horizontal distance from the centre of pressure of the rudder to the rudder stock. The second criterion corresponds with the attachment position of the rudder: a spade rudder is only connected at the rudder root, while other rudders have an additional pintle located along the span or at the tip.

2.3. Hydrodynamic Interaction

Flow around a hull with appendages is a complex phenomenon. The presence of different components influences the flow behaviour around other components. This is called hydrodynamic interaction. This interaction can be intended or unintended, beneficial or disadvantageous. The inflow field to a propeller, for instance, is often heavily influenced by the presence of the hull, and this unintended interaction needs to be accounted for in the design phase. On the other hand, an energy saving device (ESD) is designed to have a beneficial hydrodynamic effect. Insight in hydrodynamic interaction is crucial for reducing unwanted effects and maximising beneficial ones. This section will start by providing a short introduction to fluid dynamics applied to hulls and foils, and thereafter discuss different types of aft ship interaction, being propeller-hull, propeller-rudder, and rudder-hull.

2.3.1. Boundary Layers and Flow Around a Hydrofoil

A boundary layer is the region directly adjacent to a surface, where the fluid velocity equals zero at the surface and the free-stream velocity at the boundary layer border. Under potential flow assumptions, fluids are considered inviscid and irrotational, resulting in a constant velocity field profile near surfaces. However, in real viscous flows, these assumptions do not hold, and the no-slip condition must be applied, leading to the formation of a boundary layer. We can define a surface aligned along the x-axis with the y-axis perpendicular to it, and a fluid flowing over the surface with its velocity component in x-direction denoted with u. The no-slip condition imposes that at y = 0, u = 0. The distance in y-direction over which u develops from 0 to the free stream velocity defines the boundary layer. As the flow progresses in positive x-direction the derivative $\partial u/\partial y|_{y=0}$ increases. The boundary layer thickness δ typically grows as a square root of the distance from the leading edge. The boundary layer can either be laminar or turbulent. A turbulent boundary layer has more momentum and a higher thickness than a laminar boundary layer [25]. The development of a boundary layer can be seen in figure 2.4 [27].

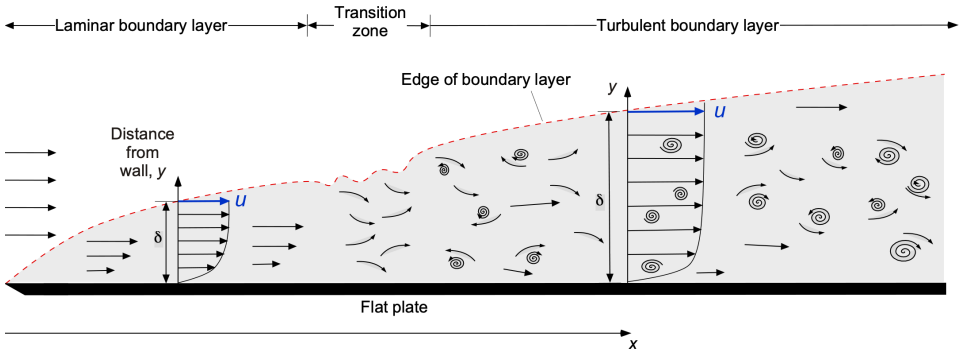


Figure 2.4: Boundary layer development. Adapted from [27].

On the suction side of a foil, the maximum velocity over the foil typically occurs near the point of maximum thickness. From the leading edge to this point the pressure decreases, so dp/dx < 0, creating a favourable pressure gradient and accelerating the flow. At the point of maximum velocity, the pressure gradient is zero. Between this point and the trailing edge the pressure increases (dp/dx > 0), and the flow decelerates. In this region of adverse pressure gradient, du/dy increases rapidly, and even flow reversal can happen close to the surface. The point at which the flow first reverses is called the separation point as the flow will separate from the foil. An example of boundary layer separation along a hydrofoil can be seen in figure 2.5 [25].

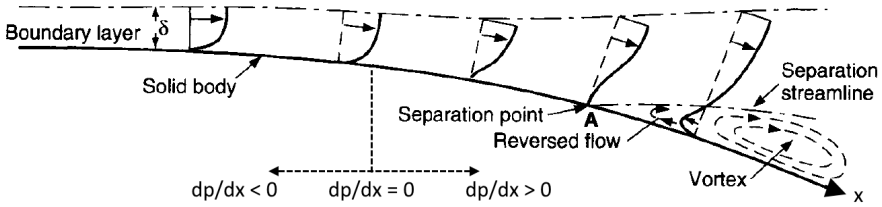


Figure 2.5: Boundary layer separation along a hydrofoil. Adapted from [25].

2.3.2. Flow Around a Hull

When sailing through the water, both viscous and inertial effects play a role. The former is responsible for the development of a boundary layer along the ship length, the development of which will be discussed. The latter is responsible for the wave system arising, and will also briefly be touched upon.

Hull Boundary Layer

For a hull, most part of the boundary layer is turbulent. Different regions in the flow around a hull are shown in figure 2.6 [28]. To determine whether a flow is laminar or turbulent, the Reynolds number shown in equation 2.5 can be used.

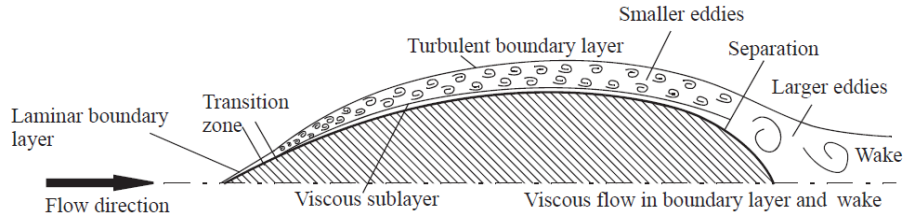


Figure 2.6: Different regions in the flow around the hull. From [28].

$$Re = \frac{\rho v L}{\mu} = \frac{v L}{\nu} \tag{2.5}$$

In equation 2.5, v is the characteristic velocity, in this case ship speed. L is a characteristic length, being ship length, ρ is the water density and μ and ν are the dynamic and kinematic viscosity, respectively. The Reynolds number is a measure for the inertial forces with respect to viscous forces. In other words: high Reynolds numbers (inertia dominates) indicate turbulence where low Reynolds numbers (viscosity dominates) indicate laminar flow. For smooth plates, the transition between laminar and turbulent flow lies around $Re \sim \mathcal{O}(10^6)$ [29]. In reality, due to surface roughness, this number can become lower. Ships typically sail at Reynolds numbers between 10^6 and 10^{10} [30]. Intuitively, at these Reynolds numbers viscosity should not play a large role. The reason why it does, is that within the turbulent boundary layer, the diameter $\mathcal{L}$ of the turbulent eddies serves as the length scale for the Reynolds number. This very low length scale relative to the ship length leads to values of $Re_{\mathcal{L}}$ possibly as low as 1 - 10^3.

Flow separation occurs at regions of adverse pressure gradient, often associated with the aft ship. The locations of flow separation are therefore related with curvatures and sharp edges in the aft ship. These design features can be tweaked. Planing hulls, for instance, trigger flow to separate at the transom and the sides by making use of a hard chine and straight stern buttock lines in order to reduce drag and enabling the hull to plane [31]. Likewise, sharp edges on for instance the propeller hub, rudder pintle or the root of foils can trigger separation.

In contrast to the 2D situation in section 2.3.1, three-dimensional effects come into play when looking at a hull boundary layer. This leads to two possible types of separation: bubble-type and vortex-type [28]. In bubble-type separation, the fluid separates in small volumes over a so-called separation bubble. In vortex-type separation, streamlines ‘roll up’ in order to form long vortices. Vortex-type separation has a large influence on the wake field, as will be seen in section 2.4.

Wave System

Inertial effects combined with the presence of a free surface cause a wave system to arise around a moving hull. As this thesis focuses on underwater behaviour in the aft ship, the phenomena driving this system will not be discussed. However, it is important to note that for ships with limited draught, the wave system can influence the direction and magnitude of the inflow to the propeller or other components. Additionally, the wave pressure field and potential resulting ship motions influence the flow around the hull, leading to highly nonlinear behaviour.

2.3.3. Propeller-Hull Interaction

One of the most crucial types of interaction in the aft ship is the two-way propeller-hull interaction. As the propeller operates in the hull wake field, its behaviour changes compared to the open-water situation. Conversely, the velocities induced by the propeller influence flow around the hull. Lastly, the propeller and hull cause pressure fluctuations on one another, potentially leading to unwanted effects.

In marine engineering the propeller-hull interaction is captured in two factors: the thrust deduction factor t and the wake fraction w [15]. They are defined in equation 2.6.

$$t = \frac{k_p T - R}{k_p T} \qquad w = \frac{v_S - v_A}{v_S} \tag{2.6}$$

In equation 2.6, k_p equals the number of propellers, T the thrust per propeller, R the towed ship resistance, and v_S and v_A the ship speed and advance velocity, respectively. The wake fraction has to do with the wake field discussed in section 2.4 and accounts for the reduction in (average) axial inflow velocity due to the presence of the hull compared to the open-water situation. The thrust deduction factor accounts for the effect of the propeller on the hull (and rudder): because the propeller accelerates both upstream and downstream areas, hull and rudder resistance increases, leading to a higher thrust demand to overcome the higher resistance. The thrust deduction and wake factors together define the hull efficiency η_H, which is defined as the ratio between (1-t) and (1-w).

The non-uniform inflow to the propeller leads to a varying angle of attack on a revolving blade segment, causing cavitation-inducing pressure fluctuations. Furthermore, if the vertical distance from the propeller tip to the hull is not too large, pressure fluctuations on that part of the hull with the blade passing frequency can arise. These pressure fluctuations and possible cavitation that results from them can lead to various unwanted effects [32], such as unwanted vibrations, on-board noise, radiated noise harmful to marine fauna and reduced stealth in the case of naval ships.

2.3.4. Propeller-Rudder Interaction

The rudder is often positioned directly in the propeller slipstream. As a result of this, the rudder encounters propeller-induced pressure fluctuations, just like the hull. Furthermore, cavities shed from the propeller, such as tip vortex cavitation or hub vortex cavitation, can implode on the rudder blade, possibly leading to erosion. Furthermore the rudder behaviour changes with changing propeller thrust loading [17]. With increasing thrust loading, the side force of the rudder also increases, as well as the angle at which stall occurs [33]. Lastly, the rudder acts as a post-swirl stator, regaining some of the rotational losses in the propeller wake field. Some rudders are optimised for this purpose, and will be discussed in section 2.6.3.

2.3.5. Rudder-Hull Interaction

The most important interaction between the hull and rudder has already been elaborated on in section 2.2.2: the rudder serves as the initiator of a turn. However, there is one other notable effect, being the flow straightening effect [34]. While neglecting the effect of the hull on the flow, sailing under a drift angle (for instance due to applying a rudder angle) will lead to a decreasing effective inflow angle to the rudder. However, the presence of the hull will lead to a straightening effect. This will lead to an increasing inflow angle and recovery of rudder lift. This recovered rudder lift translates to a higher overall efficiency because less rudder angle is needed leading to a smaller rudder drag.

2.4. The Wake Field

The region dominated by viscous flow behind the ship hull is called the wake field. While the wake field refers to this entire region, in the context of hydrodynamic interaction, the term is used to denote the flow field at the propeller plane. Due to viscous and inertial effects discussed in section 2.3.2 the wake field is usually highly non-uniform. Figure 2.7 shows an example of a hull wake field for a single-propeller container ship at the propeller plane.

A wake field plot, as seen in figure 2.7, usually uses isosurfaces or numbers in the plot to denote the ratio of local axial velocity compared to ship velocity, v_a/v_S. The arrows denote the direction and magnitude of the in-plane flow, consisting of the relative tangential velocity v_t/v_S and relative radial velocity v_r/v_S. The plot is viewed from behind the ship, looking forward towards the bow.

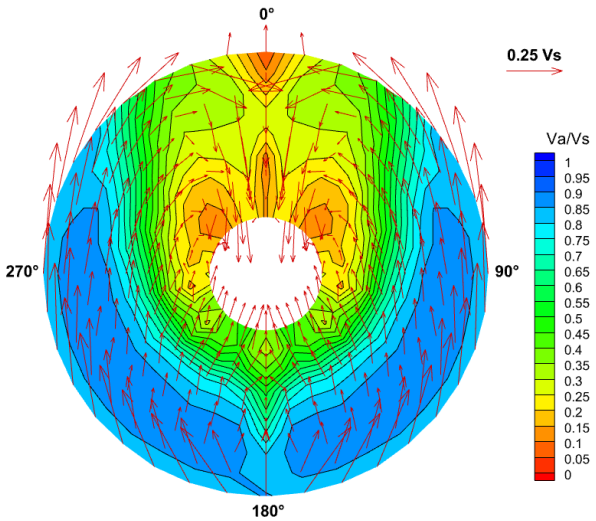


Figure 2.7: A wake field for a single-propeller container ship.

From figure 2.7 follows that due to the presence of a wake, the flow velocity diminishes locally. Sometimes, flow even reverses [20], leading to negative v_a/v_S values. Looking at the in-plane flow, it can be seen that the flow goes upward and inward, as can be expected for this single-propeller configuration. In the upper half of the wake field the influence of the hull is largest, both in terms of axial velocity and in-plane velocity. The arrows clearly show the effect of vortex-type separation leading to an inward curl in the top half plane of the wake field.

Every hull shape has its own unique wake field for every velocity [20]. The wake field at the propeller plane is dependent on the shape of the aft ship, the number of propellers and the placement of the propeller. For instance, a pulling podded propeller is likely to operate in a more uniform wake field than a propeller operating directly behind a high block coefficient oil tanker. A wake field is typically obtained by performing model tests or a model- or full-scale CFD simulation.

Like discussed in section 2.3.3, the presence of a propeller changes the wake field. Several types of wake fields can be distinguished, depending on the effects taken into account. The ITTC distinguishes between two types of wake fields: nominal and effective/total [35]. The nominal wake field is the wake field due to the hull alone. It consists of potential flow, viscous, wave action, and non-linear components. The effective or total wake field consists of the nominal wake field combined with interaction effects due to the presence of the propeller. It is an imaginary type of wake field and can be interpreted as the wake field 'felt' by the propeller. Carlton [20] splits the effective/total wake field up into an effective and a total wake field, where the latter represents the wake field as it could be measured in real-life with the propeller present.

The effective wake field is used by propeller designers to base their designs on. In the Wärtsilä in-house propeller design software Archimedes, a nominal wake field is inserted which is converted to an effective wake field using the theoretical method of Huang and Groves [36]. This theoretical method takes the nominal wake field, as well as the non-dimensional thrust coefficient C_T (see equation 2.4) as inputs [20], and outputs the effective wake field. Figure 2.8 illustrates the differences between the three wake field types.

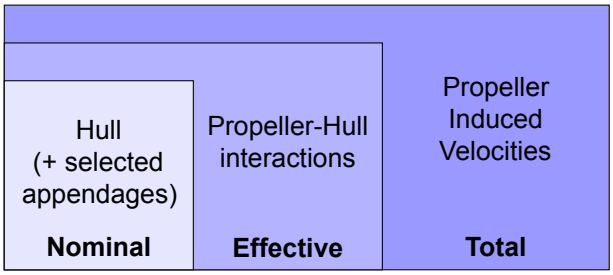


Figure 2.8: Nominal, effective and total wake field.

2.5. Cavitation

Figure 2.9 shows the phase diagram for water [20]. It can be seen that the phase of water is a function of pressure p and temperature T. This implies that at a constant temperature, such as the temperature of seawater, pressure fluctuations can cause water in its liquid form to evaporate. This phenomenon, known as cavitation, can have negative effects, including vibrations and erosion of the propeller, the hull, and other aft ship components, as well as disruptive onboard noise. Additionally, cavitation can disturb marine life and reduce stealth.

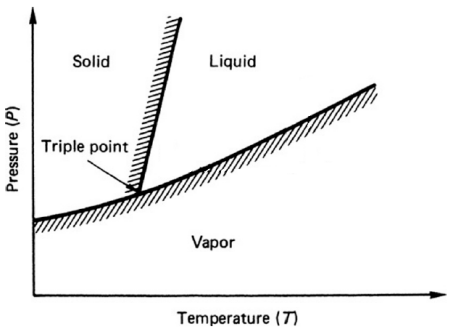


Figure 2.9: The phase diagram for water. Adapted from [20].

As propellers create pressure differences to generate thrust, cavitation should be accounted for during propeller design. In fact, propeller design is almost always a trade-off between performance and cavitation behaviour. Whether

or not cavitation will occur can be predicted using the cavitation number [21]:

$$\sigma = \frac{p_0 - p}{\frac{1}{2}\rho v_0^2} \tag{2.7}$$

Where p_0 is the ambient pressure, p the local pressure, ρ the water density and v_0 the reference velocity. The cavitation number corresponding to cavitation inception σ_v is found by substituting the vapour pressure p_v for the local pressure p in equation 2.7. If $\sigma < \sigma_v$, cavitation will occur.

Sometimes, also a pressure coefficient C_p is defined [25]. As the pressure coefficient is defined as the negative cavitation number, cavitation will occur when $\sigma_v < -C_p$:

$$C_p = \frac{p - p_0}{\frac{1}{2}\rho v^2} \tag{2.8}$$

Due to the larger pressure fluctuations, cavitation increases with higher propeller loadings. Different types of cavitation patterns on ship propellers are shown in figure 2.10 [37]. In the aft ship, propeller-induced cavitation can occur for multiple reasons:

- Increased propeller loading leads to higher lift and drag coefficients, causing lower minimum pressures.
- The interaction between the propeller and another aft ship component, such as the hull, especially if the clearance between hull and propeller is small.
- Propeller-induced pressure fluctuations due to the finite number of propeller blades [21].
- Pressure fluctuations resulting from the non-uniform inflow to a propeller blade over a revolution.

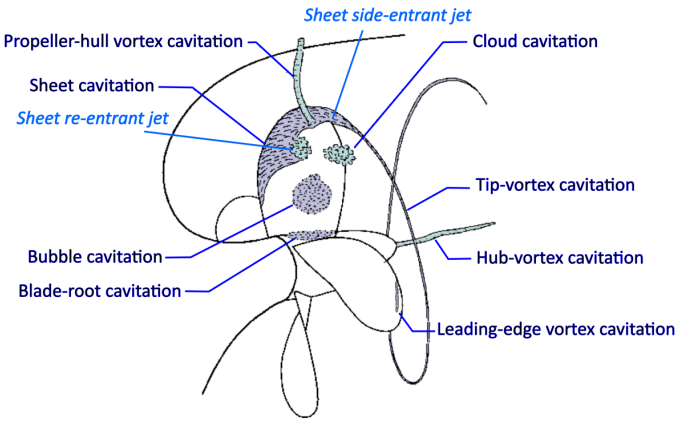
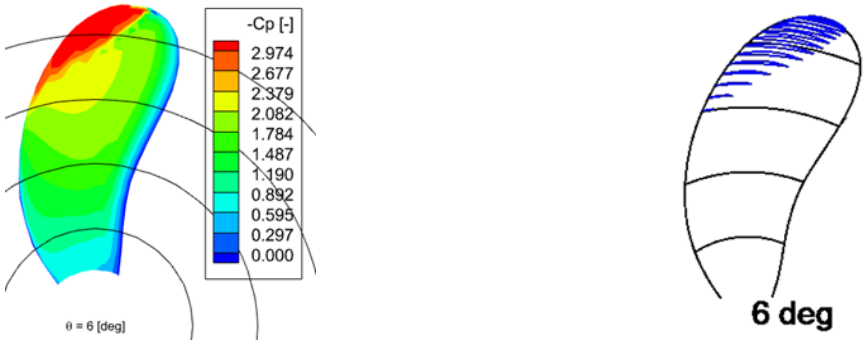


Figure 2.10: Possible cavitation patterns on ship propellers. From [37].

The latter of the mentioned reasons is highly connected with the wake field behind the ship hull, which is why it is so important to be informed about the effective wake field while designing a propeller. For a single-propeller vessel, the wake field typically has a V-shaped wake peak at the 12 o'clock position [38]. In this wake peak, velocities are low due to hull-induced flow disturbances. When a blade passes through this wake peak, the hydrostatic pressure is at its minimum. Furthermore, the reduced axial velocities in the wake peak lead to high angles of attack on the propeller. Therefore, the local pressure is at its lowest, increasing the risk of cavitation. It can be reasoned that the axial and tangential flow directions have the highest influence on pressure fluctuations, as they together determine the angle of attack on the propeller blade and subsequently the lift and drag coefficients, as well as the pressure coefficient.

While assessing wake field-induced cavitation on a blade, typically the blade position with the largest cavitation volume, or the volume of all vapour-filled cavities together, is selected. This position typically

lies close to the position with the lowest margin against cavitation (the lowest pressure on the blade). The position of the largest cavitation volume can typically be found shortly after the zero-degree position marked by the propeller generator line. At this point, the propeller has (almost) passed the wake peak, and all wake field-induced cavitation is fully developed. The total cavitation volume on a blade at this position can serve as a measure of the influence of the presence of the hull on propeller cavitation. An example can be seen in figure 2.11.



(a) Distribution of pressure coefficient C_p.

(b) Distribution of cavitation volume.

Figure 2.11: Assessment of wake field induced cavitation on a propeller blade after wake peak passing.

2.6. Energy Saving Devices

Appendages, for instance in the form of fins, ducts or stators, that are aiming to optimise the hydrodynamic behaviour of a vessel, are called energy saving devices (ESDs). All energy saving devices either aim to decrease the thrust deduction by improving the propeller-hull interaction, or to improve the propulsor's efficiency [39]. After a short period of interest during economical and oil crises in the 1970s, when oil prices dropped again in the 80s the interest in ESDs diminished. This was also due to difficulties regarding discrepancies between model- and full-scale test results [40].

Due to the goal of reducing emissions worldwide and rules and regulations that follow from this, together with increasing fossil fuel prices, energy saving devices have regained interest. As was already stated in the introduction of this report, not only fossil fuel powered vessels can benefit from increased hydrodynamic efficiency. For vessels that are powered by alternative of renewable fuels, that often still suffer from volumetric or gravimetric fuel storage constraints, the increased autonomy by making use of energy saving devices can increase their range of application.

There are various different types of ESDs, and they can be classified in various ways. Mysa, for instance, classifies ESDs based on whether they are located upstream or downstream of the propeller [42]. Carton and Jin also use this classification, and so does Anschau. However, the former authors also distinguish ESDs that are located at the propeller plane [20, 43], and the latter also distinguish ESDs that are a combination of upstream and downstream ESDs [44]. Another way of classification is working principle based. Xu divides between ESDs that aim to optimise the inflow of water to the propeller, recover propeller-induced rotative energy, reduce the tip vortex strength or are a complete substitute for a conventional propeller [45]. Turkmen [46] divides between post-swirl, pre-swirl, stern flow regulating and complex ESDs. Lastly, Spinelli [47] divides between pre-swirl, ducted and post-swirl ESDs, which is also the division that will be used in this section.

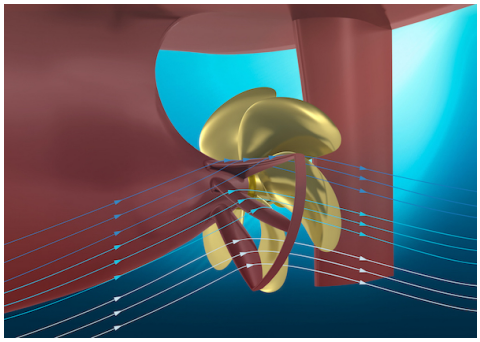


Figure 2.12: Wärtsilä EnergoFlow. From [41].

2.6.1. Pre-swirl

Apart from generating thrust, the rotating motion of a propeller leads to rotational kinetic energy losses in its wake, that get larger for higher propeller rotation rates. The idea behind pre-swirl generating ESDs is that by inducing pre-swirl the propeller is able to operate at a lower rotation rate while still encountering the same angle of attack [39]. In this way, rotational kinetic energy losses will theoretically reduce and ideally diminish. Apart from being able to decrease rotational kinetic energy losses, pre-swirl stators can also help to decrease fluctuations in blade loadings. As discussed in section 2.7, due to ship geometry the water velocity in the wake field often has a positive z-component. This can be beneficial during a downward movement of a propeller blade, but decreases the lift during upward movement and induces fluctuations in the blade loading. An asymmetric pre-swirl stator can help to reduce this effect. An example of a pre-swirl stator is the Wärtsilä EnergoFlow [41], which is shown in figure 2.12. The EnergoFlow consists of three foils that are connected at the tips. It makes sense that the EnergoFlow is located on port side: if it were placed on starboard side but still deflected the flow downward, it would actually reduce the thrust generated by the clockwise-rotating propeller.

2.6.2. Ducts

Ducts that aim to optimise the inflow to the propeller plane have a long history, described in many papers [39, 45–47]. Consisting of a curved hydrofoil, they can be either symmetrical or asymmetrical. Ducts are usually placed around the propeller (ducted propellers), or upstream of it. Ducted propellers are typically used in low-speed situations, such as towing (bollard pull) and trawling. Because they directly increase the velocity at the propeller, they reduce propeller thrust and torque. However, the duct itself generates a thrust that is higher than the decrease in propeller thrust at low speeds. In bollard pull, up to 50% of the thrust can be generated by the duct [20]. However, due to drag increase the efficiency of the duct diminishes with increasing velocity.

Upstream ducts typically aim to optimise the inflow to the propeller, improving the propeller-hull efficiency. Some examples can be seen in figure 2.13 [47]. It can be seen that some upstream ducts operate in combination with pre-swirl stators. Most upstream ducts are located at the top half of the propeller plane. By increasing the flow velocity in this top half, the wake field becomes more uniform, reducing pressure fluctuations and possibly cavitation on the propeller blades.

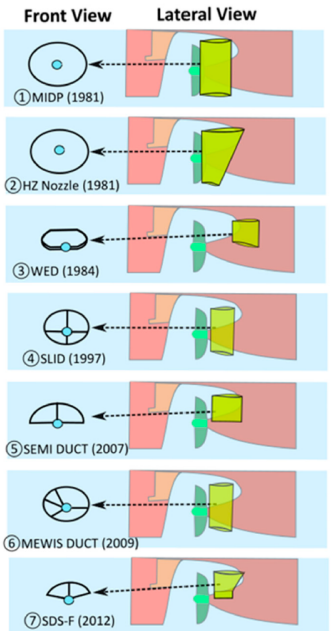


Figure 2.13: Various upstream duct designs. From [47].

2.6.3. Post-swirl

Energy saving devices of the post-swirl category aim, just like pre-swirl ESDs, to mitigate rotational kinetic energy losses. However, they act as a 'post-treatment' instead of generating pre-swirl, which is why they can also be classified as 'energy recovery' energy saving devices. There are various ESDs that are designed to operate behind the propeller, such as different kinds of blades or fins (propeller boss cap fin, hub vortex vane) that recover energy from the propeller hub vortex [47]. Another existing application is extending the propeller hub towards the rudder [45]. This can improve the flow straightening effect of the rudder. Furthermore, this decreases the size of the zone of low pressure behind the propeller hub, decreasing the effect of the propeller hub vortex.

2.7. The Gate Rudder

The gate rudder was invented by Sasaki and Kuribayashi in 2012 and first mentioned by Sasaki in 2013 [8, 9]. The main difference between the gate rudder and a conventional rudder is that the two blades of a gate rudder are placed alongside the propeller instead of in the propeller slipstream. Because of this, the gate rudder is supposedly able to generate a net thrust force instead of a net drag, which is one of the main reported working principles of the gate rudder [48]. Initial designs were a two-foil configuration on both sides of the propeller that did not wrap around the top side of the propeller, and a configuration that wrapped around the top of the propeller in which both blades were connected and

could only be pivoted together [49]. The most recent design consists of two foils that wrap around the top and sides of the propeller and are able to pivot individually, and can be seen in figure 2.14 [10]. In figure 2.14, the first full-scale gate rudder ever installed on the 2400 deadweight tonnage vessel Shigenobu is shown. The Shigenobu was equipped with a gate rudder system (GRS) in 2017, while

Figure 2.14: A gate rudder installed on the Shigenobu. From [10].

her sister ship Sakura remained sailing with a conventional rudder system (CRS), enabling comparing the performance between the vessels. Next to the Shigenobu, two other vessels were equipped with a GRS in 2021, being the Shinmon maru and Koshin maru. In 2022 the Nogami was converted, being the fourth vessel featuring a gate rudder [50].

The gate rudder is currently being investigated by the European Union-funded project GATERS, which has a sub-license agreement with the gate rudder patent holder, Wärtsilä Netherlands [11]. Within this project, MV Erge became the fifth gate rudder equipped vessel in 2023. MV Erge also has a sister ship, MV Erle, allowing for comparative studies [51].

2.7.1. Reported Benefits

The main benefit of the gate rudder over a conventional rudder is the alleged power saving. Both the Shigenobu and the Erge reported significant savings after their retrofit to a GRS. For the Shigenobu, an average power saving of 14% has been reported, with peaks up to 27% compared to her sister vessel Sakura [10, 53]. The Erge reported power savings over 35% after being retrofitted to a GRS compared to the situation just before retrofitting. Compared to the power usage just after newbuild, the savings were still over 20% [51]. However, discrepancies arise when comparing the full scale results to model tests and CFD results. For instance, in 2015 Sasaki [49] only reports savings in the range of 6-8% for model and CFD tests. Bureau Veritas even reports an adverse power saving of 3-4% while evaluating the feasibility of implementing a GRS on a container vessel [54]. On the other hand, Köksal [55] reports a 22% benefit regarding power requirements for a GRS-equipped 1:21.7 scale model of the Erge.

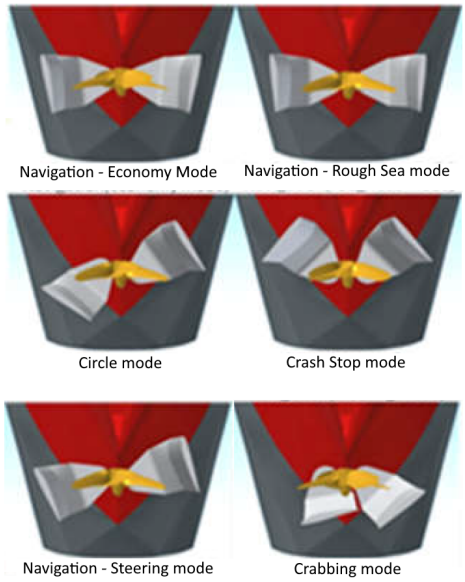


Figure 2.15: Gate rudder operation modes. Adapted from [52].

A possible explanation for the observed discrepancies are scale effects. Because of smaller Reynolds numbers for model tests, flow around model scale vessels tends to be more laminar than in real-life. Because a conventional rudder is positioned in the turbulent propeller slipstream and the gate rudder is not, the gate rudder will suffer more from this effect, leading to larger differences between model and full scale [56, 57]. Hussain et al. [58] performed a numerical study on small (3 m) and large (6 m) model scale, as well as full scale (69 m), and found that on larger scales the gate

rudder generates relatively more thrust compared to scaled-down models. Furthermore, due to the gate rudder affecting the advance velocity of the propeller and this phenomenon also being subject to scale effects, the ITTC wake scaling procedure can not be applied when a gate rudder is applied [58].

Another reported benefit of the gate rudder is its improved manoeuvrability and better seakeeping compared to a conventional rudder. Because both gate rudder blades are individually rotatable, a lot of different configurations can be used, as can be seen in figure 2.15 [52]. After the retrofit of Erge [51] it was directly reported that overshoot angles while steering were reduced, the turning circle speed increased and the crabbing mode turned out to work conveniently. However, also for manoeuvrability different studies provide discrepant findings. From MARIN research, for instance, no manoeuvrability benefits for a 214 m vessel were found during model tests. [59]. Instead, a negative outcome was found compared to a conventional rudder due to sub-optimal use of the propeller flow field.

Lastly, model tests of the Erge show that using a gate rudder less cavitation on the propeller occurs due to the lower propeller thrust. Furthermore, the gate rudder is not located in the wake of the propeller like a conventional rudder, preventing shed cavitation to collapse on the rudder blade [60]. This leads to less noise emissions. What adds to the reduction of noise is the fact that the gate rudder serves as a sound barrier around the propeller, reflecting and scattering emitted noise. Model tests of the Sakura show similar results [61].

2.7.2. Working Principles

At the time of writing, there is no consensus on the primary working principle of the gate rudder. However, the theory most commonly mentioned in GATERS and Sasaki literature is the net thrust effect. This theory states that the gate rudder delivers net thrust, in contrast to a conventional rudder, which typically generates net resistance [46].

The gate rudder is shaped like an accelerating duct. In the aft ship, water flows inward toward the propeller plane. Combined with the forward ship speed and the gate rudder's position at both sides of the propeller, the gate rudder generates a lift force that can be decomposed in a force parallel to the sailing direction and a force perpendicular to the sailing direction, oriented to the propeller [46]. This effect in combination with the accelerating duct effect leads to the presence of a region of low pressure located on the insides of the gate rudder blades at the leading edges, increasing the positive force on the gate rudder blades and unloading the propeller.

A critical point to note is the gate rudder's function as a duct: tip clearance, or the distance from the propeller tip to the inside of the duct, highly affects the duct efficiency [62]. Generally, tip clearance should be kept as low as possible [20]. This condition is not met by the gate rudder, making it less likely that double-digit efficiency improvements arise solely from the duct effect. However, a numerical study described that a beneficial effect was found when placing the gate rudder at a distance of $1.25R$ compared to $1.5R$ from the propeller, where R is the propeller diameter [46]. Therefore, the gate rudder is usually designed to operate at a distance of 1.2-1.4R from the propeller [58].

2.8. Conclusions

This chapter aimed to provide an answer to the first and third sub-question of this thesis. These sub-questions are as follows:

What is the importance of understanding the nominal wake field in propeller design, and at which points in the aft ship design process can predicted nominal wake fields be applied?

What criteria determine the quality of a predicted nominal wake field, and what are the requirements for accurate nominal wake field prediction?

This chapter has provided a hydrodynamical background for this thesis report. After introducing the main aft ship components, propellers and rudders, the complex relationship between these components and the ship hull was elaborated on. To design a propeller with optimal performance and to prevent cavitation-induced hindrance levels to be exceeded, knowledge of the wake field is essential. In particular, understanding the effective wake field is crucial. It was explained that the current incorporation of wake field information in the propeller design process involves obtaining a nominal wake field through model or CFD tests, after which the method of Huang and Groves is used to scale the nominal wake field

up to the effective wake field. As was mentioned in the introduction to this thesis, due to ever-increasing demand for higher efficiencies and the resulting developments in the field of energy saving devices and the gate rudder, iterative and integrated aft ship design processes might in the future be preferred over linear ones. As performing model tests after every small change in hull form or after every appendage is infeasible, and performing CFD calculations is costly and time-intensive, being able to quickly predict a nominal wake field would be a good first step in facilitating this modern iterative design approach.

The quality of a predicted nominal wake field regarding propeller performance is determined by the level of accuracy in representing the velocity distribution behind the ship hull. This implies that the average predicted velocity in the ship wake (represented by the wake fraction w), as well as individual predicted velocities in the wake field, should lie close to the expected values. In this project, predicted wake fields will be evaluated using a score assigned by a “loss function” (a concept that will be introduced in section 3.1.6), which will be used to fine-tune the models.

When evaluating the quality of a predicted nominal wake field in terms of cavitation behaviour, its performance depends on the extent to which a propeller designer can use the prediction to estimate the wake-induced cavitation behaviour of a propeller. This means that the wake field should accurately identify regions with strong velocity fluctuations, as these lead to pressure fluctuations that may induce cavitation. To assess the predicted nominal wake fields for cavitation performance, some predictions will serve as input to a cavitation assessment tool. This process will be elaborated on in section 7.6.

3

Machine Learning Theory

In the field of artificial intelligence (AI), machine learning (ML) involves creating algorithms that classify or predict the outcome of an event based on some input, without having previously seen the "correct" output. The models used are regression or classification models. There are different ways to train a machine learning model, but the training always involves a dataset of some kind. Although the name "artificial intelligence" suggests otherwise, machine learning algorithms are not really intelligent. Instead, statistics lie at the basis of all algorithms, varying from the simplest linear regression to the most complex deep neural network algorithms.

This chapter will present a background on machine learning and, similar to chapter 2, is partially adopted from the literature review conducted as part of this thesis project [13]. The aim of this chapter is to partially answer the second sub-question of this thesis, which was presented in the thesis introduction:

Which machine learning algorithms and structures are most suitable for predicting the nominal wake field, and what are the steps that need to be taken to develop such models?

In this chapter, part of this question will be addressed. This will be achieved by first introducing key machine learning concepts and terminology in section 3.1. Subsequently, the two machine learning algorithms used throughout the thesis project, namely support vector machines and neural networks, will be elaborated on in sections 3.2 and 3.3, respectively. Thereafter, existing literature on wake field prediction using machine learning is elaborated on in section 3.3.5. Finally, a conclusion to this chapter will be presented in section 3.4.

3.1. Fundamentals of Machine Learning

In this section, the fundamentals of machine learning will be discussed. First, the main concepts and some terminology are introduced, and an explanation of what a dataset consists of follows. Thereafter, the concepts of classification, regression, machine learning types and feature engineering are explained. Lastly, the use of a loss function will be elaborated on.

3.1.1. Main Concept and Terminology

Every machine learning model is equivalent to a function $\mathbf{y}(\mathbf{x})$ that takes input $\mathbf{x}$ and transforms it to an output $\mathbf{y}$ [63]. Two phases in the existence of a machine learning model can be distinguished: the learning phase and the production phase. During the learning phase, the function $\mathbf{y}(\mathbf{x})$ is defined: the machine learning algorithm is selected, the values of relevant hyperparameters are defined and the model parameters are determined through model training. Hyperparameters are settings that are defined before training a machine learning model, such as the amount of layers in a neural network or the learning rate. Model parameters are the parameters that are determined during the training of the machine learning model and essentially contain the "learned information" by the machine learning model. For example, $\mathbf{w}$ and $\mathbf{b}$ in the linear regression equation $\mathbf{y} = \mathbf{w}\mathbf{x} + \mathbf{b}$ are model parameters [64].

Several activities occur during the learning phase. First, a data set needs to be collected and prepared.

Thereafter, the inputs (or "features") of the machine learning model need to be extracted from the data set in a process called feature engineering. Next, the selected machine learning model is trained, meaning that all model parameters are determined by using a learning algorithm. Lastly, the trained model is evaluated and if necessary a next design iteration is started. When the machine learning model has been tested and validated, it can be deployed and enters the production phase, where it can be used on new examples that it has never encountered before. The machine learning project life cycle by Burkov [65] is shown in figure 3.1 and will be used throughout this thesis report.

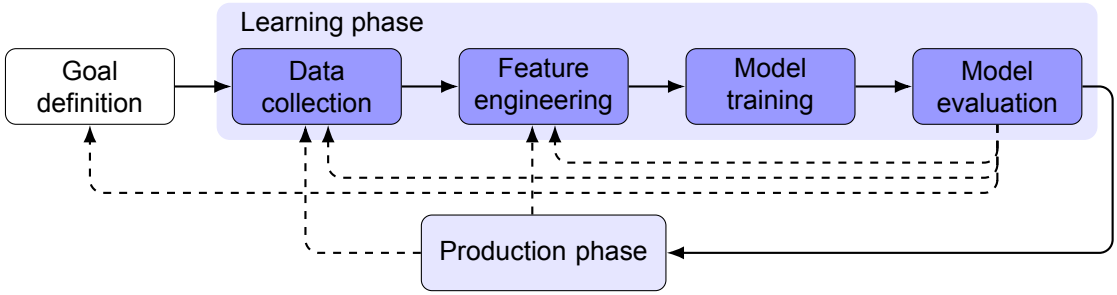


Figure 3.1: Machine learning project life cycle. Adapted from [65].

3.1.2. The Dataset

In a machine learning project, a dataset is typically represented as $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$. In this set, every vector $\mathbf{x}_i$ is called a feature vector. A feature vector $\mathbf{x}_i$ is a vector of features with length D: $\mathbf{x}_i = \left[x_i^{(1)}, x_i^{(2)}, \ldots x_i^{(D)}\right]$. Every feature corresponds to a property of the input, and each feature in every feature vector corresponds to the same property. In the case of this thesis, for instance, $x^{(1)}$ might refer to the ship velocity v_S. If so, every feature vector $\mathbf{x}_i$ would have its first entry $x_i^{(1)}$ represent ship velocity. The amount of entries in the feature vector is called the dimensionality D [64]. Note that the subscript denotes the vector in the dataset, and the superscript between brackets the feature within a vector.

Every element y_i is a label, which contains the "right answer" given the feature vector. It is used to properly train a machine learning model in supervised and semi-supervised learning. As will be discussed later, some types of machine learning do not require the data set to be labelled, or only to be partially labelled.

For machine learning purposes, the dataset is split into minimally two subsets, the largest of which (usually 80% or more) is called the training set. The training set is used to determine the model parameters, such as the weights in a neural network. In addition to the training set, there is a validation set [63]. This validation set is used to check the chosen hyperparameters and the selected algorithm. It is not used in fitting the model parameters, so it can serve as an indicator of the quality of the fit. Sometimes, a so-called test set is kept apart. Burkov [64] recommends using the test set to check the performance of the model before going into production. As it is not used to fit the model parameters or tune the hyperparameters and is therefore unseen by the machine learning model, the test set can be used to get unbiased insight in model performance. Bishop [63] recommends using a test set to evaluate the performance of the model if overfitting is suspected.

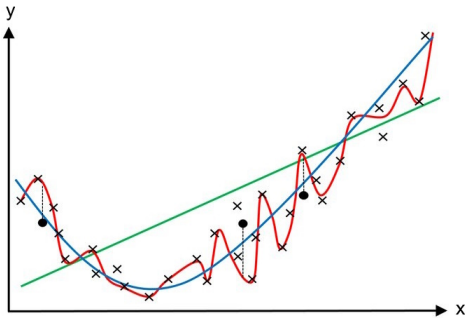


Figure 3.2: Underfit (green curve), overfit (red curve) and balanced fit (blue curve) on a dataset. Adapted from [66].

A machine learning algorithm and the tuning of its hyperparameters could be too general, leading to an underfit of the data (equivalent to using a linear function to estimate a higher-order function). On the other hand, an overfit could also occur (equivalent to using a higher-order function to estimate a

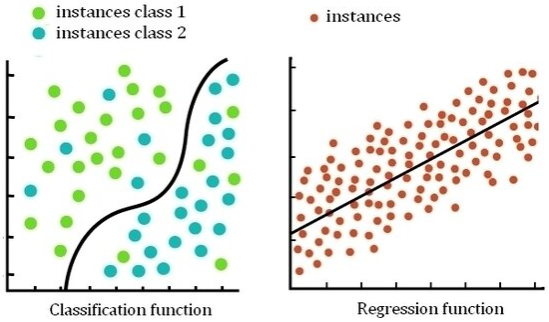


Figure 3.3: Classification versus regression. Adapted from [67].

low-order function). This means that the model is so tightly fit to the training data and its potential errors that it can predict the training data really well, but will very poorly work on validation or test data. It is important to select the algorithm and tune the hyperparameters in such a way that a balanced fit is achieved. Underfit, balanced fit and overfit are illustrated in figure 3.2 [66].

3.1.3. Classification and Regression

Machine learning models can be used for classification or regression. The difference between classification and regression can be seen in figure 3.3 [67]. In classification, a machine learning model aims to assign a label to an unlabeled example. There is a finite set of classes to which every label belongs [64]. This classification can be binary (on/off, spam/no spam, day/night) or multiclass (red/orange/green, high/medium/low). During the training phase of a classification model a decision boundary is constructed. This decision boundary divides the samples from different classes as good as possible: all data points on one side get class A, all data points on the other side get class B. The way in which the decision boundary is directed influences the classification performance of the model. In multiclass classification, more than one decision boundary is present. When the data is not perfectly separable, defining a decision boundary is always a trade-off.

In regression, a value is assigned to an unlabelled example. This is directly analogous to a regression line in a graph. The decision on which type of regression should be taken (algorithm selection) and the fine-tuning of the hyperparameters happens in the learning phase. To judge the quality of a regression, a cost function (such as least-squares) should be minimised. However, the cost function does not necessarily say anything about the level of overfitting.

3.1.4. Types of Machine Learning

There are three main branches in machine learning: supervised, unsupervised and reinforcement learning [68]. Semi-supervised learning, a hybrid method between supervised and unsupervised learning, is often considered as a fourth branch [63, 64]. The main principles of supervised, unsupervised, semi-supervised and reinforcement learning are as follows:

- **Supervised learning**
  In supervised learning, the machine learning algorithm optimises a function that maps an input to an output [69]. It does so by learning from labelled examples $(\mathbf{x}_i, y_i)$. Both regression and classification functions can be derived from supervised learning.
- **Unsupervised learning**
  Unsupervised learning takes an unlabelled dataset $\{(\mathbf{x}_1), (\mathbf{x}_2), \ldots, (\mathbf{x}_N)\}$, and analyses it. Unsupervised classifiction is also called clustering [70]. It can be used to automatically divide data points in groups, find outliers, or for generative purposes [69].
- **Semi-supervised learning**
  Semi-supervised learning is a cross-over between supervised and unsupervised learning. Usually, labelled data is rare and unlabelled data are numerous [69]. Semi-supervised learning works just like supervised learning. The unlabelled data might seem useless, but can help the algorithm identify the distribution of the data. In other words, information is added to the machine learning

model, potentially leading to better predictions [64].

- **Reinforcement learning**
  Reinforcement learning is an environment-driven approach [69], meaning that the model operates "in an environment". This environment could be, for example, a chess match. There are usually many choices to make and the optimal sequence of choices depends on external factors (the opponent). By coupling risk and reward to certain actions, the expected average reward can be calculated and the model can choose the optimal action [64].

3.1.5. Feature engineering

The process of transforming raw data into an usable dataset consisting of input vectors with features is called feature engineering [65]. Part of this process is conceptual and consists of determining which features to include. This choice is ideally based on specific knowledge on the problem, as it bounds the scope of the machine learning model: the regression is only based on the features that are taken into account. The other part of the process consists of creating the features and feature vectors. This can be done manually or automatically. Features can for instance be scaled raw variables, combinations of variables or categorical values converted to a numerical value.

According to Bengio et al., feature engineering is an important but labour-intensive process [71]. Furthermore, it displays the main weakness of most commonly-used machine learning models: a model on its own is unable to separate the wheat from the caff in a raw dataset. According to a 2016 survey from Forbes, data scientists spend 80% of their time on cleaning and organising data, which includes feature engineering [72]. Feature engineering is arguably the most critical step in the development of a machine learning model. It is relatively simple to train multiple different models and choose the best one, but without a properly engineered dataset no model will be able to generalise and produce sensible results. Good features are unique, and informative, which means that they should be highly relevant to the target variable(s).

As machine learning models can only handle numbers, all non-numerical data has to be converted to numerical values. Various methods exist to convert all types of data to numerical values, a lot of which are discussed by Burkov [65]. One used in this thesis project is one-hot encoding, which converts categorical data into numerical values without introducing ordinality. This method creates n columns, with n being the number of categories, and assigns a value of 1 to the column representing the category while keeping all other columns to 0.

It is common practice to scale numerical features, for instance by normalising to the interval [0,1] or [-1,1], or by standardising so that the mean μ of the feature equals 0 and the standard deviation σ equals 1. There are several reasons to do so. In deep neural networks, scaled data speeds up the training speed [65]. Scaling features ensures that every feature has an equal contribution, preventing domination of a single feature [73]. Lastly, by forcing the features in a specified range, the risk of numerical overflow errors due to very large or small numbers is mitigated.

3.1.6. The Loss Function and Gradient Descent

An essential component of every machine learning model is the loss function. Given a single prediction made by a machine learning model, for instance during model training, the loss function is some function of this prediction $\hat{y}$ and the label value y, which is also known as the target. Because we are usually interested in the average performance of a machine learning model on the entire validation set, a function of all losses together is calculated, usually being an average. This function is known as the objective function. An example of an objective function is shown in equation 3.1. Here, the loss function is the squared difference between the prediction and the target, making this objective function the mean squared error (MSE).

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.1}$$

Sometimes also a cost of input factor is added to the objective function. A cost of input factor is a numerical penalty that depends on the magnitude of the trained input weights. The idea behind penalising large model weights is that smaller weights lead to simpler models. Usually, a trade-off hyperparameter (for instance C) determines the balance between loss and cost of input.

The aim of the training phase is to minimise the objective function by finding the optimal model parameters. In order to do this, the loss function is derived with respect to the model parameters, which are part of the prediction $\hat{y}$. By finding the gradient, which indicates the direction of the steepest ascent of a function, the direction of largest decrease of the loss (and therefore objective) function can be found. A step with a certain size (learning rate α) can now be taken. This concept is called gradient descent, and lies at the core of many machine learning algorithms. Its process is shown in equation 3.2 [63].

$$\mathbf{w}_{j+1} = \mathbf{w}_j - \alpha \nabla L_i \tag{3.2}$$

Where $\mathbf{w}$ is the parameter vector, j is the iteration, α denotes the learning rate hyperparameter and L_i equals the loss function for sample i. The variation of gradient descent which updates the model parameters sample by sample based on the individual losses per sample, is called stochastic or sequential gradient descent. If the model parameters were updated by calculating the gradient based on the combined loss of multiple samples, it would be called batch gradient descent [74].

3.2. Support Vector Machines

One of the two machine learning algorithms applied in this thesis project is support vector machine (SVM). SVM, a supervised machine learning algorithm, is widely used for classification. Within this thesis project, the regression equivalent of SVM was used, which is called support vector regression (SVR). Section 3.2.1 provides a background and introduction to support vector machines. In section 3.2.2, the differences between support vector regression and support vector machine are explained. Later in this thesis, sections 5.3 and 6.2 will elaborate on the application of a support vector machine and support vector regression in the context of the thesis project, respectively.

3.2.1. Working Principles of Support Vector Machines

The key principle behind SVM is to consider the input data in high-dimensional space [63]. By trying to split the data with high-dimensional planes called hyperplanes, the class of a new input can be predicted by determining its location with respect to the hyperplane. The hyperplane is therefore also called the decision boundary.

The equation for a D-dimensional hyperplane, where D is the number of features in $\mathbf{x}$, is as follows:

$$\mathbf{w}\mathbf{x} + b = 0 \tag{3.3}$$

With parameters $\mathbf{w}$ and b, where $\mathbf{w}$ is a real-valued vector and b a real-valued constant.

The objective when training a SVM model is to find the optimal set of parameters $\{\mathbf{w}^*, b^*\}$ which do not only correctly classify all data in the train set, but also maximise the margin around the constructed hyperplane in order for the model to better generalise to new, unseen samples. This is visualised in figure 3.4 [74].

Something that can also be seen in figure 3.4 is that samples to the right of the hyperplane have a positive value, whereas samples to the left are negative. By checking the similarity in sign between a label y and a prediction $\hat{y}$, loss can be assigned to incorrectly labelled samples. On top of this classification loss, the margin in figure 3.4 is given by $\frac{1}{||\mathbf{w}||}$. Thus, by minimising $||\mathbf{w}||$ the margin is maximised.

The final optimisation problem, as explained by Brunton, is shown in equation 3.4 [74]. A hyperparameter C that weighs the relative importance of the classification loss with respect to the margin size is included in the objective function. High values of C will lead to a smaller margin but better classification accuracy on the training data, while low values of C will lead to a larger margin but potentially lower classification accuracy. Equation 3.5 summarises the loss function for an SVM.

$$\min C \sum_{j=1}^{m} \underbrace{\ell(\mathbf{y}_j, \hat{\mathbf{y}}_j)}_{\text{classification loss}} + \underbrace{\frac{1}{2}||\mathbf{w}||^2}_{\text{margin}} \tag{3.4}$$

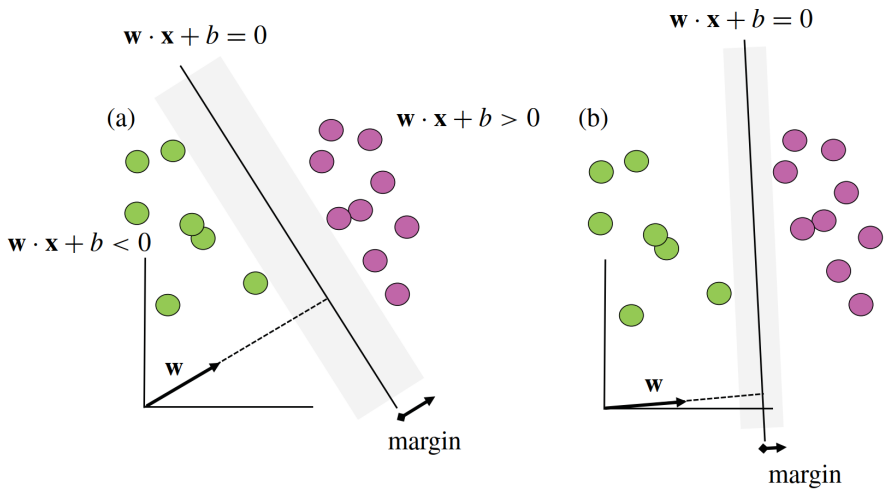


Figure 3.4: Two potential hyperplanes applied to the same set of data. In (a) the hyperplane margin is maximised, as opposed to the situation in (b). From [74].

where

$$\ell(\mathbf{y}_j, \hat{\mathbf{y}}_j) = \begin{cases} 0, & \text{if signs of } \mathbf{y}_j \text{ and } \hat{\mathbf{y}}_j \text{ match;} \\ +1, & \text{otherwise} \end{cases} \tag{3.5}$$

As can be seen in equation 3.5, the loss function is not forgiving of noise in the data set. If the data is not linearly separable, a hyperplane with as many misclassified samples *as close to* the decision boundary is preferred over a hyperplane with misclassified samples further away. This is not accounted for by the current loss function ℓ. To do so, and also prevent optimisation errors due to the discrete nature of the loss function ℓ, the smooth Hinge loss function H is introduced. This Hinge loss function is zero if the sample is correctly labelled, and proportional to the distance from the decision boundary if not [64]:

$$H = \max(0, 1 - \mathbf{y_j}(\mathbf{w}\mathbf{x_j} - b)) \tag{3.6}$$

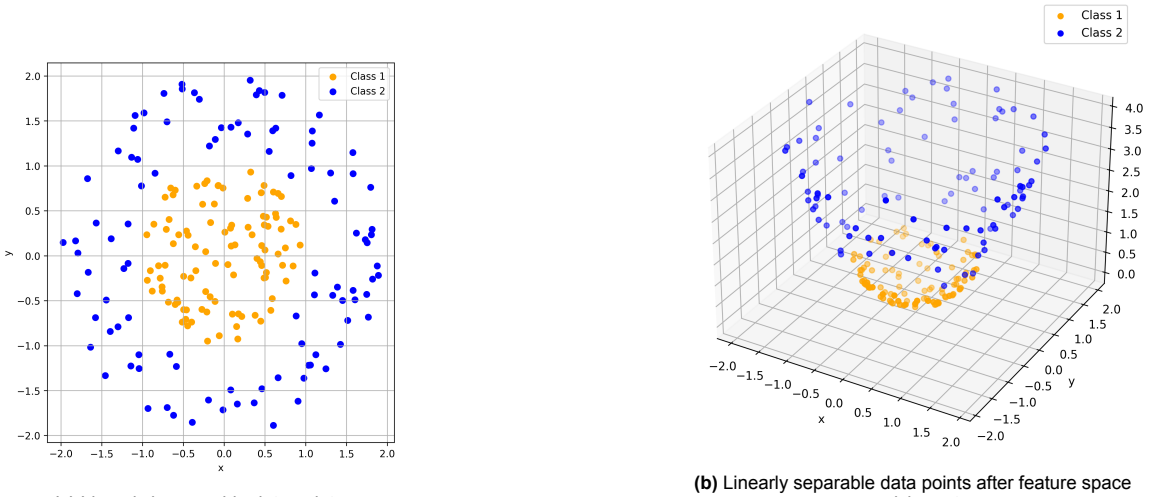
A powerful ability of SVM is its ability to enrich the feature space of the dataset in order to be able to separate and classify the samples more easily. The data points in figure 3.5a, for instance, are not linearly separable. By using a mapping $\phi : \mathbf{x} \mapsto \phi(\mathbf{x})$ [64], where ϕ is of higher order than $\mathbf{x}$, the dataset is enriched and it has become easy to fit a hyperplane between the different classes. In the case of figure 3.5b, the dataset is enriched by introducing a new dimension that is a non-linear combination of the two existing ones: $z_i = x_i^2 + y_i^2$ for each i.

Functions that map data to a higher-dimensional non-linear space are called kernel functions. Performing such calculations is time-intensive. Furthermore, selecting the appropriate kernel function for a specific problem is, although an educated guess can be made, an iterative process. Fortunately, the method of Lagrange multipliers can be used to solve the optimisation problem. This is known as the “kernel trick” [64] and essentially is a work-around to reduce computational cost associated with high-dimensional feature spaces. One of the most widely used kernel functions is called radial basis function (RBF), which measures the distance between data points and uses this information to map all points in higher-dimensional space, where they are easily separable by hyperplane. This allows the SVM to create complex and curved decision boundaries [75].

3.2.2. Support Vector Regression

Support vector regression (SVR) is the regression counterpart of SVM. Unlike SVM, the goal of SVR is not to find a hyperplane that serves as a decision boundary with a margin as high as possible. Instead, the goal is to find a hyperplane that best fits the high-dimensional relationship between the different features. This difference, as well as the slight difference in terminology, is shown in figure 3.6.

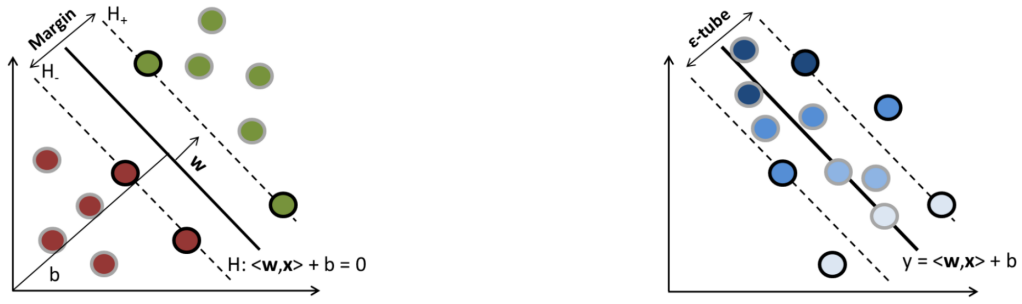
In figure 3.6 can be seen that what was previously called the margin, is now called the ε-tube. Samples that fall within this “tube” do not get penalised, even though they are not exactly aligned with the hy-



(a) Linearly inseparable data points

(b) Linearly separable data points after feature space enrichment

Figure 3.5: Transformation to a higher-dimensional space of a dataset, making it easily separable by a hyperplane.



(a) Support vector machine

(b) Support vector regression

Figure 3.6: Differences between SVM and SVR. Adapted from [76].

perplane. In this way the algorithm is forgiving to (small) errors in the dataset, and a low level of noise. Bishop [63] presents both an objective and loss function for support vector regression. The objective function is shown in equation 3.7 In this equation, the term $||\mathbf{w}||$ appears, although it has less of a physical interpretation as compared with SVM. Minimising the parameter $||\mathbf{w}||$ is equal to regularising the model, which means that model complexity is penalised. The loss function, which is essentially an updated version of the SVM Hinge loss function, is shown in equation 3.8 [63].

$$\min C \sum_{j=1}^{m} \ell(\mathbf{y}_j, \hat{\mathbf{y}}_j) + \frac{1}{2}||\mathbf{w}||^2 \tag{3.7}$$

where

$$\ell(\mathbf{y}_j, \hat{\mathbf{y}}_j) = E_\varepsilon(\mathbf{y}_j, \hat{\mathbf{y}}_j) = \begin{cases} 0, & \text{if } |\hat{\mathbf{y}}_\mathbf{j} - \mathbf{y}_j| < \varepsilon; \\ |\hat{\mathbf{y}}_\mathbf{j} - \mathbf{y}_j| - \varepsilon, & \text{otherwise} \end{cases} \tag{3.8}$$

3.3. Neural Networks

Artificial neural networks (ANN), or just neural networks (NN), are machine learning models inspired by the human brain [77]. Developed from the mid-20th century [78], they feature deeply interconnected "neurons" called units or nodes divided over multiple layers. A neural network structure is shown in figure 3.7 [79]. As information only flows from left to right in figure 3.7, this type of network is often called "feedforward neural network" (FNN).

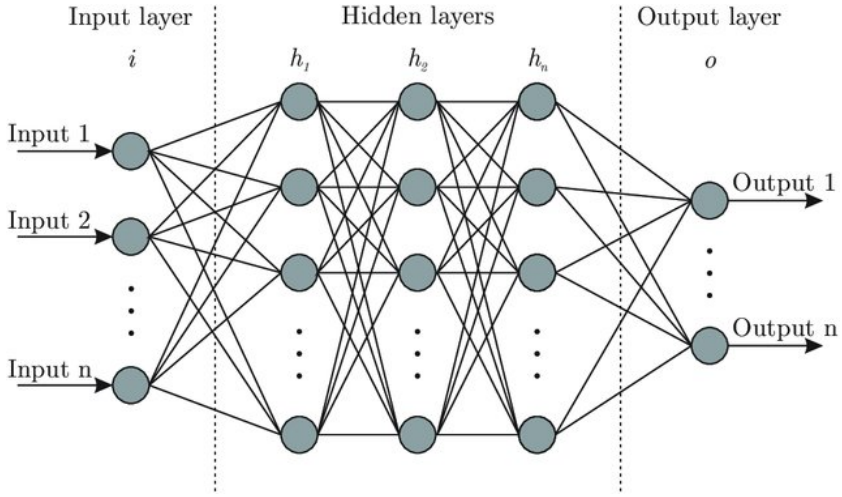


Figure 3.7: Structure of a neural network (NN). From [79].

This section discusses the theory relevant to this thesis regarding neural networks. Section 3.3.1 will explain the key concepts of neural networks. In section 3.3.2 ensemble models will be discussed. Although ensemble models are not exclusively used with neural network algorithms, in this thesis, the ensemble method's submodels will consist solely of smaller neural networks. Afterwards, section 3.3.3 will explain the concept of memory in neural networks by delving into recurrent neural networks (RNN) and long short-term memory (LSTM) networks. Lastly, section 3.3.4 will elaborate on finding the optimal set of hyperparameters in a process called hyperparameter tuning.

3.3.1. Working Principles of Neural Networks

As can be seen in figure 3.7, a neural network consists of a D-dimensional input layer, one or more hidden layers and an output layer. It is conventional to denote variables like y with a super- and subscript $y_{l,u}^{(n)}$, where l and u represent the hidden layer index and unit index within the layer, respectively. The superscript (n) is used when $\mathbf{y}$ is a vector, to denote the entry in the vector. Therefore, the network shown in figure 3.7 can be described using equations 3.9 and 3.10 (adapted from [64]).

$$\mathbf{y} = \boldsymbol{f}_{NN}(\mathbf{x}) = \boldsymbol{f}_o(\boldsymbol{f}_{h_n}(\boldsymbol{f}_{h_2}(\boldsymbol{f}_{h_1}(\mathbf{x})))) \tag{3.9}$$

$$\boldsymbol{f}_l(\mathbf{z}) \overset{def}{=} \boldsymbol{g}_l(\mathbf{W}_l\mathbf{z} + \mathbf{b}_l) \tag{3.10}$$

The bold-face functions in equations 3.9 and 3.10 are vector functions of the same size as the amount of units per layer. The function $\boldsymbol{g}$ is usually a non-linear activation function. In words, in every neuron (unit u) in a layer, a bias $b_{l,u}$ is added to the linear combination of all outputs from the previous layer multiplied by a weight $\mathbf{w}_{l,u}\mathbf{y}_{l-1}$. This result is then passed through the typically non-linear activation function g_l, which is usually, but not necessarily, the same for all units in a layer. In every layer, all weight vectors $\mathbf{w}_{l,1} \dots \mathbf{w}_{l,n}$ form the layer weight matrix $\mathbf{W}_l$ and all biases $b_{l,1} \dots b_{l,n}$ form the bias vector $\mathbf{b}_l$.

Figure 3.8 provides a closer look at this process. Unit $u_{1,1}$ is displayed. Because this is the first unit in the first hidden layer, it takes the output of the input layer $\mathbf{x}$ as inputs. After multiplying all features in $\mathbf{x}$ by their respective weights $w_{1,1}^{(n)}$, they are summed together with bias $b_{1,1}$ upon entering the unit. The summation is then passed through the activation function to obtain the numerical value of the first feature of the input to the next layer, $y_1^{(1)}$. It can already be seen that with an increasing amount of layers and nodes, the amount of parameters increases rapidly.

Activation Functions

The activation function, which processes the summation of the previous layer's outputs and the bias, is the component that introduces non-linearity to the neural network - although also linear activation functions exist. Apart from being linear or non-linear, important characteristics of activation functions include

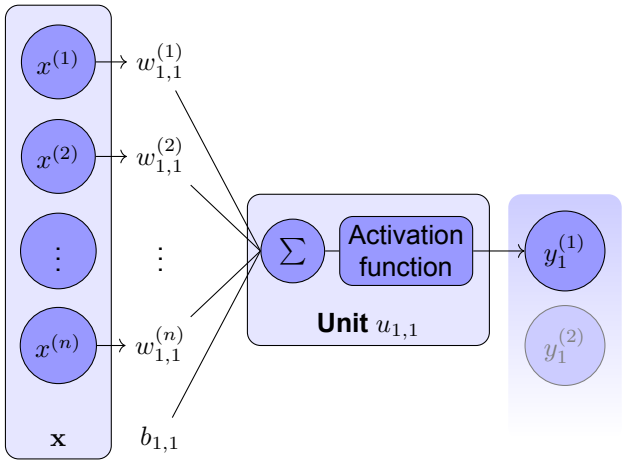
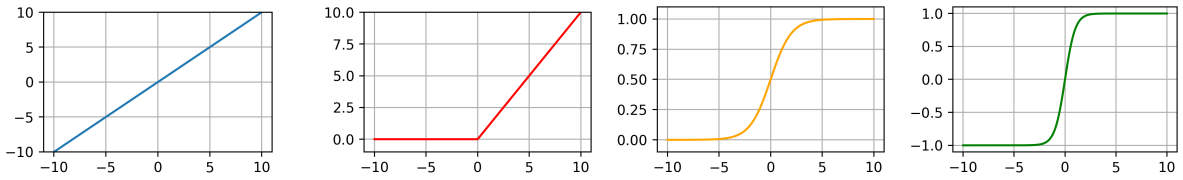


Figure 3.8: Unit $u_{(1,1)}$ at position 1 in the first hidden layer h_1 of a neural network.

their differentiability (continuous or discontinuous), their range, and whether they are saturating or non-saturating. Differentiability is required for model training, whereas being saturating or non-saturating indicates whether the limit of the activation function's derivative approaches zero as it tends to infinity. Some commonly used activation functions are listed below. Their graphs and equations are shown in figure 3.9 and equation 3.11:

- The **Linear** or identity activation function simply passes the input value unchanged. It is continuously differentiable, non-saturating and its range is $(-\infty, \infty)$.
- **ReLU** stands for rectified linear unit and is a ramp function. It outputs 0 for all $x \leq 0$ and x for all $x > 0$. It is not differentiable without manually setting the derivative at $x = 0$ to either 0 or 1, after which it becomes piecewise differentiable. ReLU has a range of $[0, \infty)$.
- The **Logistic** or sigmoid activation function is a continuously differentiable saturating function with a range of $(0, 1)$, making it suitable for binary classification.
- The **TanH** activation function is defined by the hyperbolic tangent function and behaves similarly to the logistic activation function, with the primary difference being its range: $(-1, 1)$.



(a) Linear **(b)** ReLU **(c)** Logistic **(d)** TanH

Figure 3.9: Graphs of the Linear, ReLU, Logistic and TanH activation functions.

$$\begin{array}{llll} \text{Linear:} & \text{ReLU:} & \text{Logistic:} & \text{TanH:} \\ g(x) = x & g(x) = \max(0, x) & g(x) = \frac{1}{1+e^{-x}} & g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \end{array} \tag{3.11}$$

Deep Learning

When a neural network has many hidden layers, it is referred to as a deep neural network (DNN), in contrast to a regular neural network, which is called shallow. There is no specific number of hidden layers that classifies a neural network as deep, but it generally requires at least two. The models trained in this thesis often have more than two hidden layers. However, due to the relatively shallow nature of the developed models compared to more advanced models, the term "deep" will not be used.

Model Parameter Tuning

Neural networks can easily become very complex, featuring a large number of weights and biases. Finding the optimal set of model parameters may seem like a challenging task, but in practice, it primarily requires proper book-keeping. A popular optimisation algorithm is called backpropagation and its steps are described below [80]:

1. Define a loss function: because this is a function of both the prediction and the targets, all weights and biases are incorporated in the loss function.
2. Initialise all weights and biases. Typically, small random values around 0 are used. The reason for using random values is that if all weights and biases were set to a fixed value, such as zero, there would be no differentiation between layers and nodes. As all routes through the neural network would then be identical, there would be a lack of asymmetry in the network. In that case, the algorithm would not be able to find the gradient of the loss function [81].
3. Compute the next prediction by running one sample from the training dataset through the neural network using the initialised weights and biases. Calculate the loss based on the prediction and the target. To increase training speed, the average loss of multiple samples is sometimes used to train the model. This so-called batch size is a hyperparameter of the neural network. This step is called the forward pass because information flows from input to output.
4. Using the chain rule, calculate the partial derivatives of the loss function with respect to all weights $\partial L/\partial w^{(k)}_{(i,j)}$ and biases $\partial L/\partial b_{(i,j)}$. The chain rule is used to compute all these partial derivatives, that together form the gradient, by systematically applying the derivatives through the layers of the model. This step is the reason why activation functions need to be differentiable. Because the starting point for calculating all derivatives is the output side of the network, this step is called the backward pass.
5. Update the weights $w^{(k)}_{(i,j)}$ and biases $b_{(i,j)}$ by taking a step with size α (the learning rate hyperparameter) in the direction of largest descent of the loss function, which is the negative gradient.
6. Repeat the from step 3. A complete cycle through all samples in the training set is called an epoch. Optimisation usually stops when the decrease in loss falls below a certain threshold or the maximum (pre-defined, hyperparameter) number of epochs has been reached.

It should be noted that there are several ways to calculate the updated weights and biases in step 5. An algorithm to perform this calculation is called an optimiser. In this thesis, the popular Adam optimiser was used, which adjusts the learning rates for each parameter and helps the model learn more efficiently [82].

3.3.2. Ensemble Methods

An ensemble method combines outputs from multiple base models, which may individually have low accuracy, using averaging, majority voting or a meta-model to achieve better performance than using a single model. Averaging means taking the average of outputs of all base models. Majority voting is often applied in classification and means that the class that the majority of base models predict will be selected. A meta-model is a top-layer machine learning model that is trained to combine the outputs from the base models. An ensemble method does not require either the meta-model or any of the base models to be a neural network. However, in this thesis, the only ensemble model used is one built exclusively with neural networks as both meta- and base models.

Three commonly used types of ensemble learning methods are bagging, boosting, and stacking. Bagging involves dividing the training data into multiple subsets, on which models are trained. The final result is obtained by averaging the predictions of the base models. Bagging is used to reduce the variance of model predictions. Boosting involves sequentially training multiple base models, where newer models are trained to correct the errors of the older models. It is used to reduce the bias of model predictions. Lastly, stacking involves training multiple base models, using their predictions as inputs for a top-level meta-model. Stacking aims to use the strengths of different models to reduce both the bias and variance of the predictions. Budu [83] summarises the key differences between bagging, boosting and stacking in table 3.1.

Table 3.1: Key differences between bagging, boosting and stacking. Adapted from [83].

	Bagging	Boosting	Stacking
Approach	Parallel training of weak models	Sequential training of weak models	Aggregate predictions of models in meta-model
Subset selection	Random sampling	Not required	Not required
Goal	Reduce variance	Reduce bias	Reduce variance and bias
Model combination	Majority voting or averaging	Weighted majority voting or averaging	ML model (meta-model)

3.3.3. Recurrent Neural Networks and Long Short-Term Memory

Until now, only feedforward neural networks have been discussed. Feedforward neural networks work well when the inputs have a constant shape and do not necessarily have a sequential relationship. Sometimes, however, a prediction does not only need to be a function of the current input, but also of previous inputs. A neural network architecture that can handle this kind of sequential input values is called a recurrent neural network (RNN). A more advanced solution is called long short-term memory (LSTM). Both architectures are discussed in this section.

Recurrent Neural Networks

A vanilla recurrent neural network is shown in figure 3.10 [84]. It can be seen that the structure in general is similar to a feedforward neural network, but the output from the activation function loops back into the weight and bias summation. For readability, the network in figure 3.10 has been "rolled out" to show two timesteps. Based on figure 3.10, the following statements can be made:

- The inputs x_i have a sequential (e.g. temporal) relationship. For example, in figure 3.10, x_1 could denote yesterday's temperature and x_2 today's temperature.
- Element i in the output vector $\mathbf{y}$ represents a prediction for timestep $i + 1$. Depending on the application, either the entire output vector or only its last element can be returned as the prediction. In figure 3.10, only the final prediction $y^{(2)}$, which represents tomorrow's temperature, is of interest. In this case, the prediction of today's temperature $y^{(1)}$ is of no interest because today's measurement is already available.
- All weights and biases (in this case w_1, w_2, w_3, b_1 and b_2) remain constant over all timesteps. This allows the RNN to handle inputs of variable size.

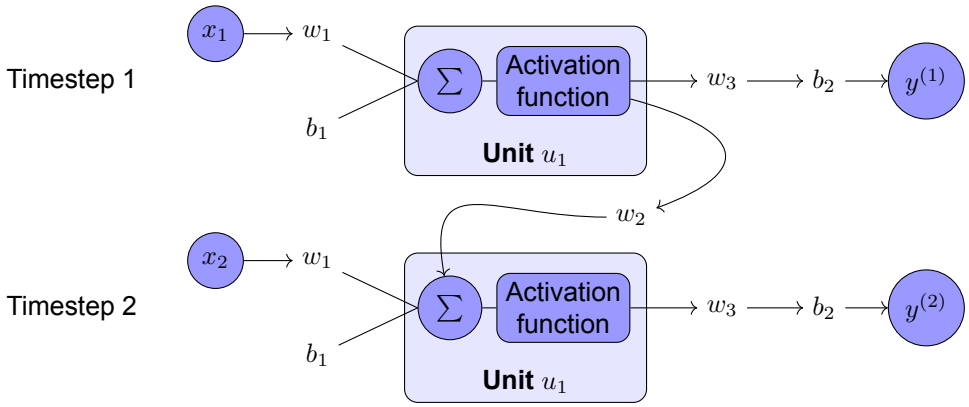


Figure 3.10: Recurrent neural network structure showing two time steps. Adapted from [84].

The fact that all weights and biases remain constant over all timesteps allows the RNN to handle variably-sized inputs, but it is also its greatest weakness. Since the output from unit u_1 is multiplied with weight w_2, which is constant over all timesteps, over time the first input gets multiplied by w_2^t, where t equals the amount of timesteps. This can lead to gradient issues:

- When w_2 < 1, the gradient of the loss function with respect to w_2 approaches zero over time. This leads to slow optimisation and is known as the vanishing gradient problem.
- When w_2 > 1, the opposite occurs. Due to the constant multiplication by a number greater than 1, the first timestep's input will become relatively more important than later inputs over time. This is known as the exploding gradient problem.

In order to prevent the vanishing/exploding gradient problem, new recurrent neural network algorithms have been developed, a popular one being long short-term memory.

Long Short-Term Memory

Long short-term memory (LSTM) is a popular recurrent neural network structure. The network features a so-called long-term memory called a "cell state" C_t, and a short-term memory called a "hidden state" h_t, where t denotes the timestep. It is designed in such a way that the cell state does not encounter the vanishing/exploding gradient problem. An overview of a long short-term memory unit is shown in figure 3.11 [85].

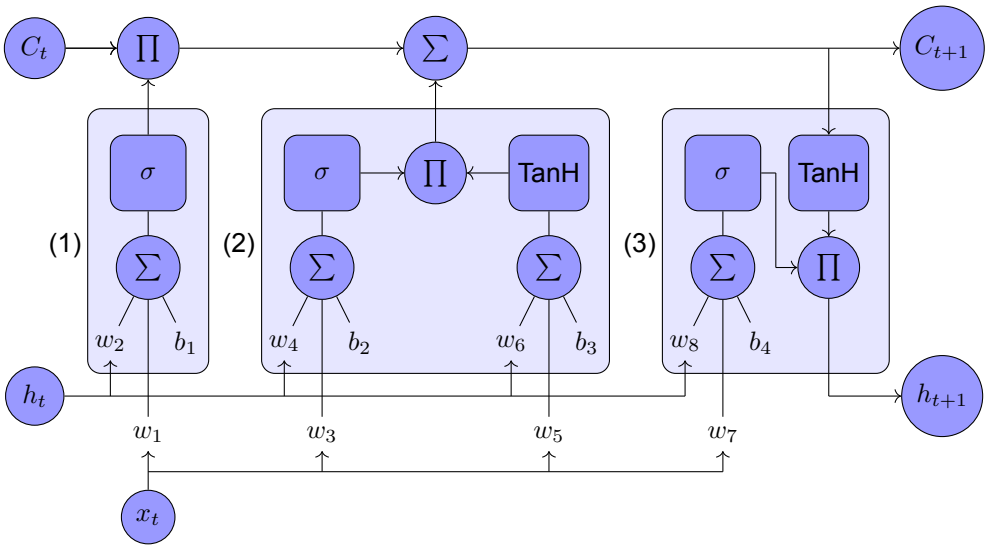


Figure 3.11: Long short-term memory unit. (1) = forget gate. (2) = input gate, where right side = potential long-term memory and left side = percentage of potential long-term memory to remember. (3) = output gate, where right side = potential short-term memory and left side = percentage of potential short-term memory to remember. Adapted from [85].

Starting at the left side of figure 3.11, the current values of the long-term memory C_t and short-term memory h_t can be seen. In combination with the input x_t, in region (1) the percentage of long-term memory to retain is determined. This is done by passing the short-term memory and input, multiplied by weights and incremented by a bias, through the sigmoid activation function, resulting in a value between 0 and 1. Because this is the step that allows the cell state to "forget" part of its content, region (1) is called the forget gate.

Region (2) is called the input gate, because in this region the input to the long-term memory is determined. At the right side of region (2) the potential long-term memory to add is determined using the weighted short-term memory and input, and a bias. The TanH activation function is used to obtain a value between -1 and 1. On the left, the actual added percentage of the potential long-term memory is determined, in a way similar to the forget gate.

Finally, in region (3), which is called the output gate, the new short-term memory is determined. For this, the long-term memory is passed through a TanH activation function and multiplied with a percentage that - again - follows from the short-term memory, input and a bias. The resulting new short-term memory h_{t+1} is also the model's output from timestep t. By adding multiple LSTM units, additional timesteps can be calculated.

3.3.4. Hyperparameter Tuning

Model parameters like weights and biases are determined in during model training. On top of this, the optimal set of hyperparameters corresponding to a model also needs to be determined. Although some hyperparameters can be determined through reasoning (a non-linear problem requires non-linear activation functions), most of the hyperparameters in a neural network are usually determined in a process called hyperparameter tuning. Among neural network hyperparameters that can be tuned are the amount of layers in the net, the amount of nodes per layer, the activation function per layer, the implementation of regularisation techniques, and the learning rate. It can be imagined that the hyperparameter space, consisting of all possible hyperparameter combinations, gets very large very easily. Therefore, multiple hyperparameter tuning algorithms exist. Some common algorithms are described in this section.

Grid Search

Grid search is the simplest of all hyperparameter tuning algorithms. It "exhaustively searches over a predefined set of hyperparameters" [86], making it a very slow algorithm when the hyperparameter space is very large. It makes no informed decisions on which parts of the grid to assess first. When time constraints are not a factor, grid search is a solid and thorough tuning algorithm.

Random Search

Random search is a hyperparameter tuning algorithm that randomly selects entries from the hyperparameter space and evaluates them. Although random search is still a relatively brute-force method, it can be more efficient than grid search, especially when the number of possible combinations to try is limited. Due to its deviation from the grid, random search allows for testing more values of the different variables with the same number of tried hyperparameter combinations.

Bayesian Search

Bayesian optimisation works by building a surrogate model using results from past attempts of hyperparameter combinations. It uses this surrogate model to select the next combination of hyperparameters that appears the most promising. In theory, the optimal set of hyperparameters should be found in less time compared to grid search and random search [65, 86].

Hyperband Search

Hyperband is a relatively new hyperparameter tuning algorithm introduced by Li et al. [87] in 2018. Instead of creating a surrogate model to increase tuning efficiency like Bayesian methods, Hyperband accelerates the random search algorithm by employing aggressive early stopping. By early stopping the trial of a hyperparameter combination when performance is already lacking after a few epochs, a much larger portion of the hyperparameter space can be explored in the same amount of time. The authors even claim that with Hyperband, an order-of-magnitude decrease in computation time compared to Bayesian tuning techniques is achieved [87].

3.3.5. Literature on Wake Field Prediction

During the literature review that was conducted as part of this thesis project [13], literature on wake field prediction using machine learning was collected and analysed. A summary of this literature is presented in this section.

Hwangbo and Shin published on the statistical prediction of wake fields in 2000 already [88], making it the first paper to delve into predicting wake fields using machine learning. They used an artificial neural network and their dataset consisted of design parameters and measured wake fields of 57 vessels, varying from container vessels to very large crude carriers. Their input consisted of a description of the hull by means of a grid, where every grid point referred to the intersection between a waterline and a station line. Every data point in the grid consisted of the angle between the centre line of the vessel and the line from the ship middle line plane at the propeller to the corresponding intersection. Hwangbo and Shin used a symmetric representation of the wake field and reached a correlation of 80% between the input and output data.

Another paper was written by Kim and Moon in 2006 [89]. This paper builds upon the work by Hwangbo and Shin: Kim and Moon use the same data set and the same way of representing the input. However,

instead of a regular neural network, they make use of a neuro-fuzzy system (NFS). This is a combination of fuzzy logic and a neural network. In fuzzy logic, an exact (“crisp”) value is fuzzified, which means that it is (potentially partially) assigned to a certain set with a value on the interval [0,1] [90]. A simple example: driving 80 km/h on a highway can be considered ’slow’ with a value of 1. Driving 120 km/h is considered ’fast’ with a value of 1. An intermediate value, say 100 km/h, is then considered 0.5 ’slow’ and 0.5 ’fast’. This is represented in figure 3.12. There can, of course, be infinitely many categories. After fuzzification, logic rules (IF-THEN, OR, AND, etc) can be applied. Combining this with a neural network, Kim and Moon achieved higher accuracy than Hwangbo and Shin. Although the authors do not provide a performance metric, it can visually be confirmed that the NFS outperforms the ordinary neural network described by Hwangbo and Shin.

In 2023, research on wake flow characteristic prediction using deep neural networks and transfer learning was published by Lee and Lee [91]. This research focuses on predicting viscous resistance and the wake distribution for container ships, with a specific type of ESD attached: the flow control fin (FCF). The input consists of six design variables specific to the FCF, as well as geometric information of the hull shape. Lee and Lee [91] assumed the circumferential distribution of the axial velocity to be symmetrical in the xz-plane. They used eight Fourier series to the tenth order to prepare the axial wake field for training. A similar approach was followed in this thesis, as will be discussed in section 4.3.2. The aim of their research was to use a method called transfer learning to extend a model trained on data from a 1000 TEU container ship to be able to predict results from 2500 and 3600 TEU container ships, with limited data.

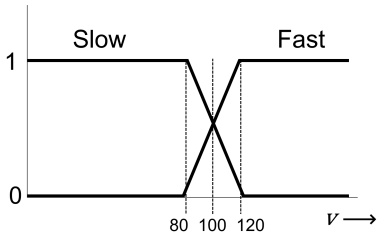


Figure 3.12: Fuzzification.

A common factor across all existing literature on wake field prediction is that the features to all described models consist of geometric representations of the ship hull. This thesis focuses on nominal wake field prediction using only basic ship parameters as features. No existing literature on nominal wake field prediction using these types of input features was found.

3.4. Conclusion

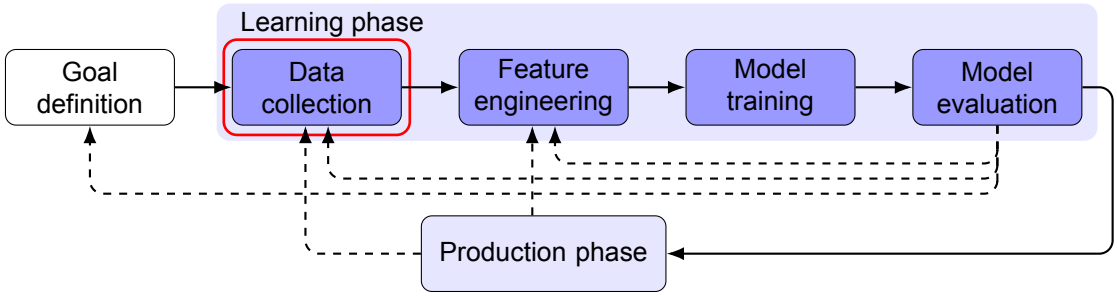
In this chapter, the main concepts of machine learning have been introduced. Firstly, the fundamentals of machine learning were discussed, followed by a more thorough introduction to both support vector machines and neural networks. Lastly, existing literature on wake field prediction was summarised, identifying that all existing literature makes use of geometric representatoins of ship hulls. Based on the information provided in this chapter, part of the second sub-question of this thesis can be answered:

Which machine learning algorithms and structures are most suitable for predicting the nominal wake field, and what are the steps that need to be taken to develop such models?

From this chapter, it has become clear that linear regression models are not suitable for predicting nominal wake fields, as the driving phenomena behind the wake field (discussed in chapter 2) are highly non-linear. Furthermore, the preferred machine learning architecture for this problem needs to have a sufficiently high level of complexity to learn the underlying patterns from the dataset. Section 6.2 later in this thesis will show that support vector regression lacks sufficient complexity. Lastly, the machine learning project life cycle has been introduced, and the following chapters will walk through this process step by step. Therefore, the remainder of this thesis will provide a full answer to the second sub-question.

4

Data Collection and Preparation



This chapter covers the “data collection” step in the machine learning project lifecycle as described by Burkov [65] in figure 3.1. An important first step after defining the goal of the machine learning project is to collect and prepare a dataset that suits the problem. This dataset can either be produced for the purpose, or an already existing dataset can be used.

In section 4.1, the collection of data will be elaborated on. This raw data needs to be filtered before it can be used; this process is described in section 4.2. Not only are the inputs to the machine learning project important, but the labels used in the supervised learning process are as well. They will be discussed in section 4.3. Finally, section 4.4 will provide concluding remarks to the data collection and preparation process, and present the characteristics of the filtered dataset.

4.1. The Dataset

As mentioned in the chapter introduction, the dataset used for a machine learning problem can either be purpose-built or an existing dataset can be taken. Advantages and disadvantages of producing a dataset or taking an existing one are summarised in table 4.1.

For this project, where there was not sufficient time to create a purpose-built dataset, an existing dataset was taken from Wärtsilä’s in-house propeller design software, Archimedes. Section 4.1.1 will elaborate on this software. Afterwards, in section 4.1.2, the used variables from the dataset will be discussed.

4.1.1. Wärtsilä Archimedes

Archimedes is Wärtsilä’s in-house propeller design software. It consists of various modules that can be used in several design stages. Among others, the available functionalities consist of cavitation analysis using Cavprop, Mpuf, PropCav, and Procal, blade thickness checks, ice class checks, corrosion fatigue analysis, hub stress and strength calculations, propeller benchmarking, and creating hydrodynamic documentation.

Table 4.1: Advantages and disadvantages of using an existing dataset or using a purpose-built one.

	Advantages	Disadvantages
Existing dataset	+ Inexpensive in terms of time and production cost + No additional complexity added to the workflow (e.g. the propeller design workflow)	- Data quality and informativeness may be lacking because the dataset was created for another purpose
Purpose-built dataset	+ Highly informative data can be generated + Dataset size can be tailored to the problem's needs	- Expensive in terms of time and production cost - Workflow might be disrupted due to the need to create the dataset

Not only does Wärtsilä Archimedes allow the propeller designer to make use of the above-mentioned tools, it also serves as a database with current and past propeller and thruster projects. It is this database that has been used as the dataset for this thesis project. This choice was not only made out of necessity: the ship parameters that are stored in the dataset are typically known early in the design process, before any complex calculations or CFD analyses have been performed. To be able to predict a nominal wake field based solely on this rudimentary dataset would therefore be of value for the propeller designer.

4.1.2. Dataset Variables

The dataset from Wärtsilä Archimedes consists of 1799 samples. Their corresponding labels will be elaborated on in section 4.3. The dataset variables that have been used, either to filter the dataset or for feature engineering (see chapter 5), are shown and explained in table 4.2. In this table, the "Used for" column denotes whether the variable has been used for filtering (F), feature engineering (FE), or both.

4.2. Data Filtering

Following the collection of the dataset, the next step is dataset filtering, which will be described in this section. Sections 4.2.1 to 4.2.5 will cover the removal of incomplete samples, duplicate samples, Product, OrderKey and WakeOrigin filtering, physical outlier filtering, statistical outlier filtering, and label filtering, respectively. This order corresponds with the order of steps taken in the project. After every step, the number of remaining samples will be presented.

4.2.1. Incomplete Sample Removal

The first step in the data filtering process is the removal of incomplete or invalid samples. A sample is considered incomplete if not all expected variables are present, and invalid if one or more variables are not of the correct datatype. For instance, the variable Cb that denotes the block coefficient C_b should be of datatype `float`.

All variables used for feature engineering (FE) in table 4.2 were checked for completeness, except for ShipType and WakeOrigin. These two "special" variables will be discussed later. Within the 1799-sample Wärtsilä Archimedes dataset, 165 invalid samples were identified, reducing the amount of samples to 1634.

With all invalid samples removed from the dataset, some raw dataset visualisations could be created. Visualisations and characteristics of the filtered dataset are discussed in section 4.4 and appendix B, while those of the raw 1634-sample dataset are presented in appendix A.

Table 4.2: Overview dataset variables used for filtering (F) and/or feature engineering (FE).

	Explanation	Used for
OrderKey	The internal order key used by Wärtsilä to identify projects. As order keys can start with different codes to denote specific series, they can be used for filtering.	F
Product	The Product variable is used for filtering. It denotes the type of product, which is either `Propeller`, `CIPS` (Coastal and Inland Propeller System, used to denote smaller FPPs of no more than 2.5 meters in diameter), or `Steerable Thruster`.	F
ShipType	Variable used to denote the vessel type.	FE
Lpp	The length between perpendiculars of the vessel in meters.	F, FE
Beam	The beam of the vessel in meters	F, FE
Cb	The block coefficient C_B at design draught of the vessel.	F, FE
Draught	The draught of the vessel in meters	F, FE
ShaftHeight	The vertical distance from the vessel bottom to the center of the shaftline in meters.	F, FE
NumberOfShaftLines	The number of shaft lines in the vessel. As this thesis project focuses solely on single-propeller vessels, the NumberOfShaftLines variable is used for filtering.	F
PropellerDiameter	The diameter of the propeller in millimeters.	F, FE
FSAH_ShipSpeed	The vessel speed in the Free Sailing Ahead condition, in knots.	F, FE
FSAH_RPM	The rotational speed of the propeller in the Free Sailing Ahead condition, in revolutions per minute.	F, FE
WakeOrigin	A variable that denotes the origin of the wake field information associated with the sample. Due to scaling effects, the wake field will look different on a model scale than on a full scale, so it is important to know this origin.	FE

4.2.2. Duplicate Removal

The data filtering step that follows on the removal of incomplete samples is removing duplicates. Duplicate samples not only influence dataset statistics, such as the mean and standard deviation of a certain variable, but they can also lead to the machine learning model overfitting. If a duplicate sample appears in both the training and validation sets, the model trained on an exact copy of that sample will "recognise" it in the validation set. This will lead to an overestimation of model performance.

For this project, the duplicate removal has been based on the subset of dataset variables consisting of Lpp, Beam, Cb, Draught, ShaftHeight, as well as the wake field information. If all of the variables in this subset had identical values to an other sample or other samples in the dataset, all duplicates were removed. From the 1634 samples in the dataset, 964 samples remained after duplicate removal.

4.2.3. Product, OrderKey and WakeOrigin Filtering

The next filtering step is based on the Product and OrderKey variables. As this thesis project focuses solely on symmetrical single-propeller vessels, all samples with Product category `Steerable Thruster` were removed from the dataset to make sure that only categories `Propeller` and `CIPS` remain.

Filtering on OrderKeys was added later in the project. Each OrderKey starts with a combination of letters. In consultation with Wärtsilä engineers, only projects starting with `PD`, `SNL`, and `SP` were selected. This was done because these are relatively new projects, and it is likely that their associated labels, the corresponding wake fields, are of higher quality. Therefore, this filtering method might help training the machine learning model on a qualitatively better dataset.

In another effort to increase dataset quality, filtering based on WakeOrigin was added. WakeOrigin denotes the origin of the labeled wake field data and can take any of the following values: `CfdCalculationFs`, `CfdCalculationMs`, `ModelTest`, `OtherProject`, `StandardDB`, or `Unknown`. The first three values are self-explanatory, denoting a full-scale CFD calculation, a model-scale CFD calculation, and a model-scale EFD test, respectively. `OtherProject` means that the wake field is directly imported from another project, such as a similar ship, and `StandardDB` indicates that the wake field information comes from a standard database. Since these two categories indicate that the wake field is only a rough estimate of the actual wake field, they are filtered out of the dataset. The `Unknown` category, which is the most frequent, has a special status that will be discussed in section 5.4.

Filtering on Product, OrderKey and WakeOrigin further reduced the amount of samples in the dataset, from 964 to 279 samples.

4.2.4. Physical Outlier Filtering

Filtering numerical variables based on physical constraints is the next step in the data preparation phase. All these numerical variables (Lpp, Beam, Cb, Draught, ShaftHeight, NumberOfShaftLines, PropellerDiameter, FSAH_ShipSpeed, and FSAH_RPM) were checked to ensure they have nonzero, non-negative values. All samples that did not meet this criterion were filtered out. Furthermore, if necessary, minimum and maximum values were set for these variables. The block coefficient variable (Cb) was constrained to the interval $[0.2, 0.9]$, and the NumberOfShaftLines was constrained to $[1, 1]$, effectively allowing only vessels with a single shaftline.

The physical outlier filtering step reduced the amount of samples in the dataset from 279 to 154.

4.2.5. Statistical Outlier Filtering

In the following filtering step, statistical outliers were detected and removed from the dataset. Z-score anomaly detection was selected to perform the statistical filtering step. The Z-score of a sample is the amount of standard deviations it is away from the sample mean. The Z-score can be calculated by subtracting the mean from each sample and dividing the result by the standard deviation. Usually, a Z-score higher than 3 indicates an anomaly [92]. Therefore, all samples with variables having a Z-score of 3 or higher were filtered out. This led to a further reduction of the dataset size to 148 samples.

4.2.6. Label Filtering

The final step in the filtering process is applying filters based on the labelled wake fields. Firstly, a check was performed on the completeness of the label information: if the label was missing, the sample was dropped. Thereafter, a symmetry check was performed. The idea behind introducing a so-called asymmetry coefficient, the calculation of which will be elaborated on in section 4.3.4, is that for a symmetric single-propeller vessel, a more or less symmetrical wake field is expected. If the Z-score of the asymmetry coefficient exceeded 3, indicating a high level of asymmetry, the sample was dropped. After the final filtering step, 142 samples remained out of the initial 1799 samples. An overview of the number of samples remaining after every filtering step is shown in figure 4.1.

4.3. Data Labels

Apart from the variables that will later be turned into features, the Wärtsilä Archimedes dataset also includes wake field information that will be used as labels. The variables that are of interest for this label engineering procedure are presented in table 4.3.

The variable types that are of interest consist of two integer variables and five arrays. nWakeRadii and nWakeAngles denote the number of radii and angles at which nondimensionalised velocities in the axial, tangential, and radial directions are given, respectively. WakeRadii and WakeAngles contain these radii and angles. WakeRadii is zero padded to a length of 13 and WakeAngles to a length of 100. Vx, Vt, and Vr are all arrays of size 1300. The first 13*nWakeAngles entries in these arrays represent the nondimensionalised velocities. For every angle in WakeAngles, the nondimensionalised velocities at every radius in WakeRadii are given, followed by zero padding until a length of 13 is reached. After iterating through every radius for every angle, the array is zero padded to a length of 1300.

The arrays Vx, Vt and Vr that describe the wake field in three directions are not usable as labels in their

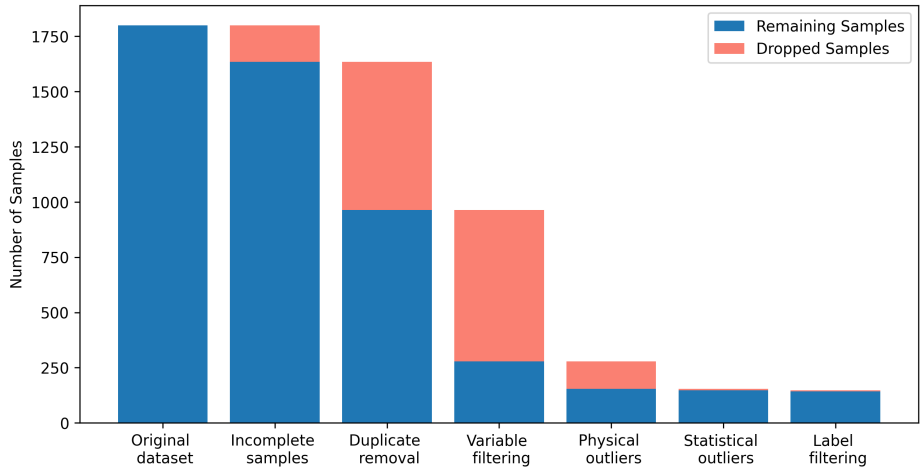


Figure 4.1: Samples from the Wärtsilä Archimedes dataset remaining after every filtering step.

Table 4.3: Overview of variables from the Wärtsilä Archimedes dataset used for label engineering.

	Explanation	**Datatype**
nWakeRadii	The amount of radii at which wake velocities are given	int
nWakeAngles	The amount of angles at which wake velocities are given	int
WakeRadii	The radii at which wake velocities are given. Normalised to the interval $[0, 1]$.	array of floats of size 13 with the last (13-nWakeRadii) entries being zero
WakeAngles	The angles at which wake velocities are given.	array of floats of size 100 with the last (100-nWakeAngles) entries being zero
Vx, **Vt** and **Vr**	Wake velocity in axial, tangential and radial direction, respectively. Nondimensionalised with ship speed.	array of floats of size 1300

original form, as they do not contain information on which location the nondimensionalised velocities refer to. Furthermore, not all wake fields are given at the same interval of radii or angles.

To be able to train and validate the machine learning model, all labels should present the nondimensionalised velocities in the wake field at the same radii and at the same angles. Section 4.3.1 delves into the process of interpolating to obtain these velocities. In section 4.3.2, the use of the discrete cosine transform to reduce output dimensionality is elaborated on. Section 4.3.3 contains an explanation on how the final sample labels have been built up. In section 4.3.4 the calculations for the asymmetry coefficient, that is used to filter the raw dataset, are shown. Lastly, in section 4.3.5 the MARIN standard for loading and saving wake field files is touched upon. It should be noted that although the procedure is the same for the velocities in the axial, tangential, and radial directions, only the calculations for the axial direction are presented because the final models have only been trained on axial data.

4.3.1. Interpolation of Radii and Angles

In order to present all wake field velocities at the same radii and angles, they need to be interpolated if not already given at the desired radii and angles. First, the target points had to be determined. For the angles, symmetry in the wake field was assumed. The justification for this assumption is that this project focuses solely on single-propeller symmetrical vessels, so symmetrical nominal wake fields are expected. The target angles were determined to be the 19 values between 0 and 180 degrees, in intervals of 10 degrees, so $\theta = [0, 10, \ldots, 170, 180]$.

In order to determine the target radii, a visual inspection of the dataset was performed. It was noticed that almost all labels provide wake field information from a radius of approximately $r = 0.4R$ onward. To prevent large extrapolation errors and to emphasize the important outer region of the propeller, the set of target radii $r/R = \{0.4, 0.6, 0.8, 0.9, 1.0\}$ was selected.

To prepare the raw labels for interpolation and extrapolation, the 1300-entry Vx array for each sample has been stripped of all padded zeroes and converted into a list of lists representing a matrix of dimensions (nWakeAngles, nWakeRadii), where each row represents a single radius and each column represents a single angle. Note that nWakeAngles and nWakeRadii can vary from sample to sample. By iterating over the columns of the Vx matrix, all nondimensionalised velocities are interpolated or extrapolated to the target radii using linear interpolation. Linear interpolation was chosen because it allows for easy extrapolation in cases where the lowest radius at which the wake field is given lies above $r = 0.4R$ for a certain sample.

After interpolation to the correct wake radii, the Vx matrix is transposed to a matrix of dimension (5, nWakeAngles). Using the WakeAngles array to match each column of Vx with its corresponding wake field angle, the smallest subset of Vx where the interval [0,180] still fits is preserved. The nondimensionalised velocities at the right angles can now be obtained. While iterating over the five target radii $r/R = \{0.4, 0.6, 0.8, 0.9, 1.0\}$, the wake angle values $\theta = [0, 10, \ldots, 170, 180]$ are calculated using cubic spline interpolation. This process can be seen in figure 4.2.

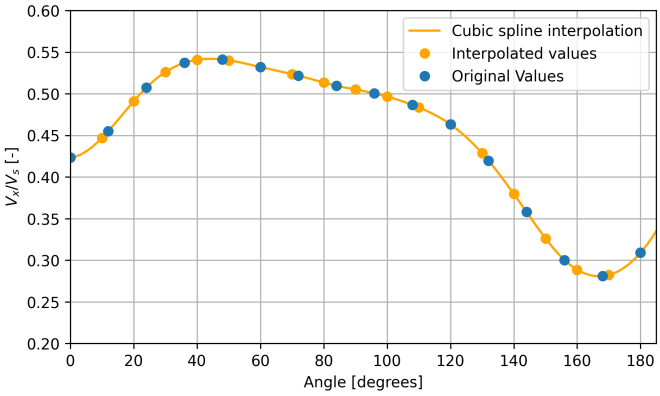


Figure 4.2: Cubic spline interpolation of the nondimensionalised velocities of a wake field for a single radius in the interval $[0, 180]$.

After iterating over all samples, all wake field velocities are given at the same five radii and 19 angles. This results in an output space dimensionality of 95, which is relatively high compared to the input dimensionality of approximately 30. One way to address this is to reduce the number of radii and angles at which the nondimensionalised velocities are labelled. However, dropping too many radii or angles leads to loss of information. Another option is to represent all wake field graphs in a way that requires fewer data points than directly logging the velocities, such as using a discrete cosine transform.

4.3.2. Discrete Cosine Transform

The discrete cosine transform (DCT) is a Fourier-like transformation first described in 1974 by Natarajan, Rao, and Ahmed in [93]. The DCT can transform any evenly-sampled signal into a sum of cosine functions with different frequencies. It is therefore a transformation from the time domain (or, in this case, the spatial domain) to the frequency domain. A property of the discrete cosine transform that makes it suitable for compression and dimensionality reduction is the fact that most energy (information) of the signal is packed in the first few coefficients [94]. Because of this, DCT is widely used in data compression, especially in digital media compression like images, audio and video. The application of a frequency domain transform like DCT to prepare wake field information for use in a machine learning model has been demonstrated before by Lee and Lee [91], who used Fourier series to the tenth harmonic in their work, as can be seen in figure 4.3.

There are multiple types of DCT. Usually, DCT Type II is referred to as “the” discrete cosine transform. DCT Type III is the inverse of DCT Type II and can be used to perform a back-transformation from the frequency domain to the time domain. The equations for DCT Types II and III are given in equations 4.1 and 4.2, respectively [95].

$$y_k = 2 \sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi k(2n+1)}{2N}\right) \quad \text{for } k = 0, \ldots, N-1 \tag{4.1}$$

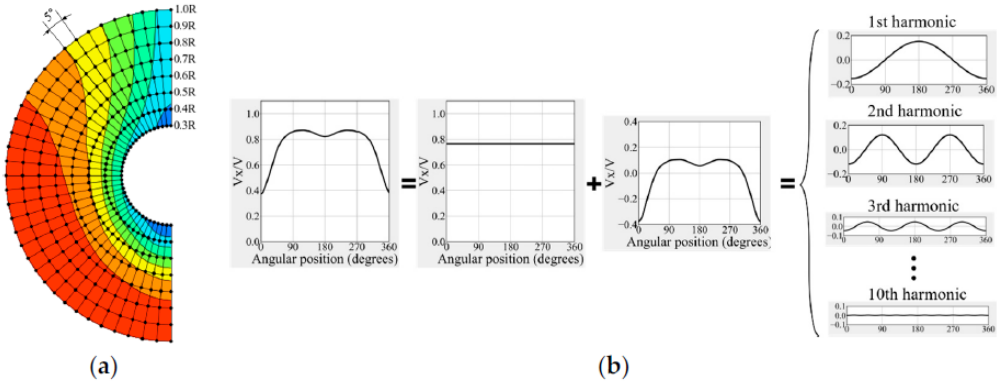
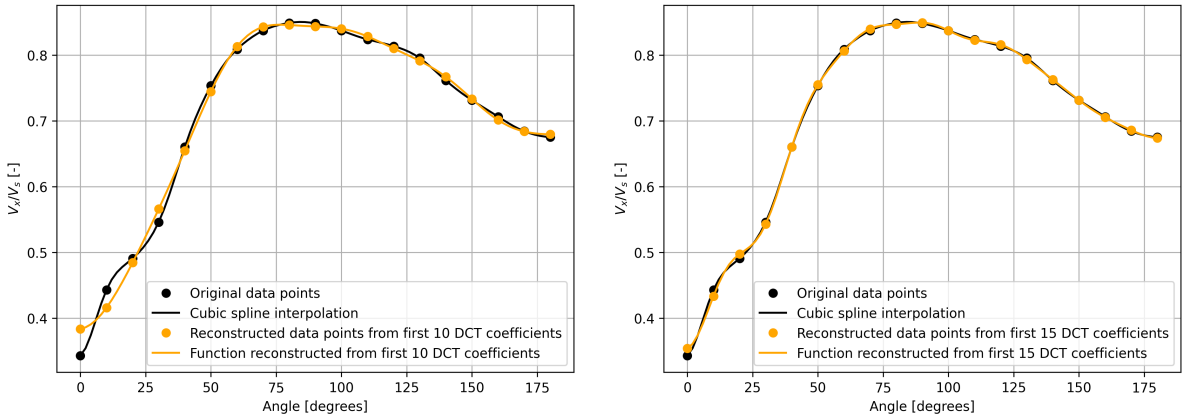


Figure 4.3: Pre-processing of wake field data for neural network: (a) bisecting the propeller plane; (b) harmonic analysis. From [91].

$$y_k = x_0 + 2\sum_{n=1}^{N-1} x_n \cos\left(\frac{\pi(2k+1)n}{2N}\right) \quad \text{for } k = 0, \ldots, N-1 \tag{4.2}$$

In equation 4.1, y_k denotes the DCT coefficients and x_n denotes the equally-spaced samples from the original signal. In equation 4.2, which represents the DCT type III equation (the inverse of the type II DCT), x_n denotes the DCT coefficients and y_k denotes the reconstructed samples.

The power of the discrete cosine transform can be seen in figure 4.4. After sampling the wake field graph at 91 equally spaced intervals, the first 91 DCT coefficients were obtained. It can be seen that with just the first 15 (figure 4.4b) or even 10 (figure 4.4a) coefficients, the graph can be recreated with reasonable accuracy. Appendix C delves deeper into representing a wake field graph using its first n DCT coefficients and shows how most of the energy is indeed stored in the first few coefficients by presenting a graph of the number of DCT coefficients taken into account versus the mean squared error of the reconstructed curve.



(a) Graph reconstructed from first 10 DCT coefficients

(b) Graph reconstructed from first 15 DCT coefficients

Figure 4.4: A wake field velocity graph reconstructed using (a) its first 10 DCT coefficients and (b) its first 15 DCT coefficients.

Considering that all preprocessed wake field velocities are sampled at five different radii for 19 different angles, it would not make sense to take the first 19 DCT coefficients or more to represent one wake field graph. The number of DCT coefficients to take into account has therefore become a trade-off between accurately representing the original wake field graph and the output space dimensionality. The number of DCT coefficients to take has become a hyperparameter of the machine learning model.

4.3.3. Label Engineering

Once the raw labels from the Wärtsilä Archimedes dataset were sampled at fixed radii and angles, and the option to perform a discrete cosine transform on individual wake field graphs at certain radii had been implemented, the next step was to create the dataset labels in a uniform manner. There were two options: label engineering with and without using DCT.

When the discrete cosine transform is not chosen to be used, the process of creating the labels is fairly simple. Using Python, the interpolation to the correct radii and angles as described in Section 4.3.1 is performed. This leads to an output space dimensionality of $j \cdot k$, where j equals the number of radii and k the number of angles to be considered. The standard values of j and k are 5 and 19, respectively, leading to a dimensionality of 95. The resulting 3D matrix has a shape of (i, j, k), where i equals the total number of samples.

The other option is to represent every wake field graph with the first n coefficients from its discrete cosine transform. When this option is chosen, instead of sampling k angles from the cubic spline interpolation described in section 4.3.1, 91 angles are sampled and the DCT process as described in section 4.3.2 is followed. This results in a 3D matrix of shape (i, j, n) where i and j again represent the amount of samples and the amount of radii, and n denotes the amount of DCT coefficients that have been taken into account.

4.3.4. Asymmetry Coefficient

To be able to filter the dataset based on the quality of the labels, a “asymmetry coefficient” was defined. Since only single-propeller symmetric vessels are considered in this research, symmetrical nominal wake fields are expected. Highly asymmetric wake fields can therefore indicate flawed data, such as wake field information from other types of thrusters mislabelled as propeller plane data or measurement errors during EFD experiments.

The input to the asymmetry coefficient calculation for each sample is the Vx array, which has been stripped of all padded zeroes and converted into a list of lists representing a matrix with dimensions (nWakeAngles, nWakeRadii). For each radius, the list containing all velocities in the interval $\theta = [0, 360]$ was split into a “right hand plane” list containing the values of θ between 0 and 180 degrees and another list (“left hand plane”) containing the values between 180 and 360 degrees. The latter list was then reversed. Since not all labelled wake fields presented velocities with a constant angular interval, both lists were cubic spline interpolated to an interval of one degree. The mean of the absolute differences between all left hand and right hand planes was then taken to be the sample’s asymmetry coefficient. Filtering was based on the Z-score of each sample’s asymmetry coefficient: if the Z-score exceeded 3, the wake field was deemed an anomaly and the sample was removed from the dataset.

4.3.5. MARIN Wake Format

Although all inputs to the machine learning model, including both features and labels, are directly sourced from the Wärtsilä Archimedes database, there should be an easy method to import predicted wake fields back into Archimedes for comparison. Wake fields can be imported into Archimedes via its graphical user interface (GUI). The MARIN wake format is used within this GUI to import and export wake fields. This wake format uses text files with a `.wak` extension and uniformly displays wake field velocities in the axial, tangential, and radial directions, respectively. The first three lines of the file indicate that it is a MARIN wake file, including the ID (OrderNo), the ship type, and its main parameters L, B, T, and C_B. Thereafter, the number of radii and angles at which the velocities are provided is presented, followed by three tables containing the Vx, Vt, and Vr information, respectively. An example is shown in figure 4.5. In this example, angles between 40 and 320 degrees have been omitted for brevity.

Using Python, a function was written that takes all required variables (order number, ship type, length, breadth, draught, block coefficient, wake field angles, wake field radii, and nondimensionalised velocities in axial, tangential, and radial direction) as inputs, and outputs a wake field in the MARIN format for further analysis in Wärtsilä Archimedes.

```
    Wake export in MARIN format
OrderNo (As engineered)Wakefield
ShipType, L = 165, B = 26.9, T = 7.4, Cb = 0.6421

    8   19
          1130  1412.5    1695  1977.5    2260  2542.5 2683.75  3107.5

     0   0.347   0.473   0.578   0.654   0.771   0.822   0.831   0.876
    20   0.483   0.667     0.8   0.856   0.884    0.87   0.854   0.816
    40   0.424   0.645   0.797    0.86   0.892   0.875   0.857   0.812

                                  ...

   320   0.424   0.645   0.797    0.86   0.892   0.875   0.857   0.812
   340   0.483   0.667     0.8   0.856   0.884    0.87   0.854   0.816
   360   0.347   0.473   0.578   0.654   0.771   0.822   0.831   0.876

     0       0       0       0       0       0       0       0       0
    20  -0.001   -0.03  -0.004   0.033  -0.012   0.076   0.153   0.348
    40   0.037   0.018   0.037   0.072   0.037   0.123   0.196   0.382

                                  ...

   320  -0.037  -0.018  -0.037  -0.072  -0.037  -0.123  -0.196  -0.382
   340   0.001    0.03   0.004  -0.033   0.012  -0.076  -0.153  -0.348
   360       0       0       0       0       0       0       0       0

     0  -0.091  -0.078  -0.072  -0.073  -0.081   -0.16  -0.216  -0.366
    20  -0.105  -0.117  -0.124  -0.122  -0.096  -0.139  -0.178  -0.275
    40  -0.085  -0.119  -0.126  -0.106  -0.088  -0.127  -0.159  -0.241

                                  ...

   320  -0.085  -0.119  -0.126  -0.106  -0.088  -0.127  -0.159  -0.241
   340  -0.105  -0.117  -0.124  -0.122  -0.096  -0.139  -0.178  -0.275
   360  -0.091  -0.078  -0.072  -0.073  -0.081   -0.16  -0.216  -0.366
```

Figure 4.5: The structure of a MARIN wake file. Angles between 40 and 320 degrees have been omitted.

4.4. Filtered Dataset Characteristics and Conclusion

This chapter has described the “data collection” step in the machine learning project life cycle. By filtering on multiple criteria, the dataset size was reduced from 1799 to 142 samples. Descriptive statistics of all variables that will be used for feature engineering, except for WakeOrigin, are shown in table 4.4. The WakeOrigin variable will be discussed in section 5.4. Table 4.4 shows for each variable its mean, standard deviation, minimum and maximum value, and the values under which 25, 50, and 75 percent of the data falls, respectively. The box plots of all variables are given in figure 4.6.

It can be seen that all variables not only have values that mostly fall within a reasonable range, and their extreme values are not considered extreme outliers. Appendix B presents several other visualisations based on the filtered dataset, including a distribution plot, density plot, and linear correlation heatmap. After a statistical and visual inspection of the filtered dataset, it can be concluded that the filtering procedure was successfully applied. The next step is to perform feature engineering, which will be described in the following chapter.

Table 4.4: Descriptive statistics of the filtered Wärtsilä Archimedes dataset.

	mean	std	min	25%	50%	75%	max
Lpp	155.4	59.6	54.4	115.9	148.8	187.3	349.5
Beam	26.1	9.4	12.4	19.2	23.4	32.3	63.0
Draught	8.4	2.4	5.2	6.7	7.9	9.2	16.1
ShaftHeight	3.0	0.8	1.4	2.3	2.9	3.4	4.9
Cb	0.7	0.1	0.4	0.6	0.7	0.8	0.9
PropellerDiameter	5631	1564	2800	4500	5400	6500	9600
FSAH_ShipSpeed	15.7	3.3	7.4	13.8	16.0	17.9	23.5
FSAH_rpm	119.2	31.1	68.2	96.7	114.0	137.6	229.4

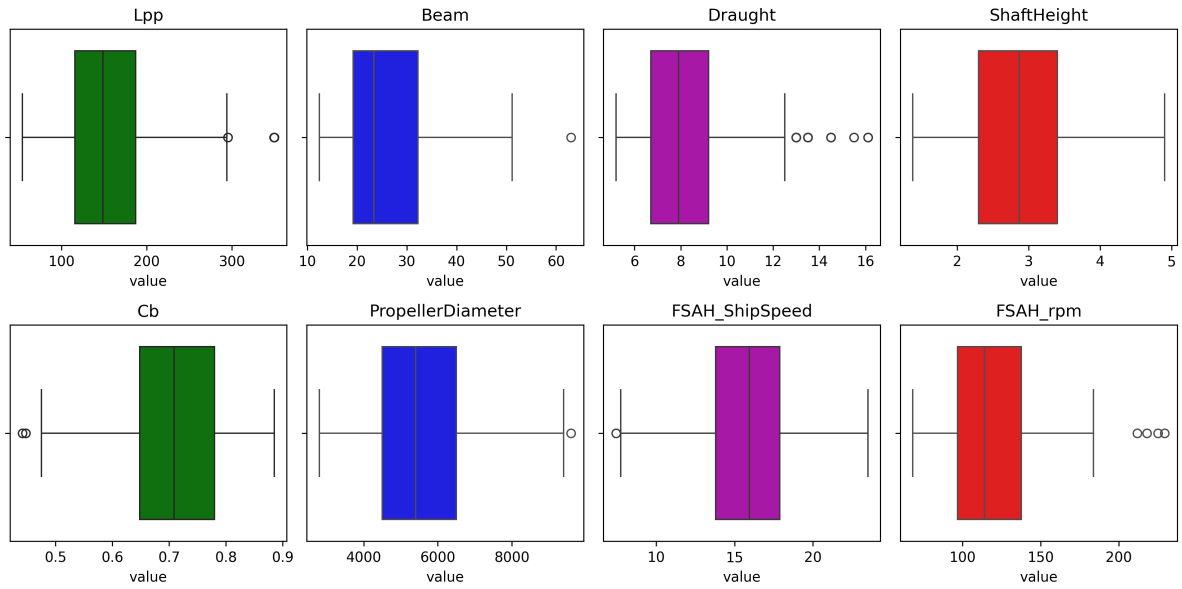
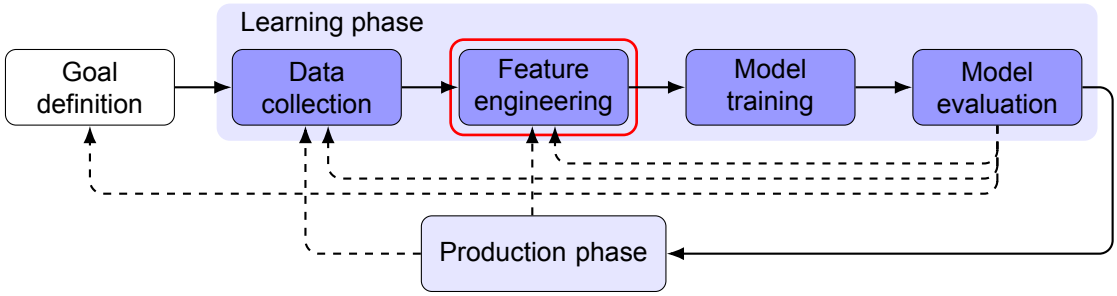


Figure 4.6: Box plots of all filtered variables in the Wärtsilä Archimedes dataset.

5

Feature Engineering



After collecting and preparing the dataset, the next step in the machine learning project life cycle [65] is feature engineering. This involves selecting, combining, and manipulating the variables in the dataset to use them as inputs to the machine learning problem. This step is crucial because the features allow the model to learn. Therefore, the features should be as informative as possible, ideally uniquely describing every sample in the dataset.

In this chapter, the process of feature engineering is described and the features that are used in this thesis are elaborated on. The engineered features are explained based on their category. Section 5.1 delves into variables from the raw dataset that are either directly used as features or combined with another variable and then turned into a feature. In section 5.2, the process of feature engineering using domain knowledge is elaborated on. Afterward, the one-hot encoding algorithm is explained in section 5.3. Feature engineering by data imputation is touched upon in section 5.4. Section 5.5 explains the importance of normalisation or standardisation. These processes are generally recommended after engineering the features. Finally, in section 5.6 the characteristics of the feature space are shown and discussed.

5.1. Uncorrelated Features from Dataset Variables

Some features directly originate from one of the variables in the raw dataset. They can either be directly turned into a feature or combined with another variable. The latter is often done when two variables are highly correlated. Strong correlations between features should be avoided. At best, correlated features only introduce redundancy and computational cost to the model, and at worst, they cause overfitting.

Linear correlation can be easily detected using a correlation matrix, such as the one discussed in section 4.4. A correlation coefficient with an absolute value close to one indicates strong linear correlation, whereas correlation coefficients with near-zero absolute values indicate no linear correlation. In contrast to linear correlations, higher-order correlations are harder to grasp in a single matrix. Instead, a scatter plot or heatmap can be created for every possible combination of potential features. Higher-order correlations can then be easily detected visually.

There are multiple options to avoid feature correlation. Firstly, one of the correlated features can simply be removed. In the case of perfect correlation, this would reduce the complexity of the model while not removing any information from it. Additionally, the values of two features can be averaged to obtain a single feature. Another option is to divide one feature by the other. If the features describe variables of equal unit, this directly nondimensionalises the feature.

Variables that have directly been turned into features are **Cb**, **PropellerDiameter**, **FSAH_ShipSpeed** and **FSAH_rpm** as they did not have a strong correlation with each other. Features created by combining variables are:

- **L/B**, or the length-to-beam ratio, is introduced. By taking the ratio between the Lpp and Beam variables from the dataset, this commonly used ratio is utilised as a feature instead of two highly correlated ones.
- **B/T**, or the beam-to-draught ratio, is created by taking the ratio of the Beam and Draught variables for the same reasons as the L/B feature.
- **ShaftHeight/T**, or the shaft height-to-draught ratio, is created in a similar manner to L/B and B/T.

5.2. Feature Engineering using Domain Knowledge

Non-linearly combining variables or existing features into new features adds information to the feature space of a machine learning problem. Domain knowledge is essential to combine variables or existing features in such a way that the newly engineered features represent quantities relevant to the problem.

In this section, the features engineered using domain knowledge for this project are discussed. Two categories of domain knowledge-informed features can be distinguished: features that represent physics-based relations, referred to as derived features, are discussed in section 5.2.1, and those that are derived (semi-)empirically are covered in section 5.2.2.

5.2.1. Derived Features

The derived features used in this project are as follows:

- **Fr** is a feature that combines FSAH_ShipSpeed (converted from knots to meters per second) and Lpp, along with the gravitational constant g = 9.81 m/s^2, to obtain the dimensionless Froude number shown in equation 5.1. As the Froude number represents the ratio of inertial to gravitational forces, it intuitively adds relevant information to the feature space.

$$Fr = U/\sqrt{gL} \tag{5.1}$$

- **Re** is a feature that combines FSAH_ShipSpeed and Lpp with the kinematic viscosity of seawater ν = 1.19·10^{-6} m^2/s, resulting in the Reynolds number (equation 5.2). Representing the ratio between inertial and viscous forces, the Reynolds number is also of interest in nominal wake field predictions.

$$Re = (UL)/\nu \tag{5.2}$$

- **Displacement**. Lpp, Beam, Draught, and Cb are multiplied with a shell appendage factor s of 0.005 to find the displacement volume ∇ in cubic meters as shown in equation 5.3.

$$\nabla = LBTC_B(1+s) \tag{5.3}$$

- **CNabla** represents the displacement volume to length ratio C_∇ as provided by Lamb [96]. This ratio denotes the “fatness” of the vessel: higher values indicate fuller vessels where lower values are an indication for more slender vessels. The dimensionless C_∇ takes displacement volume and vessel length as inputs and is shown in equation 5.4.

$$C_\nabla = \frac{\nabla}{L^3} \tag{5.4}$$

5.2.2. (Semi-)Empirical Features

The (semi-)empirical features used in this project are as follows:

- **Delta_Bertram** is an equation for displacement mass in tonnes, derived from a semi-empirical ship length estimation equation provided by Bertram [97]. All variables in this equation are present in the dataset, or can be directly calculated. By re-writing, equation 5.5 was obtained. In this equation, L is in meters, U in meters per second, and C_B and Fr are dimensionless. It was found that there is some correlation with the displacement volume feature discussed in Section 5.2.1. However, due to the semi-empirical nature of Bertram's equation, the correlation is not strong, and thus this feature was retained.

$$\Delta_{\text{Bertram}} = \left(\frac{L_{pp}}{3.2U^{0.3} \frac{C_B + 0.5}{(0.145/Fr) + 0.5}} \right)^{1/0.3} \tag{5.5}$$

- **CM_HSVA** is the empirical midship coefficient as proposed by the Hamburg Ship Model Basin HSVA. The midship coefficient C_M is defined as the area of the underwater cross-section amidships divided by the product of beam and draught. Presented by Lamb [96] as being the best empirical equation out of three and being the best practice estimation in Germany, the HSVA approximation of the midship coefficient is only dependent on the block coefficient and is shown in equation 5.6.

$$C_{M,\text{HSVA}} = (1 + (1 - C_B)^{3.5})^{-1} \tag{5.6}$$

- **KB_Schneekluth** is Schneekluths approximation of centre of buoyancy (COB) height KB, presented in Lamb [96]. It is dependent on midship coefficient C_M, block coefficient C_B and vessel draught T, and its equation is shown in 5.7.

$$KB_{\text{Schneekluth}} = (0.90 - 0.30C_M - 0.10C_B)T \tag{5.7}$$

- **CF_ITTC** denotes the dimensionless frictional resistance coefficient C_F as determined by the International Towing Tank Conference (ITTC) in 1957, described in [98]. The frictional resistance coefficient plays an important role in the frictional resistance equation $R_F = 0.5\rho C_F S U^2$, where R_F is in Newtons, S in square meters, and U in meters per second. The empirical ITTC coefficient is a function of the Reynolds number Re and can be seen in equation 5.8.

$$C_{F,\text{ITTC}} = 0.075/(\log_{10} Re - 2)^2 \tag{5.8}$$

- **PD_Volker** is an estimation for delivered power P_D in kW by the vessel engine by Völker from [98]. This estimation, shown in equation 5.9, combines displacement mass Δ in tonnes with vessel speed U.

$$P_{D,\text{Volker}} = \Delta^{0.567} U^{3.6} \cdot 10^{-3} \tag{5.9}$$

- **W_Taylor** is the empirical wake fraction (w_T) equation by Taylor, as published in [99]. Potentially being a very strong feature, multiple empirical wake fraction equations have been implemented, including the British Ship Research Association (BSRA) and the Harvald estimations, apart from the Taylor estimation. The Taylor estimate was chosen after comparison between the outcomes of the semi-empirical equations and the calculated wake fractions from the labels. The Taylor wake fraction estimation turned out to be the best match and the other wake fraction estimations have been left out of the feature space to prevent strong correlations. The Taylor, BSRA and Harvald wake fraction estimations are shown in equations 5.10 to 5.13, respectively.

$$w_{T,\text{Taylor}} = 0.50C_B - 0.05 \tag{5.10}$$

$$w_{T,\text{BSRA}} = -0.0458 + 0.03745C_B^2 + 0.1590D_W - 0.8635Fr + 14773Fr^2 \tag{5.11}$$

where

$$D_W = \frac{B}{\nabla^{1/3}} \sqrt{\frac{\nabla^{1/3}}{D_{prop}/1000}} \tag{5.12}$$

$$w_{T,\text{Harvald}} = \left(1.095 - 3.4C_B + 3.3C_B^2\right) + \left(0.5C_B^2 \left(6.5 - \frac{L_{pp}}{B}\right)\right) / \frac{L_{pp}}{B} \tag{5.13}$$

5.3. Categorical Features

All the features discussed so far have numerical values. After normalizing or standardizing, these features can be directly used as inputs to the machine learning model. However, sometimes a feature is not numerical but categorical. Such a categorical feature must first be converted to a numerical value before it can be used in a model.

There are several ways to convert categorical values to numerical values. When the categories have an ordinal relationship, this conversion is straightforward: for instance, a low-mid-high categorization would translate into 0.0-0.5-1.0 in numerical values. When the categories do not have an ordinal relationship, they are called nominal categorical values [100]. Nominal categorical values are not associated with any numerical order or hierarchy, for instance car brands or colours. When nominal categorical values are converted to numerical features by simply assigning a random value to each individual category, false hierarchical information is introduced into the machine learning model. Therefore, an alternative method of conversion is needed. One such method, used in this project, is one-hot encoding.

In one-hot encoding, all n unique categories in a column are transformed into n new columns, each representing one unique category. These n new columns are filled with zeros, except for the samples that belong to the corresponding category, which are marked with a one. In this way, the nominal categorical value is converted to numerical values without introducing ordinality to the machine learning model. A downside of one-hot encoding is that with a large number of unique categories, a large number of low-informative features are added to the feature space.

One of the one-hot encoded nominal categorical variables from the Wärtsilä Archimedes dataset is ShipType. This variable denotes the ship type associated with the project. Over the years, however, this variable has become cluttered, resulting in 145 unique categories. These categories range from actual ship types to inconsistently inserted project titles, thereby complicating the analysis. To address this issue, a new variable, “ShipGroup”, was created and provided by Wärtsilä, reducing the number of categories from 145 to 18. All ShipGroup categories have been numbered in a **ShipGroupNo** variable, which was one-hot encoded to serve as input for the machine learning models. An overview of all ShipGroups with their ShipGroupNo values is shown in table 5.1.

Of all 18 ShipGroups, 9 different ShipGroups remained in the dataset after filtering, being Bulk Carrier, Cargo Vessel, Combination Carrier, Container Vessel, Fishing Vessel, Tanker, RoRo Vessel Single, Service Vessel and Other.

5.4. Feature Engineering using Data Imputation

As mentioned in section 4.1.2, a WakeOrigin variable describing the origin of the wake label exists in the Wärtsilä Archimedes dataset. The possible origins are `CfdCalculationFs`, `CfdCalculationMs`, `ModelTest`, `OtherProject`, `StandardDB`, and `Unknown`. The `OtherProject` and `StandardDB` categories were filtered out of the dataset. In the complete dataset of 1799 samples, 1377 samples have an unknown WakeOrigin, which means that 76.5% of all samples do not have a known WakeOrigin. After the filtering process, which tends to include more recent and correctly labeled samples, this percentage drops to 16.2%. This means that 23 out of the 142 remaining samples after filtering are labelled `Unknown`. Given the significant difference between a model scale and full scale wake field due to the different Reynolds numbers, it was decided to substitute the missing data through a process called data imputation. Whether a wake field is a model scale or full scale wake field was predicted using a support vector machine. Sections 5.4.1 to 5.4.4 will discuss this data imputation process, elaborating

Table 5.1: Overview of all different ShipGroups with their respective ShipGroupNo values.

ShipGroupNo	ShipGroup	Number of vessels
1	Bulk Carrier	147
2	Cargo Vessel	122
3	Combination Carrier	64
4	Container Vessel	309
5	Cruise Vessel	68
6	Fishing Vessel	37
7	Inland Vessel	4
8	Tanker	291
9	Navy & Coast Guard Vessel	75
10	Offshore Vessel	60
11	Passenger Vessel Single	4
12	RoRo Vessel Single	21
13	RoRo Vessel Twin	47
14	Service Vessel	149
15	Passenger Vessel Twin	5
16	Yacht	82
17	Other	210
18	Ferry	104

on the tools and libraries used, the data preparation, the training of the support vector machine, and an evaluation of the model's performance, respectively.

5.4.1. Tools and Libraries Used for Data Imputation

Just like the rest of this thesis project, the data imputation of the WakeScale variable was performed using Python. For the support vector machine architecture, the `svm` module from the Scikit-learn library was used [101]. To address the imbalance between samples labeled model scale and full scale, which will be discussed in section 5.4.2, the `SMOTEENN` module from the Imbalanced-learn library was employed [102]. The Seaborn library [103] was used for evaluation, in particular for the confusion matrix visualisation in section 5.4.4.

5.4.2. Preparation of the Wake Field Data

Just like during the label engineering procedure described in section 4.3, the wake fields, which serve as inputs to the support vector machine rather than as labels in the "main" machine learning model, needed to be prepared before use. All wake fields are interpolated to fixed radii and angles, so that all wake field matrices have equal dimensionality and every entry in all matrices refers to the same position in the wake field.

The next step was to reduce the amount of different classes, in a process called "binning". As the only classes relevant for this thesis are `Model scale`, `Full scale` and `Unknown`, the `CfdCalculationMs` and `ModelTest` categories were merged into `Model scale`. The `CfdCalculationFs` category was renamed.

Thereafter, the dataset was divided into a learning and production set. The former subset consisted of all samples with a known wake scale, and were used in the learning phase to train and validate the support vector machine. The production set was filled with wake fields of unknown scale as their scale needed to be predicted.

A strong class imbalance was observed in the learning set. Class imbalance is problematic in a classification algorithm because it can lead to biased models that favour the most common (majority) class, resulting in poor performance on the less common (minority) class. This imbalance can cause the model to have high overall accuracy but fail to correctly predict samples of the minority class, which is to be avoided. Batista, Prati and Monard [104] explain that this problem can be solved by artificially oversampling the minority class and undersampling the majority class. By over- and undersampling the dataset is artificially balanced by adding and removing data points, respectively, thus reducing

prediction errors on the minority class.

Batista, Prati and Monard propose various combined over- and undersampling techniques, of which SMOTE-ENN was selected [104]. SMOTE (synthetic minority oversampling technique), as explained in Burkov [65], works by randomly selecting a sample x_i from the minority class, as well as its k nearest neighbours. A random nearest neighbour x_{zi} is then selected, and a synthetic sample is created by interpolating between x_i and x_{zi}. ENN (edited nearest neighbours) is an undersampling technique, and works by taking a random majority class sample and removing it from the dataset if at least two of its three nearest neighbours are of the minority class.

The `SMOTEENN` module from Imbalanced-learn automatically performs SMOTE-ENN over- and undersampling. Its input parameters, apart from the dataset to perform the combined over- and undersampling on, include a parameter to set the desired class ratio after re-sampling. This parameter was left empty, which leads to class balance. Furthermore, a random state parameter can be specified to reproduce results by fixing the random choice of samples for re-sampling, enabling debugging.

5.4.3. Training the Data Imputation Model

After balancing the different class in the dataset, the data imputation model could be trained. To boost model performance, a process called “repeated random subset validation” was deployed. Repeated random subset validation is a method for model evaluation and selection, which helps in reducing overfitting by picking the model that generalises best to unseen data. It involves the following steps:

1. Create n random training-validation sets.
2. Train n support vector machine models on the n training sets.
3. Validate all n models on all different validation sets, keeping track of all model’s accuracies on all validation sets.
4. Calculate the variance of all n accuracies of all n models.
5. Pick the model that has the mean highest accuracy with the lowest accuracy variance on the n validation sets.
6. Using the selected model, make predictions for the production set.

The model accuracy is calculated as shown in equation 5.14. The model accuracy is the ratio between the number of correctly classified examples and the total number of classified examples. In equation 5.14, TMS, TFS, FMS, and FFS denote number of prediction outcomes of true model scale, true full scale, false model scale, and false full scale, respectively. A perfectly performing model therefore has an accuracy of 1.

$$\text{Accuracy} = \frac{TMS + TFS}{TMS + TFS + FMS + FFS} \tag{5.14}$$

The support vector machine model was defined and trained using the `svm` module. The RBF kernel was selected. Two hyperparameters associated with the RBF kernel are C and γ. The hyperparameter C is called the regularisation parameter and can be interpreted as a cost function, trading off model accuracy and model complexity. A lower value of C makes the model more forgiving to misclassification, simplifying the model. High values of C lead to a more complex shape of the decision boundary and less classification errors, but with increased risk of overfitting. The hyperparameter γ is called the kernel coefficient and determines the influence of a single sample, meaning how much space around the sample should be classified as the same category as the sample. Low values of gamma denote a large sphere of influence around samples, smoothening the decision boundary. High values of gamma lead to complexer boundaries and can, just like high values of C, lead to overfitting. A comparison of different values of C and γ can be seen in figure 5.1 [105].

5.4.4. Evaluating and Validating the Data Imputation Model

After having obtained the best model from the repeated random subset validation procedure, the performance of the model needed to be evaluated. To do this, predictions were made using the best model based on the validation set inputs, and compared to the validation set labels. There are some random

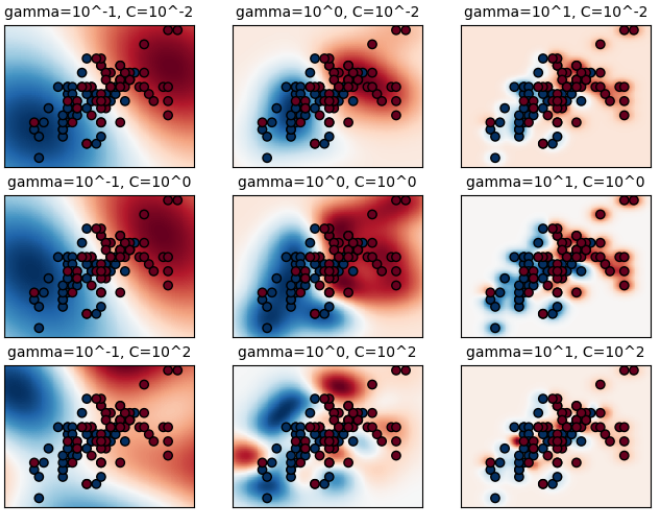


Figure 5.1: The influence of the RBF hyperparameters C and γ on the decision boundary of an SVM. From [105].

factors, being the random samples picked by `SMOTEENN` during over- and undersampling, and the random splitting into training and validation sets during the repeated random subset validation procedure. Therefore, the results discussed in this section are unique to a single run. By running the whole data imputation structure multiple times, it was verified that the results discussed here are representative for the overall model performance. The model/full scale data imputation algorithm is run during every feature initialisation step in any wake field predicting model, after which the metrics discussed below are obtained for that specific run.

To assess the model performance, both a classification report and a confusion matrix have been created. A confusion matrix is a heat map. On its x-axis the predicted classes are shown, and on its y-axis the actual (labelled) classes. In this way, the amount of correctly and falsely predicted classes can be visualised. A confusion matrix for the model/full scale classification model is shown in figure 5.2. In this confusion matrix, 0 denotes a full scale wake field and 1 a model scale wake field. As can be seen, the model has correctly predicted all wake field scales, achieving an accuracy of 1.

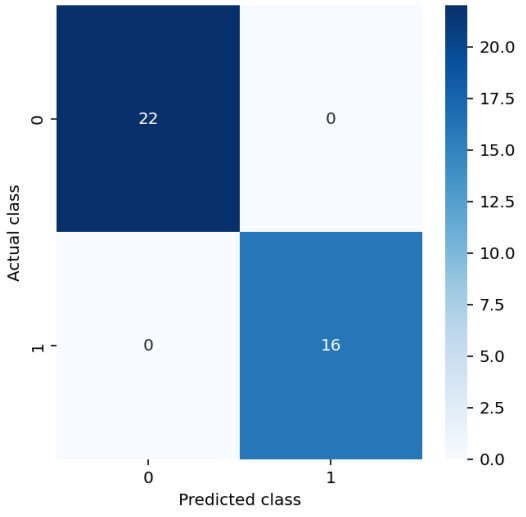


Figure 5.2: Confusion matrix of the model/full scale classification model.

Apart from a confusion matrix, a classification report was created, which can be seen in table 5.2. A classification report shows the precision, recall and F1-score for all classes. In the classification report, just as the confusion matrix, a full or model scale wake field is denoted with a 0 or 1, respectively.

Precision is the ratio between correctly predicted examples of a class and all predictions of a certain class, so for instance, precision $= TFS/(TFS + TMS)$. Recall equals the ratio between correctly predicted examples of a class and the whole size of the actual class: recall $= TFS/(TFS + FMS)$. The F1-score is the weighted average of precision and recall. Apart from precision, recall and F1-score for every class, the classification report shows the model accuracy as well as the macro average (average of the

Table 5.2: Classification report of the model/full scale classification model.

	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	22
1	1.00	1.00	1.00	16
Accuracy			1.00	38
Macro avg	1.00	1.00	1.00	38
Weighted avg	1.00	1.00	1.00	38

unweighted mean) and weighted average of the precision, recall and F1-score. The weighted average is calculated based on the amount of samples in every class, denoted in the “support” column. As the model/full scale classification model almost never makes a classification mistake, all of the values in the classification report shown in table 5.2 are equal to one. Therefore, the classification model was deemed sufficiently accurate to use the predicted wake scales for data imputation on the WakeScale variable. The one-hot encoded **WakeScale** feature columns were added to the feature space.

5.5. Feature Scaling

After obtaining a set of numerical features, the final step before proceeding to model training was to perform feature scaling. Scaling features to (more or less) equal intervals through normalisation or standardisation is a common practice in machine learning engineering, as it can improve the speed of convergence of a model and prevent feature dominance. Especially in gradient-based machine learning models like neural networks, features with orders of magnitude higher values can dominate all derivatives during the backward pass. Additionally, very large or small feature values can lead to numerical instability and integer overflow. In Sections 5.5.1 and 5.5.2, normalisation and standardisation will be discussed. Section 5.5.3 will introduce the concept of data leakage and how to prevent it.

5.5.1. Normalisation

Scaling all features back to a specified interval, usually $[0, 1]$ or $[-1, 1]$, is called normalisation. Because the features are only squeezed in or stretched out to fit the desired interval, the original distribution shape of the feature values remains the same. The normalisation equation is shown in equation 5.15, where x_{norm} is the normalised value of sample x, x_{min} and x_{max} are the minimum and maximum values of x, and l and u denote the lower and upper bound of the normalisation interval, respectively.

$$x_{norm} = \frac{(x - x_{min})(u - l)}{x_{max} - x_{min}} \tag{5.15}$$

5.5.2. Standardisation

A downside of normalisation is that in the presence of extreme outliers, the majority of data will be concentrated in a smaller sub-interval within the normalisation interval. Standardisation is the process of manipulating features in such a way that they become zero-mean and have a standard deviation of one, obtaining the properties of a standard normal distribution. Standardisation is better at dealing with extreme outliers, as it will not squeeze the rest of the data into a small subset. As there are no specific rules on whether to normalise or standardise features, both options have been tried. Neither normalisation or standardisation yielded better results, and it was randomly chosen to use standardisation as feature scaling method. Equation 5.16 shows the standardisation equation.

$$x_{std} = \frac{x - \mu}{\sigma} \tag{5.16}$$

5.5.3. Data Leakage

A phenomenon that should be avoided at all costs during machine learning engineering is data leakage. Data leakage occurs when information from the validation or test sets “leaks” into the model training process. This means that information from the validation or test sets is used by the machine learning model before it is supposed to be available to it. Data leakage can therefore lead to overfitting of the

model to the validation or test sets, making it perform well on one of those two sets while being unable to generalise to new, previously unseen data.

Data leakage can happen explicitly, for instance, by having exact duplicates in a dataset. If one duplicate sample ends up in the training set and the other in the validation set, the model will learn to remember the exact label for the duplicate input, leading to seemingly good performance. It can also happen implicitly, where not exact samples are known to the machine learning model, but other related information. Although the risk of data leakage is not unique to feature scaling, it can easily happen when the feature scaling is not applied properly.

To prevent leaking information from the validation and test sets to the machine learning model, normalisation or standardisation should first be performed on the training set only, rather than the whole dataset. If the dataset statistics (mean, standard deviation, minimum value, and maximum value) were calculated based on the whole dataset before splitting into training, validation, and test sets, the training set would contain implicit information about the validation and test sets. After normalising or standardising the training set, the training set-based statistics can be used to normalise or standardise the validation and test sets as well.

5.6. Feature Characteristics and Conclusion

This chapter has described the process of feature engineering. Some features have been directly taken from dataset variables, while other features have been derived from or are semi-empirical features based on dataset variables. Two categorical features (ShipGroupNo and WakeOrigin) have been discussed, with the latter being engineered using data imputation with a support vector machine. All features have been scaled by standardising them.

Table 5.3 presents the descriptive statistics of all engineered features, except the one-hot encoded features ShipGroupNo and WakeOrigin. These (sets of one-hot encoded) features have been omitted as they do not have any numerical meaning, only a categorical one. Table 5.3 is also shown in appendix D, where also a box plot visualisation of the statistical distribution of all features is presented. Table D.1 also defines the bounds of the feature domain, which is important to take note of. Predictions based on unseen data are more likely to be accurate when they fall within the feature domain of the model. The distribution of each feature is shown in figure 5.3.

It can be seen that some of the engineered features have a skewed distribution, such as Displacement and Delta_Bertram. This is not necessarily a problem, as standardisation rather than normalisation was used to scale all features and standardisation is less sensitive to outliers than normalisation. The most important conclusion to draw is that, according to the distributions shown in figure 5.3, and the density plots and heatmap presented in appendix D, no two features seem to have a high correlation (except for Cb and W_Taylor, which will be elaborated on in section 7.5). Having obtained a set of uncorrelated features, the machine learning models can now be defined and trained.

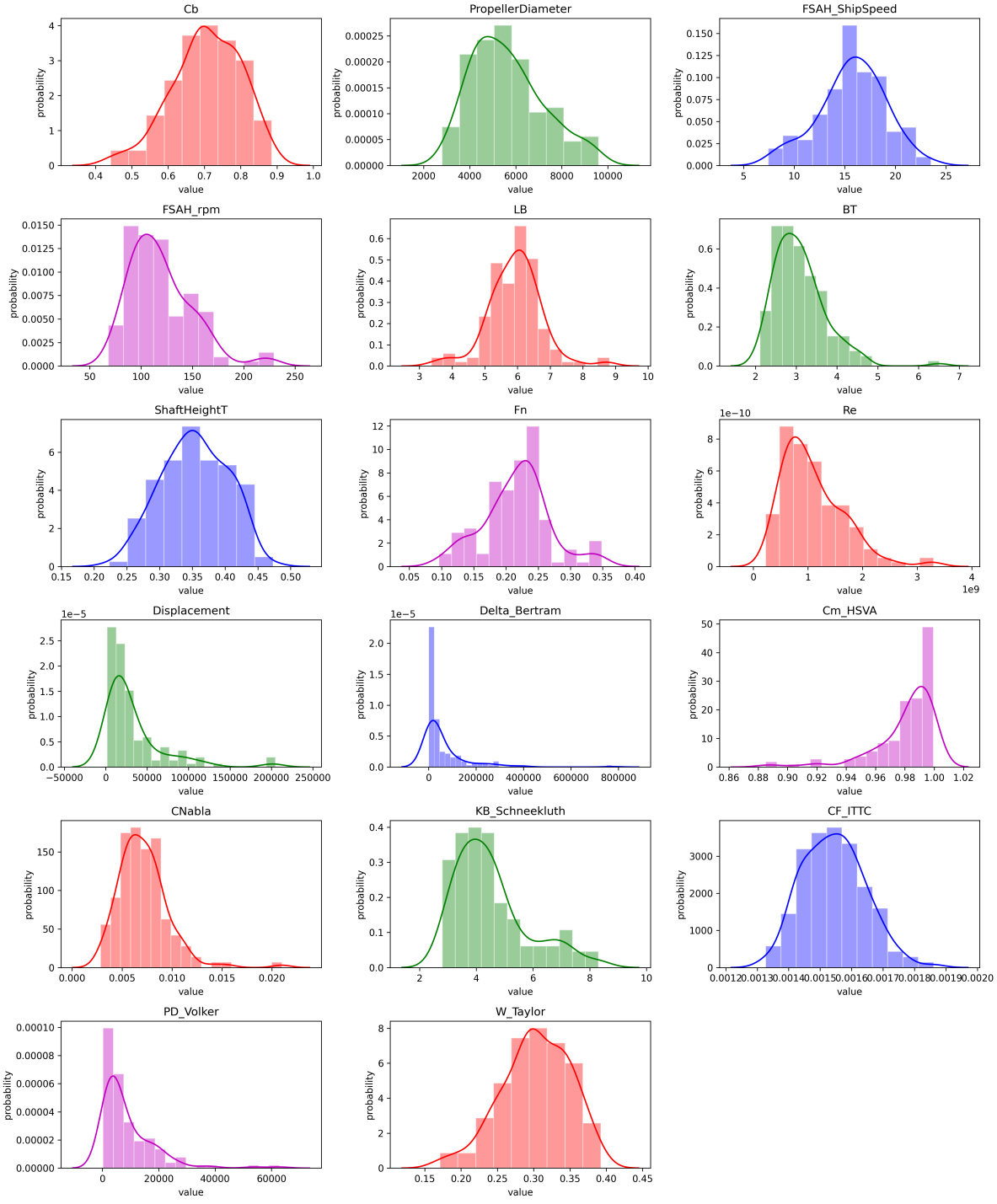


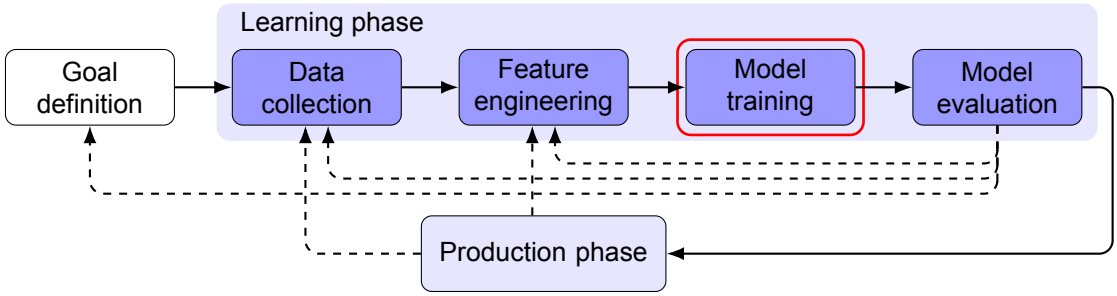
Figure 5.3: Univariate distribution plots of all engineered features.

Table 5.3: Descriptive statistics of the engineered features.

	mean	std	min	25%	50%	75%	max
Cb	0.7060	0.0938	0.4420	0.6485	0.7087	0.7800	0.8850
PropellerDiameter	5631	1564	2800	4500	5400	6500	9600
FSAH_ShipSpeed	15.73	3.281	7.435	13.79	15.95	17.90	23.51
FSAH_rpm	119.2	31.12	68.20	96.70	114.0	137.7	229.4
LB	5.925	0.8070	3.367	5.446	5.988	6.422	8.812
BT	3.086	0.6332	2.110	2.617	3.036	3.350	6.508
ShaftHeightT	0.3533	0.0499	0.2231	0.3184	0.3521	0.3933	0.4732
Fn	0.2164	0.0515	0.0960	0.1878	0.2218	0.2421	0.3486
Re	1.086e9	5.634e8	2.235e8	6.748e8	9.501e8	1.378e9	3.299e9
Displacement	32839	37188	1709	10284	20162	39574	204529
Delta_Bertram	60817	101057	585.6	8140	22209	69046	772855
Cm_HSVA	0.9805	0.0211	0.8851	0.9749	0.9868	0.9950	0.9995
CNabla	0.0071	0.0025	0.0028	0.0056	0.0068	0.0083	0.0210
KB_Schneekluth	4.503	1.275	2.800	3.612	4.214	4.836	8.307
CF_ITTC	0.0015	0.0001	0.0013	0.0015	0.0015	0.0016	0.0019
PD_Volker	8733	9609	228.9	2672	4981	12449	62769
W_Taylor	0.3030	0.0469	0.1710	0.2742	0.3043	0.3400	0.3925

6

Model Training



The step following the feature engineering step is training the machine learning model, as can be seen in the machine learning project life cycle [65]. Multiple different types of models have been trained. An explanation of the chosen machine learning structures is provided in section 6.1, together with an overview of tools and libraries used, the code framework, the training setup, visualisations, and the used loss functions and performance metrics. Section 6.2 describes a small experiment that highlights the importance of considering the interconnectedness between all data points in the wake field. Lastly, in sections 6.3, 6.4, and 6.5 the tuning and training of all feed-forward neural networks, the ensemble method and the long short-term memory model will be elaborated on, respectively.

6.1. Model Setup

This section will elaborate on everything that is constant across all trained neural networks. Section 6.1.1 will provide the overview of hardware, tools, and libraries used. In section 6.1.2, the choice for neural networks in general as well as the specific neural network "flavours" will be explained. Thereafter, in section 6.1.3 the code framework will be introduced. Section 6.1.4 will show the part of the training setup that is common across all models, and in section 6.1.5 all used standard loss functions and performance metrics are introduced and explained. Finally, sections 6.1.6 and 6.1.7 will introduce the used visualisations and custom-built loss functions.

6.1.1. Hardware, Tools, and Libraries Used

All calculations have been performed on a laptop equipped with a 12th gen Intel Core i5-1245U CPU. This CPU has 12 cores with a base speed of 1.60 GHz. Additionally, the laptop is fitted with 16 GB of memory.

All modeling, validation, and visualisation for the various model architectures was conducted using Python. In addition to the standard Python libraries such as NumPy, Matplotlib, SciPy, and Pandas, several other libraries were used. The primary library used for defining and training most models was TensorFlow [106], a popular machine learning framework compatible with Python and other program-

ming languages. On top of TensorFlow, the Keras API was employed [107]. Keras is a higher-level library than TensorFlow, providing built-in methods for specifying loss functions, model layers, training algorithms, and hyperparameter tuners. Furthermore, Scikit-learn was used to build a simple support vector regression model, and for some of its supporting features in the main models [101].

6.1.2. Model Architectures

Neural networks have been selected as the preferred machine learning architecture for this project. The reasoning behind this choice is as follows. Firstly, neural networks are well-suited for recognising complex patterns and relationships in datasets. Given the influence of complex fluid dynamics phenomena in wake field prediction, the chosen machine learning architecture must be capable of capturing this complexity. The non-linear nature of the phenomena behind wake fields is dealt with by using non-linear activation functions within the neural network structure.

Furthermore, due to the interconnectedness between all nodes in a neural network, each output node is informed about its surroundings. The gradients of all velocities within a wake field are constrained, so velocity predictions at a certain point in the wake field should be made while considering the surrounding velocity predictions. This interconnectedness is not accounted for when using less sophisticated machine learning architectures such as support vector regression. An example of single-point prediction using a support vector regression structure is provided in section 6.2 and serves as justification for employing the more complex neural network architecture.

A downside of using neural network architectures is their limited interpretability. While the concept of fitting a high-dimensional hyperplane to the dataset when using a support vector machine can (to some extent) still be visually interpreted, a neural network is typically a black box: the effect of tuning its hyperparameters and model parameters can be evaluated and interpreted, but it is not clear which underlying phenomena are exactly captured by the neural network and to what extent.

Lastly, neural network architectures are flexible by nature. When a larger dataset becomes available in the future, re-training a neural network is a straightforward task. Therefore, the architectures presented in this thesis can easily be adapted and improved by updating the dataset and re-training the models. Three types of neural network architectures have been selected and developed during this project:

Feed-Forward Neural Network

The first type of neural network architecture deployed during this project is the feed-forward neural network (FNN). This relatively simple type of neural network architecture served as a starting point for the wake field predictions. FNNs were also utilised to test the effect of different loss functions during model training, to assess the use of the discrete cosine transform (explained in section 4.3.2), and to explore hyperparameter tuning. An in-depth explanation of the trained FNNs for this project can be found in section 6.3.

Ensemble Method

The second type of neural network architecture is an ensemble method. Although it was explained in section 3.3.2 that ensemble methods do not necessarily consist solely of neural networks, in this project an ensemble of two neural network base models and a neural network meta-model was employed. The reason to try an ensemble method was to investigate whether reducing the output dimensionality of the base models would lead to better predictions by the meta-model. The ensemble method is elaborated on in section 6.4. When taken to the extreme, the output dimensionality is reduced to one, representing a single-point prediction. To demonstrate the importance of the interconnectedness between points in the wake field, section 6.2 presents a support vector regression model that aims to accurately predict single points in the wake field.

Recurrent Neural Network

The final neural network architecture used in this project is a recurrent neural network, specifically a long-short term memory (LSTM) network. LSTM networks are well-suited for handling data with a strong sequential relationship, often a strong temporal relationship. By iteratively allowing an LSTM network to cycle through all angles in a wake field, the aim was to use the abilities of an LSTM to handle sequentially dependent datasets, optimally accounting for the spatial sequential relationships between velocity points in the wake fields. The LSTM network is elaborated on in section 6.5.

6.1.3. Code Framework

For the sake of clarity, it was decided early in the project to divide all code into different Python scripts based on functionality and/or the step in the machine learning project life cycle where each block of code was utilised. This resulted in twelve Python scripts, of which eleven contain various functions, and one over-coupling "pipeline" script combines those functions to step-by-step walk through the machine learning model engineering procedure. Apart from the Python scripts, there is one JSON file containing information about the dataset. A schematic overview of all scripts is provided in figure 6.1, where arrows from one file to another indicate that the target file is called within the origin file. All Python files have self-explanatory file names. Additionally, the scripts that read from and/or write to computer storage are indicated with an asterisk (*). For simplicity, not all script calls made from function-containing scripts are shown.

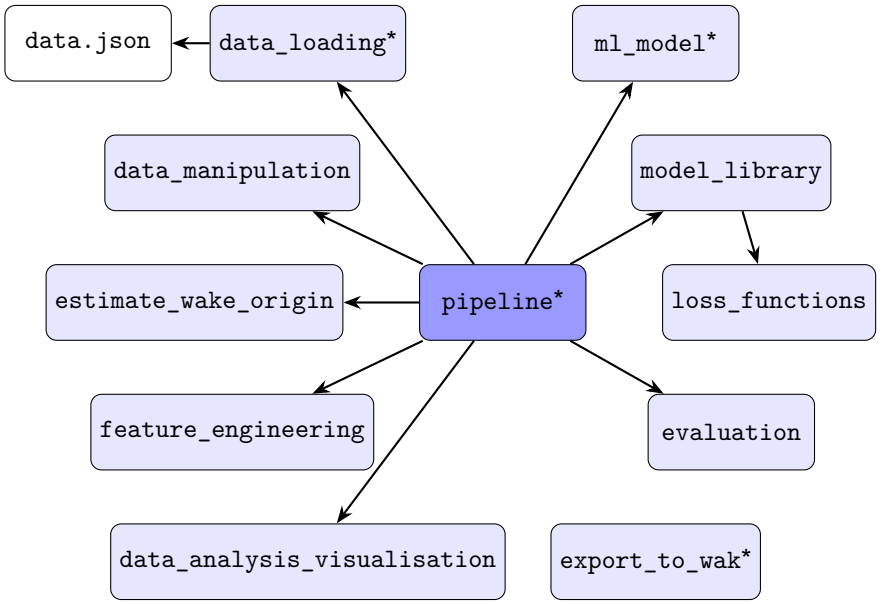


Figure 6.1: Schematic overview of the code framework.

6.1.4. Training Setup

To ensure comparability between the different model architectures that were deployed, a large part of the training setup was kept constant for all architectures. To begin with, all models were trained on the same version of the Wärtsilä Archimedes dataset, with the same filters applied and using the same set of features. All features have been standardised. A serious concern while training all of the models was the dataset size. After filtering, only 142 out of 1799 samples remained. There is no consensus on the minimum dataset size required to train an accurate, robust, and well-generalising model. Although other factors, such as the complexity of the machine learning problem and the informativeness of the features, play a significant role in this minimum requirement, there are some rules of thumb. For instance, in [108], a sample size of 10 to 100 times the number of features is mentioned as a rule of thumb for estimating the minimum dataset size. Given a dimensionality of 28 features, this would imply that the *minimum* dataset size for this project should be somewhere between 280 and 2800 samples, which the current dataset does not meet.

Two problems arise when the dataset size is too small:

1. The machine learning model might learn to remember individual samples instead of identifying underlying patterns and generalising.
2. The individual contribution of each sample becomes more significant, so if (near-)outliers end up in the training, validation, or test set, this can cause a noticeable change in performance.

An additional problem is that during hyperparameter tuning, various combinations of hyperparameters are tested, and those that result in the best performance on the validation set are selected. When a large number of combinations is tested, this may lead to data leakage from the validation set to

the model. Although this problem is not unique to small-dataset models, the risk of overfitting to the validation set increases with decreasing dataset size [65].

To assess the level of overfitting to the validation set during model training, an additional test set was used. This test set had not been "seen" by any model during training and was only used to assess a model's performance after training. Large differences between model performance on the validation and test sets may indicate overfitting to the validation set caused by data leakage during the hyperparameter tuning process.

By keeping the 75%-15%-10% train-validation-test split constant over all trained models, the effect of the second problem on the relative comparability between all models was minimised. Additionally, to assess the general effect of the small dataset size on all models, a cross-validation of all models was performed, meaning that every model was re-trained multiple times on a different training-validation split in a procedure called repeated random subset validation. The variance in model performance on the different training-validation splits provides an indication of the sensitivity of the model to the training data. Repeated random subset validation will be explained in section 7.4.

Another constant factor over all model architectures is that only models predicting the axial component of the wake field were developed and evaluated. The reasoning behind this decision is that, due to performance issues related to the dataset size and feature informativeness, a more thorough evaluation of the axial component predicting models was performed. Furthermore, multiple different model architectures (discussed in section 6.1.2) were developed in an effort to improve model performance. It should be noted that all code was written with the other velocity components in mind, as well. As the procedure of predicting the other velocity components is very similar to that of the axial components, this additional functionality could easily be added in the future. Also, it was experimentally determined that axial velocities in a nominal wake field are the most important with respect to cavitation properties, compared to tangential and radial velocities. This will be disussed in section 7.6

Furthermore, all models that had their hyperparameters tuned using a hyperparameter tuning algorithm used the same algorithm, Hyperband. The hyperparameter space was too large for grid or random search to perform enough iterations, and the Bayesian optimiser kept converging to extreme values in the hyperparameter space. Therefore, the Hyperband algorithm was the best option, preventing memory overflow during hyperparameter tuning while still enabling the testing of many different hyperparameter configurations.

The final choice in the training setup, constant across all architectures, involved the implementation of learning rate reduction and early stopping. If the reduction in the loss function stagnates during training, the learning rate is reduced by a factor of 10, with a minimum threshold of $1*10^{-6}$. This reduction occurs after 25 epochs of stagnation. If this reduction does not yield any improvement, the training process is terminated after 40 epochs of continued stagnation. This approach saves time and computational resources in cases where convergence is reached before reaching the maximum number of epochs.

6.1.5. Standard Loss Functions and Performance Metrics

This section describes all standard loss functions and performance metrics used for model training and performance assessments. In addition to the mean squared error (MSE) between predictions and targets, explained in section 3.1.6, these metrics include the mean absolute percentage error (MAPE), coefficient of determination (R^2), Pearson product-moment correlation coefficient (PPMCC), and an assessment of residuals between predictions and targets.

Apart from these standard, "out-of-the-box" loss functions and performance metrics, some custom loss functions have been developed. This was done in an effort to optimally connect the loss function to the physical problem of predicting a wake field. These loss functions are discussed in section 6.1.7.

Mean Absolute Percentage Error

The mean absolute percentage error (MAPE) is a relative error metric, based on the mean of all percentual errors between predictions $\hat{y}_i$ and targets y_i in the prediction and target arrays $\hat{\mathbf{y}}$ and $\mathbf{y}$, respectively.

Equation 6.1 shows how the mean absolute percentage error is calculated.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \cdot 100\% \tag{6.1}$$

The mean absolute percentage error is a useful performance metric due to its interpretability as a percentage difference between targets and predictions. However, this metric was not chosen for the custom loss function for several reasons. Firstly, when the target or prediction values are zero or near-zero, the MAPE value can become undefined or extremely large due to division by zero or near-zero values. Given that the relative axial velocities, which together define the wake field, typically fall within this range, MAPE was deemed unsuitable for this problem. Additionally, unlike the mean squared error (MSE), MAPE does not progressively penalise larger errors more heavily than smaller errors.

Coefficient of Determination

The coefficient of determination, better known as the R-squared or R^2 value, is indicative for the quality of a linear fit [109]. Its calculation is shown in equation 6.2. In this equation, the nominator represents the sum of squared errors between all predictions $\hat{y}_i$ and targets y_i. This is called the residual sum of squares. The denominator equals the sum of all squared differences between targets y_i and the mean target value $\bar{y}$. This is called the total sum of squares, and is closely related to the variance of the dataset labels, although while calculating the variance the average of all squared differences would be taken instead of the sum. The range of the coefficient of determination is $[0, 1]$, where a value of 0 indicates that the model does not explain any of the variability of the response data around its mean, and 1 indicates that the model explains all the variability of the response data around its mean. This means that with an R^2-value of 0, the model does not perform better than just predicting the average of all targets, and an R^2-value of 1 indicates a perfect fit, where all predictions are equal to the targets.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{6.2}$$

The coefficient of determination value is useful for evaluating the goodness-of-fit of a regression, indicating how well predictions match target values. In this project, it has been used to assess the goodness-of-fit in the “actual-vs-predicted” plot that will be explained in section 6.1.6.

Pearson Product-Moment Correlation Coefficient

The Pearson product-moment correlation coefficient (PPMCC), or Pearson correlation coefficient (PCC), is used to measure the linear correlation between variables, in this case targets and predictions. The calculation of the PPMCC is shown in equation 6.3 [110], where the numerator represents the covariance between the predictions $\hat{y}_i$ and the targets y_i. The covariance indicates the extent to which the predictions and targets change together [111]. The denominator is the product of the standard deviations of the predictions and the targets. By dividing the covariance by the product of the standard deviations of both the predictions and targets, the correlation strength is determined.

$$\text{PPMCC} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2 \sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})^2}} \tag{6.3}$$

The range of the Pearson product-moment correlation coefficient is $[-1, 1]$, where values close to -1, 0 and 1 are indicative of a strong negative, no, and strong positive correlation, respectively.

In this project, the PPMCC has been used as a metric during model training. By making predictions on the validation set and logging the PPMCC value after every epoch, an PPMCC-vs-epochs graph can be made. From the shape of this graph, the development of model performance over the training epochs can be derived. This works as follows. At the beginning of the model training, the PPMCC should be close to zero. This means that the predictions do not have any correlation with the targets. As the model has not yet learned the underlying patterns in the training data, this is expected. As the training continues, the PPMCC should increase. An increasing value of PPMCC is an indication that the model is learning from the training data, resulting in better correlations between targets and predictions after each epoch. Eventually, the PPMCC plateaus. A similar stagnation can be seen in the loss-vs-epochs

graph. This is a sign that the model does not learn much from the data anymore, suggesting that the model has learned as much as it can from the training data in its current hyperparameter configuration. Lastly, if the PPMCC starts to decrease after reaching its peak, this might be because of overfitting to the training set. When this happens, its performance on the validation set - and therefore the correlation between predictions and targets - decreases.

Assessment of Residuals

The residuals of a model prediction, defined as the difference between the targets and the predicted values ($y_i - \hat{y}_i$), can be used to assess the performance of a model. The mean and variance of the residuals provide insights into the model's bias and variance, respectively. The distinction between bias and variance is illustrated in Figure 6.2 [112]. Bias and variance are indicative of the model's complexity. This concept is known as the bias-variance trade-off, which implies that more complex models tend to have low bias but high variance, whereas simpler models show the opposite behaviour. This also means that bias and variance cannot be reduced simultaneously; reducing one typically increases the other. The "sweet spot", where bias and variance are balanced and both acceptably low, and minimise total error, is aimed to be found through hyperparameter tuning and model training.

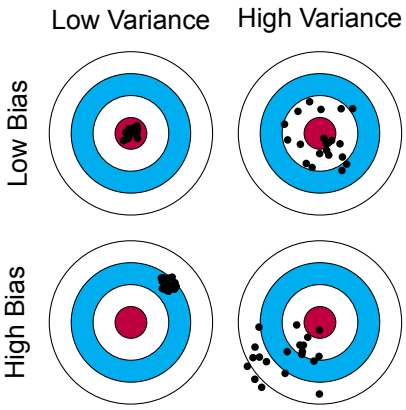


Figure 6.2: Bias and variance. Adapted from [112].

A near-zero average of residuals, indicating low model bias, is one of the factors that determine model quality. Building on a near-zero average of residuals, the distribution of residuals is also important to consider. When residuals are normally distributed, it indicates that errors are random and there is no systematic over- or under-prediction of the targets. This is a sign that the machine learning model has effectively identified all underlying patterns in the dataset.

Section 6.1.6 will present examples of visualisations based on residuals, including a visualisation that indicates the level of normality of residuals. To quantify the level of normality, the Shapiro-Wilk test was used. As a built-in function in SciPy, this test "tests the null hypothesis that the data [the residuals] was drawn from a normal distribution" [113]. The `stats.shapiro` function returns two values: the test statistic and the p-value. The test statistic is a real number between 0 and 1 that indicates the probability of the residuals being normally distributed. However, the p-value also needs to be considered. Low values (<0.05) indicate that the null hypothesis of the residuals being normally distributed should be rejected.

6.1.6. Visualisation

This section will present the various visualisations that have been developed and generated during and after each machine learning model training procedure. First, three different types of wake field plots will be introduced. Next, graphs showing the development of the loss function and performance metrics over the number of epochs during training will be presented and explained. Finally, the post-processing visualisations generated after model training will be discussed.

Single Wake Field Plots

The first type of wake field plot is the single wake field plot. This plot takes a complete, 360-degree wake field as input and outputs a visual representation of the wake field. The single wake field plot was primarily used to visualise the input wake fields from the Wärtsilä Archimedes dataset, as the labelled wake fields are provided at an interval of $\theta = [0, 360]$ degrees. The wake field plot enables the visual inspection of the labelled wake fields. An example wake field plot is shown in figure 6.3a.

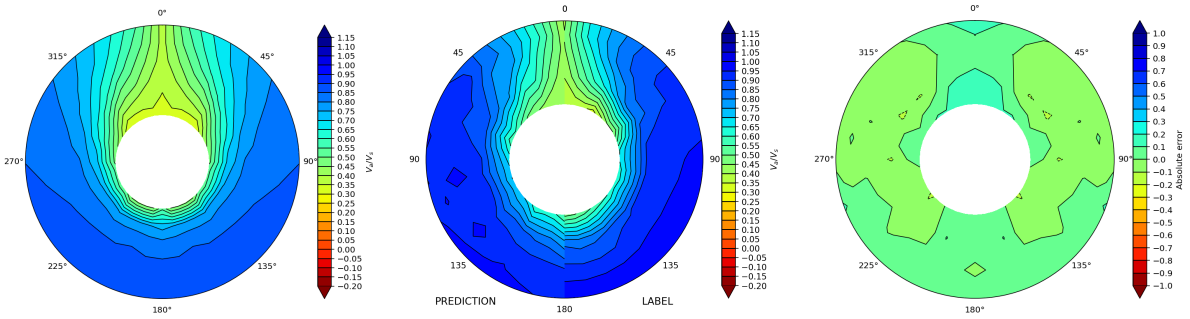
Comparison Plots

The comparison plot, or combined wake field plot, serves as an easy comparison between labelled and predicted wake fields and is shown in figure 6.3b. It should be interpreted as follows. On the left-hand

side of the comparison plot, the predicted wake field is shown. On the right-hand side, the labelled wake field is shown. Since all predictions are made at the same radii, the comparison plot always starts at the lowest standard radius of $r/R = 0.4$. As all wake fields in this thesis are deemed symmetrical (and samples with excessively asymmetrical wake fields are filtered out), the angular range of both the labels and the predictions is $\theta = [0, 180]$ degrees, allowing for side-by-side comparison as shown in figure 6.3b.

Residual Plots

The final type of wake field plot is the residual plot. The residual plot visualises the difference between the predicted wake field velocities and the labelled ones (y_i - $\hat{y}_i$). Although the angular range of the residual plot is $\theta = [0, 360]$ degrees, the left-hand and right-hand sides are equal because the angular range of all predictions is only $\theta = [0, 180]$ degrees. The residual plot can be used to assess the extent to which the residuals are random across all predictions and whether the distribution of residuals follows a pattern. An example residual plot is shown in figure 6.3c.



(a) Wake field plot **(b)** Comparison plot **(c)** Residual plot

Figure 6.3: Example wake field plot, comparison plot and residual plot.

Loss and Metric Development Over Epochs

The next type of visualisation consists of graphs that display the development of some loss function or performance metric over the training epochs. As standard loss functions and performance metrics have been elaborated on in section 6.1.5 and the custom loss functions will be explained in section 6.1.7, example plots will not be shown here. The following loss functions and metrics have been logged over the training epochs and used for visualisation:

- Custom loss on the training and validation sets
- Mean squared error loss on the training and validation sets
- Mean absolute percentage error loss on the training and validation sets
- Pearson product-moment correlation coefficient on the validation set
- Learning rate

Post-Processing Visualisations

The final type of visualisations are the post-processing visualisations, based on the model predictions, targets and residuals. They will be explained in this section.

Figure 6.4 shows an actual-versus-predicted plot. This can be interpreted as a comparison wake field plot without the visual representation of the location of the predicted velocities, instead plotting them against the labelled (actual) velocities. If the model were a perfect predictor of wake field velocities, all scatter points would fall on the sloped grey line. The coefficient of determination, printed in the top left of figure 6.4, is used as an indication of the goodness-of-fit between the predictions and the targets.

In figures 6.5a and 6.5b, a plot of residuals and a probability plot are shown, respectively. The plot of residuals is a histogram that provides insight into the mean and distribution of residuals. The probability plot, together with the Shapiro-Wilk test discussed in Section 6.1.5, shows the level of normality of the set of residuals. It works by comparing the samples, in this case the residuals, with theoretical samples

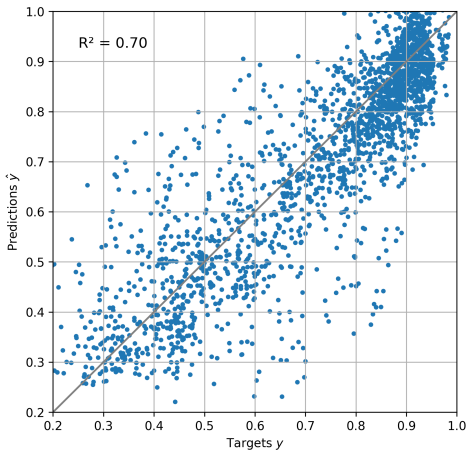
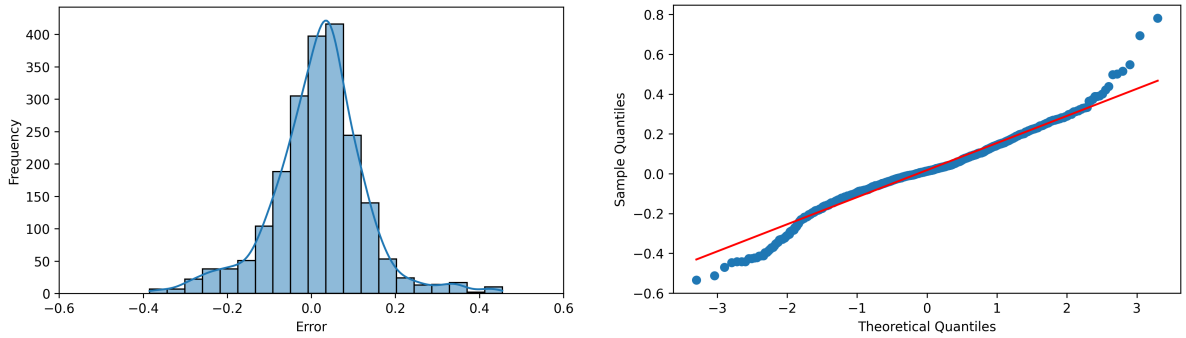


Figure 6.4: Example actual-versus-predicted plot

drawn from a standard normal distribution. A perfectly normally distributed set of residuals would result in all scatter points falling on the red line. In the case of figure 6.5, the probability plot can be used to conclude that although the set of residuals is relatively normally distributed towards the middle quantiles, the two tails of the set do not follow a normal distribution well. This is indicated by the fact that near both extremes, the scatter points begin to deviate from the red line.



(a) Plot of residuals

(b) Probability plot

Figure 6.5: Example plot of residuals and probability plot.

6.1.7. Custom Loss Functions

As mentioned in the introduction of section 6.1.5, it is important to select a loss function that is connected to the physical problem. By creating custom loss functions, the aim was to optimally assign higher loss values to physically worse and lower loss values to physically better wake fields.

In a wake field, large parts are of less interest in terms of cavitation behaviour because they are not typically located in regions where cavitation is expected. Furthermore, an out-of-the-box mean squared error loss function would be biased towards the inner radii. Due to the nature of polar plots, data points closer to the inner radius represent smaller areas than those located further towards the outer radii. When using the discrete cosine transform to reduce the dimensionality of the output space, the predictions consist of the first n coefficients calculated during the transformation. As can be seen in appendix C, the first few coefficients are typically several orders of magnitude larger than those that follow. The reasoning behind, as well as the calculation of the custom loss functions that have been developed will be discussed in the following sections.

Weighted Mean Squared Error

To account for the natural bias towards inner radii when using a standard MSE loss metric, a weighted mean squared error custom loss function was defined. First, the area weight as a function of wake field

radius was determined. This was achieved by calculating the area of the circle with the upper bound radius and subtracting the area of the circle with the lower bound radius. The upper and lower bound radii are defined as the average of the radius of interest and the radius above it, and the average of the radius of interest and the radius below it, respectively. For the lowest radius, there was no lower bound radius. For the highest radius of 1, the upper radius was set to 1. The areas were normalised to obtain the area ratios. For every radius, the mean squared error between the prediction and the target was then multiplied with its respective area ratio to obtain a weighted mean squared error loss.

In addition to correcting for the represented area, the weighted mean squared error custom loss function allows for assigning arbitrary weights to each radius. The rationale for experimenting with this custom weighting was to emphasise regions in the wake field that are sensitive to cavitation development, such as $r/R = 0.7$. However, as no significant improvement in prediction performance was observed with various weighting configurations, all custom weights were set to one.

Velocity-Based Mean Squared Error

This loss function was only defined for models that utilised the discrete cosine transform (DCT). As the first few coefficients calculated after the DCT are typically orders of magnitude larger than those that follow, an out-of-the-box mean squared error loss metric would be biased towards the first few coefficients. While this bias can be justified because the first few coefficients typically contain the most information about the wake field, the details in the wake field that distinguish an average prediction from a good one might be ignored. One way to address this issue is to standardise the DCT coefficients before calculating the loss metric. Another option is to define a custom loss function that performs an inverse discrete cosine transform before calculating the loss. This approach was implemented in the velocity-based mean squared error custom loss function. After the inverse transform, the weighted mean squared error loss function was applied to obtain the loss value.

Wake Fraction Mean Squared Error

Apart from considering every individual velocity in the wake field, the average of all velocities should also be taken into account when assessing the quality of a predicted wake field. This can be achieved by calculating the nominal wake fraction w, which is the wake fraction associated with the nominal wake field that can be measured at the propeller plane when only the bare hull is present. The average relative velocity v_a/v_s over the predicted wake field was calculated using the same weighting for all radii as explained in the weighted mean squared error. This average relative velocity was subtracted from 1 to obtain the nominal wake fraction of both the predicted and target wake fields. The wake fraction mean squared error custom loss function consists of the MSE between these two nominal wake fractions. As can be seen in the residual plot in figure 6.6, no spatial information is included in this loss function as it is only an average metric. Therefore, the wake fraction mean squared error custom loss function cannot be used on its own, but can serve as a supplement to another loss function.

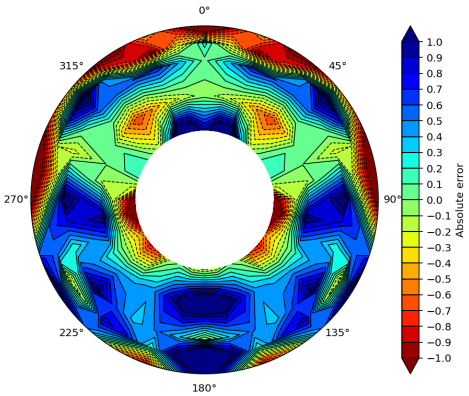


Figure 6.6: Residual plot (absolute difference between label and prediction) of a wake field predicted by a model trained using the wake fraction MSE.

Combined Loss Function

The final custom loss function is a combinator loss function, which allows different loss functions to be combined using weights. For instance, the weighted MSE can be combined with the wake fraction MSE. Most of the trained models have been trained using this combined loss function.

6.2. Support Vector Regression Model

To quantify the importance of considering the interconnectedness between all data points in the wake field, further justifying the choice for neural networks as preferred machine learning architectures for this project, a simple experiment was conducted. From the original dataset, only the relative velocities

at an angle of θ = 20 degrees at radii $r/R = \{0.4, 0.6, 0.8, 0.9, 1.0\}$ were used to train a total of five support vector regression (SVR) models that make single-point predictions.

The procedure was as follows. For every SVR model (five in total), only the label values corresponding to the respective radius were kept. From the 136-sample dataset, a 36-sample test set was drawn. With the remaining 100 samples, a so-called nested cross-validation was performed. This nested cross-validation procedure is visualised in figure 6.7 [114]. The nested cross-validation serves two purposes:

- To perform a cross-validation on the generalisation error of the SVM models.
- To perform hyperparameter tuning in such a way that overfitting is prevented.

As can be seen in figure 6.7, the 100-sample dataset (denoted with "original set") is split up in k parts, or "folds". For the SVR models, k = 10. By iterating through this loop and systematically select one fold to be the test fold and the other nine to be the training folds, the generalisation error of the SVR models can be assessed. The difference between nested cross-validation and "regular" cross-validation, is that for every iteration, the training folds are split up again. For the SVR models, this "nested split" consisted of 5 folds. By performing a 5-fold cross-validated grid search iteratively using four of these inner folds as training set and the remaining one as validation set, for every fold the optimal hyperparameters C and γ could be determined, as well as the resulting error on the validation set. All SVR models used the radial basis function (RBF) kernel. The search grid consisted of 150x150 values of C and γ, both logarithmically spaced between 10^{-5} and 10^{3}. For every model, corresponding to one of the five radii, the best hyperparameters were selected and performance metrics were collected on the 36-sample test set. The performance metrics used are MAPE and REP (relative error percentage). MAPE can be seen in equation 6.1, REP is defined as shown in equation 6.4. The resulting predictions made by the five SVR models can be seen in figure 6.8, and the performance metrics are shown in table 6.1.

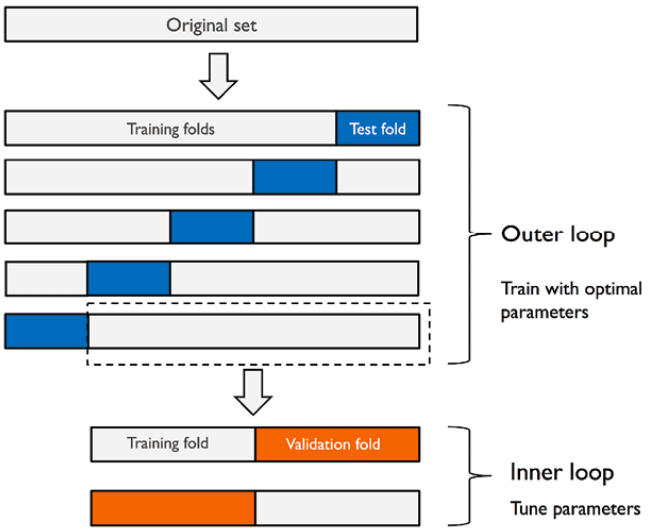


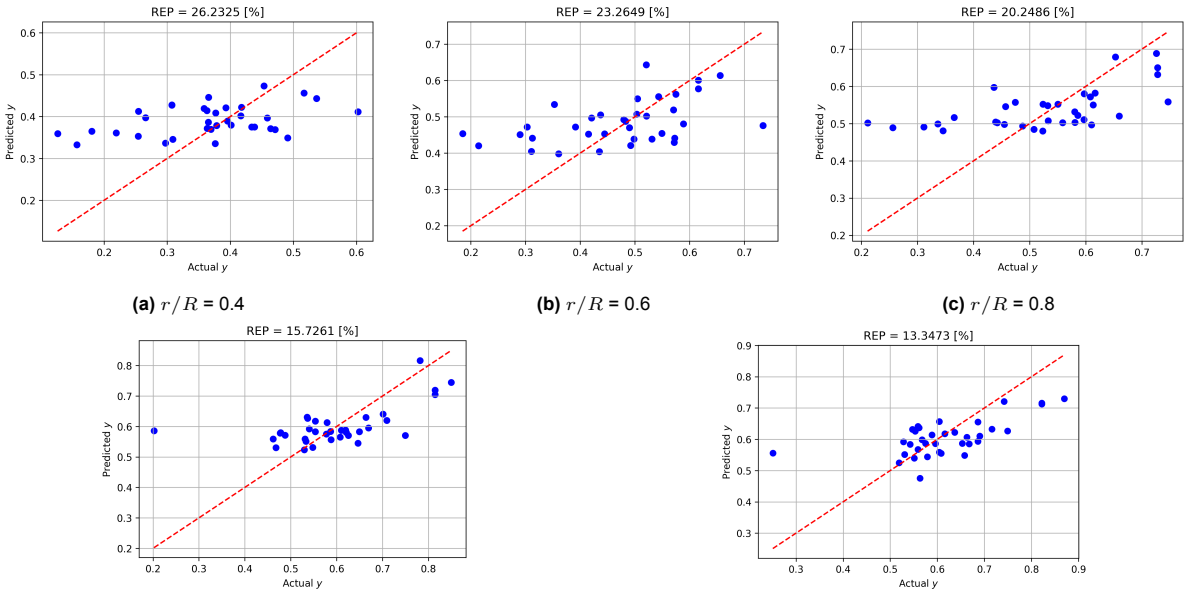
Figure 6.7: Nested cross-validation. From [114].

$$\mathrm{REP} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i)^2}} \cdot 100\% \tag{6.4}$$

Table 6.1: Cross-validated values of MAPE and REP for the five single-point predicting SVR models.

Metric [%]	r/R = 0.4	r/R = 0.6	r/R = 0.8	r/R = 0.9	r/R = 1.0
MAPE	28.69	22.96	21.10	14.99	11.80
REP	26.23	23.26	20.25	15.73	13.35

Several observations can be made from figure 6.8 and table 6.1. To start with, looking at the actual-vs-predicted plots, the SVR models seem to be unable to accurately predict single points in the wake field. A general trend that can be seen is that lower actual values are over- and higher actual values under-predicted: the models seem to predict average values. Furthermore, both figure 6.8 and table 6.1 show that velocities near the outer radii are easier to predict: the values of MAPE and REP decrease with increasing radius, and the points in the actual-vs-predicted plot are closer to the diagonal.



(a) r/R = 0.4 **(b)** r/R = 0.6 **(c)** r/R = 0.8

(d) r/R = 0.9 **(e)** r/R = 1.0

Figure 6.8: Actual-vs-predicted plots for the five single-point predicting SVR models.

In conclusion, these findings show the limitations of the support vector regression models in accurately predicting the wake field data at an angle of 20 degrees, highlighting the need for more complex modelling approaches than support vector regression to capture the underlying complexities. A neural network, for instance, is better by design at capturing complex underlying phenomena and considers the entire wake field at once, taking the values of surrounding data points into account. This finding further answers the second sub-question regarding which machine learning architecture is most suited for prediciting nominal wake fields.

6.3. Feed-Forward Neural Network Tuning and Training

The first type of neural network architecture deployed is the feed-forward neural network (FNN). As mentioned in section 6.1.2, this relatively simple type of neural network serves as a good starting point. Three different types of neural network models have been deployed, which are very similar to each other but differ in the way the labels are presented to them, leading to different predictions.

Some of the hyperparameters of the neural networks have been determined using a Hyperband optimisation algorithm. Not all hyperparameters could be optimised using this optimisation algorithm because of restrictions on the maximum size of the hyperparameter space. Therefore, the activation function was not determined through hyperparameter tuning and for all nodes the ReLU activation function was chosen. Apart from the tuned hyperparameters and variations in output space dimensionality, all neural networks are identical. All hyperparameters that are consistent across all neural networks are shown in table 6.2. The hyperparameters that are subject to tuning are listed in table 6.3.

The output layer activation function in table 6.2 is linear. This was done because predicting a nominal wake field is a regression problem. Therefore, the output layer needs to be able to produce a wide range of values. If a logistic (sigmoid) activation function were applied, for instance, all outputs would fall within the range (0,1). This activation function is often used in binary classification networks. Furthermore, all neural networks use the combined custom loss function, which combines the weighted MSE and wake fraction MSE custom loss functions, each with a factor of 1.

Table 6.3 shows all neural network hyperparameters that are subject to tuning. A range of layers between 1 and 5 was chosen to ensure the model remains relatively shallow. Deeper networks have a higher capacity to learn complex patterns, but they also have a higher risk of overfitting, especially if the dataset is not large enough. The number of units per layer is a hyperparameter that is determined for each individual layer, allowing layers of variable size to exist throughout the model depth. The possible

Table 6.2: Pre-determined hyperparameters across all feed-forward neural networks.

Hyperparameter	Value
Input dimensionality D	28
Output layer activation function	Linear
Loss function	Combined custom
Optimiser	Adam

Table 6.3: Feed-forward neural network hyperparameters to be tuned.

Hyperparameter	Possible values	Sampling method
Model depth (no. of layers)	$[1, 5]$	Steps of 1
Units per layer	$[0.5D, 150]$	Steps of 8
Dropout	$\{0, 0.025\}$	Choice
Learning rate	$[1 \cdot 10^{-6}, 1 \cdot 10^{-4}]$	Log sampling

values range between half the dimensionality (rounded off to 16) and 150, which is approximately five times the dimensionality. To prevent the hyperparameter space from becoming too large, the number of units per layer was sampled at intervals of 8.

Dropout is a regularisation measure that randomly removes a set percentage of nodes from the neural network. The philosophy behind this approach is that by simplifying the model, the chance of overfitting is reduced. The dropout hyperparameter is a choice: there is either no dropout, or a random dropout of 2.5% of all nodes. Lastly, the learning rate was set to be sampled from a logarithmic scale between minimum and maximum values of $1*10^{-6}$ and $1*10^{-4}$, respectively.

The three types of feed-forward neural networks considered in this study are: a neural network without discrete cosine transform (DCT), and two networks with DCT - one with and one without DCT coefficient standardisation. The rationale behind and the properties specific to these three types of feed-forward neural networks are as follows. Firstly, the neural network without DCT serves as the fundamental starting point of this project. The performance of this model will be used as a baseline for comparison with all other models. The output dimensionality of this network is 95, as all labelled wake fields are provided in the range of $[0, 180]$ degrees with an interval of 10 degrees. The 19 resulting angles are given at 5 different radii.

Performing a discrete cosine transform (DCT) on the wake labels was done to evaluate the effect of reducing output dimensionality on the model's performance, at the cost of introducing an error between the actual wake field graph and the DCT-transformed wake field graph. After experimenting with different numbers of initial coefficients, the first 12 DCT coefficients were selected as a compromise between output dimensionality reduction and transformation loss. This leads to an output dimensionality of 60. Initially, an out-of-the-box mean squared error loss function was applied to the DCT-transformed wake fields. However, as this loss function lacks a clear relation to physical practice, it was decided to develop a custom loss function in which an inverse DCT transformation is performed before calculating the loss based on actual velocities.

Another choice regarding the DCT-transformed FNN was to standardise the labels, consisting of DCT coefficients. Because the smallest DCT coefficients are two to three orders of magnitude smaller than the largest ones, the rationale behind standardising the DCT coefficients was to reduce the range of DCT coefficient values, thereby preventing the gradients from exploding or vanishing during backpropagation. The standardised-DCT feed-forward neural network was trained alongside the non-standardised one, enabling relative comparison.

6.4. Ensemble Model Tuning and Training

This section describes the tuning and training setup of the ensemble model. The rationale for creating an ensemble of two sub-models is to improve prediction accuracy by reducing the output dimensionality of each sub-model. By having each sub-model focus on predicting a smaller subset of coefficients

more precisely, rather than attempting to predict a large number of coefficients with less accuracy, it was aimed to enhance the overall performance of the ensemble.

The underlying trade-off being made is between output dimensionality and generalisability. While reducing the output dimensionality of the sub-models can be beneficial, taking this reduction to the extreme can stop the model from being able to generalise. This is because the model may no longer capture the underlying relationships within the predictions. For instance, this issue was observed in the single-point prediction example described in section 6.2. In section 6.4.1, the development of the two sub-models is discussed. Finally, the tuning and training of the meta-model are described in section 6.4.2.

6.4.1. Sub-Models and Out-of-Fold Predictions

It was decided to train an ensemble model consisting of two sub-models, each predicting a different subset of the 12 DCT coefficients. After inspecting figure C.7 in appendix C, which shows that the first few DCT coefficients contain the majority of the energy or information about the original curve, the decision was made to have the first sub-model predict the first four coefficients and the second sub-model predict the remaining eight. Together with the over-coupling meta-model, these sub-models form a variation on a stacking ensemble model.

The two sub-models make use of the out-of-the-box MSE loss function, as their outputs do not make physical sense. Their hyperparameters have been tuned using the exact same configuration as the feed-forward neural network with DCT-transformed labels without standardisation model described in section 6.3. The only difference between the models is the output dimensionality: the first sub-model has an output dimensionality of 20 consisting of 4 DCT coefficients provided at 5 different radii, and the second sub-model has an output dimensionality of 40, which is the product of 8 coefficients and 5 radii.

When training the sub-models, it is essential to ensure that the training data for each sub-model does not include information from the “main” validation set. This implies that the complete dataset for sub-models 1 and 2 should be of the same size as the meta-model’s training set. In other words, there should be as many predictions from sub-models 1 and 2 as there are samples. This can be achieved by using k-fold cross-validation, combining the out-of-fold predictions to match the dataset size.

In this case, by using 8-fold cross-validation, each sub-model is trained on 7 folds and makes predictions on the remaining fold. This process is repeated for all 8 folds, ensuring that each data point is used for both training and validation without overlap. All out-of-fold predictions are combined into the final set of predictions made by sub-model 1 and sub-model 2. These two final sets of predictions are, in turn, combined to form the “meta-features”, or the features that serve as input into the meta-model. The process of making out-of-fold predictions based on an 8-fold cross-validation for both sub-models 1 and 2 is visualised in figure 6.9.

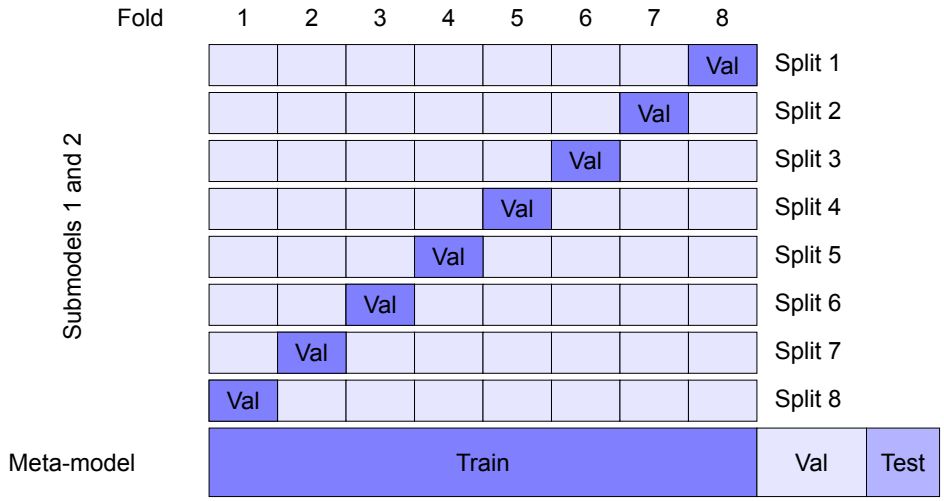


Figure 6.9: Schematic overview of the out-of-fold predicting procedure. Note that the 8-fold cross-validation takes place twice, to generate predictions for both submodel 1 and submodel 2.

6.4.2. Meta Model

The outputs from the two sub-models serve as inputs ("meta-features") to the meta model. These meta-features do not necessarily have to make physical sense, although in this case they represent the first four and last eight DCT coefficients, respectively. The dimensionality of the feature space is 12, as there are 12 intermediate DCT coefficients that serve as input to the meta model. The output space is also 12-dimensional. The hyperparameters of the meta model are to be tuned using the same Hyperband tuning algorithm as used for the feed-forward neural networks. The hyperparameters to be tuned, together with their possible values or ranges, are shown in table 6.4.

Table 6.4: Ensemble method meta model hyperparameters to be tuned.

Hyperparameter	Possible values	Sampling method
Model depth (no. of layers)	$[1, 8]$	Steps of 1
Units per layer	$[8, 160]$	Steps of 8
Dropout	$\{0, 0.025\}$	Choice
Learning rate	$[1 \cdot 10^{-6}, 1 \cdot 10^{-4}]$	Log sampling

6.5. Long Short-Term Memory Model Tuning and Training

The final type of neural network architecture employed in this project was a long short-term memory (LSTM) recurrent neural network. While an LSTM architecture is mainly used for data with a temporal sequential relationship, in this case, it was used to make predictions based on a spatial sequential relationship. To achieve this, some modifications were made to the way input data was fed into the network. Additionally, the network architecture differs from the feed-forward neural networks used in the other architectures.

Originally, the label matrices `Y_train`, `Y_val`, and `Y_test` were of shape (n, radii, angles), where n, radii, and angles denote the number of samples in a set, and the number of different radii and angles at which the wake field velocities are given, respectively. The three sets were transposed to be of shape (n, angles, radii), allowing for easy iteration over the different spatial variables.

The input data have no natural temporal relationship, although the LSTM requires an input for every entry in the sequence. Therefore, all input matrices `X_train`, `X_val`, and `X_test` were repeated 19 times, transforming them into 3D matrices. The physical interpretation of this is that the input features do not change over the angles of the wake field.

An example of a single LSTM unit was explained in section 3.3.3. In an LSTM recurrent neural network, multiple LSTM units might be present, distributed over several layers, similar to a regular neural network. When multiple LSTM units are distributed over multiple layers, the output sequence from the first layer serves as input to the second one. When units are placed in parallel, so in the same layer, they synchronously perform calculations on the input sequence, and a linear combination of their predicted output sequences is fed to all units in the following layer.

The LSTM units in the penultimate layer of the network are configured to output the entire predicted sequence instead of only the final value, ensuring that a complete wakefield is produced. This setting is disabled by default, as in other LSTM applications, the user might only be interested in the final value. The complete sequences are then passed to the final layer in the LSTM network, which is a TimeDistributed Dense layer. This is a special type of layer in Keras that applies the same Dense layer, using a linear activation function (suitable for regression problems), to all entries in the sequence.

Some of the hyperparameters of the LSTM network used in this project have been tuned using the Hyperband optimiser. The pre-determined hyperparameters can be found in table 6.5. All tuned hyperparameters, together with their possible values, are shown in table 6.6.

Table 6.5: Pre-determined hyperparameters for the LSTM network.

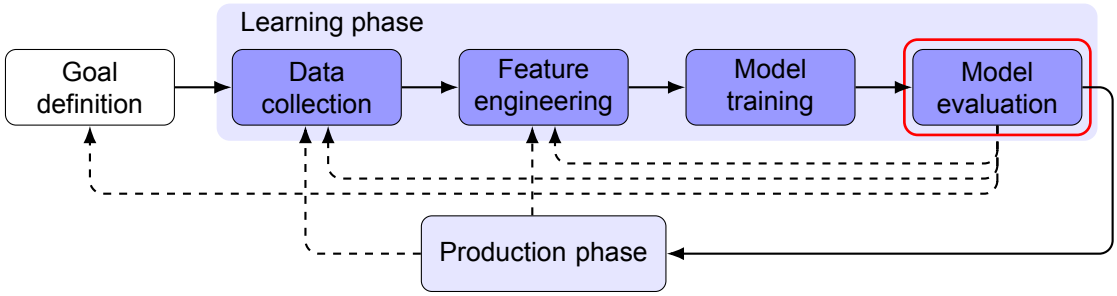
Hyperparameter	Value
Input dimensionality D	28*19
Output dimensionality	5*19
Output layer activation function	Linear
Loss function	Combined custom
Optimiser	Adam

Table 6.6: LSTM model hyperparameters to be tuned.

Hyperparameter	Possible values	Sampling method
Model depth (no. of layers)	$\{1, 2\}$	Choice
Units per layer	$\{16, 23, 64, 128, 256, 512\}$	Choice
Learning rate	$[1 \cdot 10^{-6}, 1 \cdot 10^{-4}]$	Log sampling

7

Model Evaluation



The fourth step in the machine learning project lifecycle, and the final one to be discussed in this thesis report, consists of a model evaluation. After this important step, the performance of all machine learning models will be analysed and quantified. Based on these findings, a conclusion to this thesis report can be drawn by answering the main research question.

This chapter presents an evaluation of all trained models, as well as an investigation of the ability of all models to generalise to unseen data. Based on this ability, the models will be scored relatively to each other. Furthermore, a feature importance study will be explained and presented, providing insight in the relative contribution of every individual feature to the model's performance. Lastly, some predictions will be subjected to a cavitation check and compared with their respective labels in terms of predicted cavitation behaviour.

Sections 7.1, 7.2, and 7.3 will present evaluations of the tuned and trained feed-forward neural networks, ensemble method, and long short-term memory recurrent neural network, respectively. For readability, only the actual-vs-predicted, residual and probability plots for every model will be given in this chapter. For all training metrics, including all epochs-vs-loss graphs, the reader is referred to appendix E.

The results of the subsampling validation to assess generalisability will be discussed in section 7.4. The feature importance study will be elaborated on in section 7.5, and finally, section 7.6 will present the results of the cavitation check.

7.1. Feed-Forward Neural Network Evaluation

The first three models to be evaluated are the feed-forward neural networks without DCT, with DCT, and with DCT and coefficient standardisation, respectively. Their tuned hyperparameters and evaluation will be presented in sections 7.1.1, 7.1.2, and 7.1.3.

7.1.1. Feed-Forward Neural Network Without DCT

The hyperparameters of the feed-forward neural network without discrete cosine transform that have been tuned by the Hyperband optimiser, are presented in table 7.1. The metrics obtained after training are shown in table 7.2. For the development of these metrics over the number of epochs during training, the reader is referred to appendix E. There, in figure E.1, it can be seen that the model has converged after approximately 40 epochs.

In table 7.1, it can be seen that the model depth was optimised to five layers. Despite this being at an extreme, it was decided not to extend the hyperparameter space for another round of optimisation. This decision was made to keep the model relatively shallow to prevent overfitting, and because in the five optimal sets of hyperparameters, lower model depths were also found, indicating that shallower models can also yield optimal results.

Table 7.1: Tuned hyperparameters for the feed-forward neural network.

Hyperparameter	Value
Model depth	5
Units in layer 1	126
Units in layer 2	134
Units in layer 3	86
Units in layer 4	86
Units in layer 5	86
Dropout	0
Learning rate	$7.647 \cdot 10^{-5}$

Table 7.2: Training metrics for the feed-forward neural network.

Metric	Value
Validation loss	0.0055
Validation MSE	0.0126
Validation MAPE	13.79

Figure 7.1 presents a comparison plot of a prediction made by the FNN. All predictions generated by the FNN are shown in appendix F.1. It can be observed that the predictions are quite rough, and not as smooth as in most labels. This is due to the fact that the feed-forward neural network without discrete cosine transform directly outputs non-dimensionalised velocities, rather than DCT coefficients that allow for the reconstruction of a smooth graph at every radius. Upon examining all test set predictions made by this model in Appendix F.1, it appears that the feed-forward neural network is capable of recognising general patterns present across most wake fields: a reduction in velocities towards the inner radii, and a wake peak in the top half plane. However, aside from the rather rough distribution of velocities over the wake field, an important observation is that, in general, detail is lacking in the predictions.

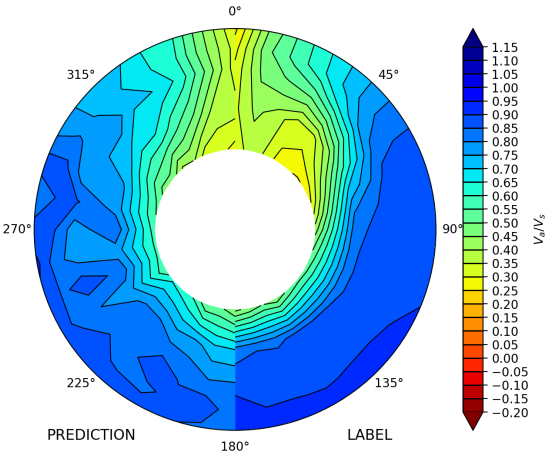
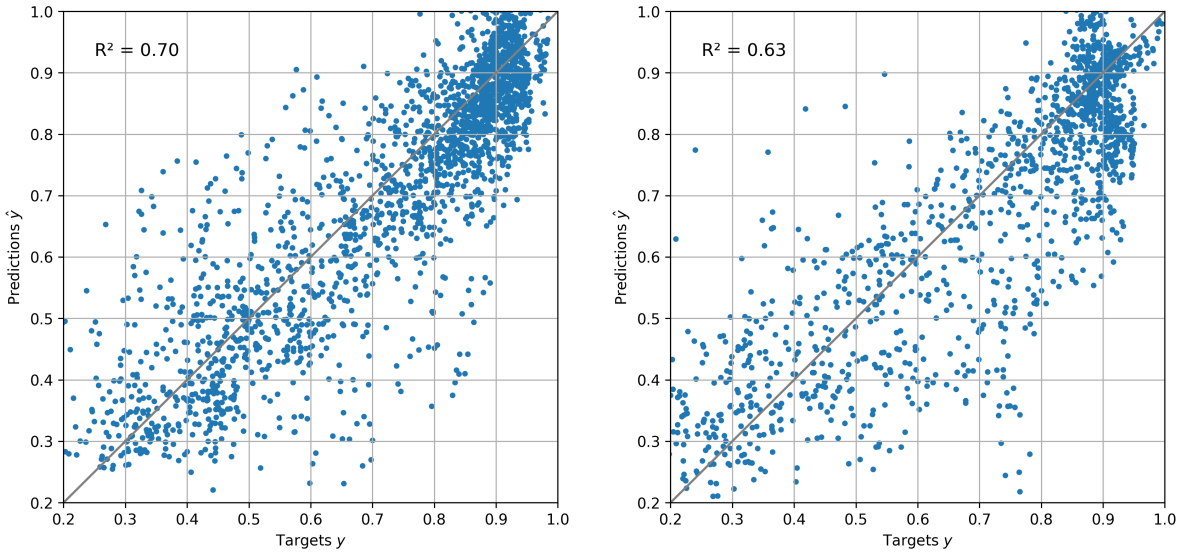


Figure 7.1: Comparison plot of a prediction of the feed-forward neural network.

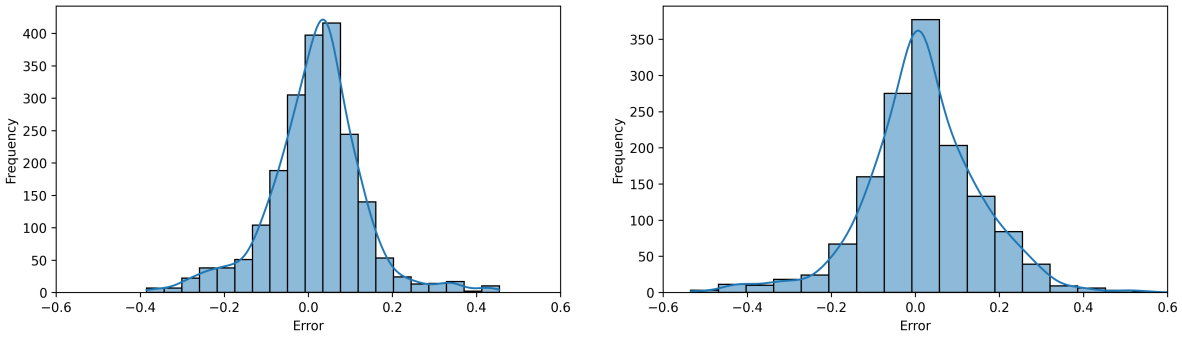
This observation is reflected in the actual-vs-predicted plots presented in figure 7.2. It can be seen that there is a medium linear correlation between targets y and predictions $\hat{y}$, which is also indicated by an R^2-value of 0.70 on the validation set and 0.63 on the test set. The dense clusters in the top right corners of both graphs represent the large part of the wake fields where the axial velocity is relatively close to the ship speed. Whether or not the decrease in correlation of the test set with respect to the validation set is due to overfitting of the model to the validation set, can not be stated with certainty based on the two graphs in figure 7.2. The reason for this is that the validation and test sets are so small (22 and 15 samples, respectively), that individual, particularly “hard-to-predict” samples may have a high influence on the model performance. Therefore, an assessment of which models might be overfitted to the validation set will be made after all different models have been discussed and they can be compared on a relative basis.



(a) Validation set **(b)** Test set

Figure 7.2: Actual-vs-predicted plots for the feed-forward neural network model.

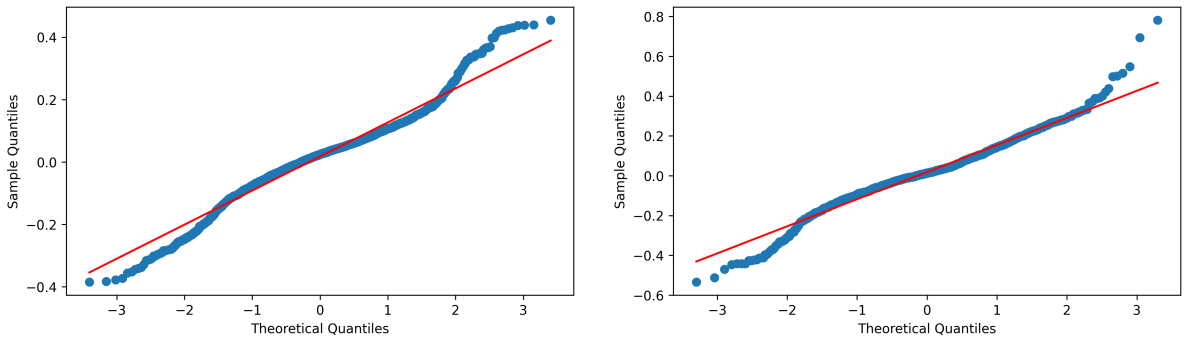
Figure 7.3 presents plots of the model's residuals of predictions for both the validation and test sets. These residual plots show no significant bias: both plots are roughly symmetrical around zero and appear to follow a normal distribution. In figure 7.4, the probability plots related to the residual plots are shown. It can be observed that both residual plots deviate from normality in both tails, although the residuals of both the validation and test sets are approximately normally distributed between residuals of -0.2 and +0.2. Specifically, the probability plots indicate heavy tails on both sides of both residual plots, suggesting that the extreme values in the set of residuals are more extreme than would be expected with a normal distribution.



(a) Validation set residual plot **(b)** Test set residual plot

Figure 7.3: Residual plots for the feed-forward neural network model.

The observations made based on the actual-vs-predicted, residual, and probability plots are reflected numerically in table 7.3. It can be seen in this table that for both the validation and test sets, the means and standard deviations of residuals are approximately equal, although slightly higher for the test set. The Shapiro-Wilk test statistics and corresponding p-values confirm that although the residual plots roughly follow a normal distribution (indicated by a test statistic close to one), the null hypothesis of normality should be rejected (indicated by a very low p-value). This suggests that while the residuals appear to be normally distributed, the deviations in the tails are significant enough to reject the assumption of normality.



(a) Validation set probability plot

(b) Test set probability plot

Figure 7.4: Probability plots for the feed-forward neural network model.

Table 7.3: Performance metrics for the feed-forward neural network.

Metric	Validation	Test
Mean of residuals	0.01750	0.01807
Standard deviation of residuals	0.1110	0.1380
Shapiro-Wilk test statistic	0.9651	0.9710
P-value	$4.6 \cdot 10^{-22}$	$2.6 \cdot 10^{-16}$
R^2	0.70	0.63

7.1.2. Feed-Forward Neural Network With DCT

Similar to the evaluation of the feed-forward neural network without DCT, the tuned hyperparameters and training metrics of the FNN with discrete cosine transformed labels are shown in tables 7.4 and 7.5. The development of the training metrics over the number of epochs can be found in appendix E. These metrics show regular behaviour on the custom combined and mean squared error loss, as well as on the the Pearson product-moment correlation coefficient development. However, the mean average percentage error loss metric shows strange behaviour. This can be explained by the fact that the FNN with DCT predicts non-standardised DCT coefficients. While the custom combined loss function performs inverse DCT after every epoch to obtain a loss that is comparable with other networks, this does not happen in the MAPE calculation, leading to division by small numbers and irregular behaviour.

Table 7.4: Tuned hyperparameters for the feed-forward neural network using DCT.

Hyperparameter	Value
Model depth	5
Units in layer 1	134
Units in layer 2	102
Units in layer 3	142
Units in layer 4	14
Units in layer 5	118
Dropout	0.025
Learning rate	$5.366 \cdot 10^{-5}$

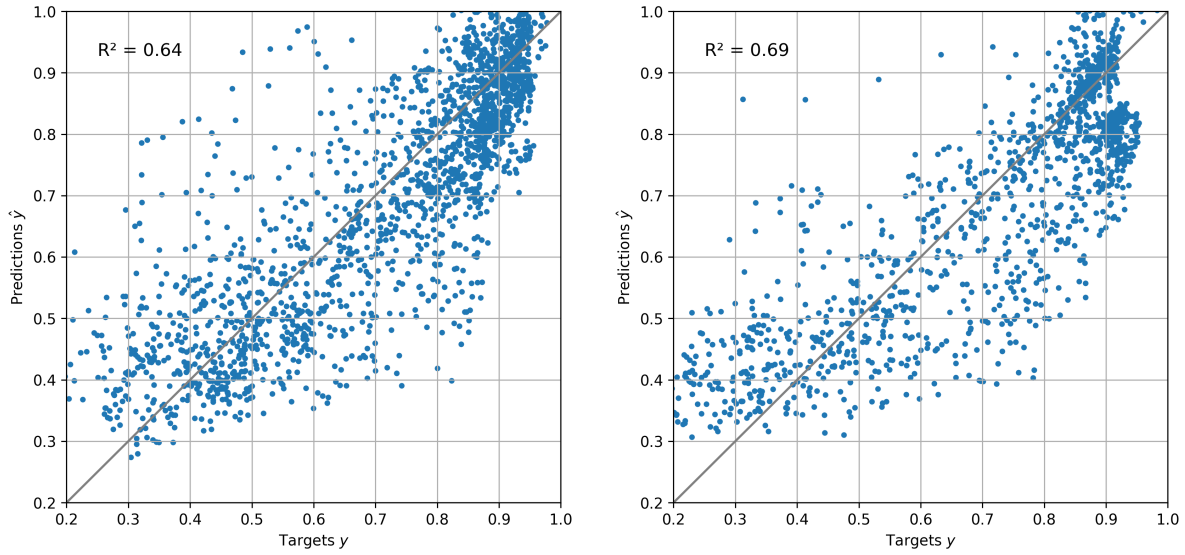
Table 7.5: Training metrics for the feed-forward neural network using DCT.

Metric	Value
Validation loss	0.0088
Validation MSE	0.1130
Validation MAPE	76152

Table 7.4 shows that the model depth has been optimised to five, similar to the FNN without DCT. Notably, there is a ten-fold reduction in the number of units between hidden layers 3 and 4, which could act as a “bottleneck layer”. This might help the network learn a more meaningful representation of the data by compressing it into a lower-dimensional space, forcing it to select the most important features. The dropout rate of 0.025 suggests that the model requires some regularisation to prevent overfitting.

Figure 7.5 presents the actual-vs-predictions plots of the FNN using DCT on the validation and test sets. It can be observed from the coefficient of determination that the model performs better on the test

set than on the validation set. Furthermore, according to the coefficient of determination, the model's performance on the validation set is worse than that of the FNN without DCT, whereas its performance on the test set is slightly better.

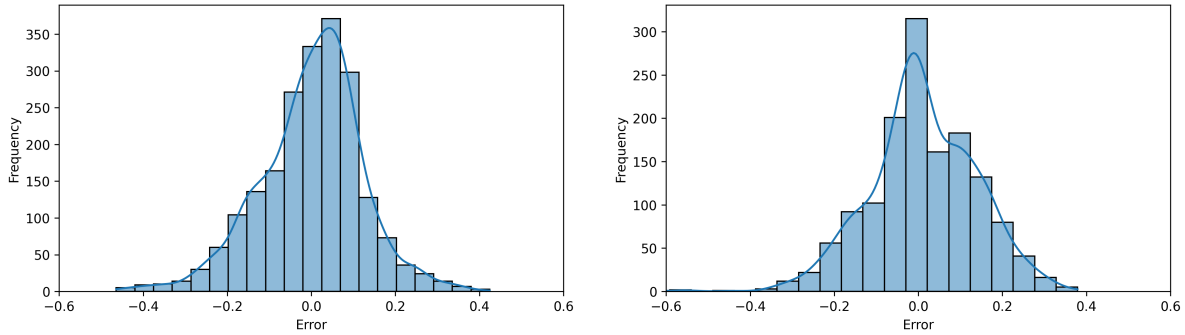


(a) Validation set

(b) Test set

Figure 7.5: Actual-vs-predicted plots for the feed-forward neural network model using DCT.

The residual plot based on the DCT FNN's predictions on the validation set, shown in figure 7.6, reveals a slight left-skewed distribution. This is reflected in figure 7.5a by the fact that more predictions fall above and further away from the diagonal compared to the predictions below it. The test set residual plot has a dip just to the right of the mean but otherwise seems to follow a normal distribution relatively well. Figure 7.7 and table 7.6 reflect the observations in the actual-vs-predicted and residual plots, although the low mean of residuals suggests that the validation set residuals are not left-skewed.

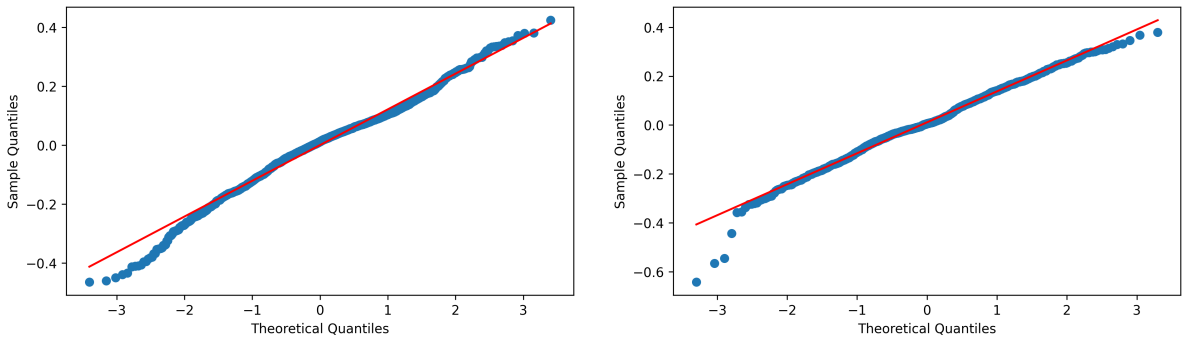


(a) Validation set residual plot

(b) Test set residual plot

Figure 7.6: Residual plots for the feed-forward neural network model using DCT.

Finally, a comparison plot of a prediction made by the feed-forward neural network using DCT can be seen in figure 7.8a. Compared to figure 7.1, it can be observed that the representation of the wake field using DCT coefficients inherently causes the wake field graph to behave more truthfully, without scattered velocity differences. However, the same limitation applies to the feed-forward neural network using DCT as to the one without DCT: the predictions made are merely scaled-and-shifted "averages", slightly adapted based on the input values and unable to capture unique details in wake fields.



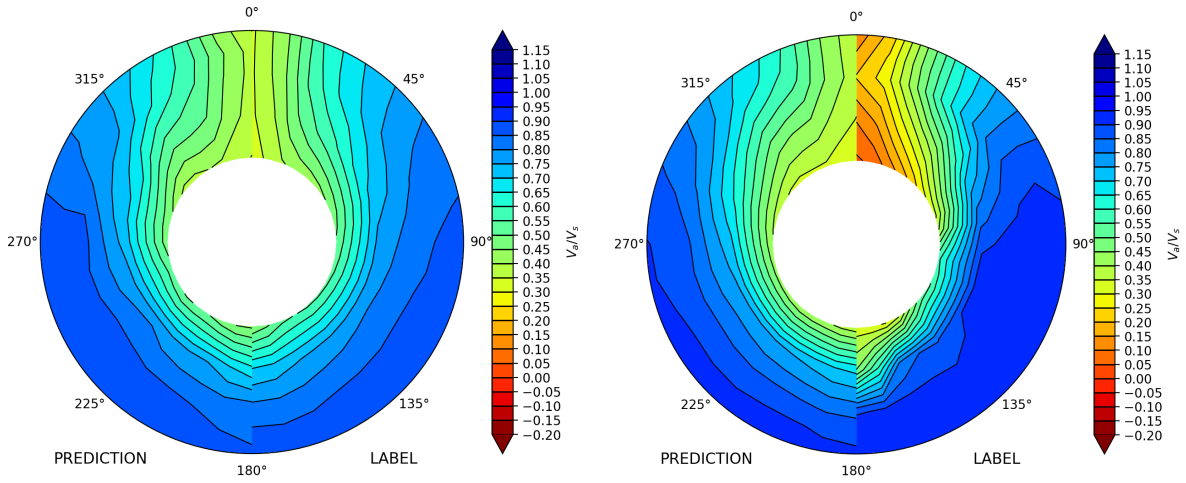
(a) Validation set probability plot

(b) Test set probability plot

Figure 7.7: Probability plots for the feed-forward neural network model using DCT.

Table 7.6: Performance metrics for the feed-forward neural network using DCT.

Metric	Validation	Test
Mean of residuals	0.000684	0.01108
Standard deviation of residuals	0.1220	0.1271
Shapiro-Wilk test statistic	0.9856	0.9906
P-value	$1.1 \cdot 10^{-13}$	$6.7 \cdot 10^{-8}$
R^2	0.64	0.69



(a) FNN + DCT

(b) FNN + DCT + standardisation

Figure 7.8: Comparison plots of predictions of the feed-forward neural networks using DCT.

7.1.3. Feed-Forward Neural Network With DCT and Standardisation

The final regular feed-forward neural network model that has been tuned and trained, and will therefore be evaluated, is the feed-forward neural network with DCT and coefficient standardisation. This model leverages both the discrete cosine transform and standardisation of the its coefficients to potentially enhance the performance of the model. Table 7.7 provides the tuned hyperparameters for this model. Notably, the model depth has been reduced to two layers, which suggests a simpler architecture compared to the previous models. The first layer contains 126 units, while the second layer has 30 units. This indicates some compression, although not as much as the FNN+DCT model. The dropout rate remains at 0.025, indicating that some regularisation is deemed needed.

Table 7.8 presents the training metrics for the feed-forward neural network using DCT and coefficient standardisation. The development of these metrics over the epochs is illustrated in figure E.3. The final

validation loss is 0.0053, which is comparable to the other models. The progression of the MAPE and PPMCC metrics over the epochs appears unusual. This behaviour is likely due to the standardisation of the DCT coefficients, which brings all coefficients close to zero.

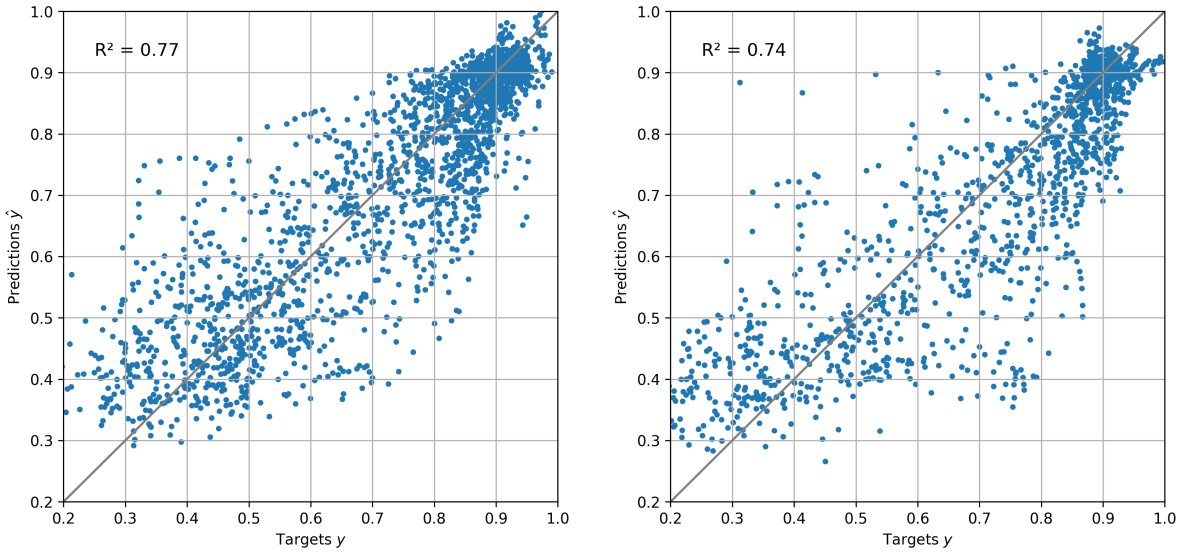
Table 7.7: Tuned hyperparameters for the feed-forward neural network using DCT and coefficient standardisation.

Hyperparameter	Value
Model depth	2
Units in layer 1	126
Units in layer 2	30
Dropout	0.025
Learning rate	$9.445 \cdot 10^{-6}$

Table 7.8: Tuned HPs for the FNN using DCT and coefficient standardisation.

Metric	Value
Validation loss	0.0053
Validation MSE	0.7543
Validation MAPE	206

Standardising the DCT coefficient seems beneficial to model performance; in figure 7.9 it can be seen that the R^2-values for the validation and test sets are 0.77 and 0.74, respectively. A comparison plot of a prediction made by the model is shown in figure 7.8b. It can be observed that just like the FNN + DCT model, this model predicts velocities in the wake field as smooth curves for every radius, leading to smooth wake field graphs.



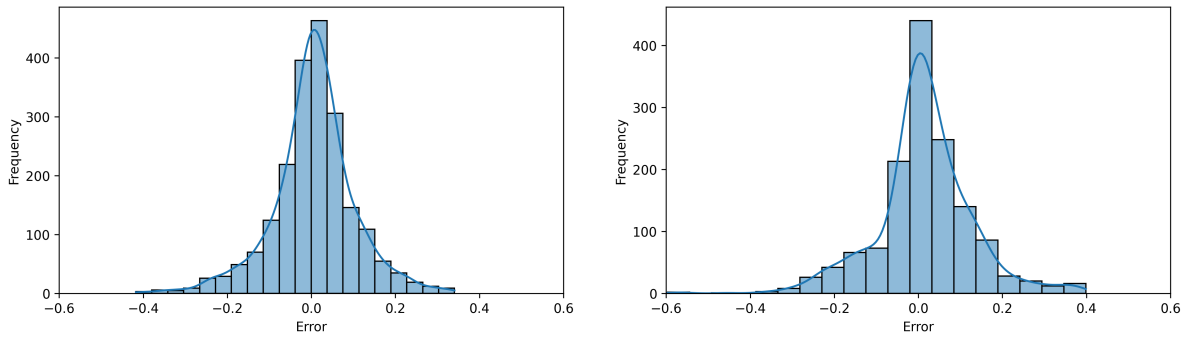
(a) Validation set **(b)** Test set

Figure 7.9: Actual-vs-predicted plots for the feed-forward neural network model using DCT and coefficient standardisation.

The residual and probability plots of the feed-forward neural network using DCT and coefficient standardisation (figures 7.10 and 7.11) show no notable differences compared to the other discussed models: they mostly follow a normal distribution, except for the tails. This is also reflected in the performance metrics in table 7.9. However, what can be observed is that the mean and standard deviation of residuals, both of the validation and test sets, is the lowest of all feed-forward neural network models.

Table 7.9: Performance metrics for the feed-forward neural network using DCT and coefficient standardisation.

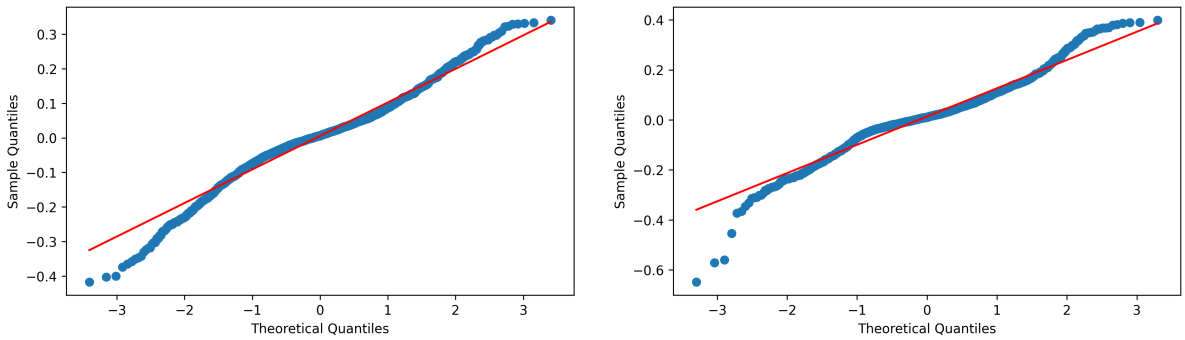
Metric	Validation	Test
Mean of residuals	0.005657	0.0134
Standard deviation of residuals	0.09844	0.1154
Shapiro-Wilk test statistic	0.9705	0.9565
P-value	$2.552 \cdot 10^{-20}$	$3.505 \cdot 10^{-20}$
R^2	0.77	0.74



(a) Validation set residual plot

(b) Test set residual plot

Figure 7.10: Residual plots for the feed-forward neural network model using DCT and coefficient standardisation.



(a) Validation set probability plot

(b) Test set probability plot

Figure 7.11: Probability plots for the feed-forward neural network model using DCT and coefficient standardisation.

7.2. Ensemble Model Evaluation

The following model that will be evaluated is the ensemble model consisting of two neural network sub-models and one meta-model. Sub-models 1 and 2 predict the first 4 and remaining 8 DCT coefficients, respectively, after which the DCT coefficients are combined by a meta-model.

The two sub-models have been tuned individually. Their tuned hyperparameters can be seen in tables 7.10 and 7.11, respectively. The tuned hyperparameters of the meta-model are presented in table 7.12, and the training metrics of the meta-model are shown in table 7.13.

Table 7.10: Tuned hyperparameters for the ensemble sub-model 1.

Hyperparameter	Value
Model depth	5
Units in layer 1	150
Units in layer 2	142
Units in layer 3	134
Units in layer 4	54
Units in layer 5	94
Dropout	0.025
Learning rate	$1 \cdot 10^{-4}$

Table 7.11: Tuned hyperparamters for the ensemble sub-model 2.

Hyperparameter	Value
Model depth	52
Units in layer 1	22
Units in layer 2	126
Units in layer 3	134
Units in layer 4	134
Units in layer 5	86
Dropout	0.025
Learning rate	$1 \cdot 10^{-4}$

When examining the validation loss, it can be seen that the loss of the ensemble meta-model is almost four times as high as that from the best-performing feed-forward neural network. The effect of this is clearly visible in figure 7.12, which shows that the performance of the ensemble method, expressed in R^2, is subpar compared to the other models, especially for the validation set. Data points appear to

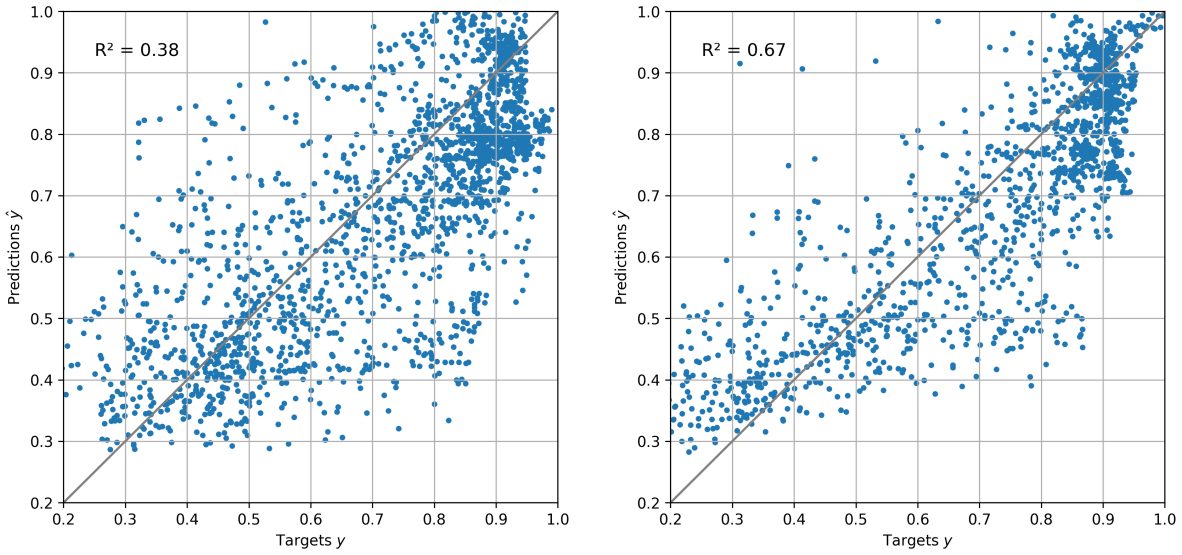
Table 7.12: Tuned hyperparameters for the ensemble meta-model.

Hyperparameter	Value
Model depth	7
Units in layer 1	56
Units in layer 2	128
Units in layer 3	120
Units in layer 4	72
Units in layer 5	16
Units in layer 6	8
Units in layer 7	144
Dropout	0.025
Learning rate	$8.25 \cdot 10^{-5}$

Table 7.13: Training metrics for the ensemble model.

Metric	Value
Validation loss	0.020
Validation MSE	0.20
Validation MAPE	87734

be randomly scattered across the actual-vs-predicted plot. For both the validation and test sets, the dense cluster in the top right corner seems to be asymmetrically located with respect to the diagonal.



(a) Validation set

(b) Test set

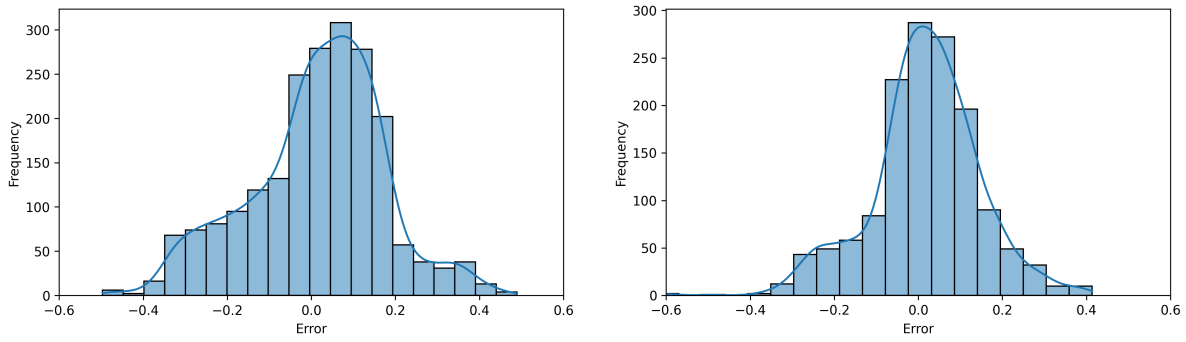
Figure 7.12: Actual-vs-predicted plots for the ensemble model.

A possible explanation for this is that although the first four and final eight DCT coefficients are "stacked together" by the meta-model, this implementation of an ensemble method can not be considered a pure stacking model. A pure stacking ensemble would combine outputs of various source and quality (e.g. models trained on different datasets, using different features, or using different machine learning algorithms) in an aim to obtain a meta-model that performs better than any of its sub-models. Therefore, whole models are stacked in a stacking model, rather than subsets of the desired final outputs, such as the first n DCT coefficients. By "cutting up" the 12 DCT coefficients that were to be predicted for each of the five radii that together define the wake field, the physical meaning of the labels was diminished for both sub-models, making it harder for the two sub-models to identify underlying patterns that connect certain inputs to the labelled values.

As can be seen in figurs 7.13, the residual plots (especially the residual plot based on the validation set) have heavy tails. Because of this clear non-normality, no probability plots are presented for the ensemble method. Both the distributions of the validation and test set residuals mainly deviate from normality at both tails. Furthermore, both distributions are slightly left-skewed. The performance of the ensemble model is quantified using the performance metrics in table 7.14. Apart from the low

R^2-values, especially the high standard deviation of residuals for both the validation and test sets is indicative of the ensemble model's subpar performance.

The reader is referred to appendix F, particularly section F.4, to get an overview of the practical effect of this relatively low performance. It can be observed that the ensemble model has converged to a more-or-less general "average shape". Although other models, such as the DCT transformed feed-forward neural networks, exhibit this behaviour as well, this effect seems to be stronger for the ensemble method.



(a) Validation set residual plot

(b) Test set residual plot

Figure 7.13: Residual plots for the ensemble model.

Table 7.14: Performance metrics for the ensemble model.

Metric	Validation	Test
Mean of residuals	0.01435	0.01565
Standard deviation of residuals	0.1604	0.1307
Shapiro-Wilk test statistic	0.9829	0.9782
P-value	$3.7 \cdot 10^{-15}$	$7.1 \cdot 10^{-14}$
R^2	0.38	0.67

7.3. Long Short-Term Memory Model Evaluation

The following machine learning model to be evaluated is the recurrent neural network, more specifically the long short-term memory model. The results of the hyperparameter tuning are presented in table 7.15, the LSTM training metrics in table 7.16.

Table 7.15: Tuned hyperparameters for the LSTM model.

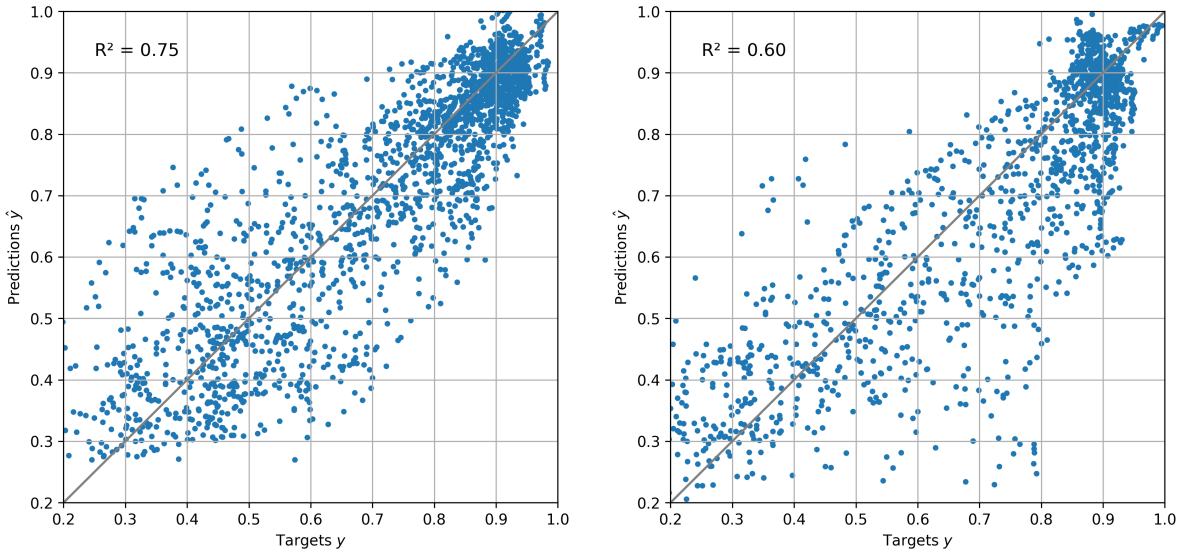
Hyperparameter	Value
Model depth	1
Units in layer 1	512
Learning rate	$9.445 \cdot 10^{-6}$

Table 7.16: Training metrics for the LSTM model.

Metric	Value
Validation loss	0.0047
Validation MSE	0.0104
Validation MAPE	13.3

Figure 7.14 shows the actual-vs-predicted plots for the LSTM model on both the validation and test sets. It can be seen that the validation set plot in figure 7.14a looks quite symmetric around the diagonal, obtaining a R^2 of 0.75, while the predictions on the test set look less coherent, which also translates to a lower R^2 of 0.60.

A first explanation for this big (0.15) difference in R^2 that might come to mind is overfitting of the model to the validation set. This can happen during a long and extensive hyperparameter tuning process, because the performance of every combination of hyperparameters is assessed using the validation set. However, only four hyperparameters (of which the "amount of units in layer 2" hyperparameter was not used in the final tuned model) have been tuned. Furthermore, the hyperparameter tuning

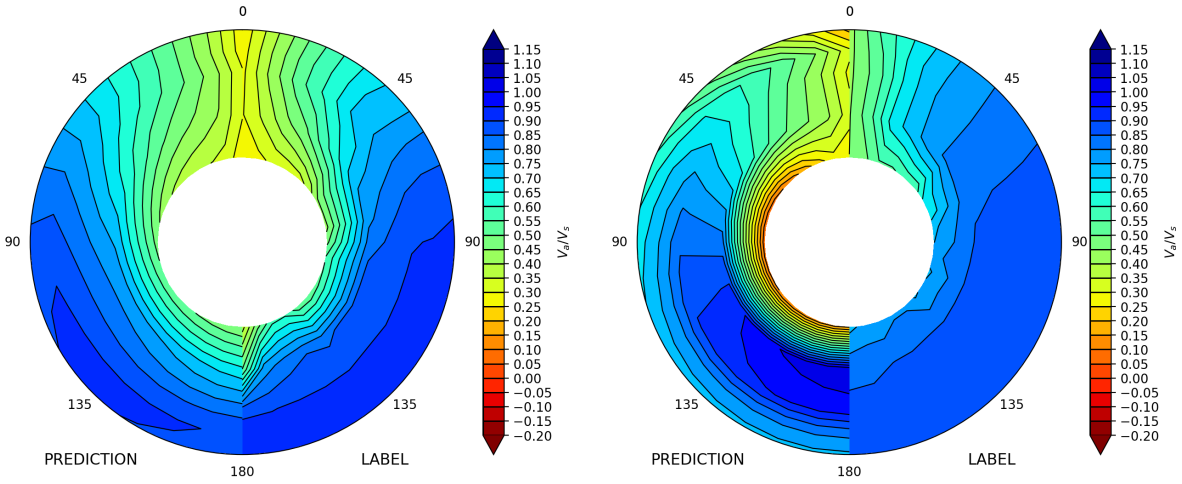


(a) Validation set **(b)** Test set

Figure 7.14: Actual-vs-predicted plots for the LSTM model.

was limited, both in total number of trials and in maximum number of epochs per trial, due to limited computational resources and the relatively long time it takes to train a LSTM model. Therefore, another explanation for the difference in performance had to be found.

When examining the predictions made by the LSTM model, as presented in section F.5, the fundamentally different approach of a recurrent neural network compared to a feed-forward neural network becomes visible. As the velocities in the wake field are sequentially predicted for every angle between 0 and 180 degrees, the model inherently takes surrounding wake field velocities into account. This informativeness-by-design can contribute to more accurate predictions, as demonstrated by the example shown in figure 7.15a. However, if the LSTM model exhibits erratic behaviour at the initial angles, these errors are likely to propagate through all following angles of prediction, with errors sometimes even “exploding” over the following angles. An example of such a prediction is illustrated in figure 7.15b. Since “very good” or “very bad” predictions appear to occur without a clear reason, the variation in the R^2-value is likely to be attributed to the sensitivity of the LSTM model to *this specific* training-validation-testing split as well as limited generalisation.



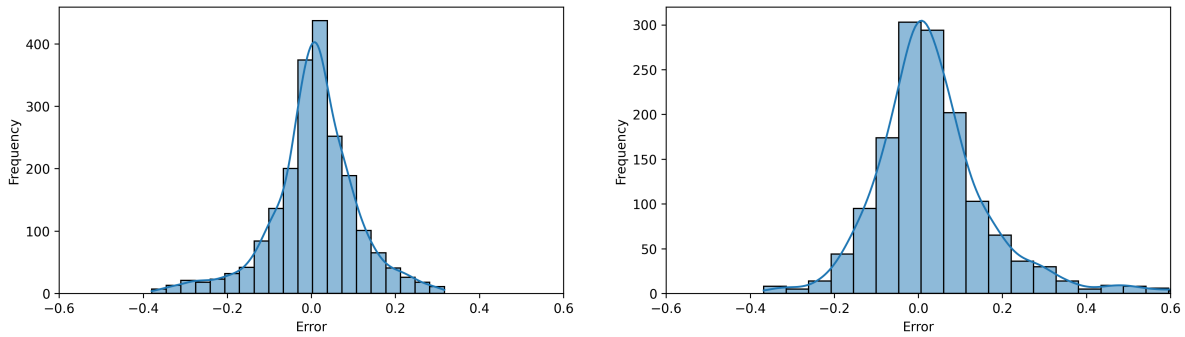
(a) Good prediction **(b)** Bad prediction

Figure 7.15: Example of a good and a bad prediction made by the LSTM model.

The residual plots and corresponding probability plots based on the LSTM model's predictions and targets are shown in figures 7.16 and 7.17. The LSTM model does nog significantly deviate from the other trained models in terms of these plots: a low zero-mean indicates low model bias, especially on the validation set. Furthermore, a roughly normal distribution of errors is observed, except for the tails. Particularly, the validation set residual plot's left and test set residual plot's right tails are heavier than would be expected. Finally, the performance metrics of the LSTM model are presented in table 7.17.

Table 7.17: Performance metrics for the LSTM model.

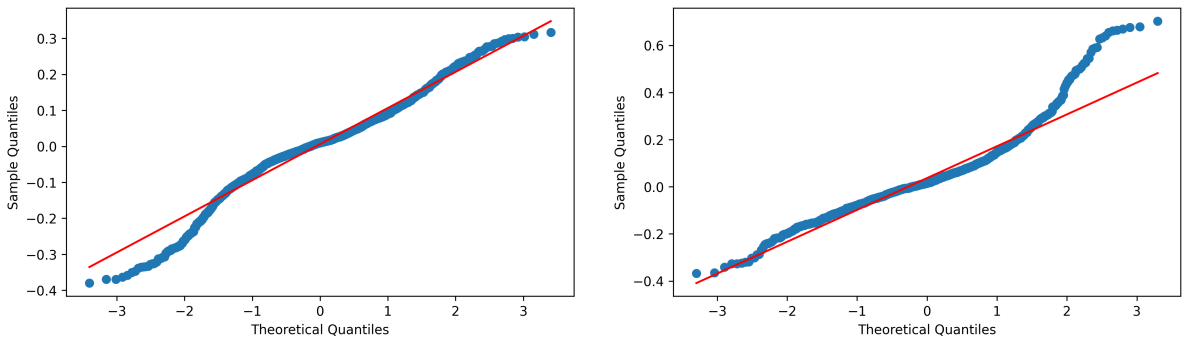
Metric	Validation	Test
Mean of residuals	0.006125	0.03679
Standard deviation of residuals	0.1020	0.1406
Shapiro-Wilk test statistic	0.9656	0.9198
P-value	$6.3 \cdot 10^{-22}$	$7.5 \cdot 10^{-27}$
R^2	0.75	0.60



(a) Validation set residual plot

(b) Test set residual plot

Figure 7.16: Residual plots for the LSTM model.



(a) Validation set probability plot

(b) Test set probability plot

Figure 7.17: Probability plots for the LSTM model.

7.4. Repeated Random Subsampling Validation

After all individual machine learning models had been evaluated, a random repeated subsampling validation was performed. The idea behind a random repeated subsampling validation is that by repeatedly splitting the data into training and validation sets, it can be proven or disproven that a model's performance is not overly dependent on a particular dataset split. In other words, the sensitivity of a model to various splits, and therefore its ability to generalise to unseen data is assessed.

When a model is highly sensitive to a certain split, averaging the model's performance over multiple iterations of splits will provide a more stable and reliable estimate of its performance. High sensitivity to a specific split can indicate that the model is overfitted to the validation or test set. However, it could also be due to issues with the dataset. For instance, when the dataset is relatively small, a random train-validation-test split is more likely to have non-similar statistical characteristics, which can affect the model's performance and is one of the reasons that without a sufficiently large dataset, a model will not be able to generalise.

The repeated random subsampling validation was performed for all models. All models, with tuned hyperparameters, were re-trained eight different times on eight different splits. The performance, measured in validation loss, was recorded for every split and every model. The descriptive statistics of the repeated random subset validation is shown in table 7.18 and visualised in figure 7.18.

Table 7.18: Descriptive statistics of the models' performances on 8 random splits. All values have been multiplied with 10^3 for readability.

values * 10^3	**mean**	**std**	**min**	**50%**	**max**
FNN	6.845	0.928	5.535	6.682	8.189
FNN DCT	8.153	1.594	6.285	8.083	0.672
FNN DCT STD	6.838	1.746	3.809	6.902	9.874
Ensemble	11.81	4.777	7.091	9.845	19.245
LSTM	6.205	1.353	4.597	6.258	8.227

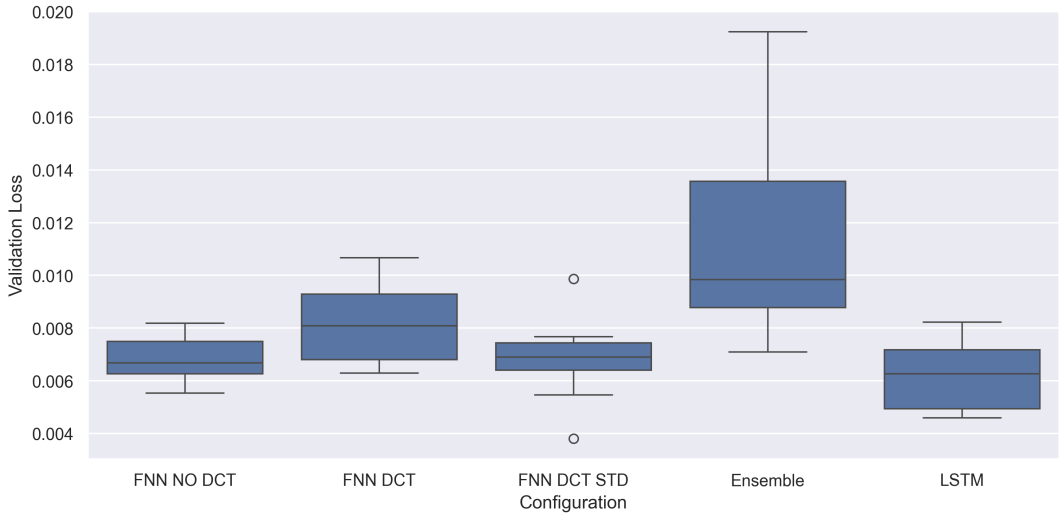


Figure 7.18: Comparison of validation loss distribution for all machine learning models with 8-time repeated random subsampling validation.

From table 7.18, it follows that the model with the lowest mean loss is the LSTM model. This model also performs well in terms of standard deviation, as only the standard deviation of the FNN without DCT is lower. Figure 7.18 shows that although the interquartile range of the FNN+DCT+STD configuration is the lowest, and it ranks second in terms of lowest mean loss, an upper and lower outlier cause its standard deviation to be rather high. The only models that are, based on this validation, not worth further examination are the ensemble method and, to a smaller extent, the DCT FNN. An important

observation is that for all models, the standard deviation is quite high, as the percentage of standard deviation relative to the average loss ranges between 14% for the FNN and 40% for the ensemble method. This is an indication that all models are to some extent sensitive to specific training-validation splits and no single model is able to generalise perfectly.

7.5. Feature Importance Study

To obtain better insight into which features have the highest influence on the performance of the machine learning models, a feature importance study by permutation has been conducted on the feed-forward neural network model. This feature importance study by permutation works by loading the trained model and calculating its baseline performance using the validation set and the custom combined loss function. After determining the baseline performance, an iteration over all features takes place. During each iteration, the input values of one specific feature are permuted randomly. In other words, a random shuffling of that particular feature with values of that same feature for other samples takes place, while the other features' values in the input matrix are kept the same. By comparing the change in performance after permuting each feature iteratively, a relative indication of feature importance is obtained. This whole process is repeated n times to obtain a converged relative feature importance, in this case a value of n of 100 was taken. The results of the feature importance study are presented in figure 7.19.

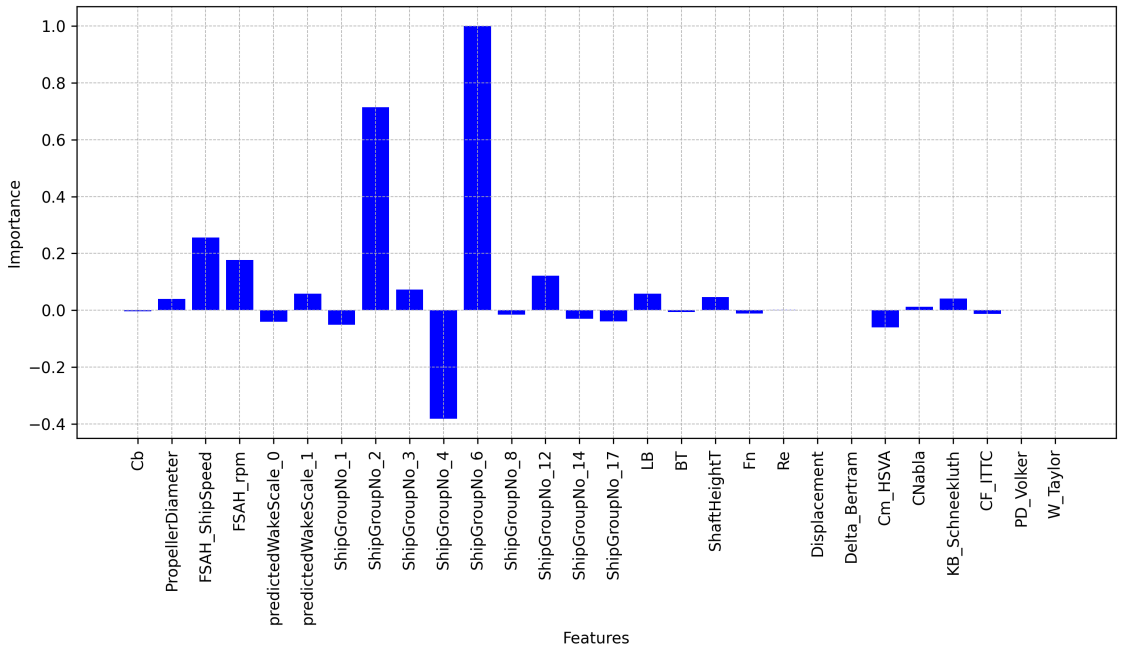


Figure 7.19: Results of the feature importance study using feature permutation.

In figure 7.19 can be seen that all features that are descriptive of the hull geometry generally low. Particularly some of the ShipGroupNo features have high relative importances, while some features have a negative importance.

From a conceptual point of view, the latter observation indicates that some features actually "confuse" the model more than they add value to it. As negative feature importance is not expected in a well-tuned and well-trained machine learning model, it is assumed that these negative importances are the consequence of overfitting of the machine learning model to the training set and high sensitivity to different splits or permuted features. The fact that the features descriptive of the hull geometry (such as ratios or the block coefficient) score relatively low supports the observation that all predicted wake fields seem to be "generalised" representations, lacking specific details. Instead of capturing the underlying phenomena that drive a wake field, the machine learning model acts like a "glorified regressor".

Another explanation for low feature importance could be correlation. When two features are highly correlated, the informative value of one feature that gets lost after permutation is substituted by the

other feature, resulting in a misleading indication of importance. As can be seen in the Taylor wake fraction equation (equation 5.10), and as has been briefly touched upon in section 5.6, the Taylor wake fraction feature, which has a perfect correlation with the block coefficient, has been preserved in the feature space erratically. To assess the effect of this correlation on the feature importance study, the feed-forward neural network has been re-trained using the same training-validation-test split but without the Taylor wake fraction feature. The results can be seen in figure 7.20.

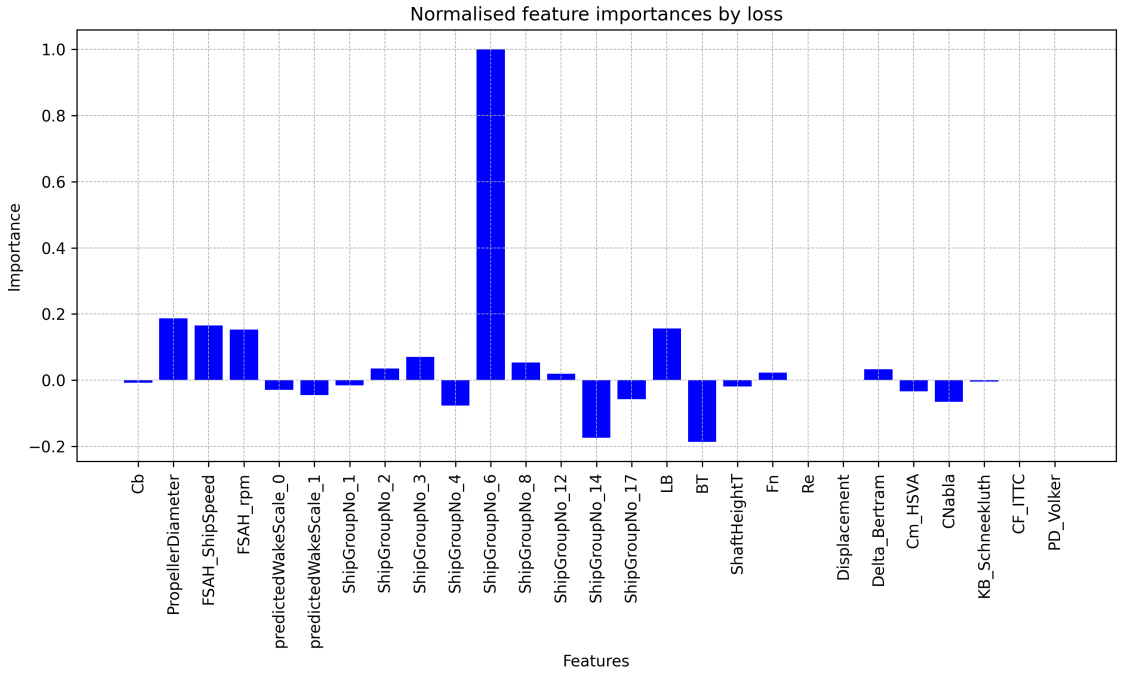


Figure 7.20: Results of the feature importance study using feature permutation, for a FNN without Taylor wake feature.

As can be concluded, no significant difference in importance of the block coefficient feature can be observed. However, this feature importance study based on a newly-trained feed-forward neural network further confirms the assumption that the model has not successfully identified the (complex) underlying hydrodynamic phenomena driving the nominal wake field.

7.6. Mpuf3a Cavitation Check

The final evaluation that has been performed is a cavitation assessment using Mpuf3a. Mpuf3a is a vortex lattice method (VLM) potential flow solver that can be accessed through Wärtsilä Archimedes. It was developed within the Cavitation Consortium lead by prof. S.A. Kinnas [115]. It takes propeller geometry and a nominal wake field as inputs, the latter of which is transformed to an effective wake field using the theoretical method of Huang and Groves. It outputs a cavitation calculation as well as the predicted propeller performance, where the values for thrust and torque are corrected empirically to account for the inviscid assumption of the potential flow solver.

As was mentioned in section 6.1.4, only models predicting axial velocities have been tuned and trained. To investigate the effect of omitting the tangential and radial velocity components and assess if Mpuf3a can be used for cavitation prediction when only axial velocities are provided, Jannicke Gjerdevik of Wärtsilä Norway performed an experiment, comparing calculations based on “complete” wake fields with calculations based on wake fields only featuring axial velocities. The outcomes of the calculations performed for this experiment are presented in appendix G. From these outcomes, it can be concluded that the influence of radial and tangential velocities on the cavitation calculation is noticeable but small enough to justify using only the axial velocities as inputs to Mpuf3a.

The Mpuf3a cavitation check was set up as follows. The FNN using DCT and FNN using DCT and coefficient standardisation models were selected. From those models, two “good” and two “bad” predictions

on the test set were taken. The IDs corresponding to those in appendix F of the specific predictions that have been selected for the Mpuf3a cavitation check are presented in table 7.19.

Table 7.19: IDs of the predictions that have been selected for the Mpuf3a cavitation check.

	FNN DCT	FNN DCT STD
Good predictions	1	7
	9	11
Bad predictions	8	0
	10	13

The selected predictions and their corresponding labels have been turned into wake files in the MARIN format, after which Jannicke Gjerdevik performed a cavitation and performance analysis using Mpuf3a for all different cases at both the Free Sailing Ahead (FSAH) condition and a service condition with lower pitch (to assess possible pressure side cavitation) [116]. Apart from different metrics, the extent and height of cavitation on the blade for the angle with lowest margin against cavitation was visualised.

As a metric for the quality of a wake field prediction, the predicted and labelled lowest margins against cavitation (lowest values of σ) have been compared for all FSAH conditions, and their percentage difference calculated. The results can be seen in table 7.20.

Table 7.20: IDs of the predictions that have been selected for the Mpuf3a cavitation check.

	FNN DCT		FNN DCT STD	
	ID	**diff. σ_{min} [%]**	**ID**	**diff. σ_{min} [%]**
Good predictions	1	-13.20	7	-4.132
	9	-12.11	11	0.8103
Bad predictions	8	-33.93	0	20.29
	10	-25.17	13	-4.040

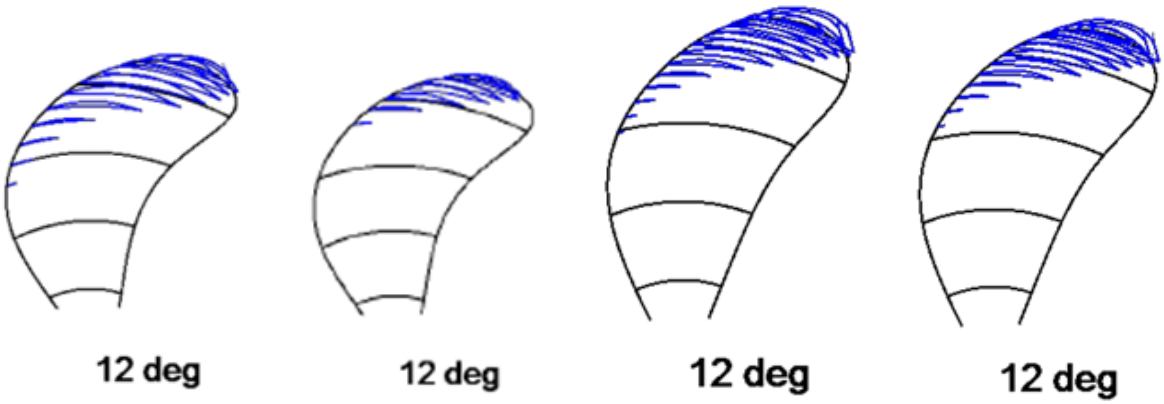
It can be seen that on average the FNN model using DCT and coefficient standardisation has lower differences between predicted and actual lowest margins against cavitation than the FNN model that only uses DCT, which could have been expected given the lower average perfomance of the latter model. Especially all “good predictions” score reasonably well, with differences in σ_{min} between -13.20% and 0.8103%. On average, the absolute difference between predicted and labelled σ_{min} equals 21.1% for the DCT FNN and 7.32% for the DCT FNN using standardisation.

Not only the minimum value of σ is of importance when performing a cavitation assessment. Also the shape and distribution of the cavitation over the propeller blades is of interest. Therefore, comparative visualisations between predictions and labels for the worst performing (ID 10) and best performing (ID 11) wake field in terms of lowest margin against cavitation are shown. Figures 7.21 and 7.22 show the distribution of cavitation volume and distribution of pressure coefficient C_P on the propeller blades at the angle of lowest margin against cavitation.

It can be seen that both the distribution of cavitation volume and pressure coefficient on the blade of the “good prediction” of ID11 is nearly identical. The minor differences in predicted and labelled nominal wake field have resulted in very minor differences in distribution, nearly impossible to distinguish in these figures. The performance of the “bad prediction” by the DCT FNN, ID10, has more distinctions between label and prediction: most notably, the total cavitation volume is clearly higher according to the label-based calculation in comparison with the prediction-based one. Furthermore, some differences in the pressure distribution on the blade can be seen. However, the general shape of cavitation, as well as the angle of lowest margin against cavitation, are quite well predicted, even if the margin against cavitation itself is not predicted accurately.

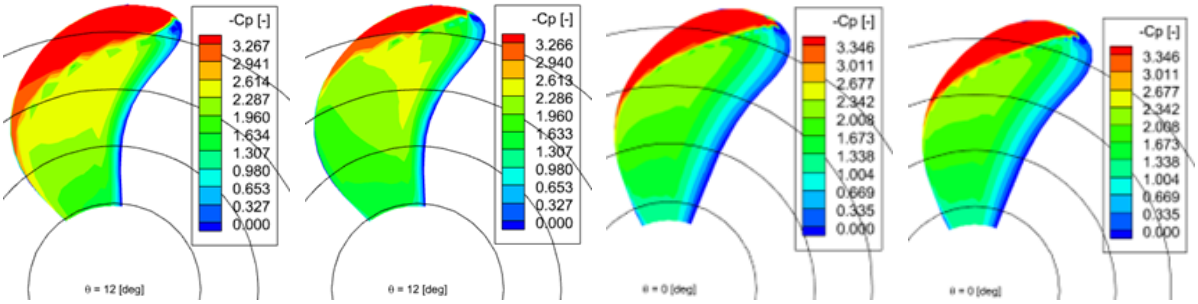
The current way of assessing cavitation behaviour when no nominal wake field is present within Wärtsilä, is to manually look for a project whose hull parameters correspond with that of the project of interest, and copy its nominal wake field to the project of interest for performance and cavitation assessment. Based

on the comparison between calculations based on predicted and labelled wake fields, presented in the Mpuf3a cavitation check report [116], Wärtsilä propeller designers have confirmed that the predicted wake fields serve as a better substitute for a missing nominal wake field than this “manual regression”.



(a) ID 10: Label **(b)** ID 10: Prediction **(c)** ID 11: Label **(d)** ID 11: Prediction

Figure 7.21: Comparison of distribution of cavitation volume over the propeller blades between the labels and predictions of IDs 10 and 11



(a) ID 10: Label **(b)** ID 10: Prediction **(c)** ID 11: Label **(d)** ID 11: Prediction

Figure 7.22: Comparison of distribution of C_p over the propeller blades between the labels and predictions of IDs 10 and 11

8

Discussion

In this chapter, a discussion of the results of the thesis project will be presented. To start with, the limitations of this project due to the scope and decisions made along the project will be elaborated on in section 8.1. Thereafter, section 8.2 will discuss this thesis project in the light of the development of new energy saving devices, in particular the gate rudder.

8.1. Limitations

In this section, scope limitations and limitations due to choices made along the way will be discussed, as well as their effects on the research.

To start with, the scope of this project limited the research to single-propeller vessels with symmetric hulls. This choice was made because in this way, this research can serve as a starting point for nominal wake field predictions. The prediction of multiple-propeller wake fields was deemed more complex due to the expected asymmetry in the wake fields. Additionally, the Wärtsilä Archimedes dataset contains a larger number of single-propeller projects compared to multiple-propeller projects. The effect of this limitation is that all trained models predict symmetric wake fields, and are not suited for multiple-propeller wake field prediction.

The choice to predict the nominal wake field, instead of the effective wake field, which is of much higher interest to a propeller designer, has a similar reason. As the effective wake field captures the interaction effects between the propeller and the hull, effective wake field prediction would have required much more complex models, including specific features that capture the geometry of the propeller used as well as propeller loading information. Furthermore, most of the wake fields included in the Wärtsilä Archimedes dataset are nominal wake fields, which are converted to effective wake fields by the theoretical method of Huang and Groves during the cavitation and performance assessments while designing a propeller. If a model predicting effective wake fields had been engineered, these target wake fields would have had to be converted to effective wake fields, introducing errors due to limitations in the method of Huang and Groves, which only takes a nominal wake field and the propeller loading coefficient C_T as inputs.

A limitation of the hyperparameter tuning procedure is that all tuning processes were performed in-memory on a computer with limited resources. This meant that the extent to which the hyperparameter space could be searched was constrained by the working memory of the PC used, as well as the speed of the CPU. This limitation was mitigated as much as possible by reducing the size of the hyperparameter space and employing the Hyperband optimisation algorithm, which is more time- and memory-efficient but less thorough compared to grid search or random search. Another limitation of the tuning procedure is that, due to computational limitations, the hyperparameter tuning for all models, except for the server model, was not cross-validated. This can result in limited generalisability and suboptimal hyperparameter selections. Despite these limitations, all models had their generalisability evaluated through repeated random subset validation.

Regarding the introduction of errors, it was decided to mitigate error introduction as much as possible. For instance, it was decided that all labelled and predicted wake fields would be given at radii starting from r/R = 0.4, to prevent having to extrapolate labels that are given at higher lowest radii, thereby introducing extrapolation errors. Furthermore, both the use of the discrete cosine transform (as can be seen in appendix C) and, to a lesser extent, the interpolation to equal radii and angles of all labelled wake fields have introduced some errors. To prevent these errors from leading to a distorted view of model performance, the predictions and the labels they were compared with were manipulated in similar ways. This means, for instance, predictions made by a feed-forward neural network using n DCT coefficients were compared with labels that were also transformed by taking their first n DCT coefficients. A downside of this approach is that there has not been an assessment of the "true" error between the predicted wake fields and the wake fields as they are provided in the Wärtsilä Archimedes dataset. However, for the aim of this research (to train and evaluate a machine learning model that is able to predict nominal wake fields), this choice was deemed reasonable.

A final choice was to focus solely on predicting the axial velocities in the nominal wake field. While this was not a scope limitation at the start of the project, it was soon discovered that small dataset size and limited feature informativeness would highly affect the performance of the trained models. Therefore, it was decided to perform a more thorough training, comparing various different neural network architectures, instead of focusing on building three models for axial, tangential, and radial velocities. The choice for the axial velocity as the direction of velocity to build all models for was justified by assuming that axial velocities have a higher influence on cavitation behaviour than tangential and radial velocities, which was confirmed by the experiment described in section 7.6.

8.2. Implementation in Gate Rudder Project

In the introduction, as well as in the conclusion to the second research question, it was stated that quick nominal wake field predictions are essential in an iterative integral aft ship design approach. Therefore, nominal wake field prediction using machine learning techniques might be a good solution. There are some remarks to this:

Firstly, the machine learning models developed during this project predict the nominal wake field for symmetrical single-propeller vessels. The assumption of having a symmetrical wake field cannot necessarily be applied when energy saving devices and/or the gate rudder are also taken into account. Furthermore, the feature space of the trained models does not include any descriptive features of any energy saving device or the gate rudder. It was observed, for instance, that the presence of appendages like a skeg was often not considered by the models, simply because there was no feature describing them. Using machine learning to predict nominal wake fields that include the presence of energy saving devices or a gate rudder would require both a re-engineering of the models used and an upgrade of the dataset, in terms of dataset size as well as feature informativeness.

In addition, the trained models predict nominal wake fields. These nominal wake fields can be converted to effective wake fields using the theoretical method of Huang and Groves. However, this theoretical method has its limitations. For instance, the intended use of the method is for axisymmetrical nominal wake fields, and energy saving devices and the gate rudder are therefore not covered by the method. As energy saving devices and the gate rudder aim to optimise the hydrodynamic interaction between them, the hull, and the propeller, the effective wake field plays an even more important role in wake field prediction in the presence of these appendages. Therefore, it is advisable to first extend the current models to accurately predict effective wake fields before implementing energy saving devices or the gate rudder.

9

Conclusion

The main research question that is answered with this research is:

To what extent is it possible to train and validate a machine learning model to accurately predict the nominal wake field of a vessel, given a database of single-propeller vessels with only basic parameters describing hull geometries?

For this thesis, five machine learning models that aim to predict nominal wake fields for single-propeller vessels have been developed. To support in answering the main question and developing these five models, three sub-questions have been formulated. These sub-questions have been answered throughout the chapters of this thesis. They will be discussed first, after which the final conclusions are drawn.

What is the importance of understanding the nominal wake field in propeller design, and at which points in the aft ship design process can predicted nominal wake fields be applied?
To design a propeller with optimal performance and to prevent cavitation hindrance levels to be exceeded, knowledge of the wake field is essential. The nominal wake field is representative of the influence of the hull on the flow field at the propeller plane. The current incorporation of the nominal wake field in a rather linear design process is that it is obtained by CFD or EFD experiments, and converted to an effective wake field by a theoretical method like that of Huang and Groves [36]. When an integral design approach of the whole aft ship is applied, the propeller design process is incorporated in this iterative design process. In such an iterative process, being able to quickly predict the wake field is essential.

Which machine learning algorithms and structures are most suitable for predicting the nominal wake field, and what are the steps that need to be taken to develop such models?
Due to the highly non-linear nature of the flow phenomena that drive the nominal wake field, a non-linear regression algorithm should be used when predicting the nominal wake field. Furthermore, all velocities in the nominal wake field are influenced by surrounding velocities. There are, for instance, limits to the velocity gradients in the nominal wake field. To demonstrate the importance of using a regression algorithm that is informed about surrounding velocities when making a velocity prediction at a certain point, an experiment was conducted. This experiment involved training a non-linear support vector regression model using the radial basis function kernel. Its inability to accurately predict single points in the wake field proved the necessity of using a learning algorithm that is informed of all velocity points at once. This reasoning led to the justification of using neural network architectures in this project. In order to develop a machine learning model that predicts nominal wake fields, multiple steps need to be taken. In this thesis, the pre-processing phase was elaborated on, consisting of data collection and preparation and feature engineering. Thereafter, the machine learning models were defined and trained. After training all models, an evaluation of the trained models was performed.

What criteria determine the quality of a predicted nominal wake field, and what are the requirements for accurate nominal wake field prediction?
The quality of a predicted nominal wake field regarding propeller performance is determined by the level of accuracy in representing the velocity distribution behind the ship hull. This implies that the average velocity in the ship wake (represented by the wake fraction w), as well as individual velocities in the wake field, should closely match the real values. These two criteria have been accounted for by using a custom loss function while training all different models.

The main research question, *To what extent is it possible to train and validate a machine learning model to accurately predict the nominal wake field of a vessel, given a database of single-propeller vessels with only basic parameters describing hull geometries?*, will now be answered.
Five different machine learning models have been developed for this thesis project: a feed-forward neural network (FNN), a feed-forward neural network using discrete cosine transformation (DCT), a FNN using DCT with coefficient standardisation, an ensemble method consisting of two neural network sub-models and one neural network meta-model, and a long short-term memory (LSTM) recurrent neural network model. All machine learning models have been trained using a dataset taken from Wärtsilä's in-house propeller design software, Archimedes. After filtering, 142 samples remained in the dataset. The variables from the dataset were transformed into 28 separate features. Out of these, 17 features are numerical, representing quantities or amounts. The remaining 11 features are binary, and are used to represent two different categorical variables in a way that the model can easily process.

The predicted wake fields generated by the five trained machine learning models have been analysed by comparing them with the target wake fields. Based purely on the resulting loss, the LSTM model performed best in terms of minimising the loss between predictions and targets, followed by the FNN using DCT and coefficient standardisation, and the "vanilla" FNN. A visual inspection of the predictions made by all models, however, revealed that all models mainly predict highly generalised wake fields. These predictions feature elements typical of nominal wake fields, such as decreasing velocities towards the inner radii and a "wake peak" in the top half-plane, but lack details specific to individual cases. These details are not necessarily important for (for instance) cavitation predictions, but are a strong indication of the inability of the models to effectively grasp the underlying physical phenomena driving the nominal wake field.

A relative comparison between all models was conducted by performing repeated random subsampling validation. This repeated random subsampling validation served two purposes. The first purpose was to gain insight into the generalisability of each individual model to different splits in the dataset. The lower the standard deviation of a model's performance across different splits, the better its ability to generalise. The second purpose was to obtain an average performance score for each model, leading to a more reliable indication of the relative performance of each model. While the vanilla FNN had the lowest standard deviation across all eight random splits, and therefore generalises best to unseen data, the LSTM model had the lowest mean loss: almost half that of the worst-performing model, the ensemble model. An important observation is that the standard deviations of loss over the different random splits for all models is rather high, varying between 14% and 40% of the average loss. A conclusion that can be drawn from that observation is that all models are quite sensitive to different training-validation splits, an indication of limited generalisability.

Feature importance was assessed using a feature importance study by permutation on the feed-forward neural network. Two main observations from this feature importance study support the conclusion to this project. First and foremost, the features that supposedly contribute the most to understanding the hydrodynamic phenomena that drive the wake field, such as the block coefficient and features that are directly descriptive of the hull shape, generally do not have the highest importance. Furthermore, some features have a negative importance, indicating that they confuse the model more than that they contribute to its performance. The latter is an indication of the lack of generalisation of the model, while the former is an indication of the limited understanding of the model of the underlying phenomena.

Lastly, from the FNN using DCT model and the FNN using DCT with coefficient standardisation, two visually good- and bad-looking wake field predictions have been subjected to a cavitation assessment using the vortex lattice based potential flow method Mpuf3a. Because the influence of the nominal wake field on the calculations performed by Mpuf3a is limited to the wake-induced contribution to cavitation behaviour, the results looked promising, with a difference in predicted and actual lowest margin against

cavitation of 21.1% for the FNN using DCT model, and 7.32% for the FNN model using DCT and coefficient standardisation. Based on the comparison between calculations based on predicted and labelled wake fields, Wärtsilä propeller designers have confirmed that the predicted wake fields serve as a better substitute for a missing nominal wake field than performing a manual regression.

Despite the fact that the predicted wake fields serve as a better substitute than manual regression in the case of cavitation analysis with missing nominal wake field data, it can be concluded that, using the provided dataset, the developed machine learning models have not been able to predict nominal wake fields to the desired accuracy. This conclusion is supported by a visual inspection of the predicted wake fields, which show signs of being “generalised” or averaged out, with detailed characteristics lacking. The small dataset size and the lack of feature informativeness are identified as the main reasons for this behaviour, supported by a repeated random subsampling validation and a feature importance study using feature permutation, respectively.

9.1. Recommendations

Some recommendations for future research can be made. Firstly, it is advised to increase both the dataset size and the informativeness of features. The consequences of the limited dataset size and feature informativenesses in this project were clearly reflected in the sensitivity of all models to specific training-validation splits, as well as their limited generalisability to unseen data. Although the required dataset size is dependent on the informativeness of the features, at least a ten-fold increase in dataset size is recommended. With regards to features, specifically features better describing the aft ship geometry would help to improve the performance of the models.

Additionally, it is recommended for future research to utilize a computer with greater computational capabilities than the one used in this study. With sufficient computational resources, a cross-validated grid search across the entire hyperparameter space can be performed, along with more extensive cross-validation of the tuned models. If the dataset size and feature informativeness are increased, and when combined with enhanced computational resources, this approach could lead to more accurate predictions of axial velocities in the nominal wake field. Subsequently, incorporating radial and tangential velocities might be feasible, enabling a complete prediction of the nominal wake field.

Another interesting extension to this research would be a deeper investigation into assessing the quality of a predicted wake field. Although a custom loss function was defined, visually bad-looking predictions could sometimes still get assigned a fairly low loss. A possible extension of this quality assessment of a wake field extension could be to look at velocity gradients. When it is known what common velocity gradients are in certain parts of the wake field, this information could be used to assign more loss to very “scattered” wake field predictions.

Furthermore, it would be interesting to look at effective wake field prediction. Incorporating information about the propeller geometry into the feature space and developing a model to predict effective wake fields would be very useful in the field of propeller design, as well as ESD and gate rudder development.

References

[1] IPCC. *Summary for Policymakers.* Geneva, Switzerland: Intergovernmental Panel on Climate Change (IPCC), 2023. URL: https://doi.org/10.59327/IPCC/AR6-9789291691647.001.

[2] International Maritime Organization. *Fourth IMO GHG Study 2020, Executive Summary*. London: International Maritime Organization, 2020.

[3] United Nations Conference on Trade and Development. *Review of Maritime Transport 2023*. New York: UNCTAD, 2023.

[4] International Maritime Organization. *Improving the energy efficiency of ships*. 2019. URL: https://www.imo.org/en/OurWork/Environment/Pages/Improving%20the%20energy%20efficiency%20of%20ships.aspx (Accessed 18/06/2024).

[5] International Maritime Organization. *Initial IMO Strategy on Reduction of GHG Emissions from Ships*. London: International Maritime Organization, 2018. URL: https://unfccc.int/sites/default/files/resource/250_IMO%20submission_Talanoa%20Dialogue_April%202018.pdf.

[6] International Maritime Organization. *2023 IMO Strategy on Reduction of GHG Emissions from Ships*. London: International Maritime Organization, 2023. URL: https://wwwcdn.imo.org/localresources/en/OurWork/Environment/Documents/annex/MEPC%2080/Annex%2015.pdf.

[7] A. Foretich et al. 'Challenges and opportunities for alternative fuels in the maritime sector'. In: *Maritime Transport Research* 2 (1st Jan. 2021), p. 100033.

[8] N. Sasaki et al. 'Measurements and calculations of gate rudder performance'. In: The 5th International Conference on Advanced Model Measurement Technology for the Maritime Industry (AMT'17). Glasgow, United Kingdom, 2017.

[9] N. Sasaki. 'ZEUS and NOAH Projects of NMRI'. In: Third International Symposium on Marine Propulsors. Tasmania, Australia, May 2013. (Accessed 15/05/2024).

[10] N. Sasaki, S. Kuribayshi and A. Miles. 'Full scale performance of Gate Rudder'. In: *Propellers & Impellers: Research, Design, Construction and Application* (28th Mar. 2019). DOI: 10.3940/rina.pro.2019.10.

[11] GATERS. *GATERS: Objectives and Impact*. 2024. URL: https://www.gatersproject.com/objectives-and-impact (Accessed 20/06/2024).

[12] Z. Zhang et al. 'Application of CFD in ship engineering design practice and ship hydrodynamics'. In: *Journal of Hydrodynamics, Ser. B*. Proceedings of the Conference of Global Chinese Scholars on Hydrodynamics 18.3 (1st July 2006), pp. 315–322.

[13] S. Vervoordeldonk. 'Modelling Hydrodynamic Interaction'. Literature study. Delft: Delft University of Technology, Aug. 2024.

[14] International Towing Tank Conference. *Model Manufacture, Propeller Models Terminology and Nomenclature for Propeller Geometry*. 2008.

[15] J. Klein Woud and D. Stapersma. *Design of propulsion and electric power generation systems*. London: IMarEST, 2002. ISBN: 978-1-902536-47-7.

[16] W. P. A. van Lammeren, J. D. van Manen and M. W. C. Oosterveld. 'The Wageningen B-Screw Series'. In: 77.8 (12th Nov. 1969).

[17] J. Liu and R. Hekkenberg. 'Sixty years of research on ship rudders: effects of design choices on rudder performance'. In: *Ships and Offshore Structures* 12.4 (19th May 2017), pp. 495–512.

[18] W. F. Edgerton. 'Ancient Egyptian Steering Gear'. In: *The American Journal of Semitic Languages and Literatures* 43.4 (1927). Publisher: University of Chicago Press, pp. 255–265. ISSN: 1062-0516.

[19] Wärtsilä. *Waterjet propulsion*. Wartsila.com. 2024. URL: https://www.wartsila.com/encyclopedia/term/waterjet-propulsion (Accessed 25/06/2024).

[20] J. S. Carlton. *Marine propellers and propulsion*. Fourth edition. Oxford, United Kingdom ; Cambridge, MA: Butterworth-Heinemann, an imprint of Elsevier, 2019. 585 pp. ISBN: 978-0-08-100366-4.

[21] V. Bertram. *Practical Ship Hydrodynamics*. 2nd ed. Oxford: Butterworth-Heinemann, 2012. ISBN: 978-0-08-097150-6. (Accessed 23/05/2024).

[22] J. Grevink. 'Manoeuvring and rudders'. Lecture from the lecture series MT44006 Future Marine Propulsion Systems at Delft University of Technology. Delft, 2023. (Accessed 25/06/2024).

[23] American Bureau of Shipping. *Guide for Vessel Maneuverability*. Houston, Texas: ABS, 2017.

[24] F. K. Fuss. 'The dynamics of a turning ship: mathematical analysis and simulation based on free body diagrams and the proposal of a pleometric index'. In: *Dynamics* 3.3 (Sept. 2023), pp. 379–404.

[25] A. F. Molland and S. R. Turnock. *Marine Rudders and Control Surfaces*. Oxford: Butterworth-Heinemann, 2007. ISBN: 978-0-7506-6944-3. (Accessed 23/05/2024).

[26] E. C. Tupper. 'Manoeuvring'. In: *Introduction to Naval Architecture (Fifth Edition)*. Ed. by E. C. Tupper. Oxford: Butterworth-Heinemann, 1st Jan. 2013, pp. 277–298. ISBN: 978-0-08-098237-3.

[27] J. Gordon Leishman. 'Boundary Layer Flows'. In: (1st Jan. 2023). URL: https://eaglepubs.erau.edu/introductiontoaerospaceflightvehicles/chapter/introduction-to-boundary-layers/.

[28] L. Larsson, H. C. Raven and J. R. Paulling. *Ship resistance and flow*. Principles of naval architecture. Jersey City, N.J: Society of Naval Architects and Marine Engineers, 2010. 230 pp. ISBN: 978-0-939773-76-3.

[29] P. K. Kundu et al. *Fluid mechanics*. Sixth Edition. London: Elsevier, Academic Press, 2016. 921 pp. ISBN: 978-0-12-405935-1.

[30] G. Weymouth. *Course Notes Lesson 1*. Lecture notes from the lecture series MT44025 Numerical Ship Hydrodynamics at Delft University of Technology. 2024. (Accessed 07/01/2024).

[31] D. Radojcic et al. 'Resistance Prediction for Hard Chine Hulls in the Pre-Planing Regime'. In: *Polish Maritime Research* 21 (26th July 2014), pp. 9–26.

[32] S. Gaggero et al. 'Ship propeller side effects: Pressure pulses and radiated noise'. In: *Noise Mapping* 3 (23rd Dec. 2016).

[33] A.F. Molland and S. Turnock. 'Wind tunnel investigation of the influence of propeller loading on ship rudder performance'. In: *Transactions of the Royal Institution of Naval Architects* 38 (1st Jan. 1991).

[34] A. F. Molland and S. R. Turnock. 'Flow Straightening Effects on a Ship Rudder due to Upstream Propeller and Hull'. In: *International Shipbuilding Progress* 49.3 (1st Jan. 2002), pp. 195–214.

[35] International Towing Tank Conference. *The Specialist Commitee on Scaling of Wake Field, Final Report and Recommendations to the 26th ITTC*. Rio de Janeiro, Brazil, 2011.

[36] T. T. Huang and N. C. Groves. *Effective wake: Theory and Experiment*. DTNSRDC-8i/033. Bethesda, Maryland: David W. Taylor Naval Ship Research and Development Center, Apr. 1981.

[37] Johan Bosschers. 'Propeller tip-vortex cavitation and its broadband noise'. PhD thesis. 21st Sept. 2018. URL: https://research.utwente.nl/en/publications/propeller-tip-vortex-cavitation-and-its-broadband-noise.

[38] P.B. Regener, Y. Mirsadraee and P. Andersen. 'Nominal vs. Effective Wake Fields and Their Influence on Propeller Cavitation Performance'. In: *Journal of Marine Science and Engineering* 6.2 (June 2018), p. 34. ISSN: 2077-1312. DOI: 10.3390/jmse6020034.

[39] B. Schuiling and T. Van Terwisga. 'Hydrodynamic working principles of Energy Saving Devices in ship propulsion systems'. In: *International Shipbuilding Progress* 63.3 (2017), pp. 255–290. ISSN: 0020-868X. DOI: 10.3233/ISP-170134.

[40] H. J. Prins et al. ‘Green Retrofitting through Optimisation of Hull-propulsion Interaction – GRIP’. In: *Transportation Research Procedia* 14 (1st Jan. 2016), pp. 1591–1600. ISSN: 2352-1465.

[41] Wärtsilä. *Wärtsilä EnergoFlow boosts propulsion efficiency*. 2017. URL: https://www.wartsila.com/insights/article/wartsila-energoflow-boosts-propulsion-efficiency (Accessed 21/08/2024).

[42] R. Mysa et al. ‘Numerical Investigation of Energy Saving Device Effects on Ships With Propeller-Hull Interactions’. In: ASME 2018 37th International Conference on Ocean, Offshore and Arctic Engineering. 17th June 2018. DOI: 10.1115/OMAE2018-78600.

[43] W. Jin et al. ‘Optimization of Blade Position on an Asymmetric Pre-Swirl Stator Used in Container Ships’. In: *Journal of Marine Science and Engineering* 11.1 (Jan. 2023), p. 50.

[44] P. Anschau, N. Kornev and S. Samarbakhsh. ‘Unsteady hydrodynamic loads on energy saving ducts’. In: *Ocean Engineering* 269 (1st Feb. 2023), p. 113431.

[45] L. Xu, D. He and D. Wan. ‘Research Status and Analysis of the Ship Hydrodynamic Energy-saving Devices’. In: 2017.

[46] S. Turkmen, A. Carchen and M. Atlar. ‘A new energy saving twin rudder system - Gate Rudder’. In: SCC 2015 International Conference on Shipping in Changing Climates. Ed. by N. Sasaki. 24th Nov. 2015.

[47] F. Spinelli et al. ‘Shipping Decarbonization: An Overview of the Different Stern Hydrodynamic Energy Saving Devices’. In: *Journal of Marine Science and Engineering* 10.5 (May 2022). Number: 5 Publisher: Multidisciplinary Digital Publishing Institute, p. 574. ISSN: 2077-1312. DOI: 10.3390/jmse10050574. URL: https://www.mdpi.com/2077-1312/10/5/574 (Accessed 27/05/2024).

[48] N. Bulten. *Gate Rudder performance estimate of the different aspects based on full-scale CFD simulations*. 2024.

[49] N. Sasaki, M. Atlar and S. Kuribayashi. ‘Advantages of twin rudder system with asymmetric wing section aside a propeller’. In: *Journal of Marine Science and Technology* 21.2 (29th Dec. 2015), pp. 297–308.

[50] N. Sasaki et al. ‘Design concept and application of open type ducted propeller (Gate Rudder System)’. In: *Journal of Marine Science and Engineering* (2021).

[51] M. Atlar. ‘A Review of Gate Rudder System Retrofitting of the GATERS Target Ship ”MV ERGE”’. GATERS. Glasgow, 2023. (Accessed 15/05/2024).

[52] A. Carchen et al. ‘A prediction program of manoeuvrability for a ship with a Gate Rudder system’. In: A. Yücel Odabaþý Colloquium Series - 2nd International Meeting on Recent Advances in Prediction Techniques for Safe Manoeuvring of Ships and Submarines. Istanbul, 17th Nov. 2016.

[53] N. Sasaki et al. ‘Joint sea trial of ships with Gate Rudder and conventional rudder’. In: Full Scale Ship Performance online conference. 2021.

[54] Bureau Veritas. ‘CMA Ships - Gate Rudder Evaluation on 2300 YZJ’. 30th Aug. 2023.

[55] Ç. Köksal et al. ‘Experimental powering performance analysis of M/V ERGE in calm water and waves’. In: 4th international meeting on ship design & optimization and energy efficient devices for fuel economy. Istanbul, Turkey, 2022.

[56] N. Sasaki and M. Atlar. ‘Scale Effect of Gate Rudder’. In: Sixth International Symposium on Marine Propulsors. Rome, Italy, 2019.

[57] N. Sasaki et al. ‘Towards a Realistic Estimation of the Powering Performance of a Ship with a Gate Rudder System’. In: *Journal of Marine Science and Engineering* 8.1 (Jan. 2020), p. 43.

[58] M. Hussain, M. Karim and N. Sasaki. ‘Numerical assessment of the scale effects on the propulsive performance of a ship with gate rudder system’. In: *Ocean Engineering* 249 (Apr. 2022).

[59] R. Tonelli. *9000 CEU PCTC; Manoeuvring analysis of the gate rudder*. 34716-5-RD. Wageningen: MARIN, May 2023, p. 50.

[60] I. Santic et al. ‘Systematic experimental survey of propulsive and acoustic performances of a gate rudder system in relation to a conventional rudder’. In: 7th International Conference on Advanced Model Measurement Technology for the Maritime Industry. Istanbul, Turkey, 2023.

[61] S Ozsayan et al. ‘Effects of the Gate Rudder System (GRS) on the experimental cavitation observations and noise measurements’. In: The 7th AMT’23 Conference. Istanbul, Oct. 2023.

[62] D. Yongle, S. Baowei and W. Peng. ‘Numerical investigation of tip clearance effects on the performance of ducted propeller’. In: *International Journal of Naval Architecture and Ocean Engineering* 7 (2015).

[63] C. M. Bishop. *Pattern Recognition and Machine Learning*. 1st ed. New York: Springer, Sept. 2016. 778 pp. ISBN: 978-0-387-31073-2.

[64] A. Burkov. *The hundred-page machine learning book*. Quebec City, Canada: True Positive Inc, 2019. 141 pp. ISBN: 978-1-9995795-0-0.

[65] A. Burkov. *Machine learning engineering*. Quebec City, Canada: True Positive Inc, 2020. 282 pp. ISBN: 978-1-9995795-7-9.

[66] T. Daniel. ‘Machine learning for nonlinear model order reduction’. PhD thesis. 24th Sept. 2021.

[67] S. Belkacem. ‘Machine learning approaches to rank news feed updates on social media’. PhD thesis. 2021.

[68] S. V. Mahadevkar et al. ‘A Review on Machine Learning Styles in Computer Vision - Techniques and Future Directions’. In: *IEEE Access* 10 (2022).

[69] Iqbal H. Sarker. ‘Machine Learning: Algorithms, Real-World Applications and Research Directions’. In: *Sn Computer Science* 2.3 (2021), p. 160. ISSN: 2662-995X. DOI: 10.1007/s42979-021-00592-x. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7983091/ (Accessed 05/08/2024).

[70] R. Xu and D. WunschII. ‘Survey of Clustering Algorithms’. In: *IEEE Transactions on Neural Networks* 16.3 (May 2005), pp. 645–678.

[71] Y. Bengio, A. Courville and P. Vincent. *Representation Learning: A Review and New Perspectives*. 23rd Apr. 2014. arXiv: 1206.5538[cs].

[72] G. Press. ‘Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says’. In: *Forbes* (May 2016). URL: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/ (Accessed 05/08/2024).

[73] D. Singh and B. Singh. ‘Feature wise normalization: An effective way of normalizing data’. In: *Pattern Recognition* 122 (1st Feb. 2022).

[74] S. L. Brunton and J. N. Kutz. *Data-driven science and engineering: machine learning, dynamical systems, and control*. 2nd edition. Cambridge: Cambridge university press, 2022. ISBN: 978-1-00-909848-9.

[75] S. Eskandar. *Introduction to RBF SVM: A Powerful Machine Learning Algorithm for Non-Linear Data*. Medium. 26th Mar. 2023. URL: https://medium.com/@eskandar.sahel/introduction-to-rbf-svm-a-powerful-machine-learning-algorithm-for-non-linear-data-1d1cfb55a1a (Accessed 14/01/2025).

[76] R. Rodríguez-Pérez and J. Bajorath. ‘Evolution of Support Vector Machine and Regression Modeling in Chemoinformatics and Drug Discovery’. In: *Journal of Computer-Aided Molecular Design* 36.5 (2022), pp. 355–362. DOI: 10.1007/s10822-022-00442-9.

[77] R. Qamar and B. Ali Zardari. ‘Artificial Neural Networks: An Overview’. In: *Mesopotamian Journal of Computer Science* (2023), pp. 130–139.

[78] C. Clabaugh, D Myszewski and J Pang. *Neural Networks History*. 2000. URL: https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html (Accessed 23/12/2024).

[79] F. Bre, J. Gimenez and V. Fachinotti. ‘Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks’. In: *Energy and Buildings* 158 (2017).

[80] A. Burkov. ‘The Hundred-Page Language Models Book’. 2025. URL: https://thelmbook.com (Accessed 29/12/2024).

[81] Eumenedies. *Answer to ”Why should the initialization of weights and bias be chosen around 0?”* Data Science Stack Exchange. 9th Aug. 2017. URL: https://datascience.stackexchange.com/a/22097 (Accessed 29/12/2024).

[82] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 30th Jan. 2017. DOI: 10.48550/arXiv.1412.6980.

[83] E. Budu. *Bagging, Boosting, and Stacking in Machine Learning*. July 2023. URL: https://www.baeldung.com/cs/bagging-boosting-stacking-ml-ensemble-models (Accessed 30/12/2024).

[84] J. Starmer. *Recurrent Neural Networks (RNNs), Clearly Explained*. July 2022. URL: https://www.youtube.com/watch?v=AsNTP8Kwu80 (Accessed 29/12/2024).

[85] J. Starmer. *Long Short-Term Memory (LSTM), Clearly Explained*. Nov. 2022. URL: https://www.youtube.com/watch?v=YCzL96nL7j0&list=PLblh5JKOoLUIxGDQs4LFFD--41Vzf-ME1&index=16 (Accessed 30/12/2024).

[86] J. A Ilemobayo et al. ‘Hyperparameter Tuning in Machine Learning: A Comprehensive Review’. In: *Journal of Engineering Research and Reports* 26.6 (7th June 2024), pp. 388–395. DOI: 10.9734/jerr/2024/v26i61188. URL: https://journaljerr.com/index.php/JERR/article/view/1188.

[87] L. Li et al. ‘Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization’. In: *Journal of Machine Learning Research* 18.185 (2018), pp. 1–52.

[88] S. Hwangbo and H. Shin. ‘Statistical Prediction of Wake Fields on Propeller Plane by Neural Network using Back-Propagation’. In: *Journal of Ship and Ocean Technology* 4.3 (2000), pp. 1–12.

[89] S.-Y. Kim and B. Y. Moon. ‘Wake distribution prediction on the propeller plane in ship design using artificial intelligence’. In: *Ships and Offshore Structures* 1.2 (1st Feb. 2006), pp. 89–98.

[90] E. Kayacan and M. A. Khanesar. ‘Chapter 2 - Fundamentals of Type-1 Fuzzy Logic Theory’. In: *Fuzzy Neural Networks for Real Time Control Applications*. Ed. by E. Kayacan and M. A. Khanesar. Butterworth-Heinemann, 1st Jan. 2016.

[91] M. Lee and I. Lee. ‘Transfer Learning with Deep Neural Network toward the Prediction of Wake Flow Characteristics of Containerships’. In: *Journal of Marine Science and Engineering* 11.10 (29th Sept. 2023), p. 1898.

[92] DataHeroes. *Z-Score for Anomaly Detection*. URL: https://dataheroes.ai/glossary/z-score-for-anomaly-detection/ (Accessed 06/01/2025).

[93] N. Ahmed, T. Natarajan and K.R. Rao. ‘Discrete Cosine Transform’. In: *IEEE Transactions on Computers* C-23.1 (Jan. 1974), pp. 90–93. ISSN: 0018-9340. DOI: 10.1109/T-C.1974.223784. URL: http://ieeexplore.ieee.org/document/1672377/.

[94] S. Theodoridis and K. Koutroumbas. ‘Chapter 6 - Feature Generation I: Data Transformation and Dimensionality Reduction’. In: *Pattern Recognition (Fourth Edition)*. Ed. by S. Theodoridis and K. Koutroumbas. Boston: Academic Press, 1st Jan. 2009, pp. 323–409. ISBN: 978-1-59749-272-0. DOI: 10.1016/B978-1-59749-272-0.50008-6.

[95] SciPy. *dct - SciPy v1.15.0 Manual*. SciPy Docs. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.fft.dct.html (Accessed 01/07/2025).

[96] T. Lamb, ed. *Ship design and construction*. Jersey City, NJ: Society of Naval Architects and Marine Engineers, 2003. ISBN: 978-1-61583-301-6.

[97] V. Bertram. ‘Estimating Main Dimensions and Coefficients in Preliminary Ship Design’. In: *Shiffstechnik* 45 (1998).

[98] A. Papanikolaou. *Ship Design: Methodologies of Preliminary Design*. Dordrecht: Springer Netherlands, 2014. 1 p. ISBN: 978-94-017-8750-5.

[99] A. F. Molland, S. R. Turnock and D. A. Hudson. *Ship resistance and propulsion: practical estimation of ship propulsive power*. New York: Cambridge University Press, 2011. 537 pp. ISBN: 978-0-521-76052-2.

[100] E. Stevens. *What is Nominal Data? Definition, Characteristics, Examples*. Mar. 2021. URL: https://careerfoundry.com/en/blog/data-analytics/what-is-nominal-data/ (Accessed 14/01/2025).

[101] F. Pedregosa et al. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830.

[102] G. Lemaître, F. Nogueira and C.K. Aridas. 'Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning'. In: *Journal of Machine Learning Research* 18.17 (2017), pp. 1–5. URL: http://jmlr.org/papers/v18/16-365.html.

[103] M. Waskom. 'seaborn: statistical data visualization'. In: *Journal of Open Source Software* 6.60 (6th Apr. 2021), p. 3021. DOI: 10.21105/joss.03021. URL: https://joss.theoj.org/papers/10.21105/joss.03021.

[104] G.E. Batista, R.C. Prati and M.C. Monard. 'A study of the behavior of several methods for balancing machine learning training data'. In: *ACM SIGKDD Explorations Newsletter* 6.1 (June 2004), pp. 20–29. DOI: 10.1145/1007730.1007735.

[105] Scikit-learn. *RBF SVM parameters*. scikit-learn. URL: https://scikit-learn/stable/auto_examples/svm/plot_rbf_parameters.html (Accessed 17/01/2025).

[106] TensorFlow Developers. *TensorFlow*. Version v2.18.0. 25th Oct. 2024. DOI: 10.5281/ZENODO.4724125. URL: https://zenodo.org/doi/10.5281/zenodo.4724125 (Accessed 22/01/2025).

[107] F. Chollet. *Keras*. 2015. URL: https://keras.io.

[108] A. Alwosheel, S. Van Cranenburgh and C. G. Chorus. 'Is your dataset big enough? Sample size requirements when using artificial neural networks for discrete choice analysis'. In: *Journal of Choice Modelling* 28 (Sept. 2018), pp. 167–182. ISSN: 17555345. DOI: 10.1016/j.jocm.2018.07.002.

[109] C.-L. Cheng, Shalabh and G. Garg. 'Coefficient of determination for multiple measurement error models'. In: *Journal of Multivariate Analysis* 126 (Apr. 2014), pp. 137–152. DOI: 10.1016/j.jmva.2014.01.006.

[110] SciPy Developers. *pearsonr — SciPy v1.15.1 Manual*. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html (Accessed 24/01/2025).

[111] S. Taylor. *Covariance*. Corporate Finance Institute. URL: https://corporatefinanceinstitute.com/resources/data-science/covariance/ (Accessed 24/01/2025).

[112] S. Fortmann-Roe. *Understanding the Bias-Variance Tradeoff*. June 2012. URL: https://scott.fortmann-roe.com/docs/BiasVariance.html (Accessed 25/01/2025).

[113] SciPy Developers. *shapiro — SciPy v1.15.1 Manual*. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.shapiro.html (Accessed 25/01/2025).

[114] Ajitesh Kumar. *Nested Cross Validation for Algorithm Selection*. Analytics Yogi. Aug. 2020. URL: https://vitalflux.com/python-nested-cross-validation-algorithm-selection/ (Accessed 26/02/2025).

[115] S.A. Kinnas. *Cavitation Consortium*. URL: https://cavity.caee.utexas.edu/kinnas/mission.html (Accessed 02/04/2025).

[116] J. Gjerdevik. *Cavitation comparison using Mpuf-3a for wake fields predicted with Machine Learning*. TDH000007746. Wärtsilä, 31st Jan. 2025.

A

Raw Dataset Characteristics

This appendix contains the characteristics of the raw Wärtsilä Archimedes dataset and supports section 4.2.1. Similar statistics and visualisations for the filtered dataset are presented in section 4.4 and appendix B. The statistics and visualisations shown in this appendix are based on 1634 samples; only incomplete samples, from which no statistics or visualisations can be derived, have been removed from the dataset. Table A.1 shows descriptive statistics of the unfiltered variables. The OrderKey, Product, ShipGroupNo, and WakeOrigin variables have been omitted from this table as they are either categorical variables or their values are irrelevant for statistical analysis.

Table A.1: Descriptive statistics of the raw Wärtsilä Archimedes dataset.

	mean	std	min	25%	50%	75%	max
Lpp	124.2	70.49	0.000	83.00	123.8	174.0	378.4
Beam	20.998	15.05	0.000	14.00	20.20	28.00	266.0
Draught	6.689	3.645	0.000	5.000	6.700	8.500	30.60
Cb	4.343	153.7	0.000	0.4922	0.6520	0.7774	6212
ShaftHeight	2.450	1.033	-2.580	1.752	2.45	3.100	7.300
NumberOfShaftLines	1.346	0.5083	0.000	1.000	1.000	2.000	8.000
PropellerDiameter	4580	1805	0.000	3200	4500	5700	10500
FSAH_ShipSpeed	13.09	8.377	-0.2939	8.391	15.11	18.50	40.00
FSAH_rpm	168.4	117.1	0.000	113.8	145.0	185.0	2400

Table A.1 is visualised using box plot graphs in figure A.1. Both table A.1 and figure A.1 clearly show outliers and non-physical values in the variables. For instance, the block coefficient C_B has a maximum value of over 6000, while by definition, the block coefficient cannot be larger than one. Furthermore, many outliers in the Draught and FSAH_RPM variables can be identified.

Figure A.2 shows univariate distribution plots for each variable. The specific distribution of each variable can be identified in these plots, revealing potential bias and the location of outliers in the dataset, as well as enabling the identification of non-physical values. A bivariate assessment of distribution for every combination of variables is shown in the density plots in figure A.3. These plots can be used to identify strongly correlated variables, such as ShaftHeight and PropellerDiameter, and Lpp and Beam. This information will be used in the feature engineering phase to avoid strong correlation between features. Lastly, figure A.4 shows a linear correlation heatmap of all variables, quantifying the strength of linear correlations that can be seen in figure A.3.

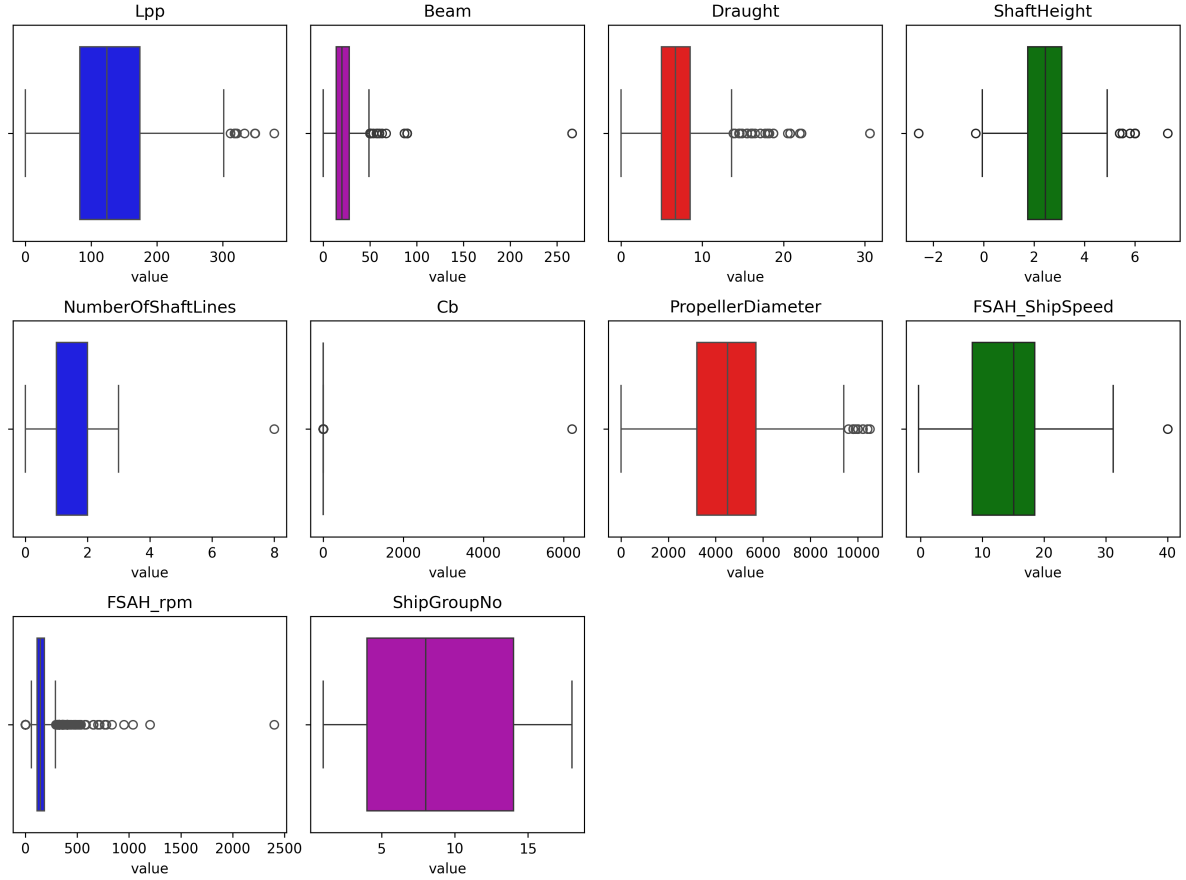


Figure A.1: Box plots of all unfiltered variables in the Wärtsilä Archimedes dataset.

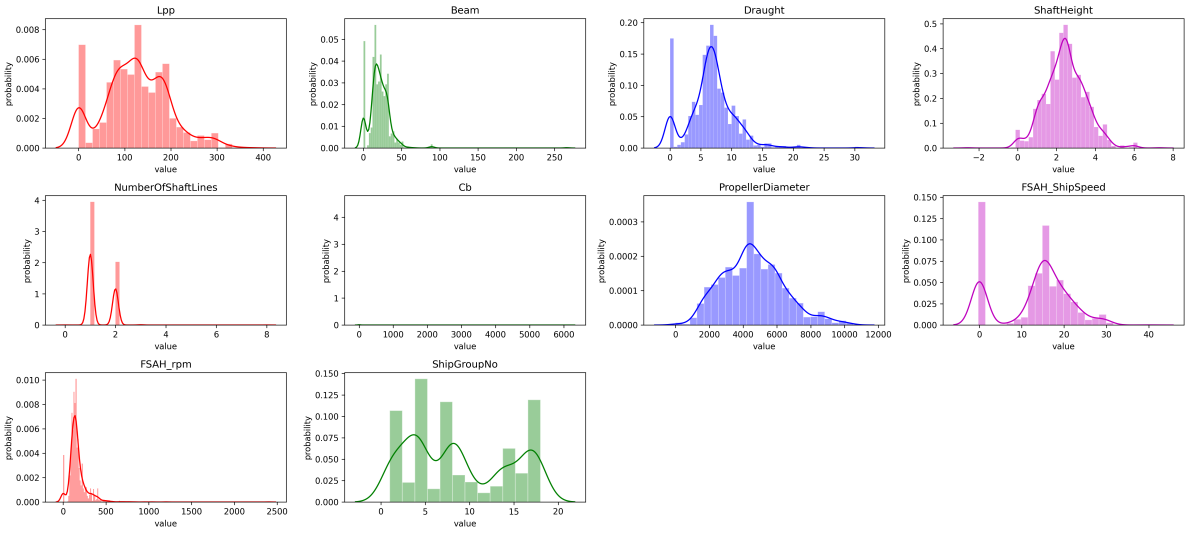


Figure A.2: Univariate distribution plots of all unfiltered variables in the Wärtsilä Archimedes dataset.

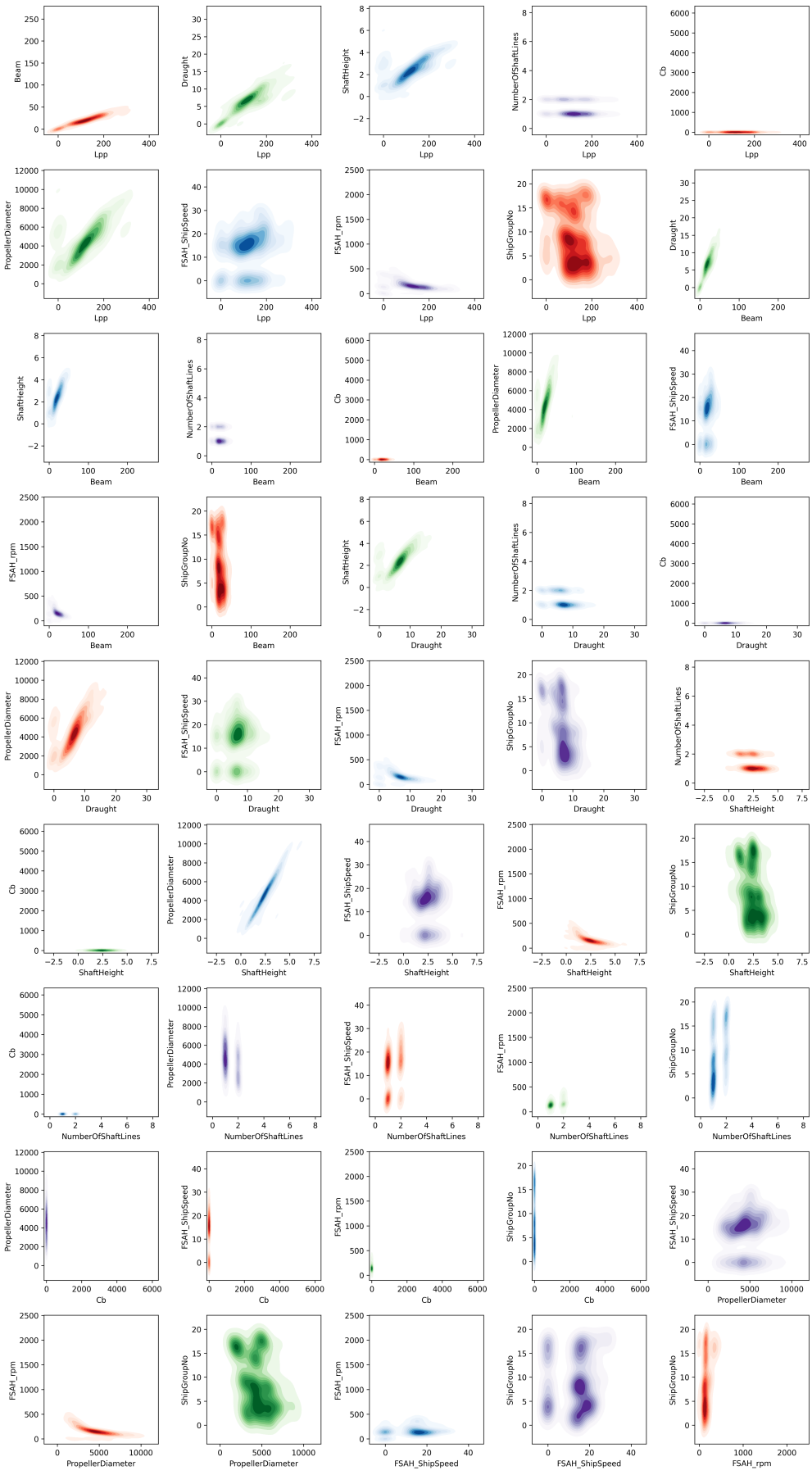


Figure A.3: Density plots of all unfiltered variables in the Wärtsilä Archimedes dataset.

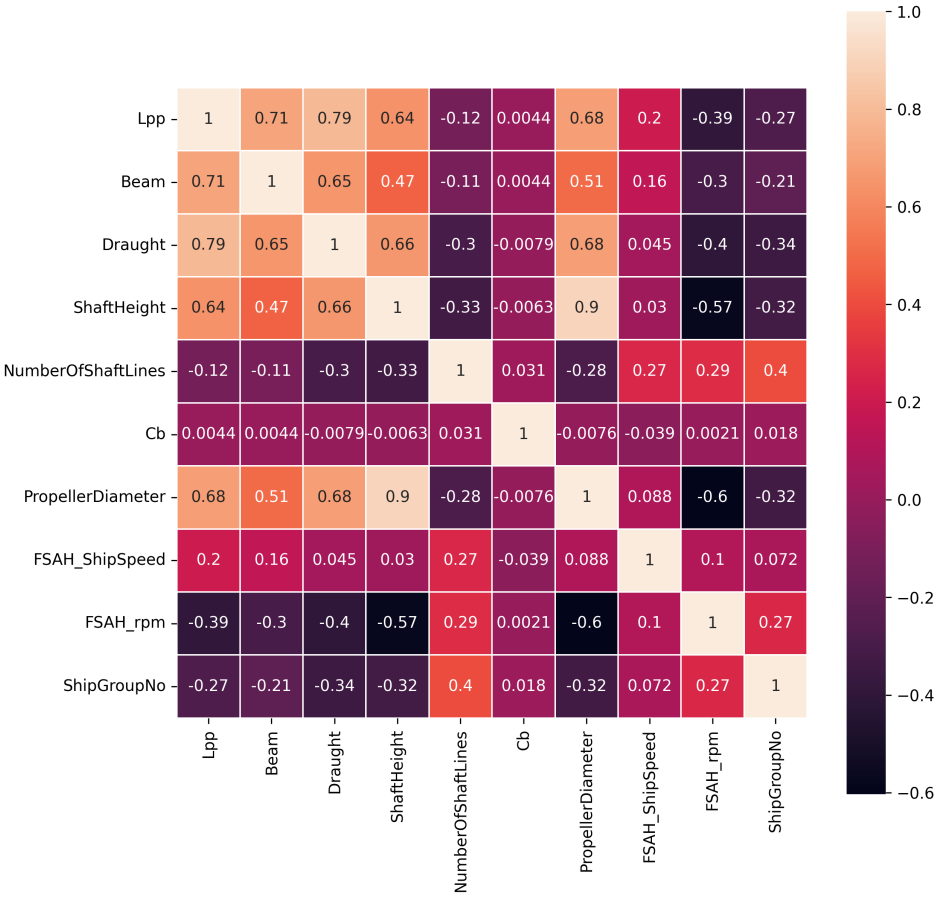


Figure A.4: Linear correlation heatmap of all unfiltered variables in the Wärtsilä Archimedes dataset.

B

Filtered Dataset Characteristics

This appendix contains the characteristics of the filtered Wärtsilä Archimedes dataset and supports section 4.4. The statistics and visualisations shown in this appendix are based on the 142 samples remaining after filtering. Table B.1, also presented in section 4.4 as table 4.4, shows descriptive statistics of the filtered variables. Figure B.1, also found in section 4.4 (figure 4.6), shows the box plots of all filtered variables.

Table B.1: Descriptive statistics of the filtered Wärtsilä Archimedes dataset.

	mean	std	min	25%	50%	75%	max
Lpp	155.4	59.6	54.4	115.9	148.8	187.3	349.5
Beam	26.1	9.4	12.4	19.2	23.4	32.3	63.0
Draught	8.4	2.4	5.2	6.7	7.9	9.2	16.1
ShaftHeight	3.0	0.8	1.4	2.3	2.9	3.4	4.9
Cb	0.7	0.1	0.4	0.6	0.7	0.8	0.9
PropellerDiameter	5631	1564	2800	4500	5400	6500	9600
FSAH_ShipSpeed	15.7	3.3	7.4	13.8	16.0	17.9	23.5
FSAH_rpm	119.2	31.1	68.2	96.7	114.0	137.6	229.4
ShipGroupNo	6.5	4.1	1.0	4.0	7.0	8.0	17.0

Table B.1 is visualised using box plot graphs in figure B.1. They can be compared with table A.1 and figure A.1, and it can easily be seen that the most extreme outliers and non-physical values have been filtered out. This is also represented in figure B.2, in which can be seen that distributions lie closer together and follow logical patterns within reasonable ranges.

Figures B.3 and B.4 show the density plots of all combinations of variables and a heatmap showing linear correlation between all variables, respectively. It should be noted that although the dataset has been filtered, not all variables can be used directly as features because of the strong (linear) correlation between some of the variables. Chapter 5 elaborately describes the process of feature engineering, and the characteristics of the engineered features are given in section 5.6 as well as appendix D.

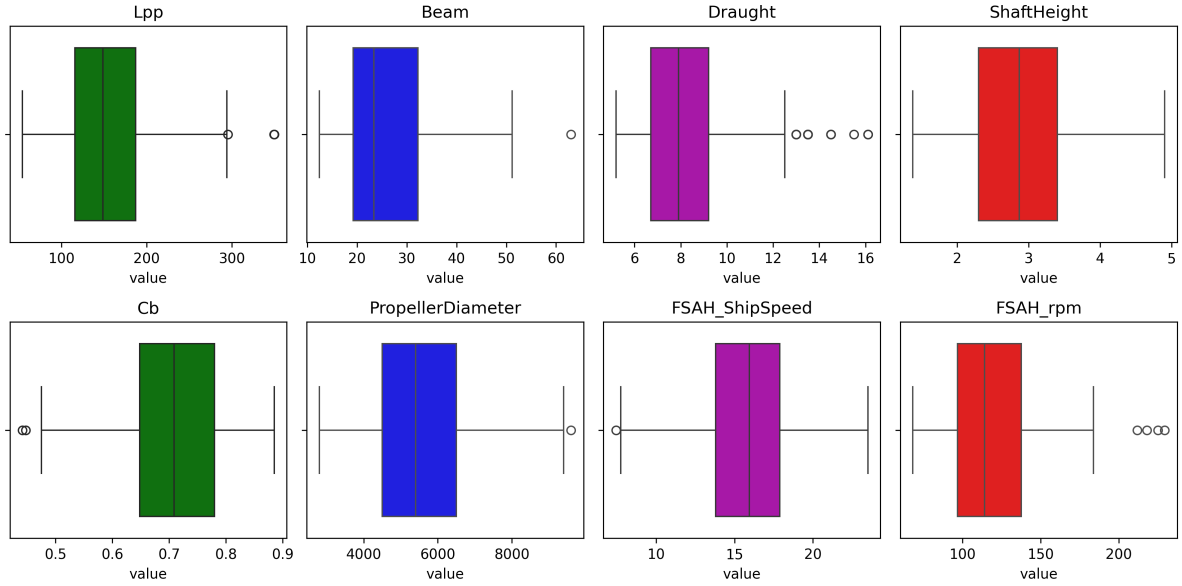


Figure B.1: Box plots of all filtered variables in the Wärtsilä Archimedes dataset.

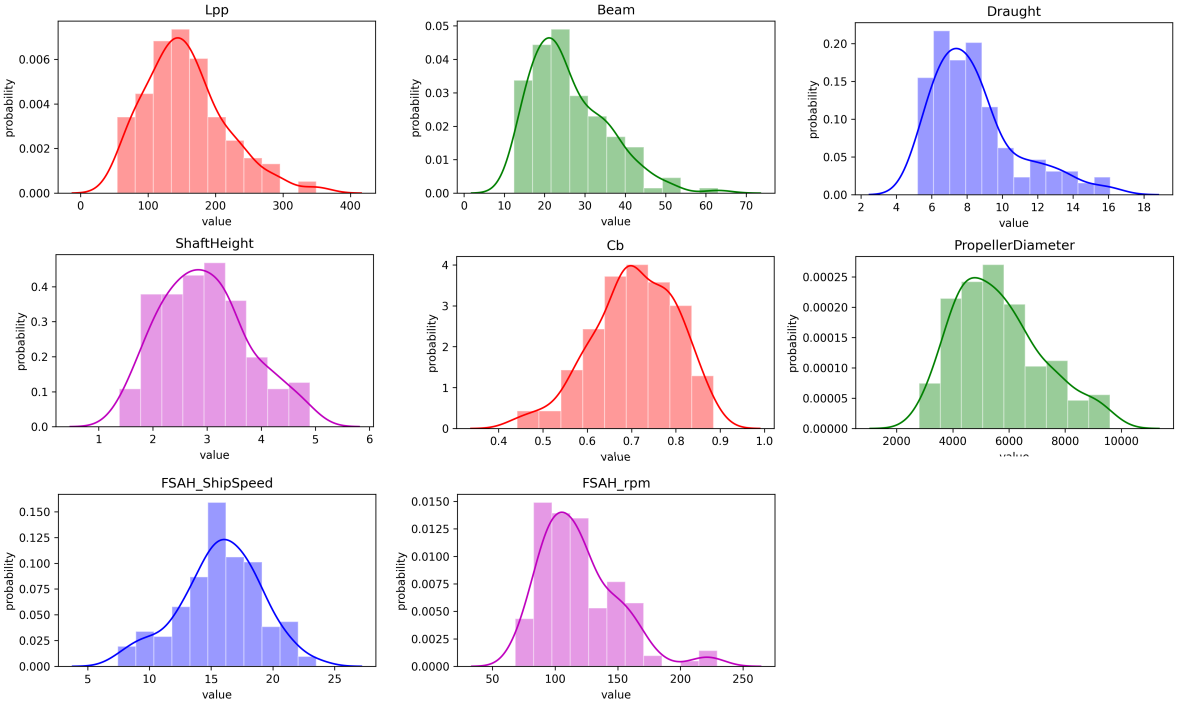


Figure B.2: Univariate distribution plots of all filtered variables in the Wärtsilä Archimedes dataset.

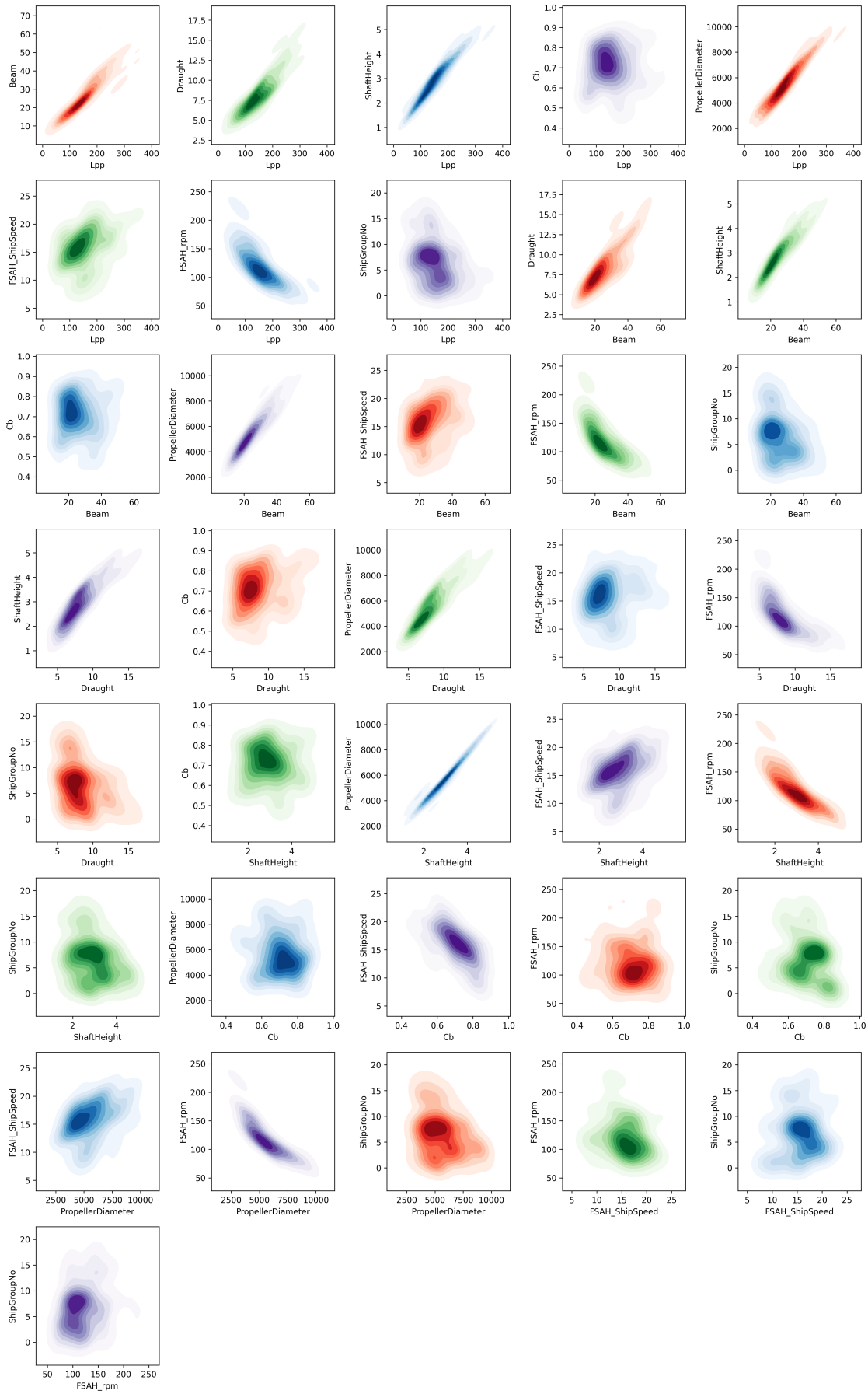


Figure B.3: Density plots of all filtered variables in the Wärtsilä Archimedes dataset.

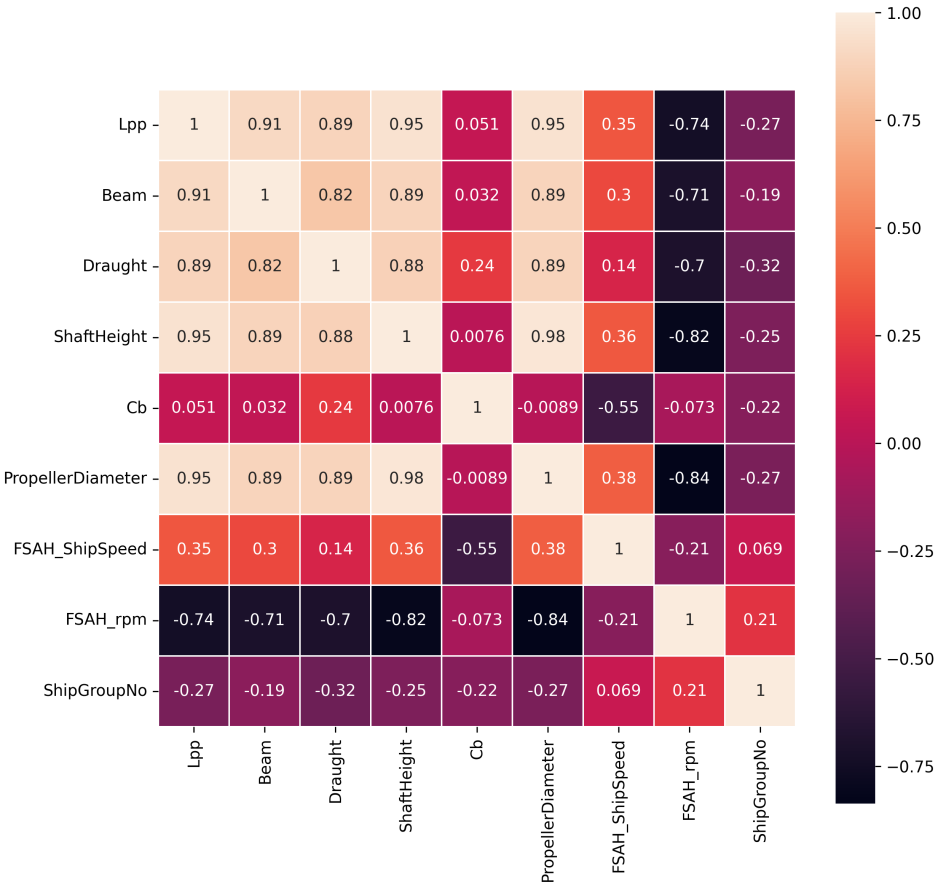


Figure B.4: Linear correlation heatmap of all filtered variables in the Wärtsilä Archimedes dataset.

C

Discrete Cosine Transform Example

This appendix supports section 4.3.2 by providing additional information and figures related to the DCT transformation shown in figure 4.4.

The original wake field graph was given at $\theta = [0, 10, \ldots, 180]$ degrees and had the following values at those angles: V_x/V_s = `[0.343, 0.443, 0.491, 0.546, 0.660, 0.753, 0.809, 0.837, 0.849, 0.848, 0.837, 0.824, 0.814, 0.795, 0.761, 0.731, 0.706, 0.684, 0.675]`.

To obtain the first 91 DCT coefficients, a cubic spline interpolation was performed on this V_x/V_s graph, and the resulting spline was sampled at 91 equally spaced intervals corresponding to taking 91 steps from 0 to 180 degrees with a step size of 2. Using the `scipy.fft.dct` Python function [95], the first 91 DCT coefficients were obtained. The first 25 DCT coefficients are shown below.

```
[6.816, -0.663, -1.072, -0.308, -0.169, 0.00645, 0.0196, -0.00477, -0.0133, -0.0293
-0.0578, 0.0452, -0.0420, -0.0377, -0.0230, -0.0191, -0.0115, -0.00837, -0.00724,
-0.00625, -0.00644, -0.00600, -0.00493, -0.00394, -0.00239]
```

To compare the influence of taking the first n DCT coefficients on the quality of the reconstructed graph, the first n DCT coefficients were taken for $n = [3, 5, 10, 12, 15, 19]$ and zero-padded to a length of 91. The `scipy.fft.idct` inverse DCT function was then used to back-transform the graph. The results are shown in figures C.1 to C.6. The mean squared error between the original and reconstructed data points was calculated for an increasing number of DCT coefficients taken into account. This can be seen in figure C.7. It can be observed that the first coefficients contain the majority of the information, as the error drops rapidly after the first few coefficients.

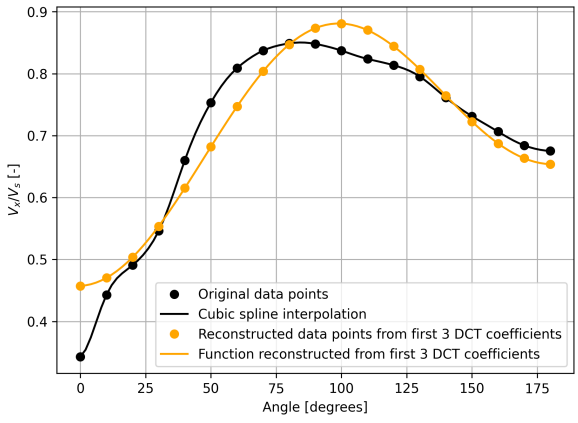


Figure C.1: Wake field graph approximated using its first 3 DCT coefficients.

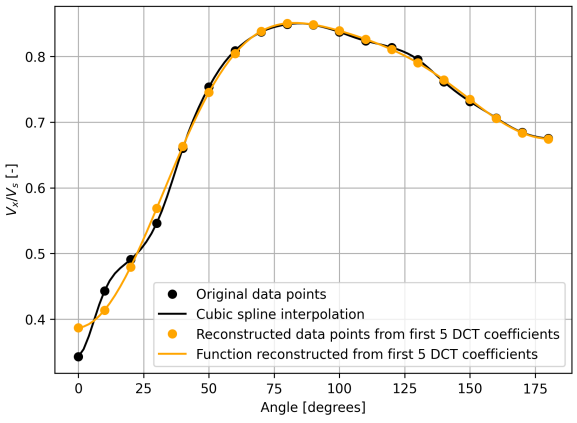


Figure C.2: Wake field graph approximated using its first 5 DCT coefficients.

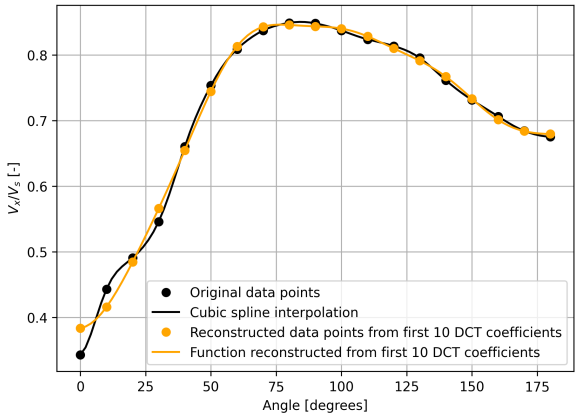


Figure C.3: Wake field graph approximated using its first 10 DCT coefficients.

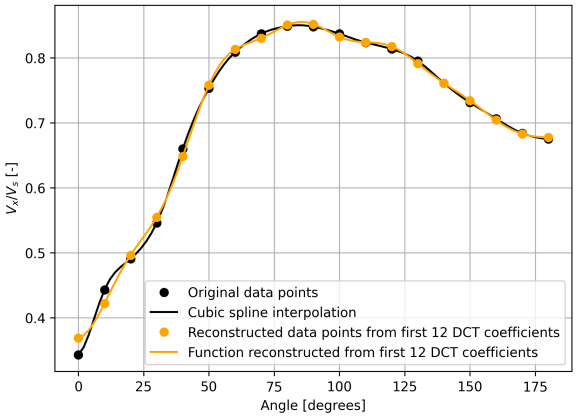


Figure C.4: Wake field graph approximated using its first 12 DCT coefficients.

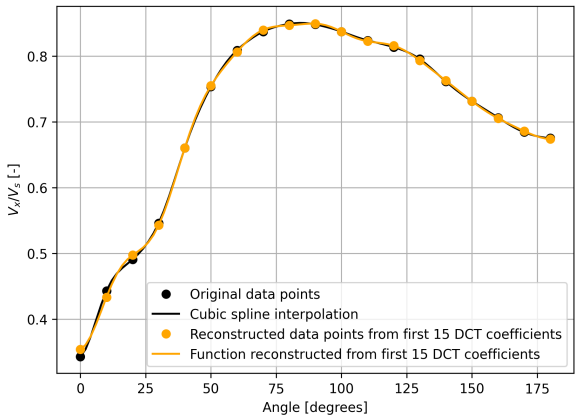


Figure C.5: Wake field graph approximated using its first 15 DCT coefficients.

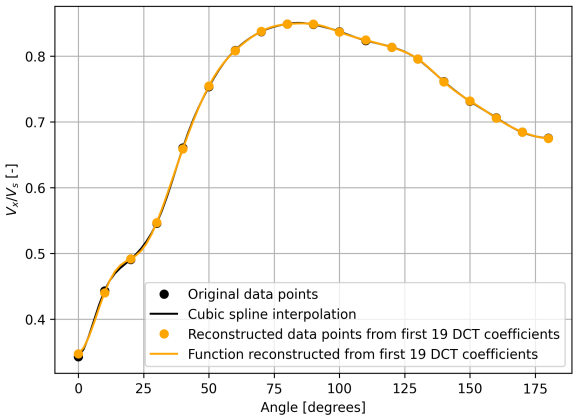


Figure C.6: Wake field graph approximated using its first 19 DCT coefficients.

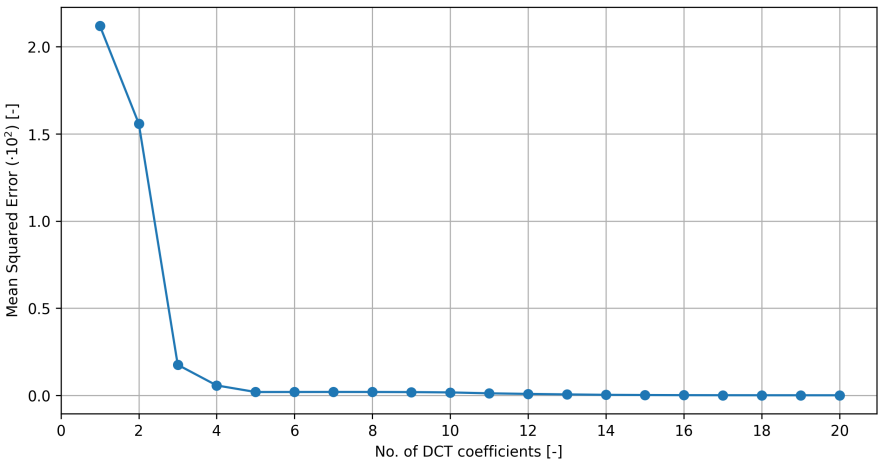


Figure C.7: Mean squared error (MSE) between the original and reconstructed data points of the discrete cosine transformed wake field graph as a function of the number of DCT coefficients used.

D

Feature Characteristics

After presenting characteristics on the unfiltered and filtered Wärtsilä Archimedes dataset in appendices A and B, respectively, in this appendix the characteristics of the engineered features will be presented. This appendix supports section 5.6, where table D.1 and figure D.1 are also shown. In the following table and visualisations the one-hot encoded categorical features ShipGroupNo and WakeScale have been omitted, as their categorical and not their numerical value is of interest.

Table D.1: Descriptive statistics of the engineered features.

	mean	std	min	25%	50%	75%	max
Cb	0.7060	0.0938	0.4420	0.6485	0.7087	0.7800	0.8850
PropellerDiameter	5631	1564	2800	4500	5400	6500	9600
FSAH_ShipSpeed	15.73	3.281	7.435	13.79	15.95	17.90	23.51
FSAH_rpm	119.2	31.12	68.20	96.70	114.0	137.7	229.4
LB	5.925	0.8070	3.367	5.446	5.988	6.422	8.812
BT	3.086	0.6332	2.110	2.617	3.036	3.350	6.508
ShaftHeightT	0.3533	0.0499	0.2231	0.3184	0.3521	0.3933	0.4732
Fn	0.2164	0.0515	0.0960	0.1878	0.2218	0.2421	0.3486
Re	1.086e9	5.634e8	2.235e8	6.748e8	9.501e8	1.378e9	3.299e9
Displacement	32839	37188	1709	10284	20162	39574	204529
Delta_Bertram	60817	101057	585.6	8140	22209	69046	772855
Cm_HSVA	0.9805	0.0211	0.8851	0.9749	0.9868	0.9950	0.9995
CNabla	0.0071	0.0025	0.0028	0.0056	0.0068	0.0083	0.0210
KB_Schneekluth	4.503	1.275	2.800	3.612	4.214	4.836	8.307
CF_ITTC	0.0015	0.0001	0.0013	0.0015	0.0015	0.0016	0.0019
PD_Volker	8733	9609	228.9	2672	4981	12449	62769
W_Taylor	0.3030	0.0469	0.1710	0.2742	0.3043	0.3400	0.3925

Table D.1 is visualised in figure D.1. It can be seen that some of the engineered features have a skewed distribution, such as Displacement and Delta_Bertram. This is not necessarily a problem, as standardisation rather than normalisation was used to scale all features. Standardisation is less prone to outliers in the dataset. The specific distributions of all features can be seen in figure D.2. Density plots between all combinations of features are shown in figure D.3. It can be seen that no feature has a strong linear relationship with another feature, although some higher-order relations seem to exist. Lastly, in figure D.4 a linear correlation heatmap is shown.

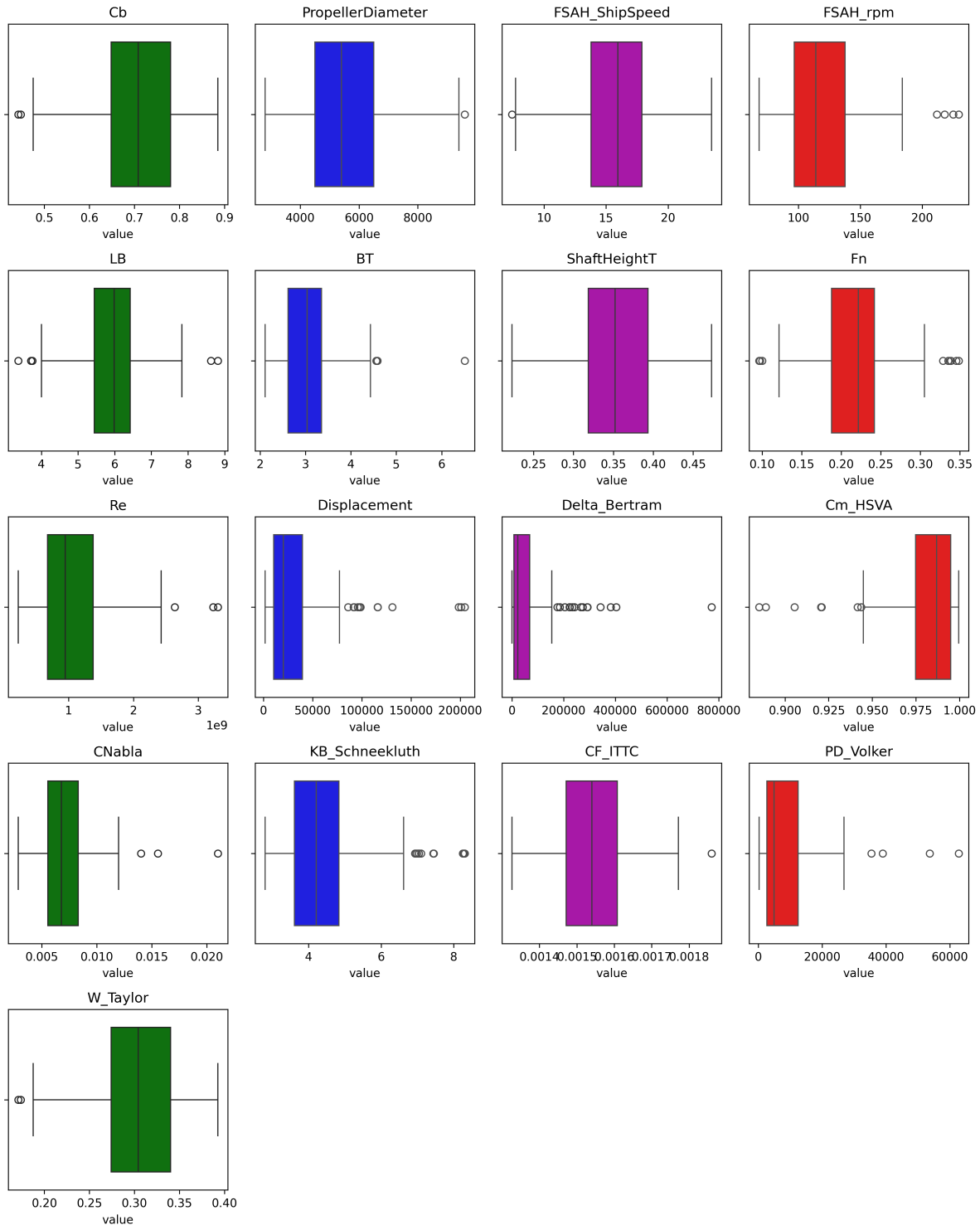


Figure D.1: Box plots of all engineered features.

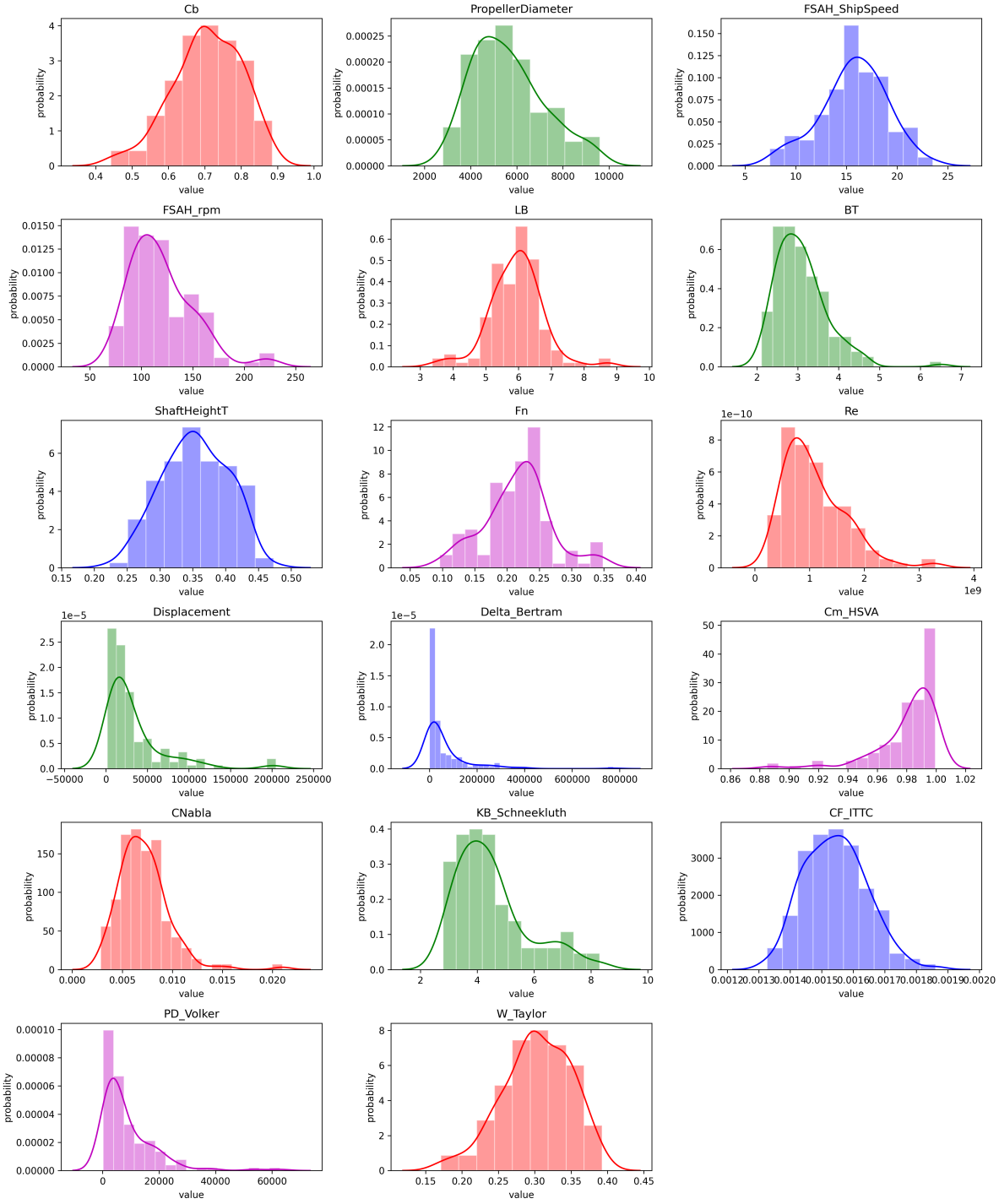


Figure D.2: Univariate distribution plots of all engineered features.

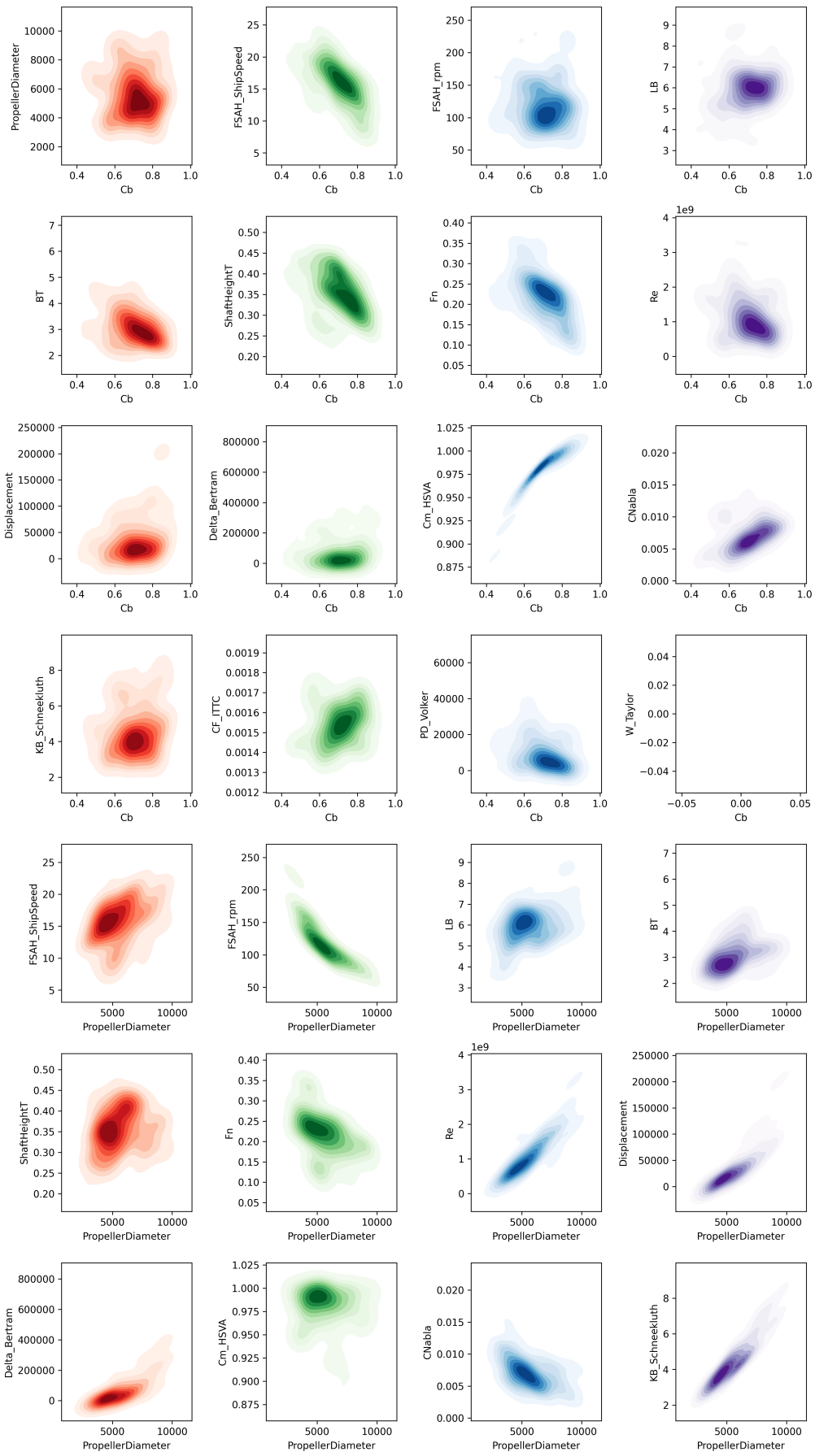


Figure D.3: Density plots of all engineered features.

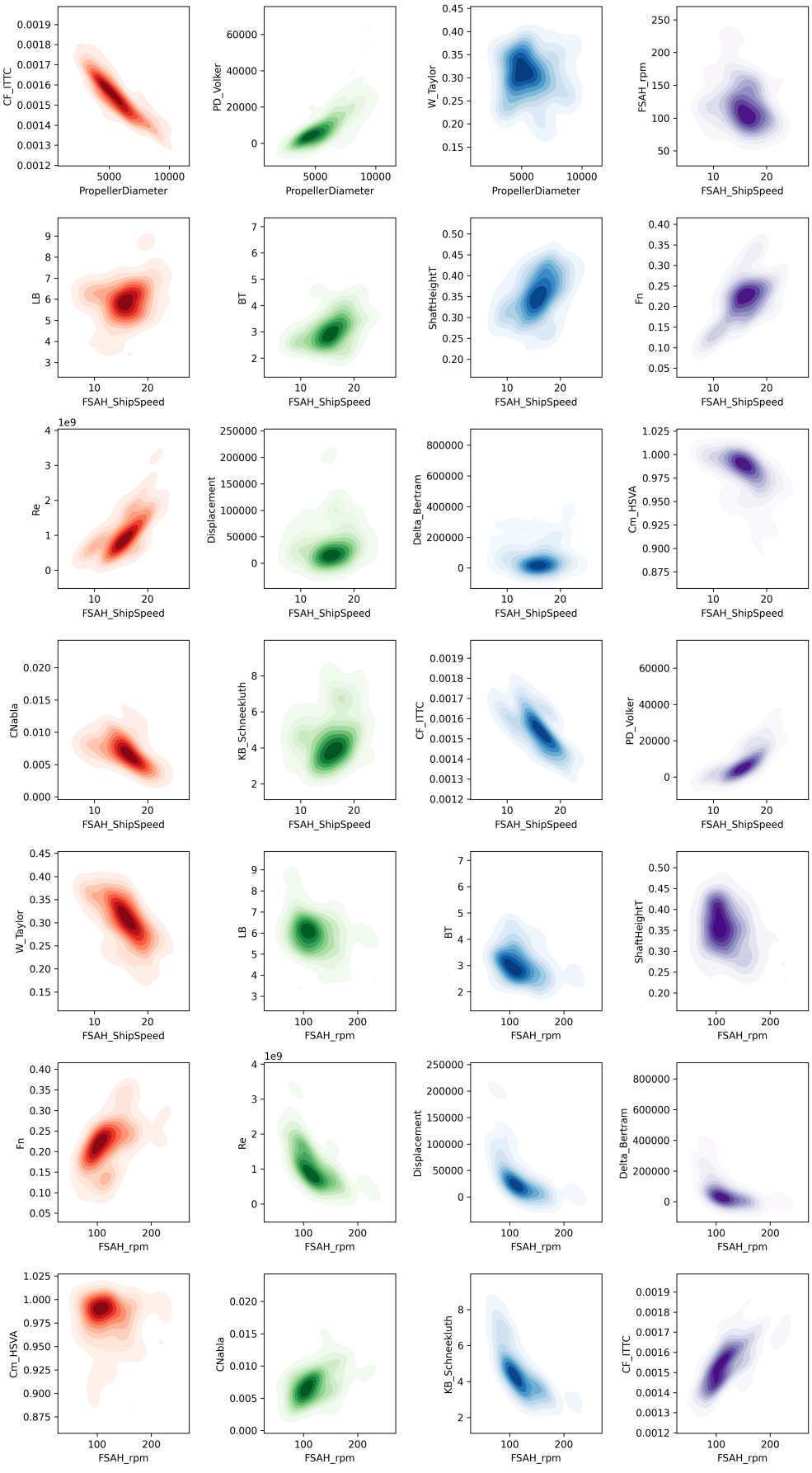


Figure D.3 (continued).

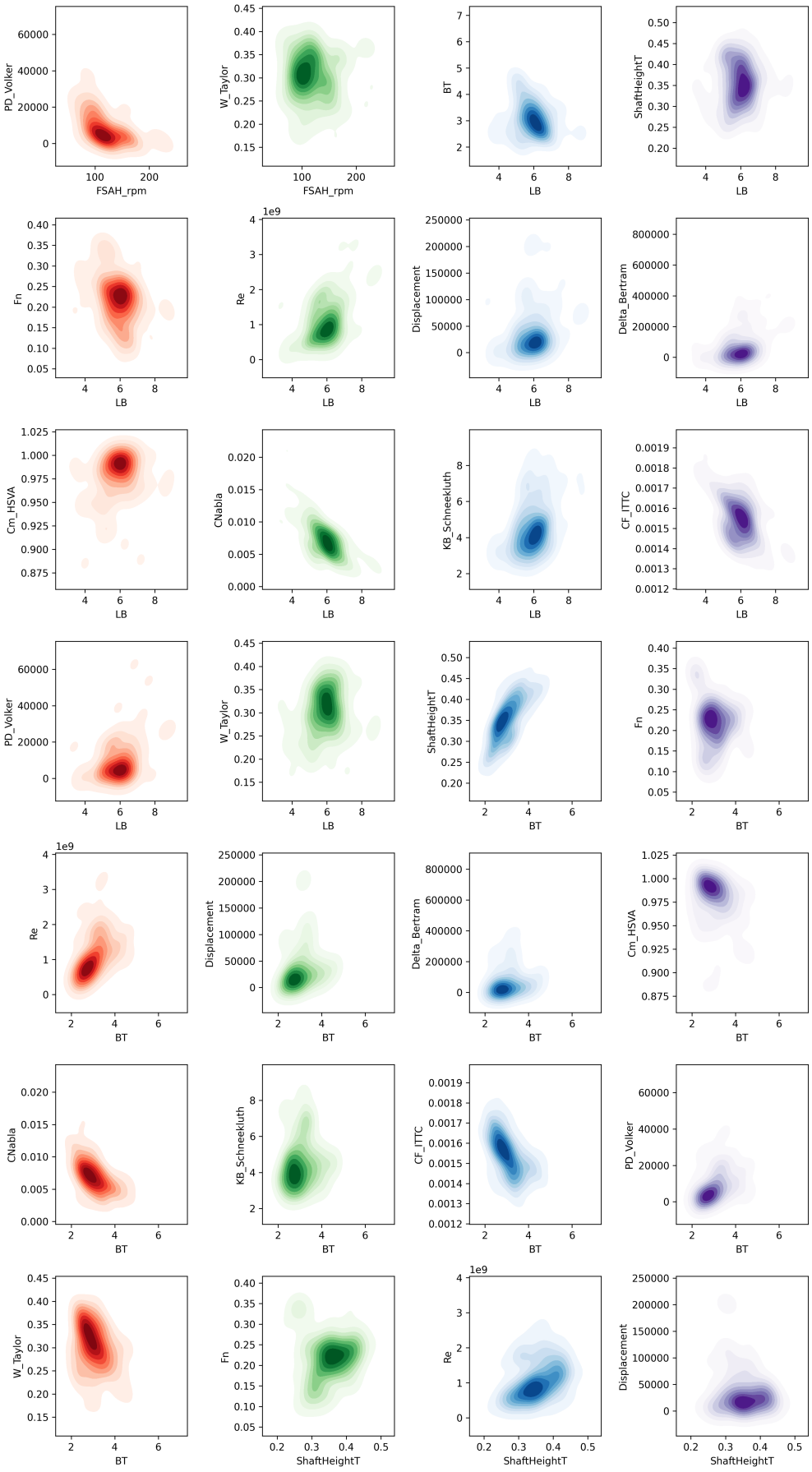


Figure D.3 (continued).

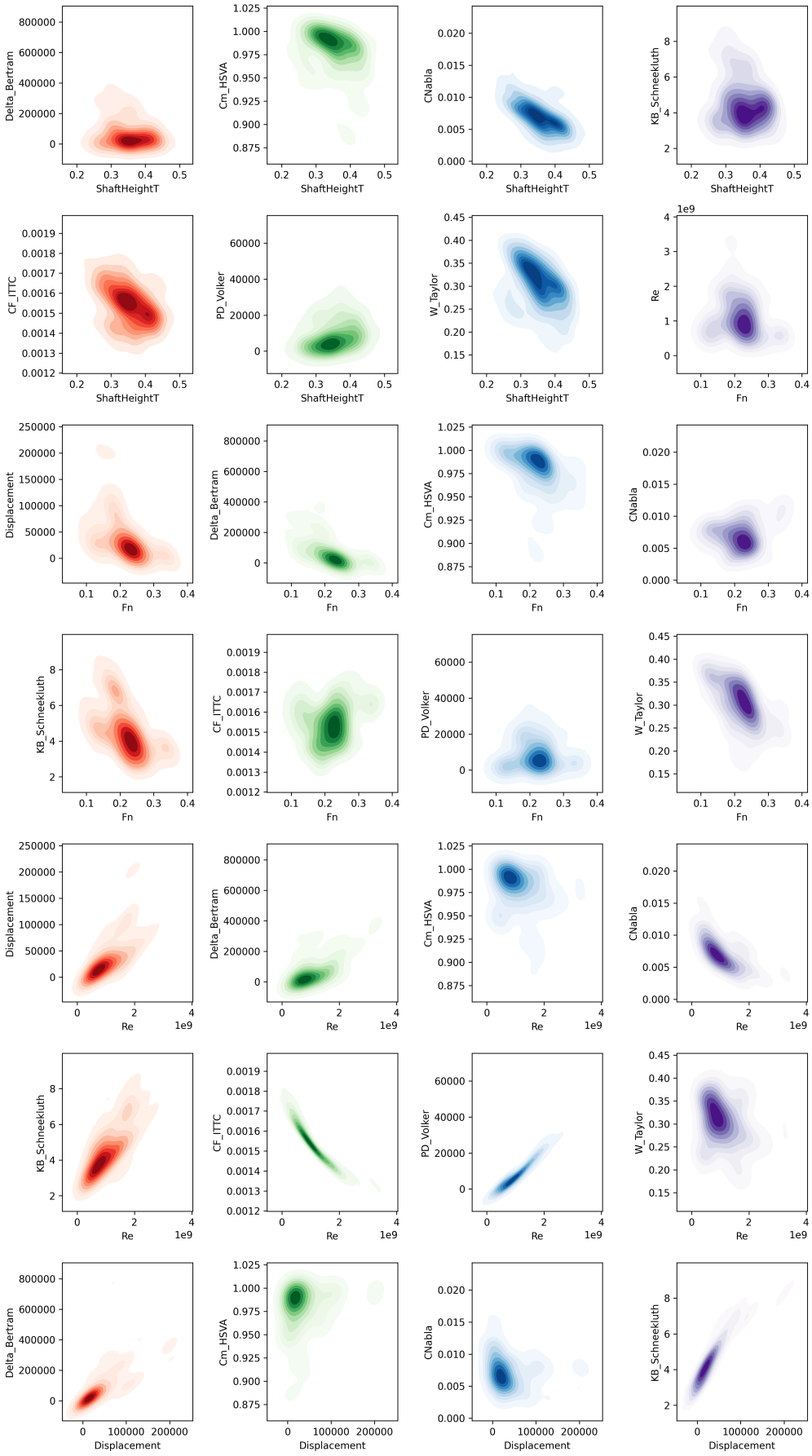


Figure D.3 (continued).

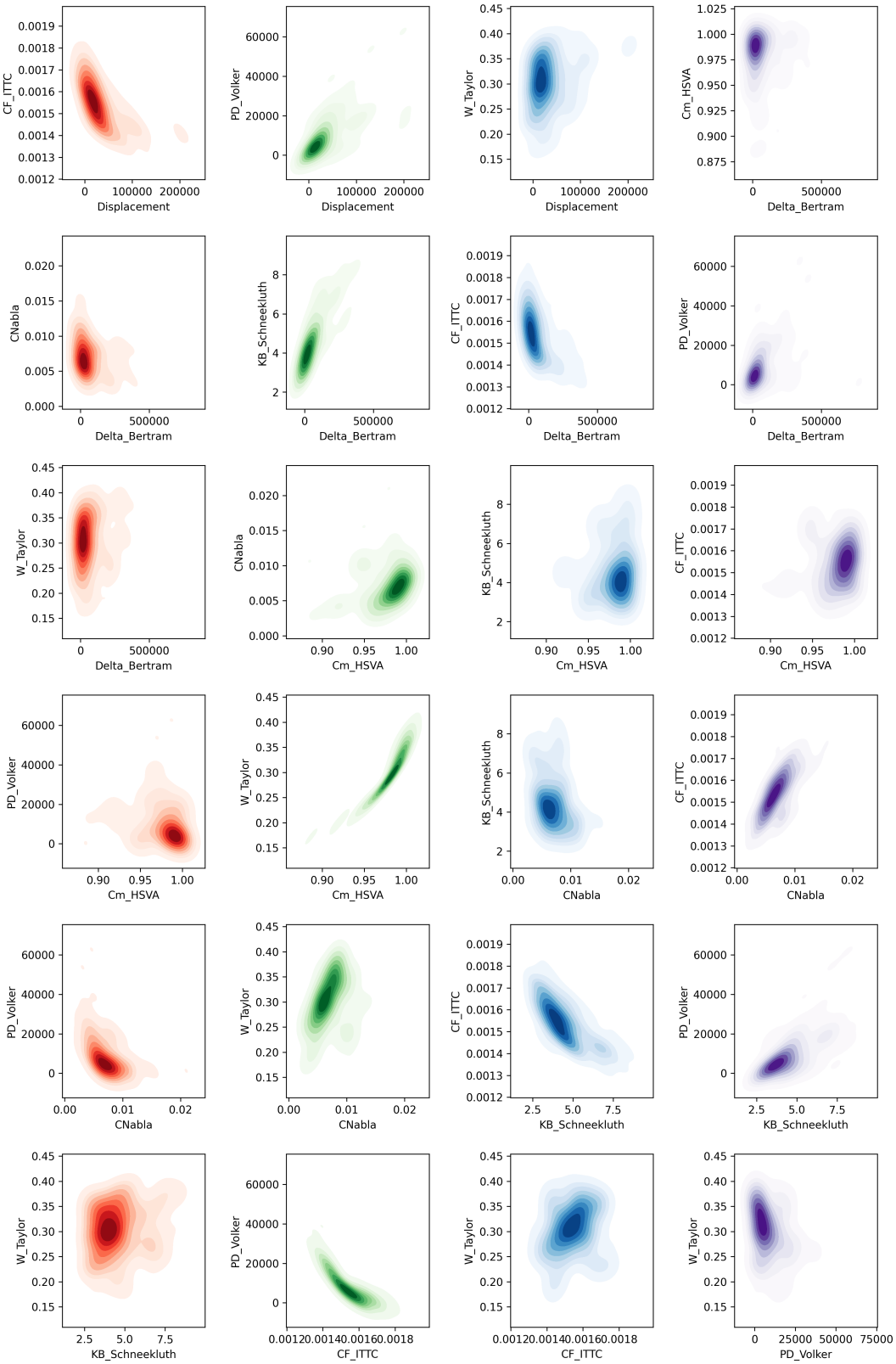


Figure D.3 (continued).

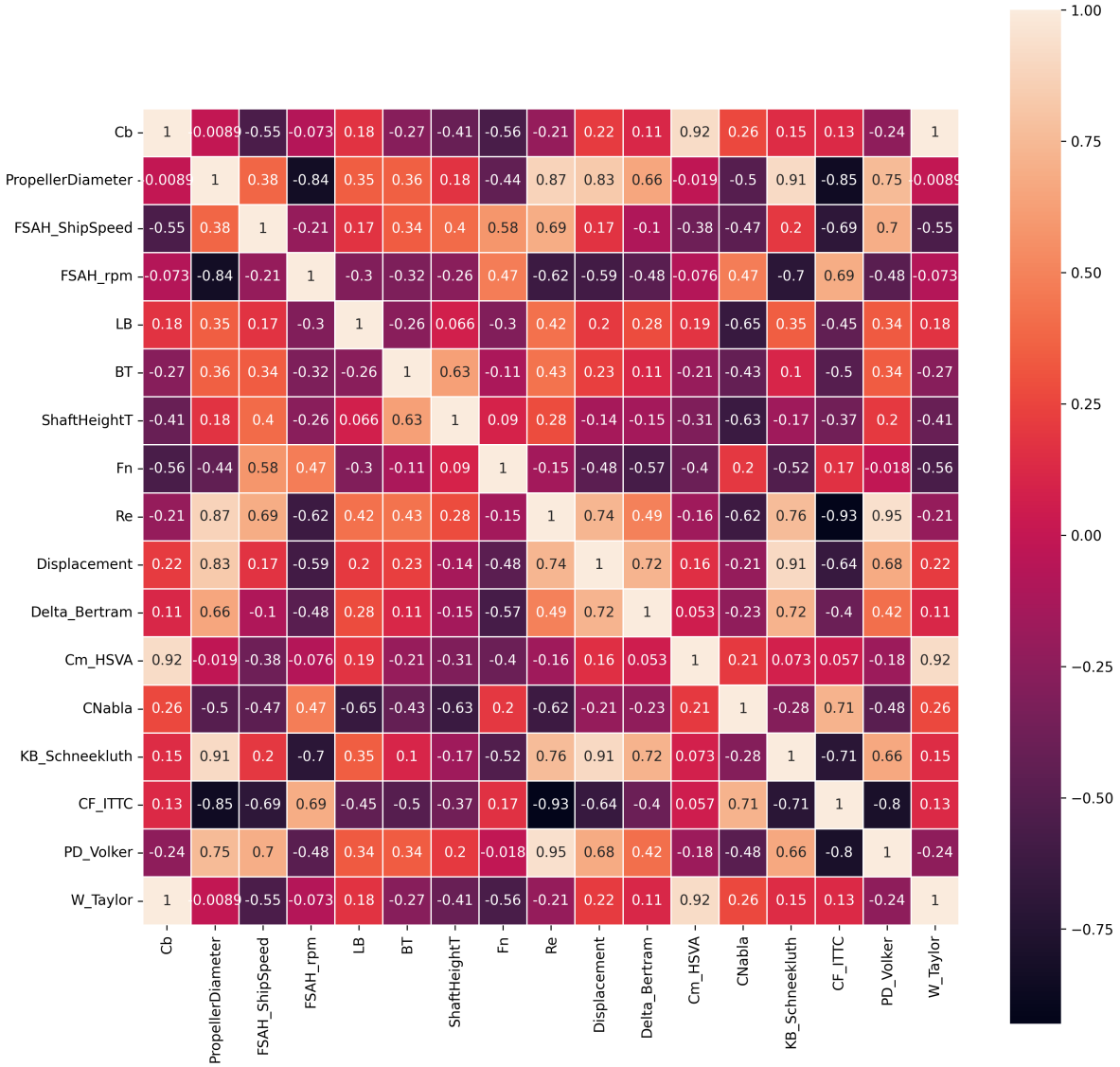


Figure D.4: Linear correlation heatmap of all engineered features.

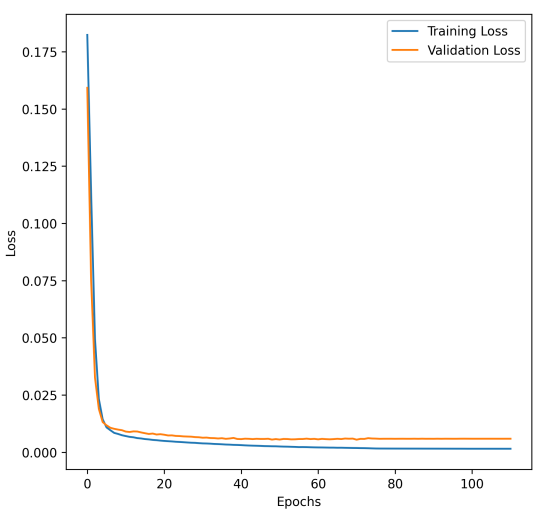
E

Model Training Metrics

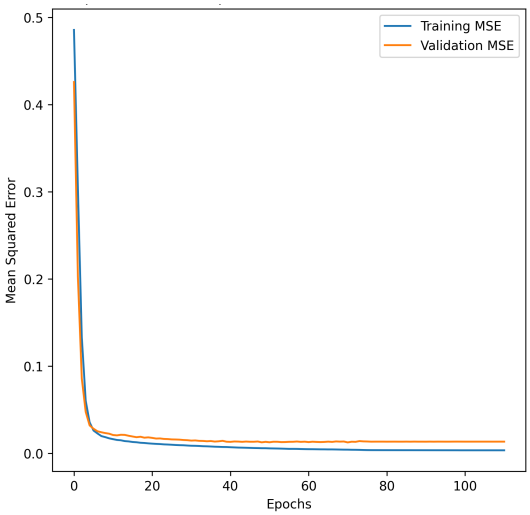
In this appendix, the training metrics for all trained models are presented. The following metrics are logged during training:

- Training and validation loss (custom combined loss function)
- Training and validation MSE
- Training and validation MAPE
- Validation PPMCC
- Learning rate

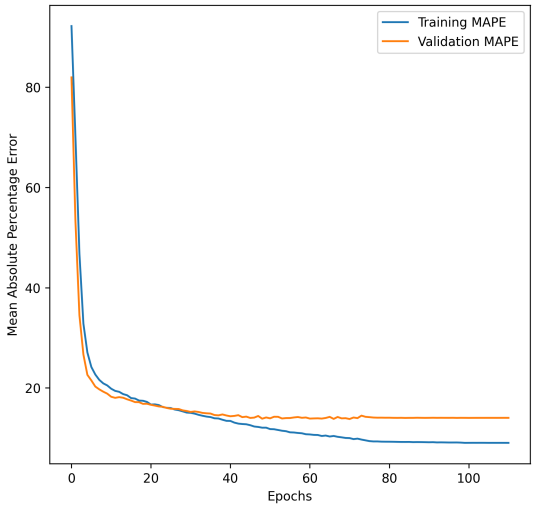
The metrics for the “vanilla”, DCT and DCT+standardisation feed-forward neural networks can be seen in figures E.1, E.2, and E.3. The metrics for the ensemble method are shown in figure E.4, and those for for the long short-term memory model in figure E.5.



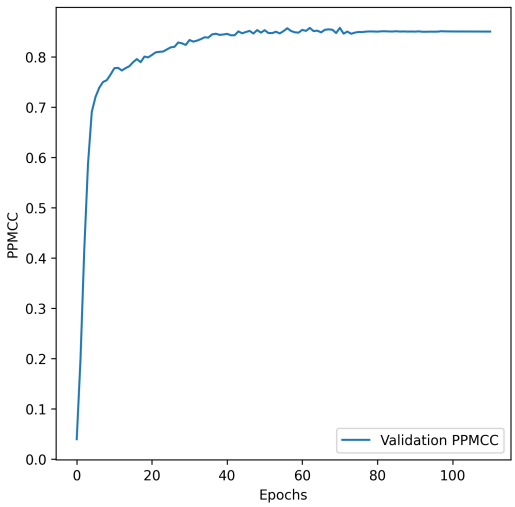
(a) Custom combined loss



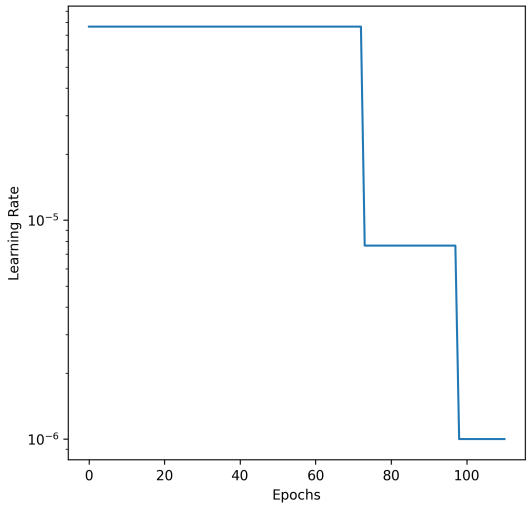
(b) Mean squared error



(c) Mean absolute percentage error

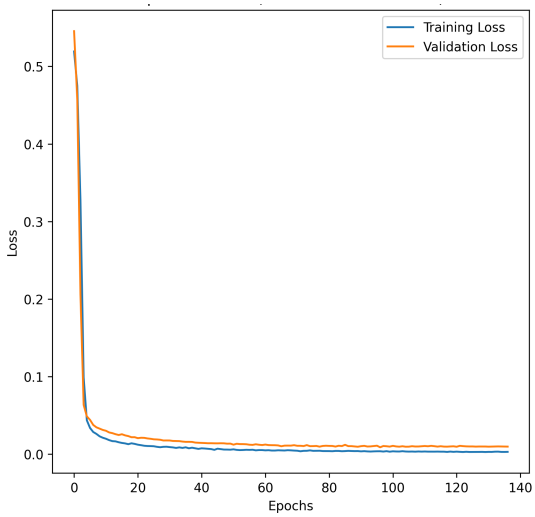


(d) PPMCC

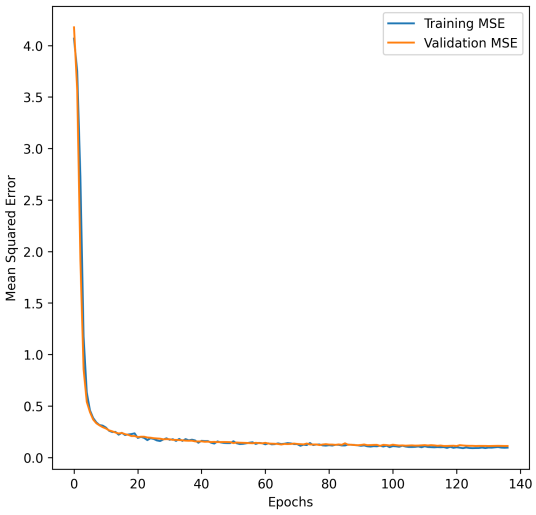


(e) Learning rate

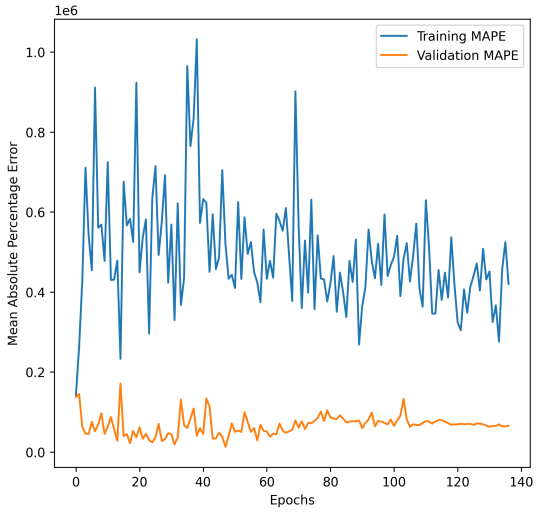
Figure E.1: Feed-forward neural network training metrics.



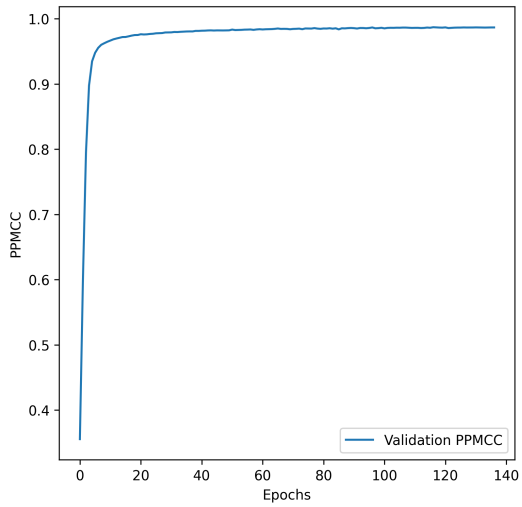
(a) Custom combined loss



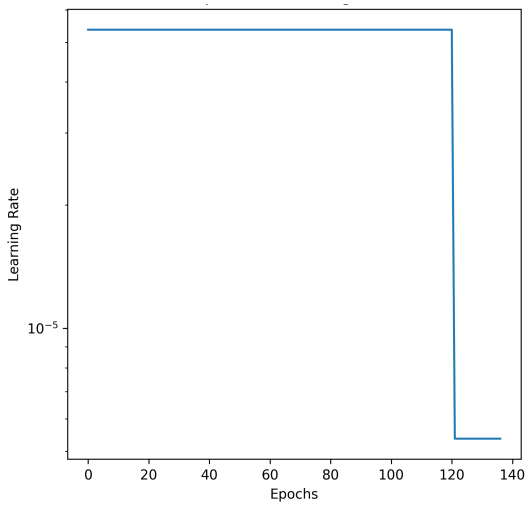
(b) Mean squared error



(c) Mean absolute percentage error

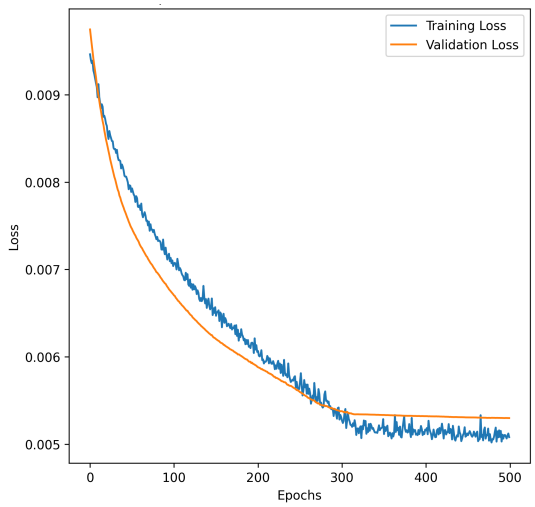


(d) PPMCC

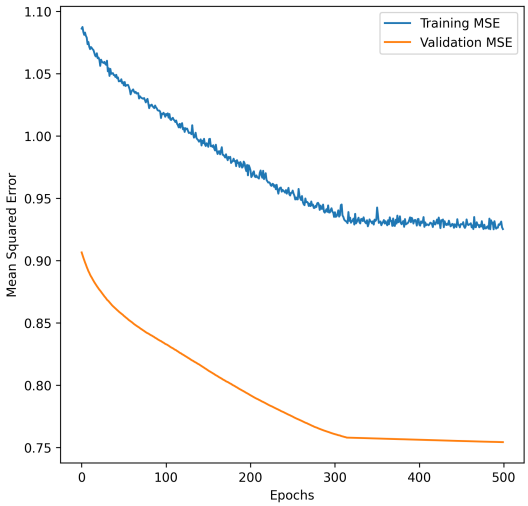


(e) Learning rate

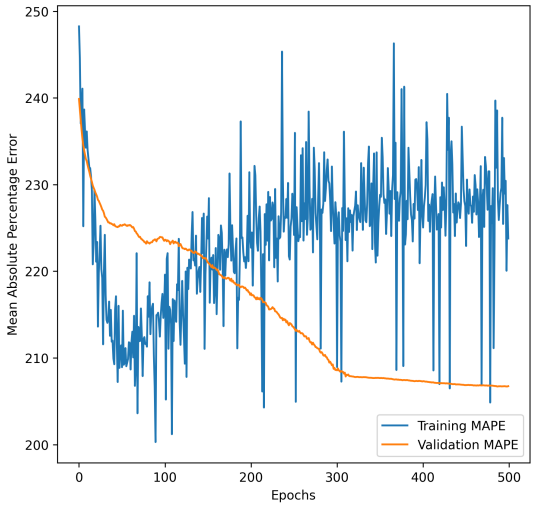
Figure E.2: Feed-forward neural network training metrics, using DCT.



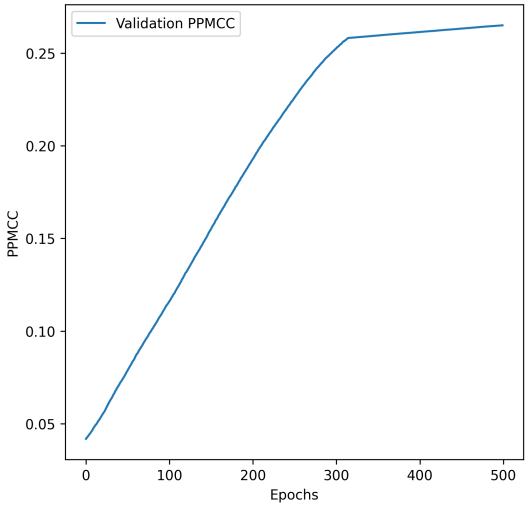
(a) Custom combined loss



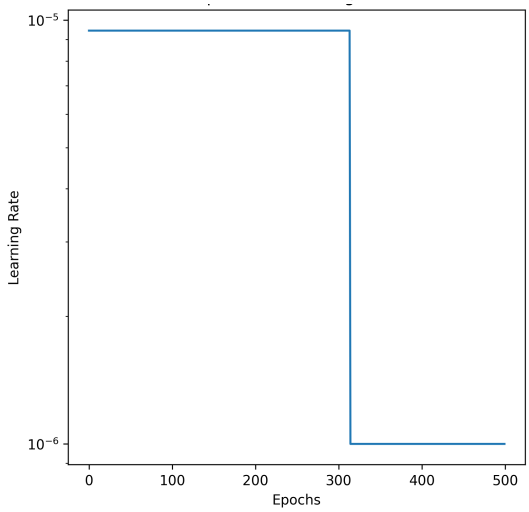
(b) Mean squared error



(c) Mean absolute percentage error

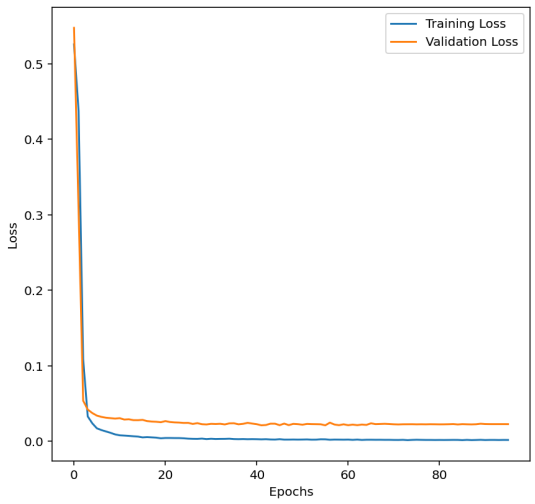


(d) PPMCC

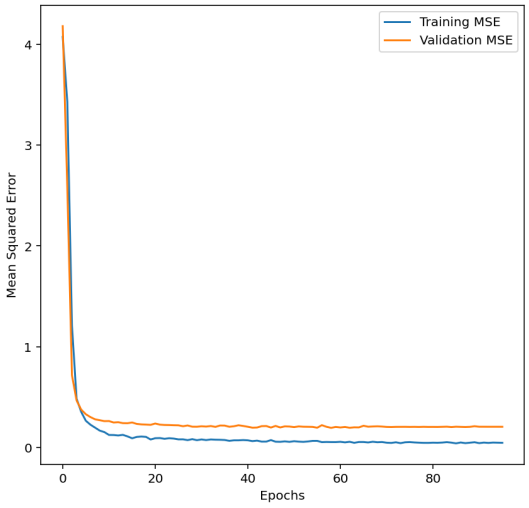


(e) Learning rate

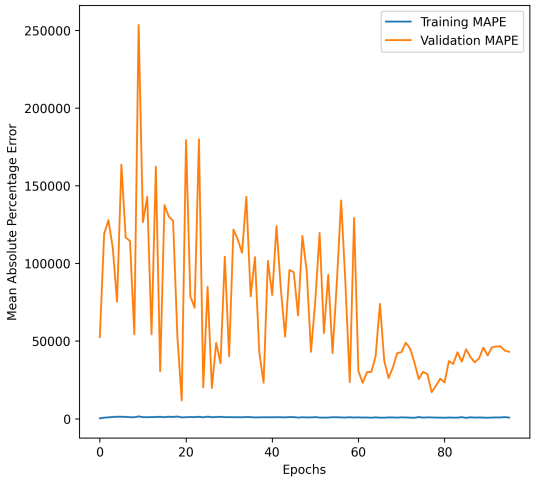
Figure E.3: Feed-forward neural network training metrics, using DCT and coefficient standardisation.



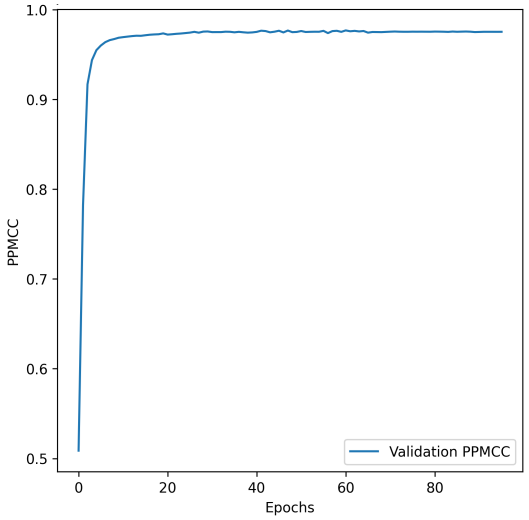
(a) Custom combined loss



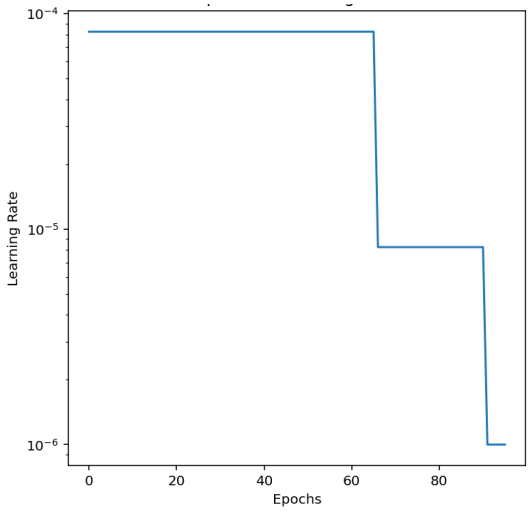
(b) Mean squared error



(c) Mean absolute percentage error

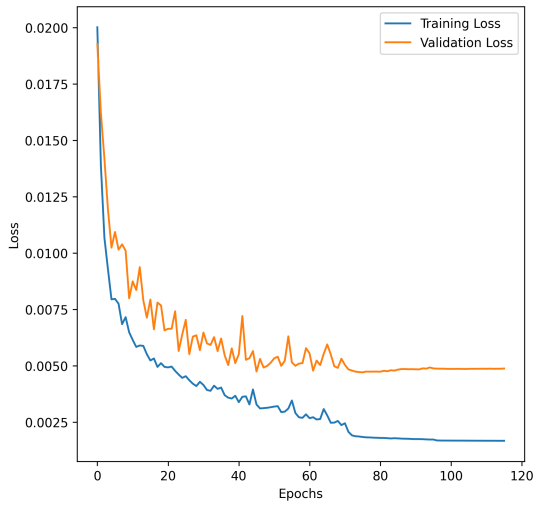


(d) PPMCC

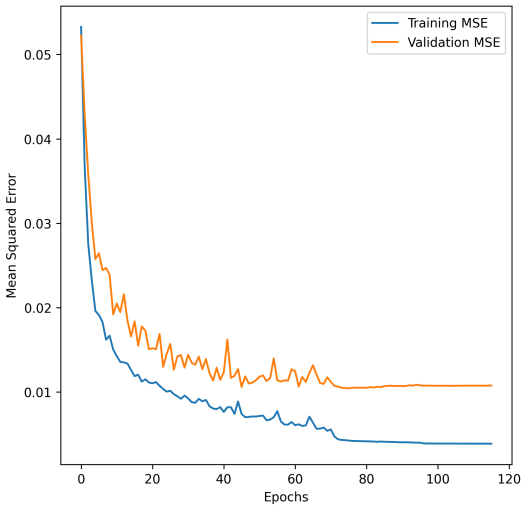


(e) Learning rate

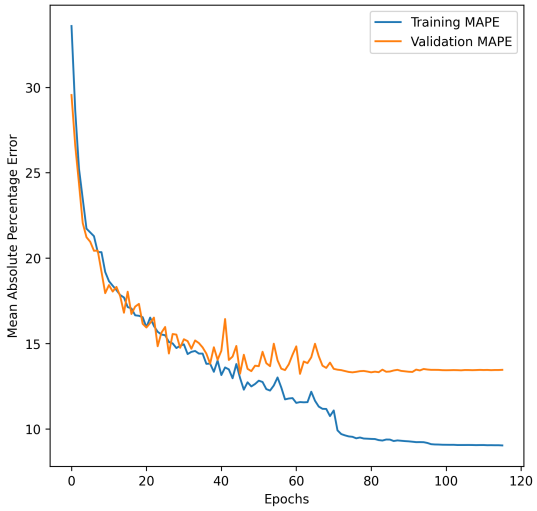
Figure E.4: Ensemble model training metrics.



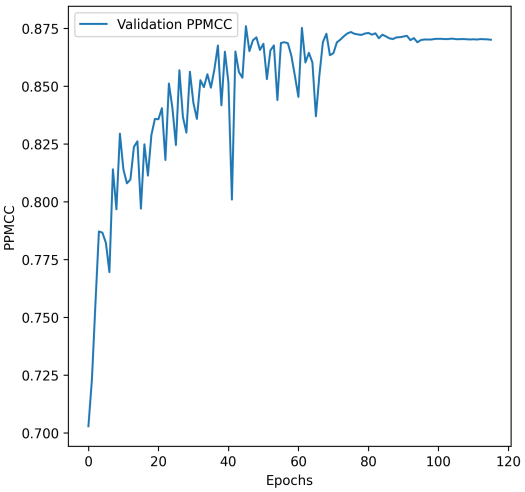
(a) Custom combined loss



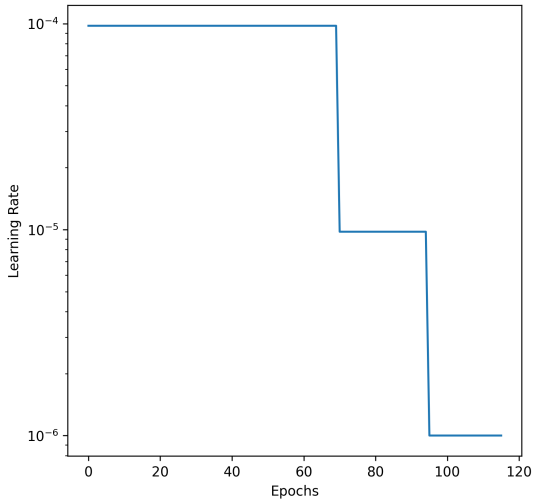
(b) Mean squared error



(c) Mean absolute percentage error



(d) PPMCC



(e) Learning rate

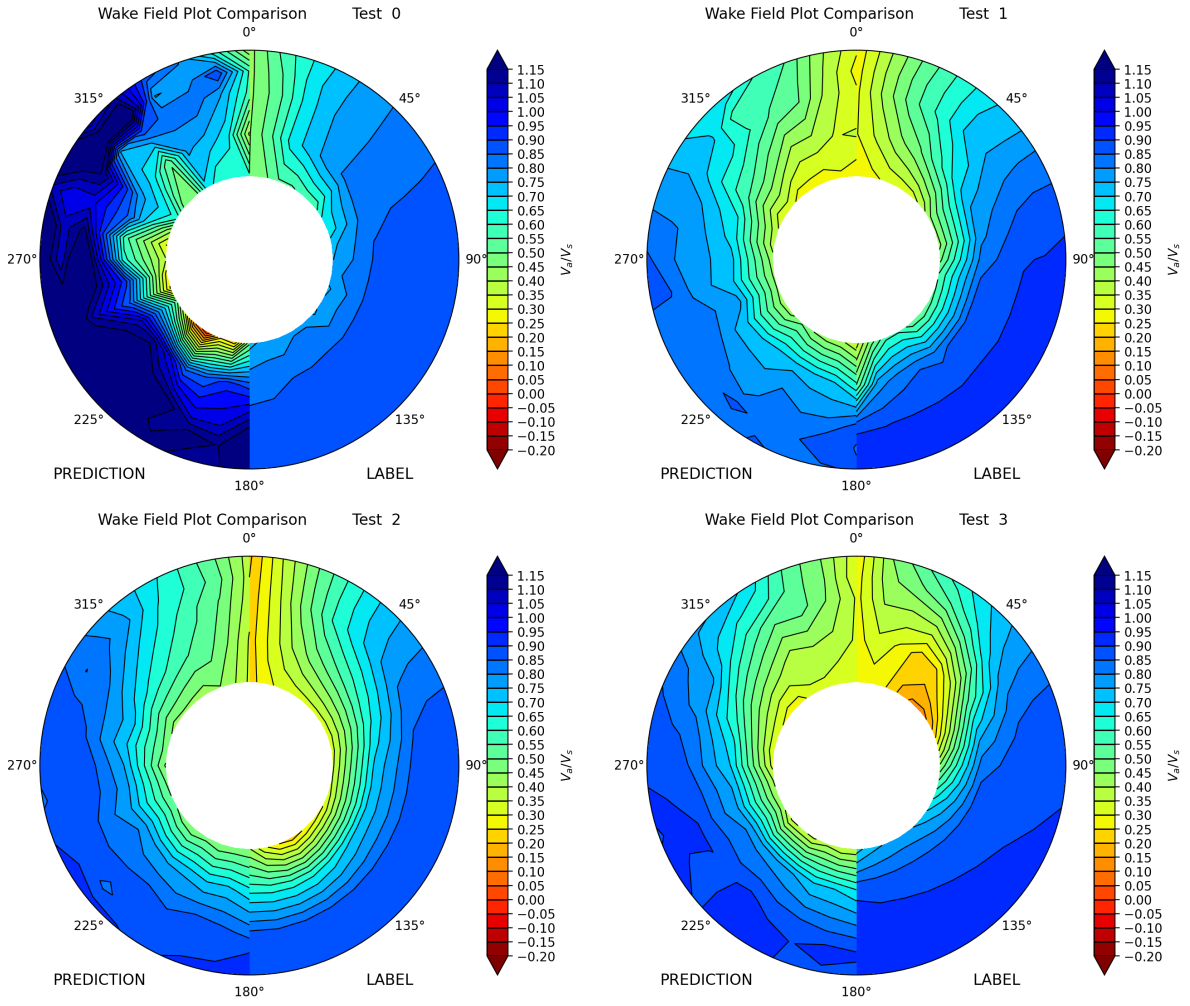
Figure E.5: Long short-term memory model training metrics.

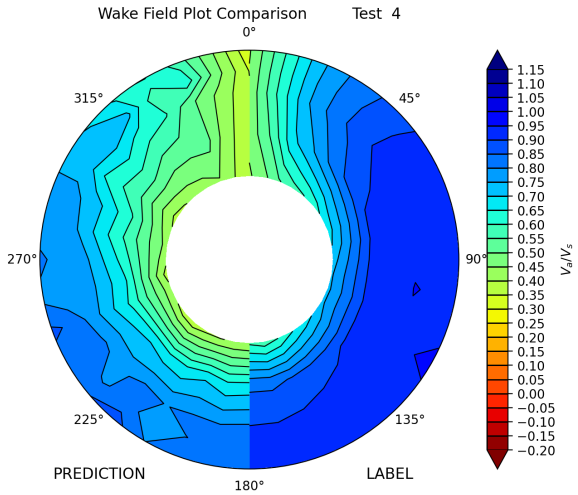
F

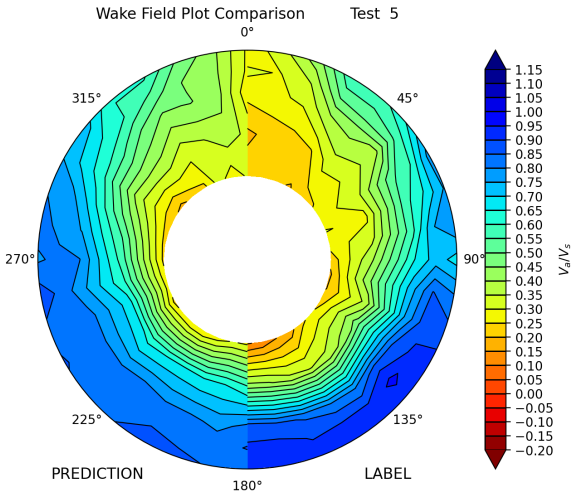
Predicted Wake Fields

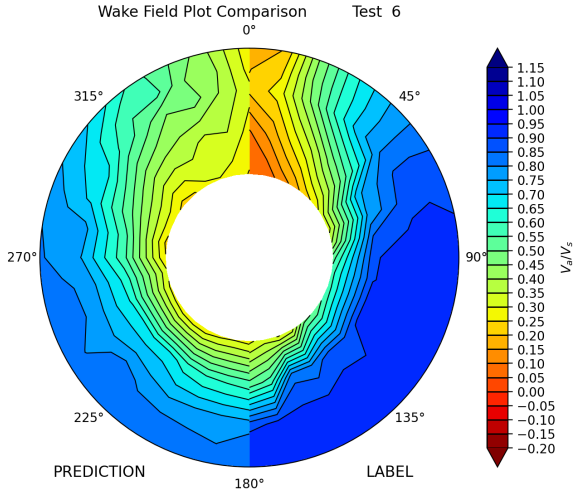
In this appendix, all predictions performed on the test set by all trained models are shown. In sections F.1, F.2, and F.3 the predictions made by the different feed-forward neural networks can be found. The ensemble and LSTM model predictions can be seen in sections F.4 and F.5, respectively.

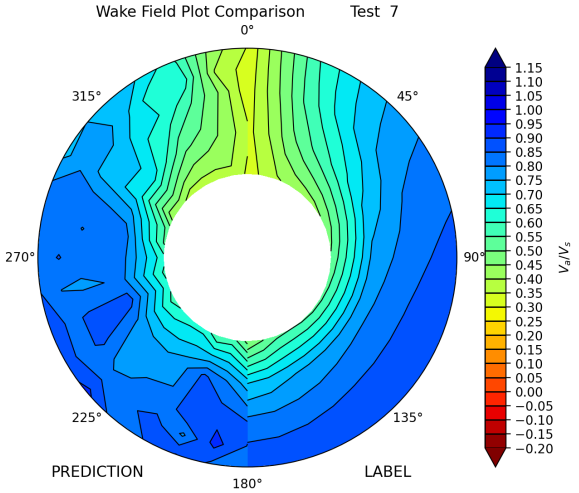
F.1. Feed-Forward Neural Network Predictions

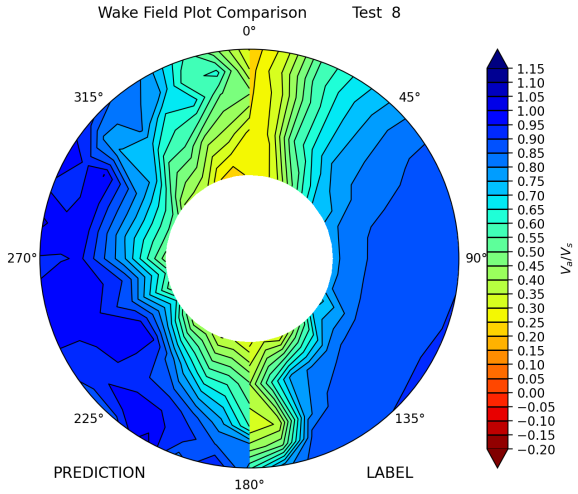


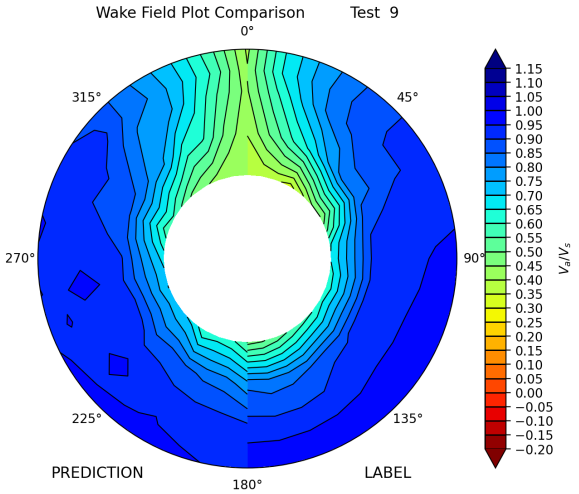

Wake Field Plot Comparison Test 4
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15 1.10 1.05 1.00 0.95 0.90 0.85 0.80 0.75 0.70 0.65 0.60 0.55 0.50 0.45 0.40 0.35 0.30 0.25 0.20 0.15 0.10 0.05 0.00 −0.05 −0.10 −0.15 −0.20
V_a/V_s

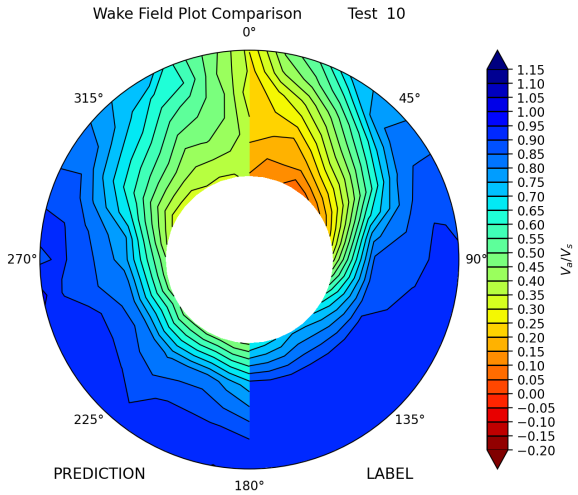

Wake Field Plot Comparison Test 5
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15 1.10 1.05 1.00 0.95 0.90 0.85 0.80 0.75 0.70 0.65 0.60 0.55 0.50 0.45 0.40 0.35 0.30 0.25 0.20 0.15 0.10 0.05 0.00 −0.05 −0.10 −0.15 −0.20
V_a/V_s

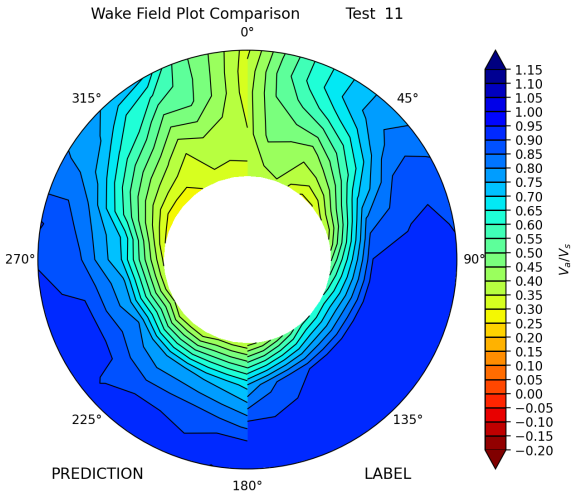

Wake Field Plot Comparison Test 6
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15 1.10 1.05 1.00 0.95 0.90 0.85 0.80 0.75 0.70 0.65 0.60 0.55 0.50 0.45 0.40 0.35 0.30 0.25 0.20 0.15 0.10 0.05 0.00 −0.05 −0.10 −0.15 −0.20
V_a/V_s

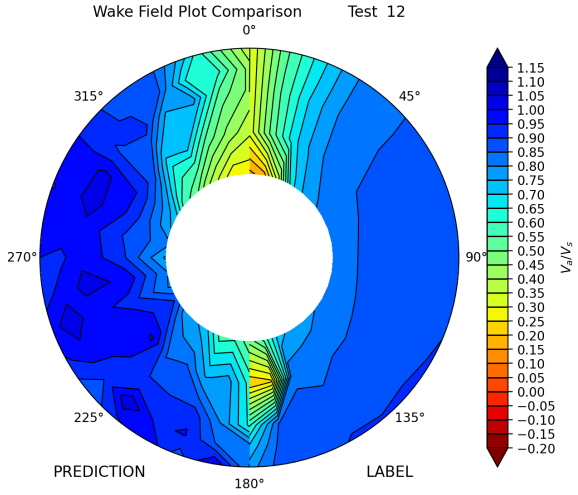

Wake Field Plot Comparison Test 7
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15 1.10 1.05 1.00 0.95 0.90 0.85 0.80 0.75 0.70 0.65 0.60 0.55 0.50 0.45 0.40 0.35 0.30 0.25 0.20 0.15 0.10 0.05 0.00 −0.05 −0.10 −0.15 −0.20
V_a/V_s

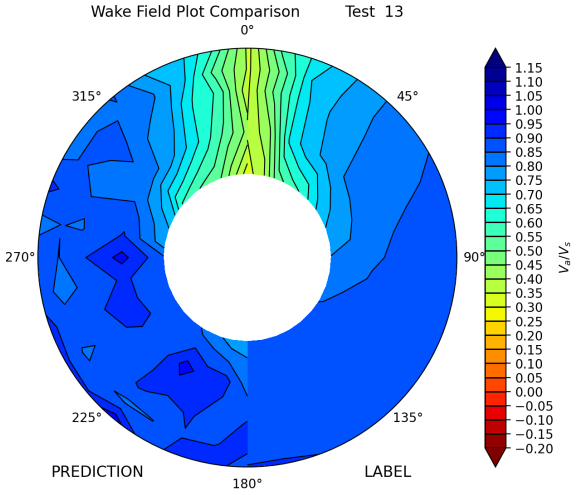

Wake Field Plot Comparison Test 8
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15 1.10 1.05 1.00 0.95 0.90 0.85 0.80 0.75 0.70 0.65 0.60 0.55 0.50 0.45 0.40 0.35 0.30 0.25 0.20 0.15 0.10 0.05 0.00 −0.05 −0.10 −0.15 −0.20
V_a/V_s

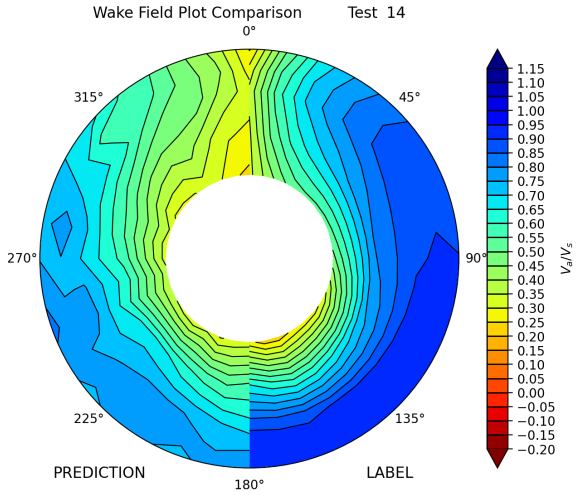

Wake Field Plot Comparison Test 9
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15 1.10 1.05 1.00 0.95 0.90 0.85 0.80 0.75 0.70 0.65 0.60 0.55 0.50 0.45 0.40 0.35 0.30 0.25 0.20 0.15 0.10 0.05 0.00 −0.05 −0.10 −0.15 −0.20
V_a/V_s


Wake Field Plot Comparison Test 10
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
V_d/V_s

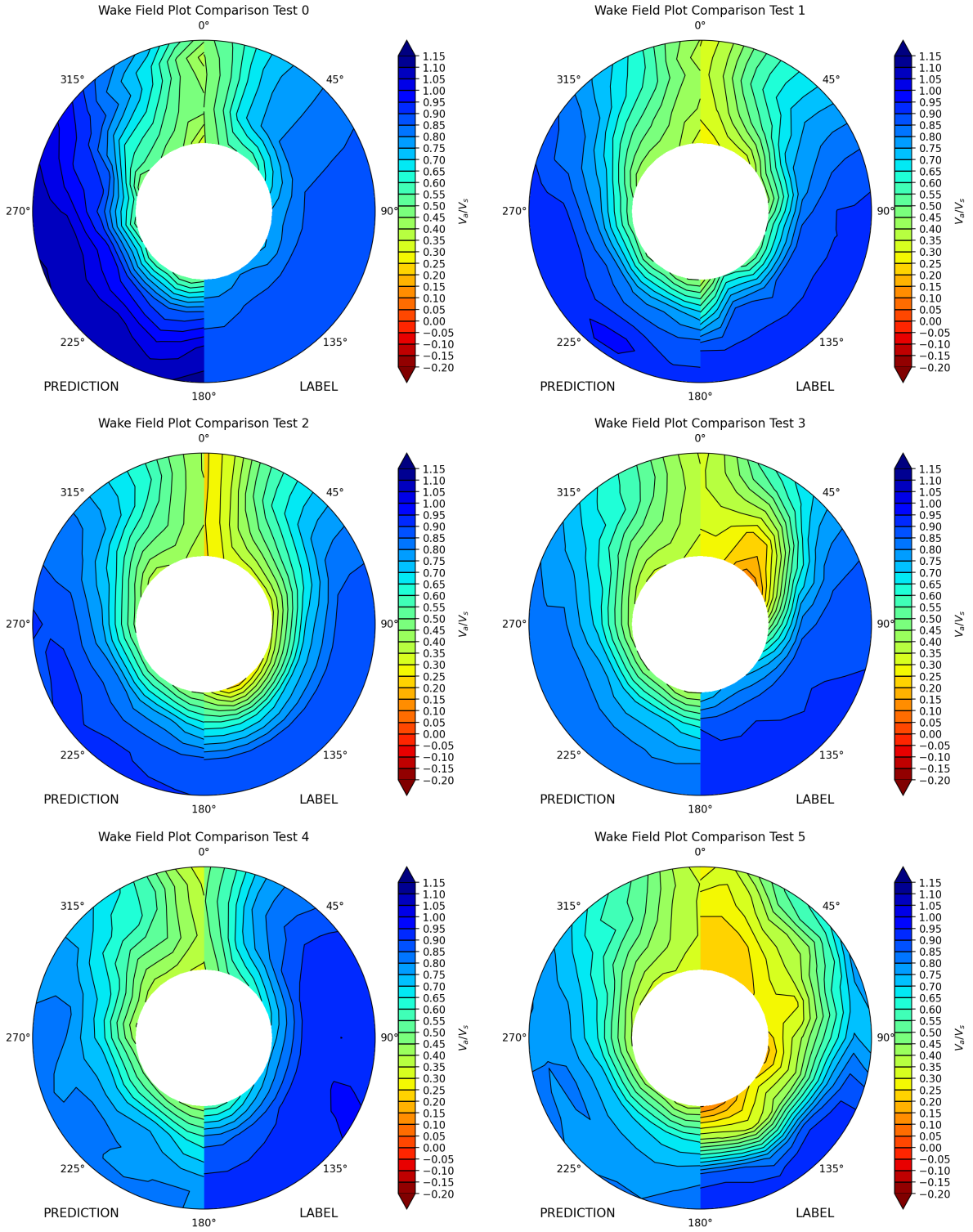

Wake Field Plot Comparison Test 11
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
V_d/V_s

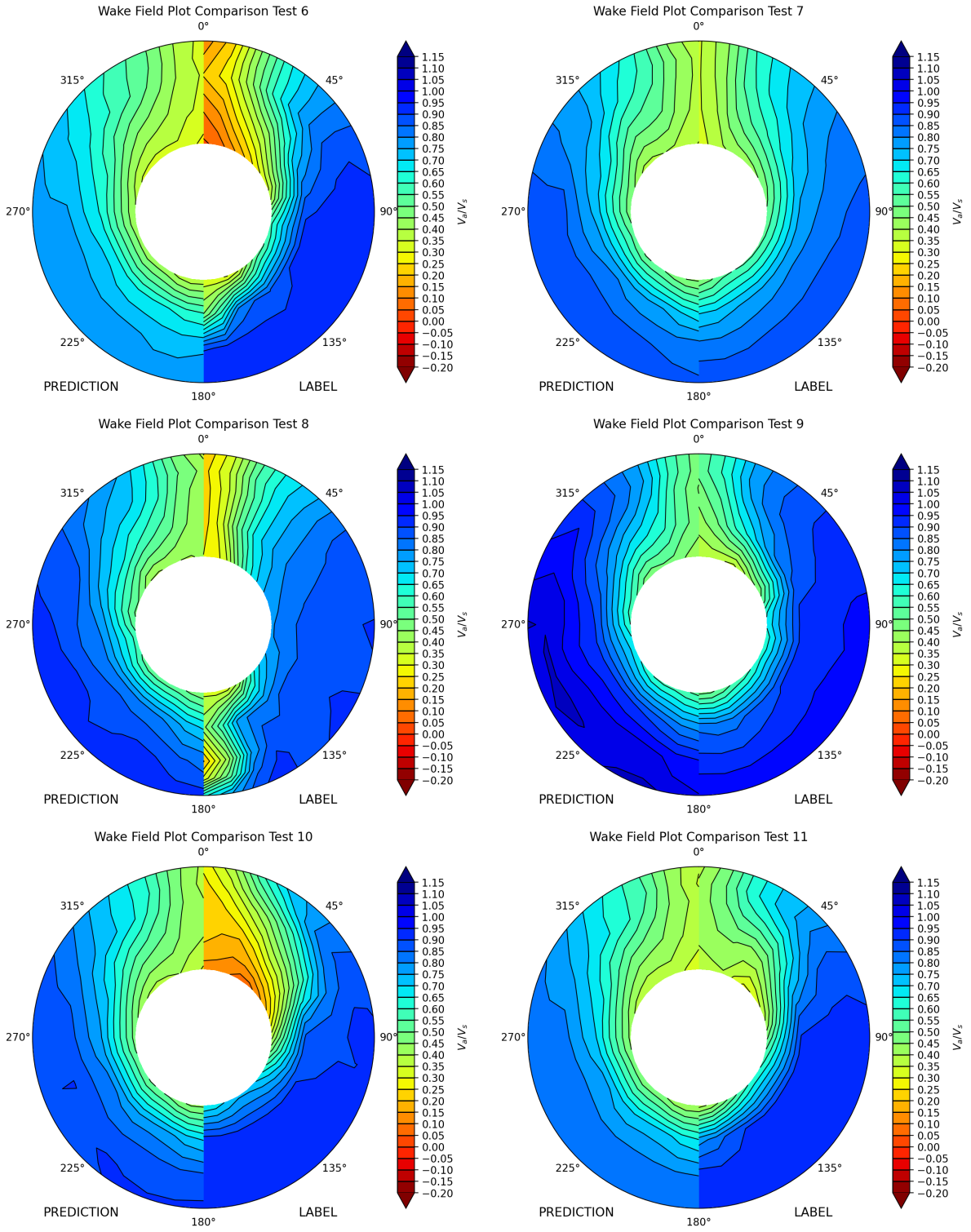

Wake Field Plot Comparison Test 12
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
V_d/V_s

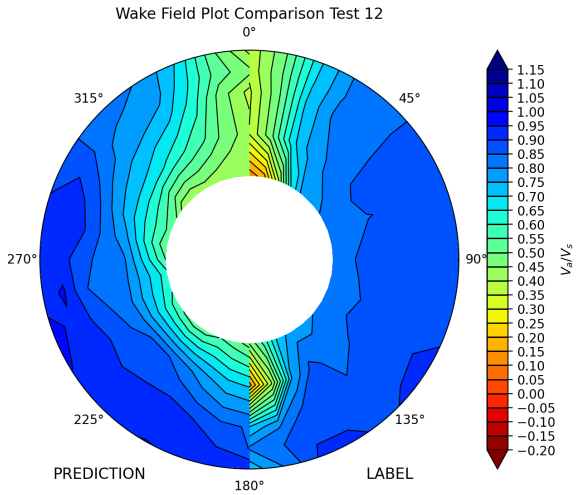

Wake Field Plot Comparison Test 13
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
V_d/V_s

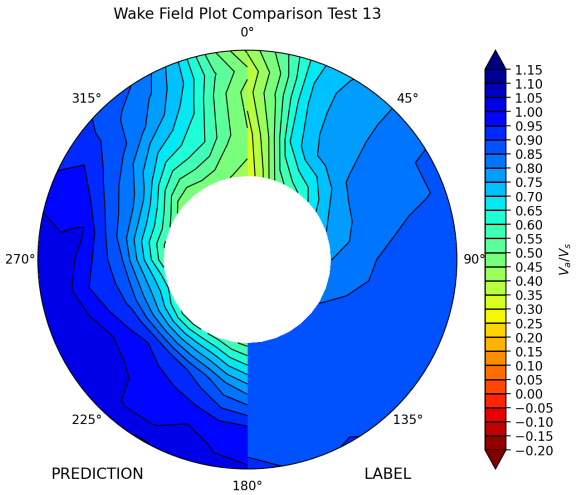

Wake Field Plot Comparison Test 14
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
V_d/V_s

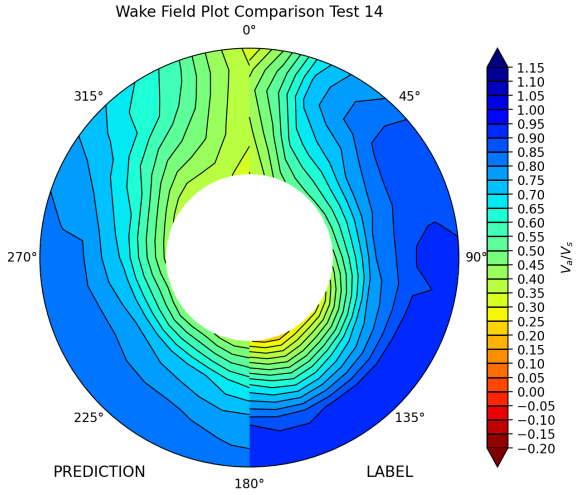
F.2. Feed-Forward Neural Network with DCT Predictions



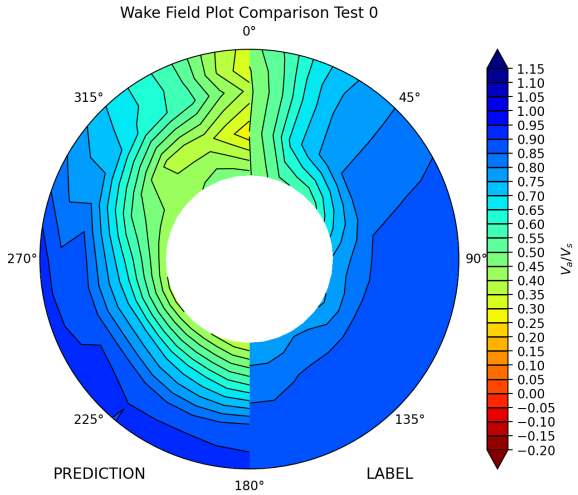

Wake Field Plot Comparison Test 6
Wake Field Plot Comparison Test 7
Wake Field Plot Comparison Test 8
Wake Field Plot Comparison Test 9
Wake Field Plot Comparison Test 10
Wake Field Plot Comparison Test 11
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
V_d/V_s
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20

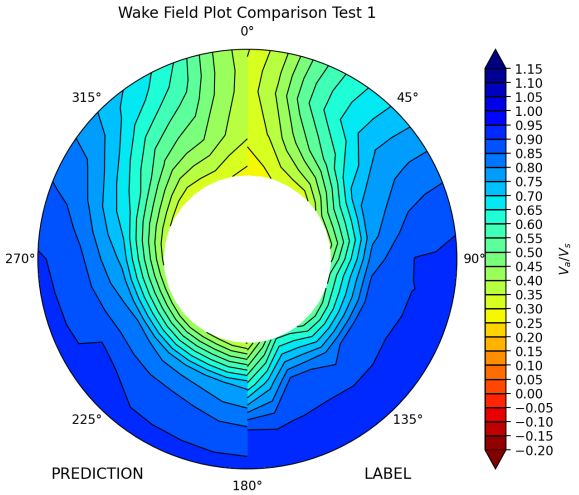


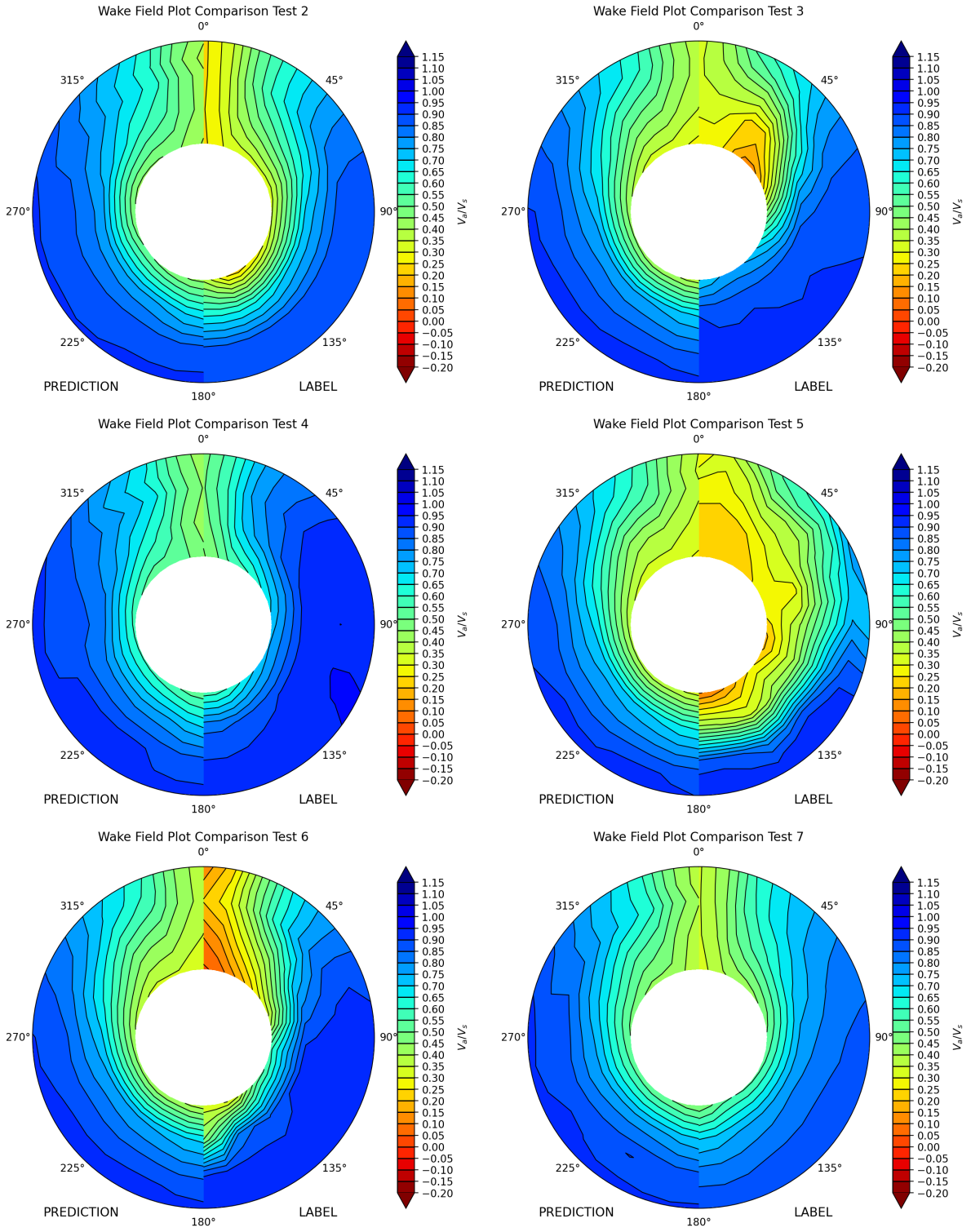


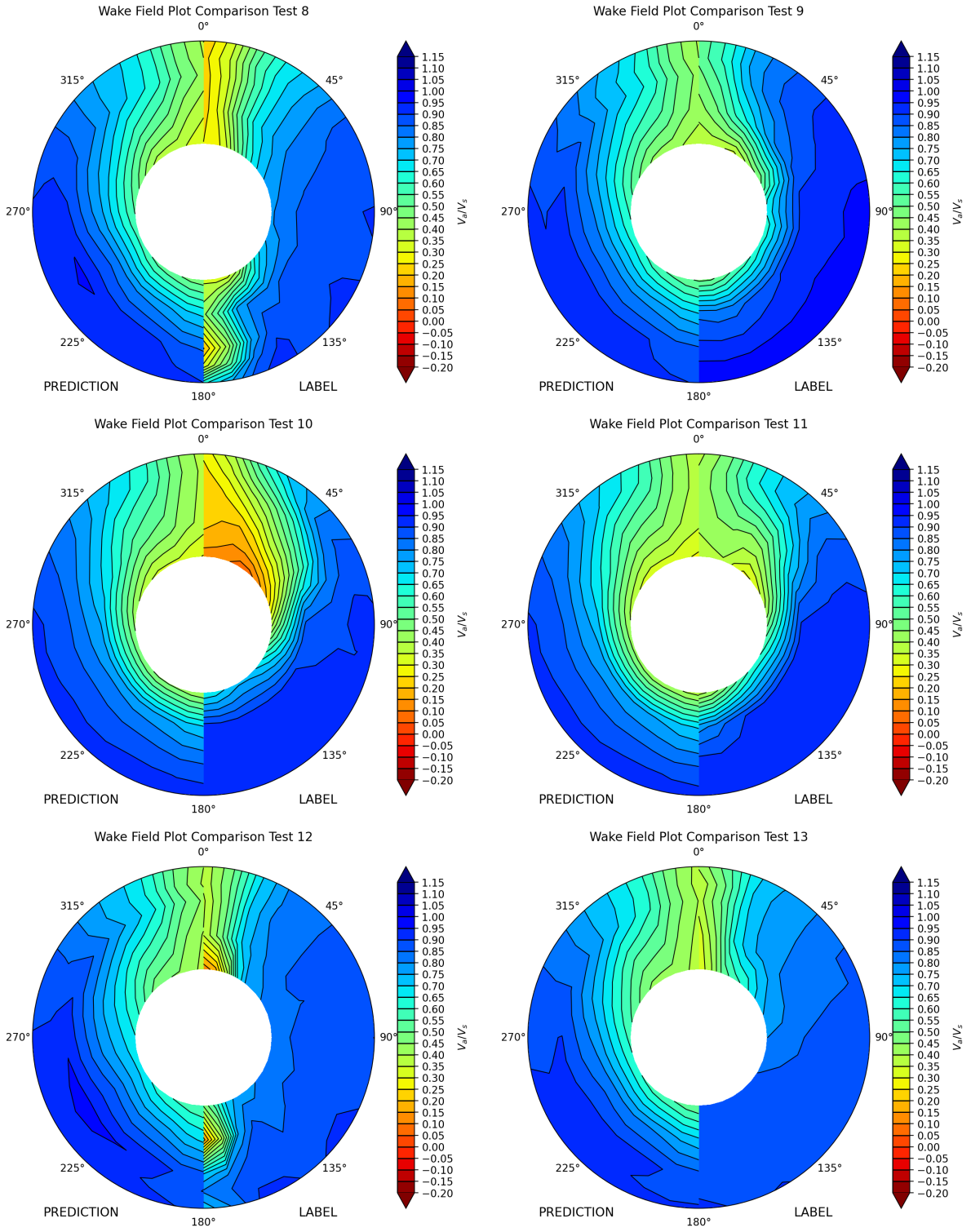


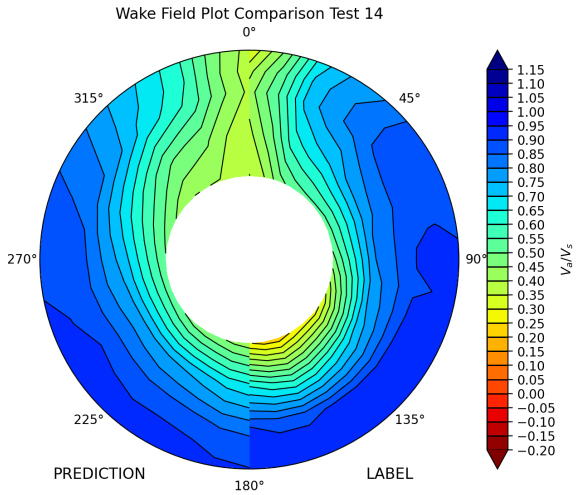
F.3. Feed-Forward Neural Network with DCT and Standardisation Predictions



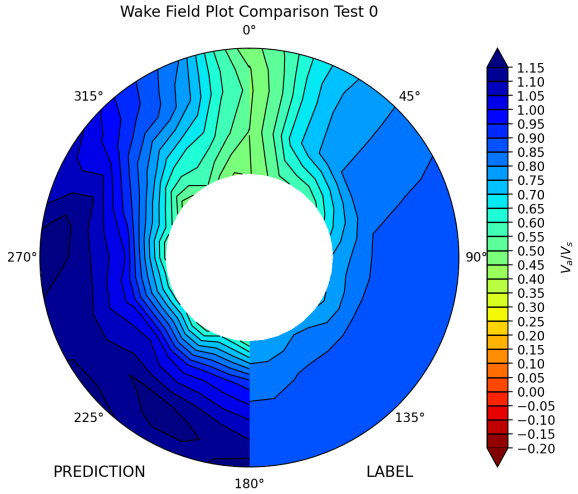


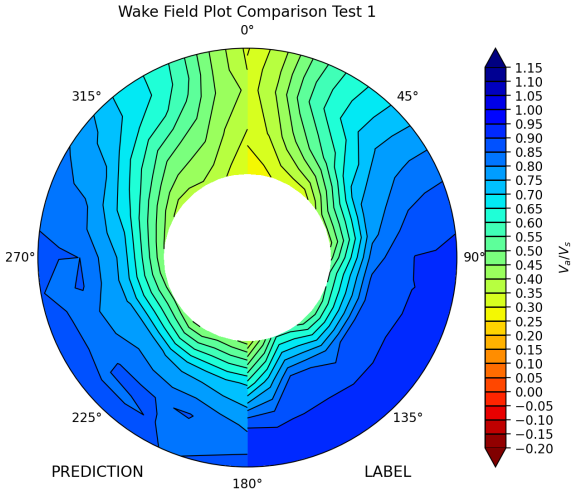

Wake Field Plot Comparison Test 2
Wake Field Plot Comparison Test 3
Wake Field Plot Comparison Test 4
Wake Field Plot Comparison Test 5
Wake Field Plot Comparison Test 6
Wake Field Plot Comparison Test 7
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
V_d/V_s
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20

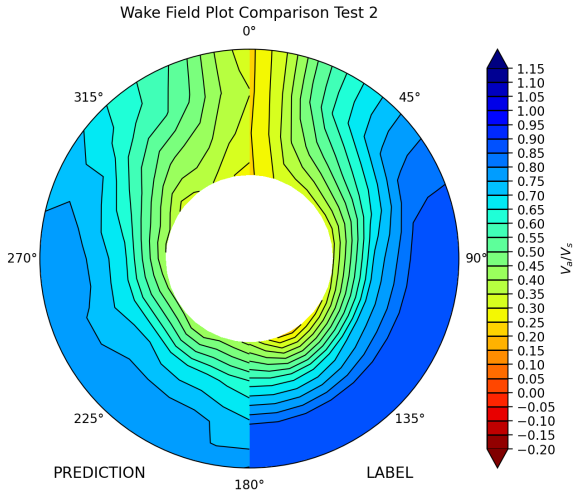

Wake Field Plot Comparison Test 8
Wake Field Plot Comparison Test 9
Wake Field Plot Comparison Test 10
Wake Field Plot Comparison Test 11
Wake Field Plot Comparison Test 12
Wake Field Plot Comparison Test 13
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
V_d/V_s
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20

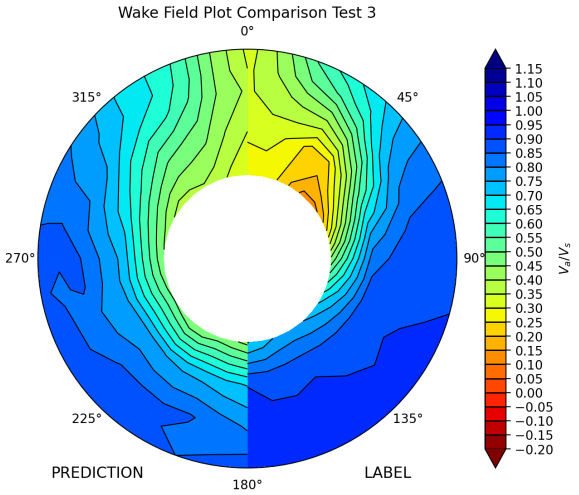


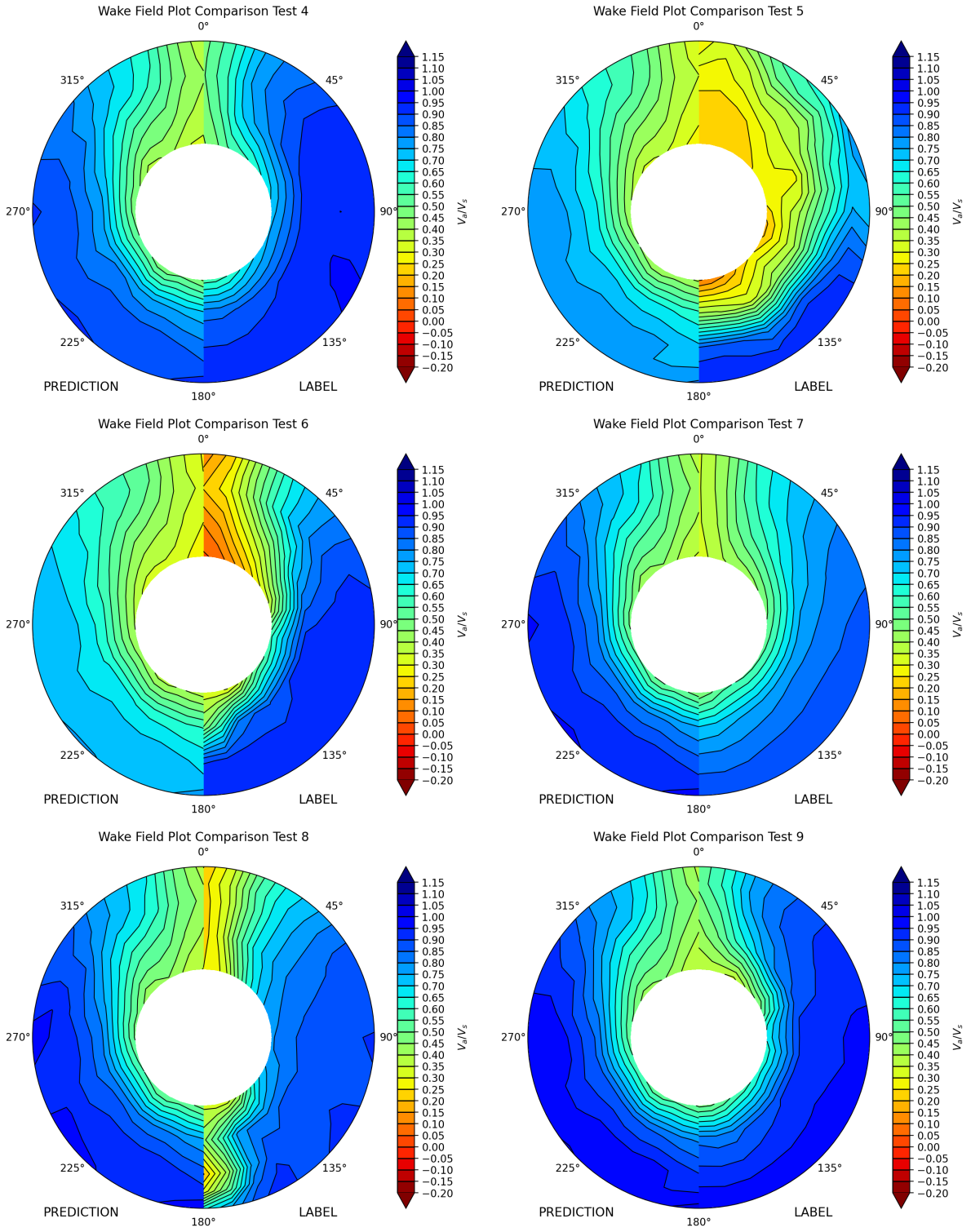
F.4. Ensemble Model Predictions

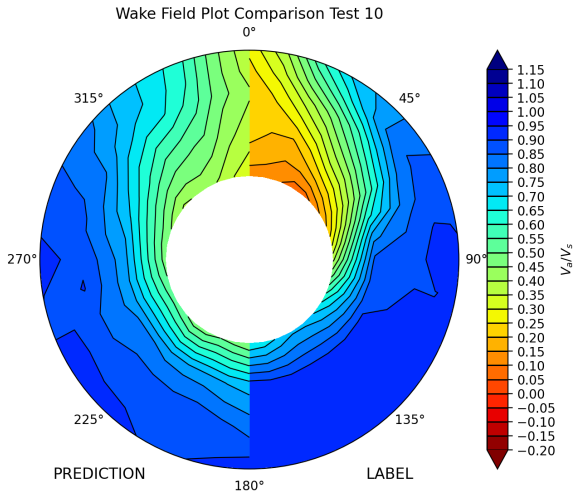


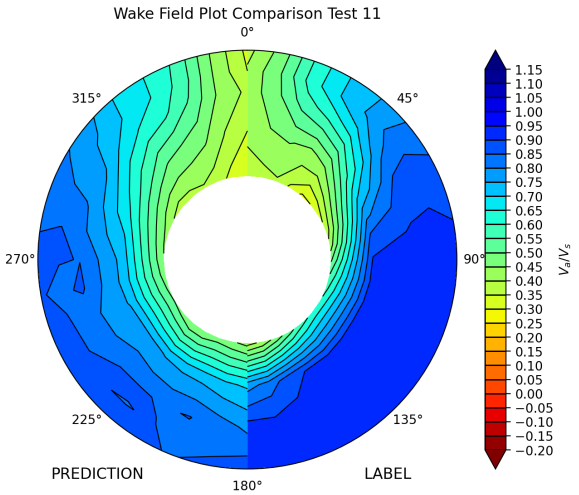


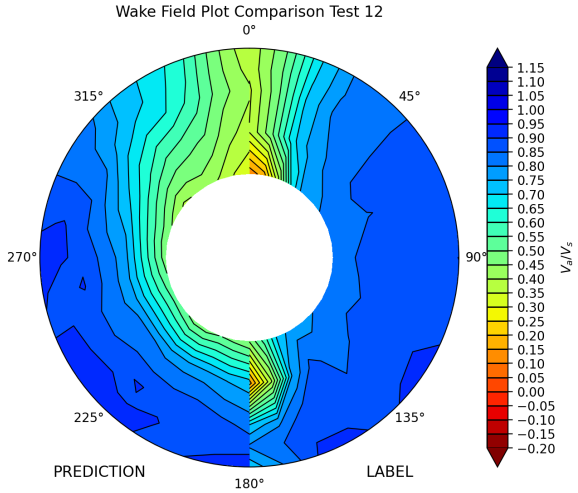


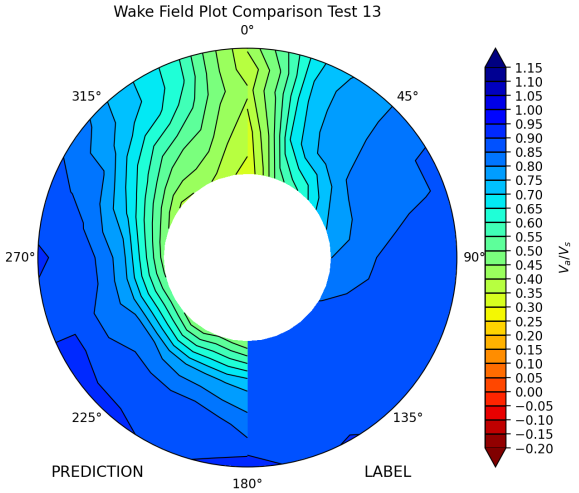


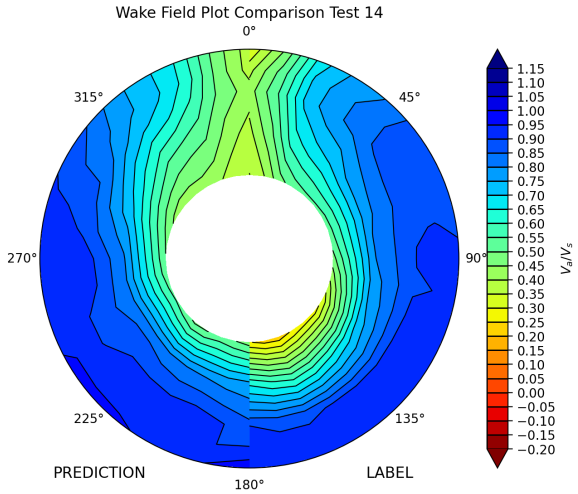

Wake Field Plot Comparison Test 4
Wake Field Plot Comparison Test 5
Wake Field Plot Comparison Test 6
Wake Field Plot Comparison Test 7
Wake Field Plot Comparison Test 8
Wake Field Plot Comparison Test 9
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
V_a/V_s
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20


Wake Field Plot Comparison Test 10
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
V_a/V_s

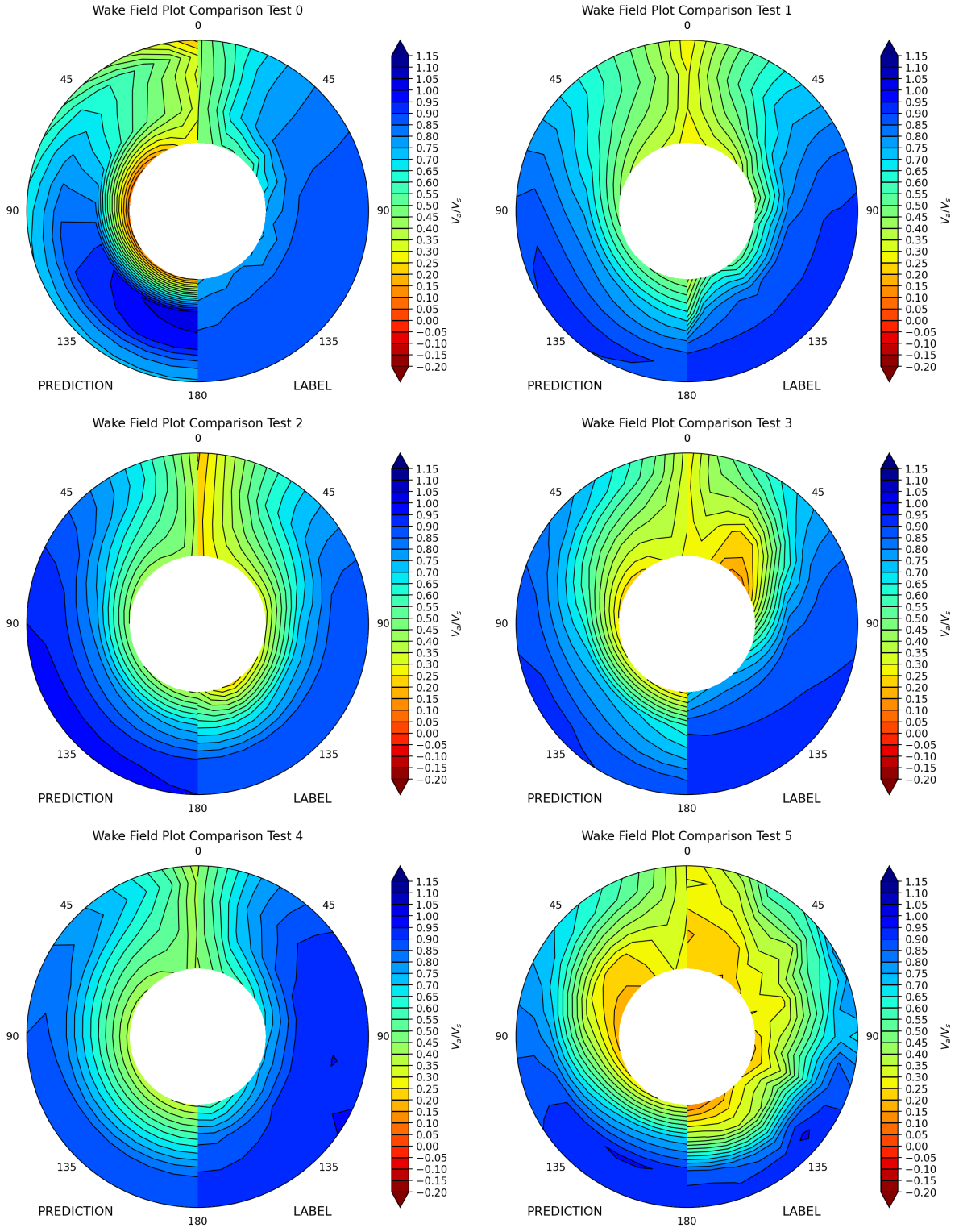

Wake Field Plot Comparison Test 11
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
V_a/V_s

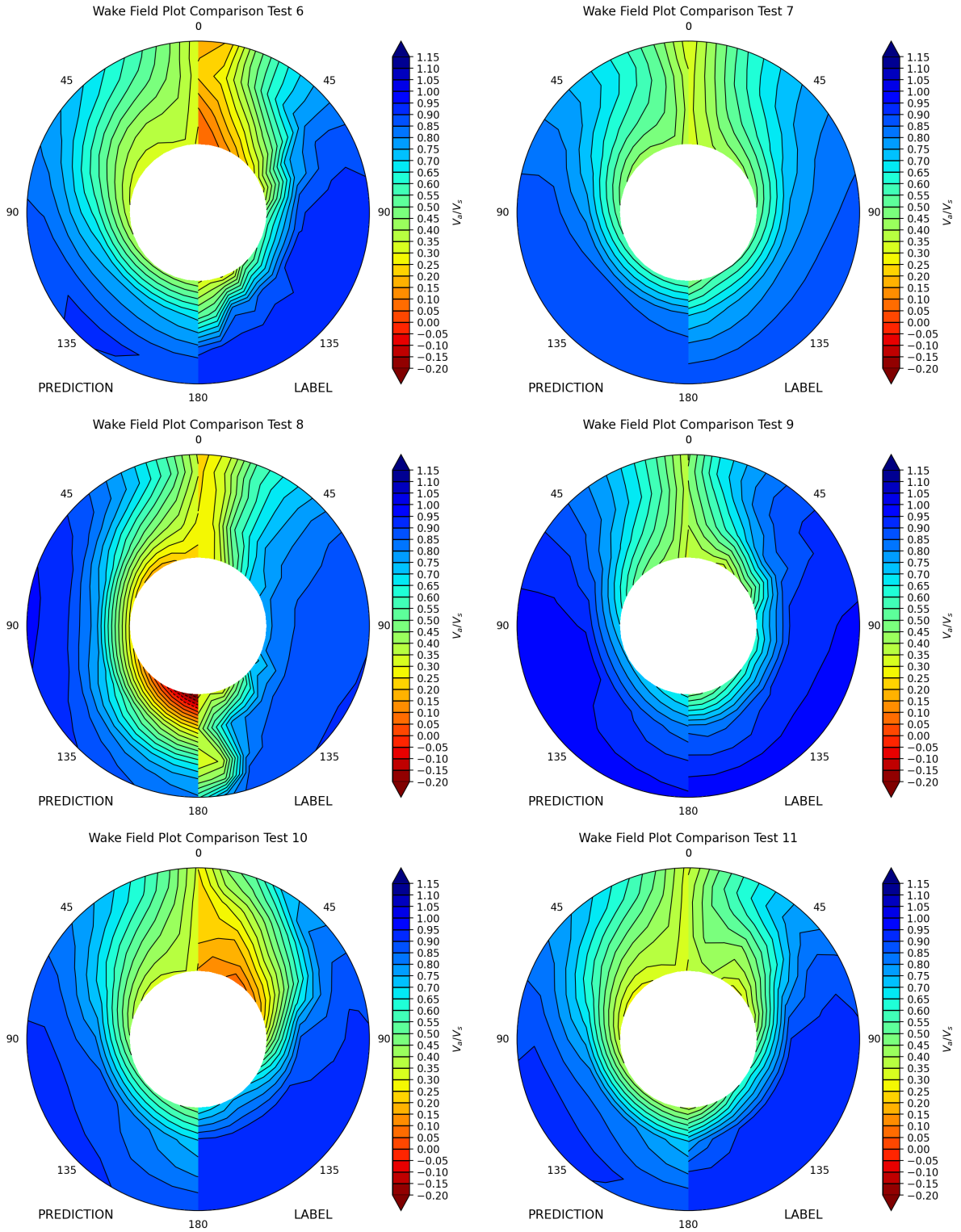

Wake Field Plot Comparison Test 12
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
V_a/V_s

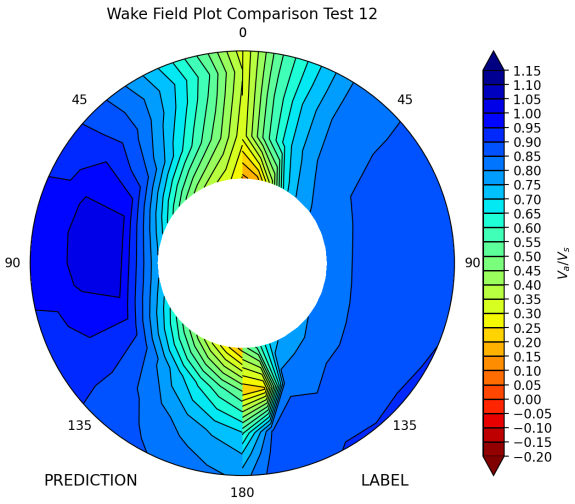

Wake Field Plot Comparison Test 13
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
V_a/V_s

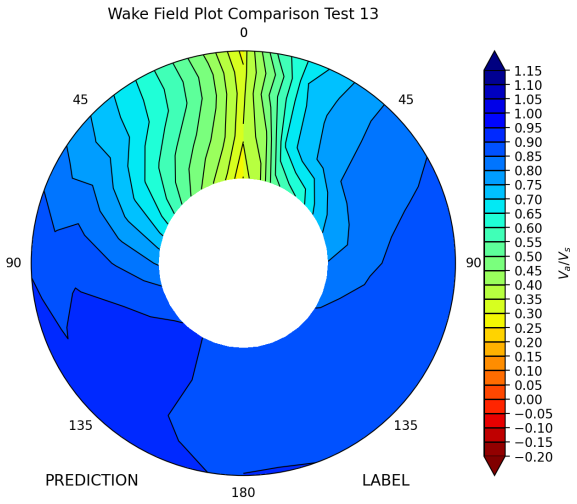

Wake Field Plot Comparison Test 14
0°
45°
90°
135°
180°
225°
270°
315°
PREDICTION
LABEL
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
V_a/V_s

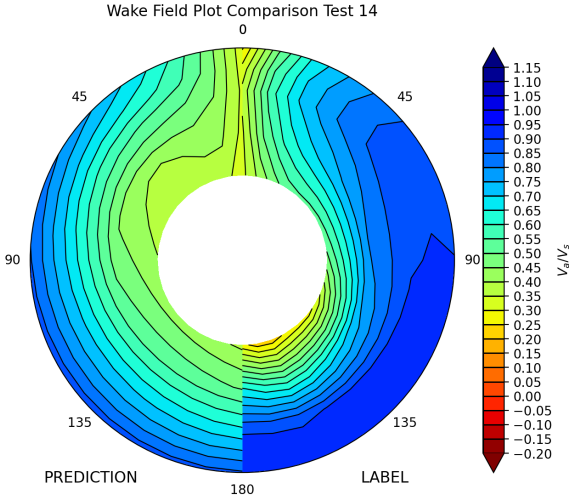
F.5. LSTM Model Predictions




Wake Field Plot Comparison Test 6
Wake Field Plot Comparison Test 7
Wake Field Plot Comparison Test 8
Wake Field Plot Comparison Test 9
Wake Field Plot Comparison Test 10
Wake Field Plot Comparison Test 11
0
45
90
135
180
PREDICTION
LABEL
V_a/V_s
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20


Wake Field Plot Comparison Test 12
0
45
45
90
90
135
135
180
PREDICTION
LABEL
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
V_a/V_s


Wake Field Plot Comparison Test 13
0
45
45
90
90
135
135
180
PREDICTION
LABEL
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
V_a/V_s


Wake Field Plot Comparison Test 14
0
45
45
90
90
135
135
180
PREDICTION
LABEL
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
V_a/V_s

G

Cavitation Plots With and Without Tangential Velocities

This appendix presents the comparison of cavitation plots with and without tangential velocities calculated by Mpuf3a. This small experiment was conducted by Jannicke Gjerdevik, to evaluate if only providing axial velocities in a wake field, rather than also tangential and possibly also radial velocities, has a large influence on the calculated cavitation by Mpuf3a.

Comparing cavitation plots for wake with and without tangential velocities using Mpuf3a

Background:

The Master thesis to Stijn Vervoordeldonk has up till now been focusing on predicting the axial velocities of a nominal wake field with Machine Learning. To evaluate if time and effort should also be used to predict tangential and possibly radial velocities, a small study was done to evaluate the effect of these velocities on the cavitation properties.

The 3 following projects, conditions and wakefields were selected:

Project	**Condition(s)**	**Wakefield name**		
SP/07774H1	FSAH	SSSRI New	SSRI New Vx	
SP/07310H1	FSAH	Measured wake	Measured wake Vx	SSRI New Vx
	Pressure side check ballast, trial	Measured wake	Measured wake Vx	
SP/06620H1	FSAH	HSVA-meas-44ang	HSVA-meas-44ang-Vx	

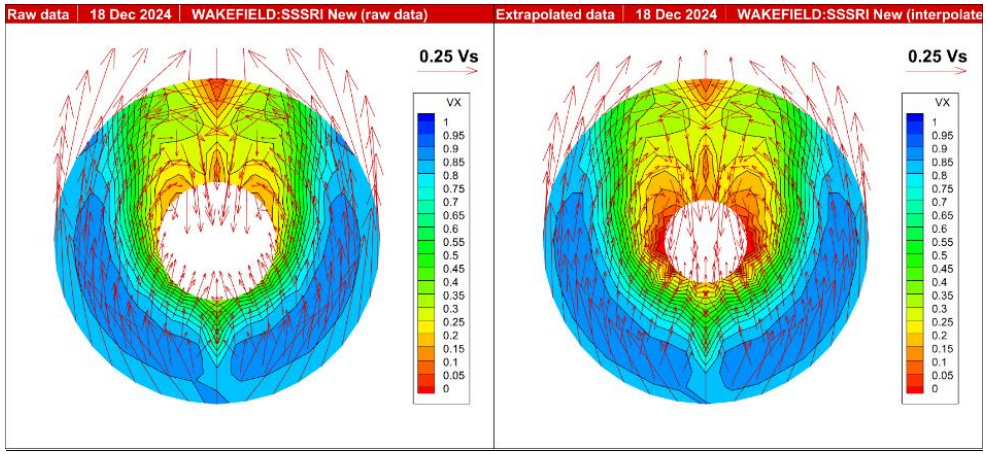
All projects are analysed for the FSAH condition with the corresponding FSAH wakefield and the same FSAH wakefield with only axial velocities (indicated with Vx). In addition, the SP/07310H1 was analysed for FSAH with the axial wakefield of SP/07774H1, to compare also the influence of changes in the axial wake field, and one additional condition. For all conditions and wakefield, the method of Huang and Groves were used to find the effective wake, which again were used to find the cavitation.

Below you will find pictures of the wakefields and cavitation plots for the different conditions and wakefields.

Conclusion:

The influence of the tangential and radial velocities is notable but rather small, and clearly much smaller than the influence of a wakefield with different axial velocities. The conclusion is therefore that it is wisest to focus the effort on improving the prediction of the axial velocities.

SP/07774H1

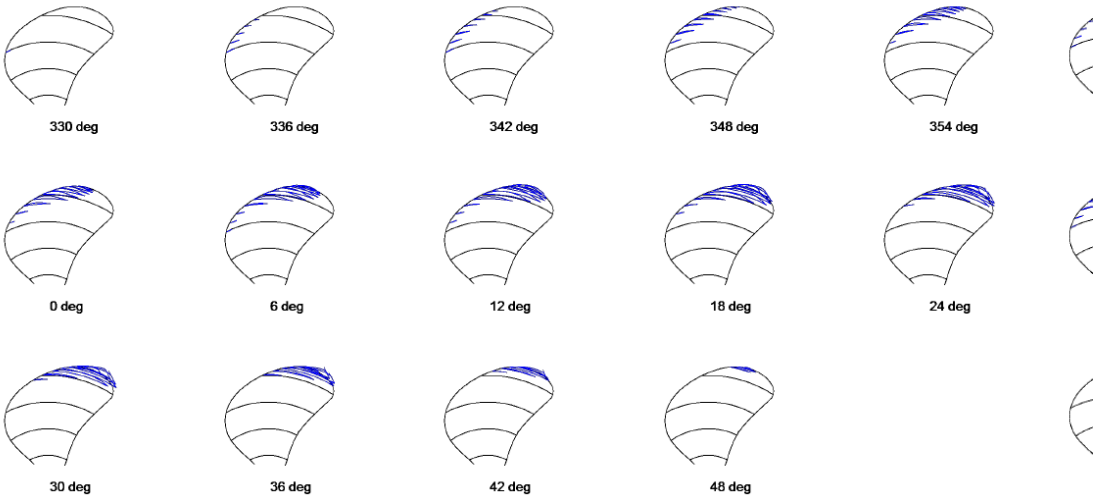


FSAH - Original wake (SSSRI New):

Mpuf (Mpuf-3aV3.3.exe)
Suction side cavitation

N = 109.00 RPM	WE = 0.2800	TA = 8.20 M	KT = 0.1576	T = 625 KN
VS = 17.57 KTS	DPHI = 0.73 DEG	J = 0.6124	10KQ = 0.2489	P = 6590 KW

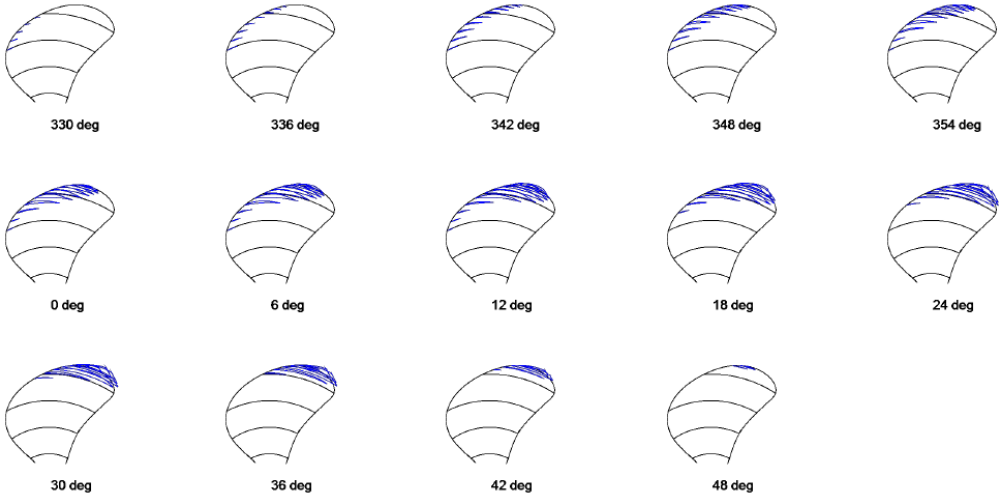
Only axial wake:



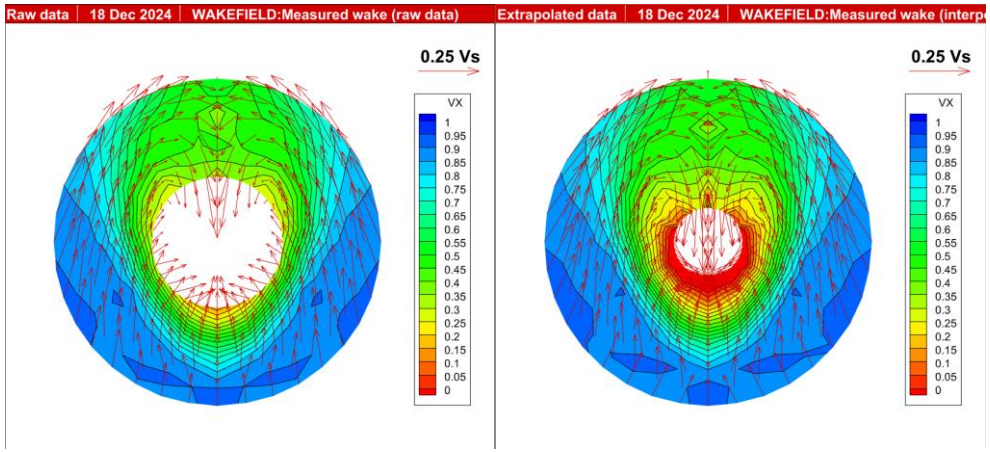
FSAH - Only axial wake (SSSRI New Vx):

Mpuf (Mpuf-3aV3.3.exe)
Suction side cavitation

N = 109.00 RPM	WE = 0.2800	TA = 8.20 M	KT = 0.1581	T = 627 KN
VS = 17.57 KTS	DPHI = 0.72 DEG	J = 0.6124	10KQ = 0.2487	P = 6585 KW



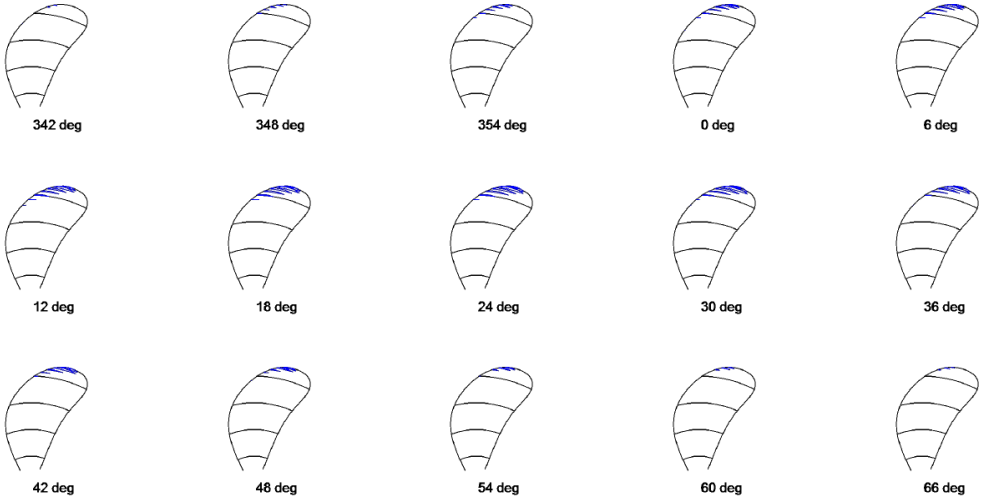
SP/07310H1:



FSAH - Original wake (Measured wake):

Mpuf (Mpuf-3aV3.3.exe)
Suction side cavitation

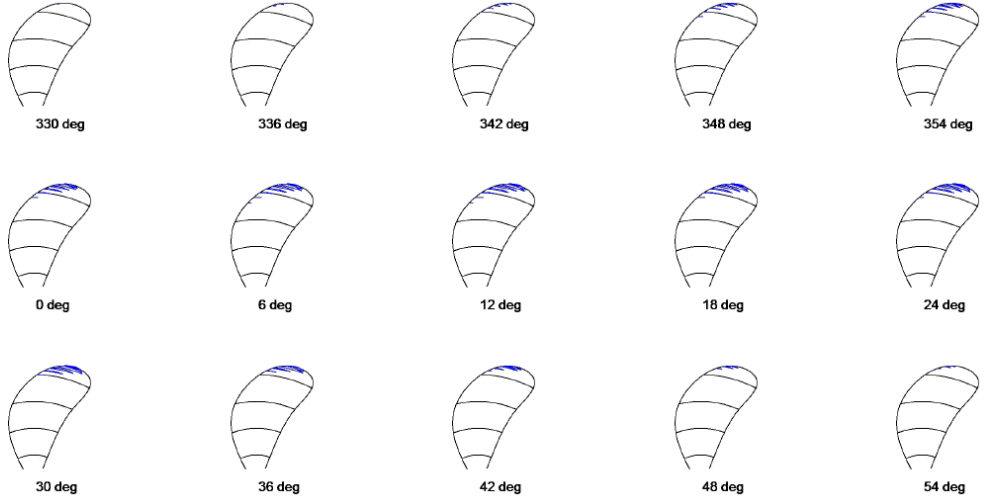
N = 104.00 RPM	WE = 0.2712	TA = 8.50 M	KT = 0.1463	T = 444 KN
VS = 14.96 KTS	DPHI = 1.38 DEG	J = 0.5778	10KQ = 0.2087	P = 3857 KW



FSAH - Only axial wake (MeasuredWakeVx):

Mpuf (Mpuf-3aV3.3.exe)
Suction side cavitation

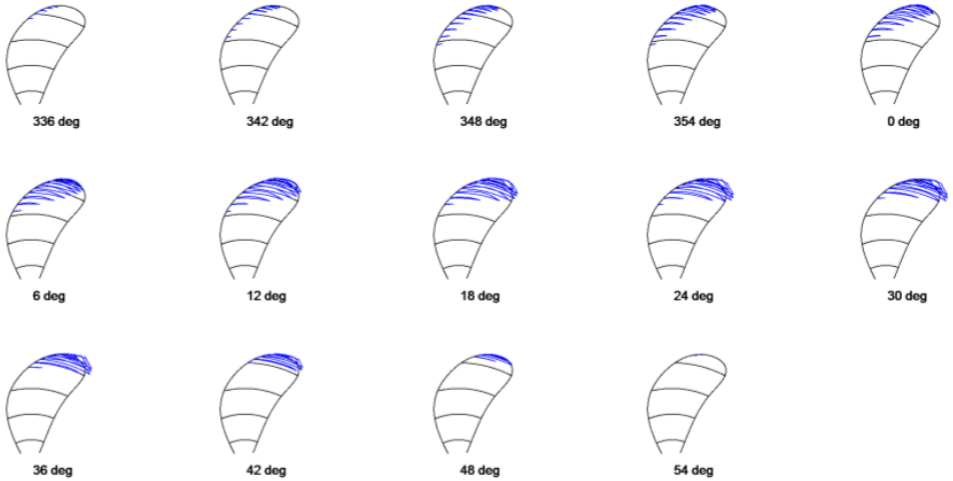
N = 104.00 RPM	WE = 0.2712	TA = 8.50 M	KT = 0.1463	T = 444 KN
VS = 14.96 KTS	DPHI = 1.40 DEG	J = 0.5778	10KQ = 0.2086	P = 3857 KW



FSAH – Axial wake from SP/07774H1 (SSSRI New Vx):

Mpuf (Mpuf-3aV3.3.exe)
Suction side cavitation

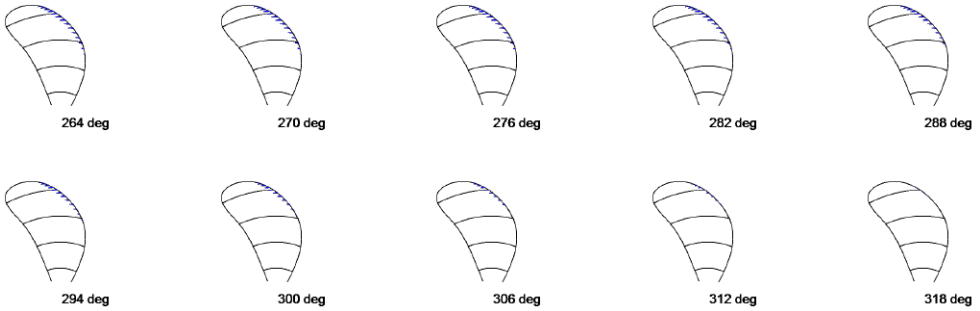
N = 104.00 RPM	WE = 0.2712	TA = 8.50 M	KT = 0.1464	T = 444 KN
VS = 14.96 KTS	DPHI = 1.45 DEG	J = 0.5778	10KQ = 0.2103	P = 3888 KW



Pressure side ballast trial - Original wake (Measured wake):

Mpuf (Mpuf-3aV3.3.exe)
Pressure side cavitation

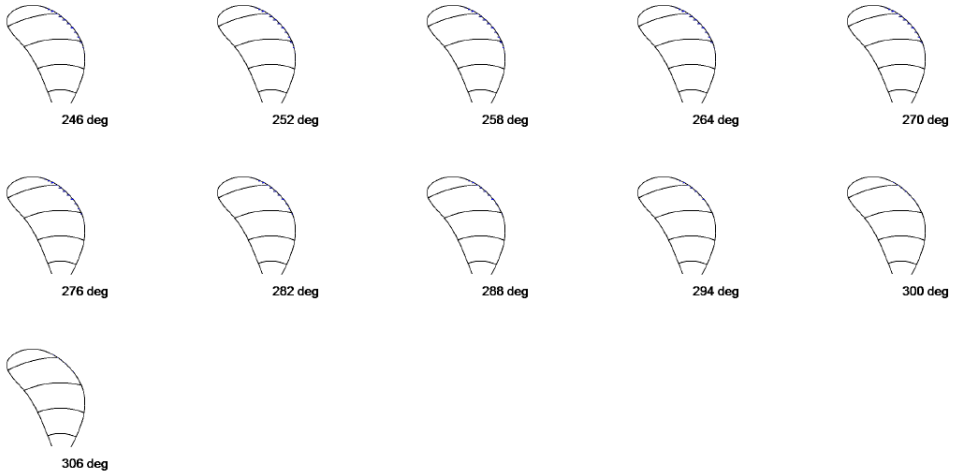
N = 104.00 RPM	WE = 0.2559	TA = 8.50 M	KT = 0.0955	T = 290 KN
VS = 16.64 KTS	DPHI = 0.00 DEG	J = 0.6560	10KQ = 0.1475	P = 2727 KW



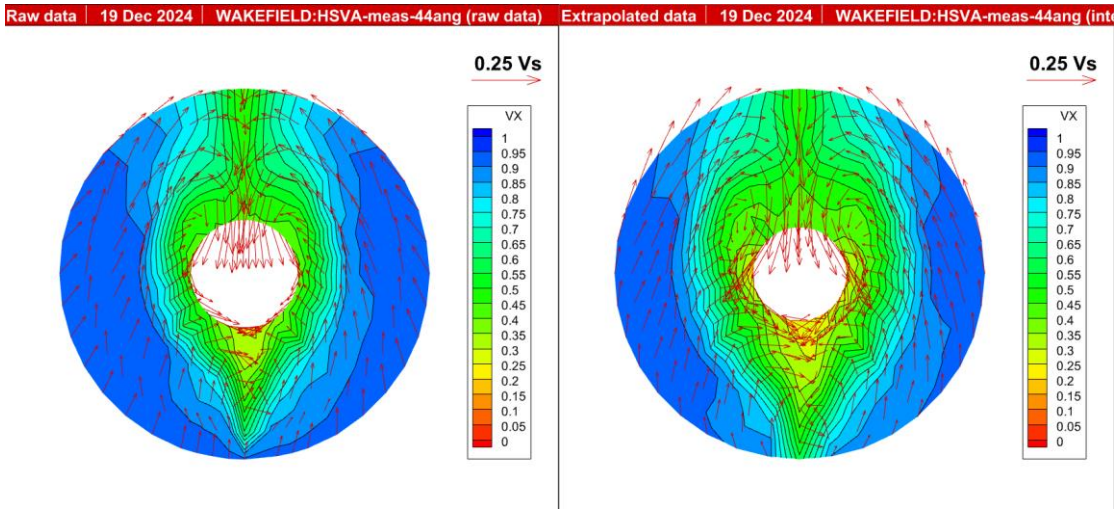
Pressure side ballast trial - Only axial wake (MeasuredWakeVx):

Mpuf (Mpuf-3aV3.3.exe)
Pressure side cavitation

N = 104.00 RPM	WE = 0.2581	TA = 8.50 M	KT = 0.0955	T = 290 KN
VS = 16.64 KTS	DPHI = 0.00 DEG	J = 0.6542	10KQ = 0.1473	P = 2723 KW



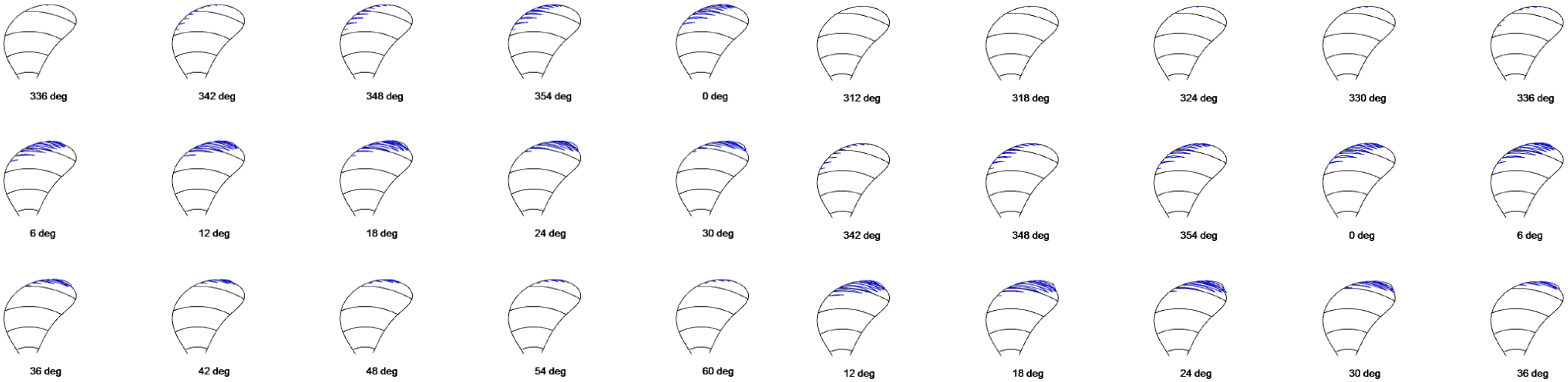
SP/06620H1:



FSAH - Original wake (HSVA-meas-44ang):

Mpuf (Mpuf-3aV3.3.exe)
Suction side cavitation

N = 110.47 RPM	WE = 0.2220	TA = 9.30 M	KT = 0.1949	T = 821 KN
VS = 18.96 KTS	DPHI = 0.59 DEG	J = 0.6986	10KQ = 0.3292	P = 9470 KW



FSAH - Only axial wake (HSVA-meas-44ang-Vx):

Mpuf (Mpuf-3aV3.3.exe)
Suction side cavitation

N = 110.47 RPM	WE = 0.2220	TA = 9.30 M	KT = 0.1940	T = 817 KN
VS = 18.96 KTS	DPHI = 0.71 DEG	J = 0.6986	10KQ = 0.3294	P = 9475 KW