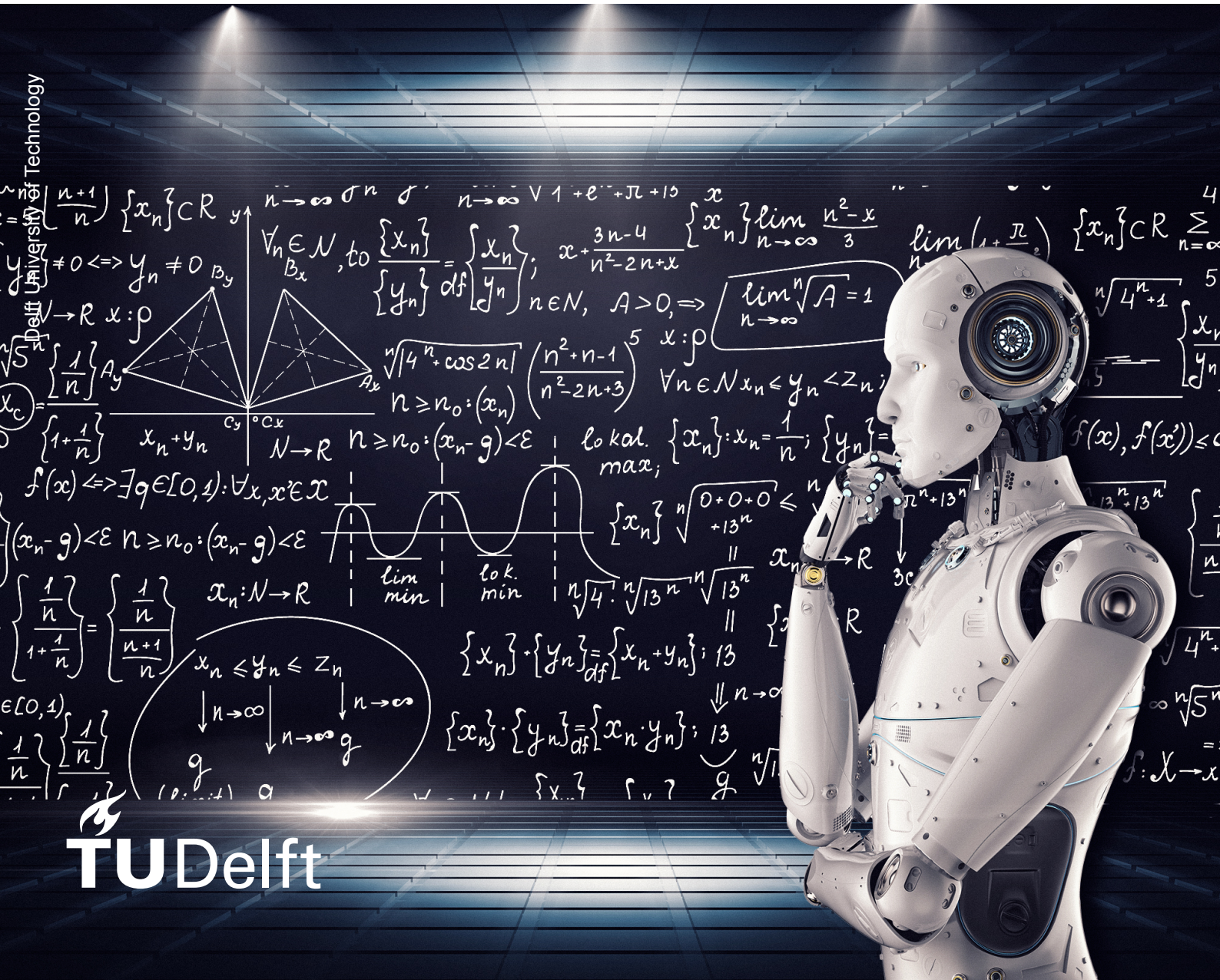
On leveraging physical knowledge to augment neural network-based surrogate models for simulations.

Master Thesis

Tadeusz Kaniewski



Delft University of Technology

TUDelft

On leveraging physical knowledge to augment neural network-based surrogate models for simulations.

by

Tadeusz Kaniewski

Student Name	Student Number
Tadeusz Kaniewski	5282640

TU Delft Supervisor:	Anh Khoa Doan, PhD
Additional Supervisor:	Jonathan Donier, PhD
Institutions:	Delft University of Technology
Place:	Lausanne, Switzerland
Project Duration:	February, 2022 - February, 2023

Cover Image: Source: [MacKenzie, 2018]

TU Delft

Preface

I would like to begin by expressing my deepest gratitude to my university supervisor Dr. Anh Khoa Doan and my work colleagues for their unwavering support and guidance throughout my master's thesis. Their invaluable insights and expertise have greatly contributed to the success of this project. This past year has been a great learning experience, which also allowed me to move to a new country and meet very interesting people.

I also would like to extend my heartfelt thanks to my family for their constant love and encouragement. Their patience and understanding during the long hours and late nights spent working on this thesis were greatly appreciated and will never be forgotten.

Lastly, I would like to take a moment to remember and honor my beloved dog, Bono. He was a faithful companion throughout the last 13 years of my life, and his passing during the course of my thesis was a great loss to me. I apologize to him that, due to my absence from home, I was unable to comfort him during his dying days, and I will always remember him as a loyal and loving friend.

Tadeusz Kaniewski
Lausanne, March 2023

Abstract

The computational cost of high-fidelity engineering simulations, for example CFD, is prohibitive if the application requires frequent design iterations or even fully fledged optimization. A popular way to reduce the computational cost and enable fast iteration cycles is to use surrogate models that are trained to predict simulation results from historical simulation data. While most traditional methods are parametric, ANNs are able to process geometries directly and are thus agnostic to the parametrization of the geometric models, which makes them appealing when working on multiple design campaigns. However, ANNs may fail to transfer the learned knowledge when used on new design campaigns that are significantly different from those used to train the model or when the size of the training data set is too small.

The goal of this project is to increase the reliability of ANN-based surrogate models on new design campaigns and on small datasets. One main direction to achieve this goal is to incorporate prior physical knowledge into the learning process. Methods to supplement training data with a simplified solution, meaningful physical scaling, and governing equation-based losses were used. The data set used for this study was based on a real-life-inspired complex 3D parametrized geometry of an automotive HVAC system.

The methods were tested in generalization, transfer learning, and single-campaign inference tasks. Moreover, physics-informed losses were tested in an ill-posed setting as a prediction outlier correction tool. While the initial findings suggest that incorporating prior physical knowledge may improve model performance, especially in low-data regimes, further investigation is needed to draw any definitive conclusions. Additionally, the preliminary results related to the use of physics-informed losses as a correction method are inconclusive and require further experimentation in the future. Therefore, more research is needed to determine the effectiveness of these approaches.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
CFD	Computational Fluid Dynamics
ROM	Reduced Order Modeling
NN	Neural Network
ML	Machine Learning
MLP	Multilayer Perceptron
CNN	Convolutional Neural Network
LBFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm
SGD	Stochastic gradient descent algorithm
RGB	Red, green, blue color model
SVD	Singular value decomposition algorithm
POD	Proper orthogonal decomposition
PDE	Partial differential equation
PCA	Principle component analysis
N-S	Navier-Stokes equation
SDF	Sign-distance function
LBM	Lattice-Boltzmann method
RANS	Reynolds-Averaged Navier-Stokes
AoA	Angle of attack
PINN	Physics Informed Neural-Network
ANN	Artificial Neural Network
S-A	Spalart Allmares turbulence model
TBNN	Tensor-Based Neural Network
LES	Large eddy simulation
DNS	Direct numerical simulation
PCG	Preconditioned conjugate gradient scheme
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
TF	TesnorFlow
SVM	Support vector machine
BC	Boundary conditions
IC	Initial conditions
ESN	Echo state network
MSE	mean squared error
VP	velocity-pressure
VV	velocity-vorticity
FVM	Finite Volume Method
GNN	Graph Neural Network
MINST	Modified National Institute of Standards and Technology database
RMS	Root-mean squared
LSTM	Long short-term memory neural networks
SINDY	Sparse identification of nonlinear dynamics
ODE	ordinary differential equation
RAR	Residual-based Adaptive Refinements
FEM	Finite Element Method

Abbreviation	Definition
MRI	magnetic resonance imaging
HVAC	Heat, ventilation and air-conditioning unit
PI	physics-informed

Symbols

Symbol	Definition
u	Velocity
k	Turbulent Kinetic Energy
Re	Reynolds number
w	neuron weight
b	neuron bias
f	body force terms
$\mathcal{L}$	loss function
J	loss function
R^2	coefficient of determination
p	pressure
t	time
v_T	eddy viscosity
S	strain rate
Ri	Richardson number
g	gravitational acceleration
L	reference length
g	kernel weights
A	area
l_m	mixing length
$\sqrt{G}$	mean strain rate tensor
$\mathbf{m}$	momentum vector
$\hat{\mathbf{s}}$	ADAM optimizer bias correction term
$\hat{\mathbf{m}}$	ADAM optimizer bias correction term
$\langle\rangle$	time averaged component
$\prime$	time varying component
$_0$	reference value
ρ	Density/ local geodesic polar coordinates
ω	vorticity
Ω	penalty function term
α	L2/L1 regularization penalty weight
σ	activation function
θ	neural network model parameters / local geodesic polar coordinates
ν	kinematic viscosity
β	momentum constant
ϵ	ADAM optimizer smoothing term
τ	stress
δ	Kronecker delta
μ	mean / dynamic viscosity
η	learning rate

List of Figures

List of Tables

1 Introduction

The computational costs of high-fidelity engineering simulations, for example, computational fluid dynamics (CFD), are very high if the application requires frequent design iterations or fully fledged optimization. A popular way to reduce computational costs and enable fast iteration cycles is to use techniques such as reduced-order models (ROM) or adjoint solvers. Those methods can be effective solutions, but their applications are often very case-specific and have a limited range of design exploration capabilities [Li et al., 2022]. This work utilizes a machine learning (ML) software that aims to improve the design product pipeline using neural network-based surrogate modeling techniques.

This work utilizes a deep learning-based computer-aided engineering software that utilizes ML techniques to accelerate design processes. This ML software allows the user to create neural network (NN) models that can be trained later to predict simulation results from historical simulation data. While most traditional methods are parametric, this one allows to build models that are able to process geometries directly and is thus agnostic to the parameterization and meshing of geometric models, which makes it appealing when working on multiple design campaigns[1]. Another benefit of this type of solution is that such models predict the results much faster than traditional simulation methods, allowing for building of fast optimization cycles for rapid design landscape exploration.

However, NN models require enough data to at least sparsely cover the desired design envelope. This can be done by utilizing past simulation results from older campaigns of a given product, but may also require some new simulations from the current to-be-designed campaign. If the coverage of the design space is too sparse or irregular, the model may fail to provide useful results. Additionally, prior usage of the ML software used in this work was purely data-driven, which on some occasions led to locally non-physical results and poor extrapolation capabilities.

The goal of this project is to tackle these limitations by incorporating prior physical knowledge into the models. The creators of the ML software used for this thesis see this as a key development area as it can lead to more accurate models that need lower amounts of data to train. Moreover, many customers, who have used CFD simulations in their design processes for years, have large libraries of simulations of their past campaigns. The information captured in those simulations can potentially be used to train models for new campaigns and limit the costly usage of CFD simulations for training data creation. This could lead to faster time to market for the products and also better designed products due to more effective design space exploration. Such improvements are seen as highly valuable to customers, and as this methodology is still in the stage of early adoption, improvements like this will hopefully allow for more widespread production use of the ML software in major target markets.

[1] In this work, **campaign** refers to a separate design cycle that aims, by iterating on the design by creating different samples, to arrive at the final optimized design.

Scientific deep learning is a new research field, and it evolves extremely dynamically. Every month, new papers are published in the field and thus the state-of-the-art evolves rapidly. Figure 1.1 illustrates this landscape, with an almost exponential growth in the number of publications in the field of physics-informed neural networks (PINNs), which is one of the fields of particular interest for this work.

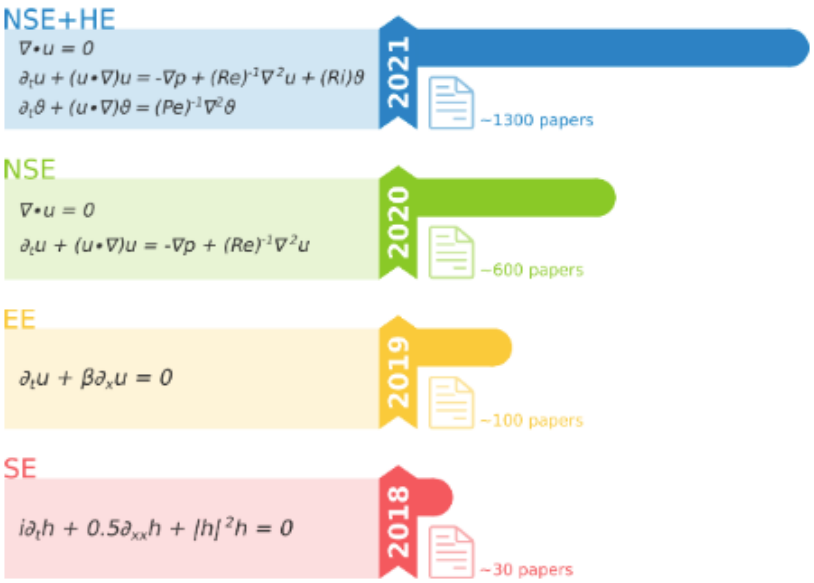


Figure 1.1: The number of publications in the field of PINNs, since the original paper of Raissi et al. [2019]. With passing years, more and more complex problems were tackled, firstly 1D Schrodinger equations (SE) [Raissi et al., 2019], subsequently Euler equations [Mao et al., 2019], than Navier-Stokes (N-S) equations [Jin et al., 2021] and finally N-S coupled with heat equations [Cai et al., 2021b]. The graphic shows the exponential growth of the interest in the field measured by new publications.

This thesis will explore a series of methods for adding physical prior knowledge to neural network-based surrogate models trained with CFD simulations. It is important to note here that this work aims to combine research and industrial applications. The models, unlike in other literature found, are trained on a complex, 3D, industrial-like application of automotive HVAC geometry. The use case of complex, 3D industrial-like automotive HVAC geometry presents significant challenges, particularly in maintaining scientific rigor, with some aspects of this work requiring compromises between the academic and research realities.

This thesis is structured as follows. Firstly, a condensed description of prior knowledge and state of the art will be described in Chapter 2. Subsequently, having identified potential research directions, Chapter 3 will present the research questions to be addressed in this thesis. Chapter 4 presents the base methodology using which the different aspects of the research questions will be tackled. Chapter 5 will present the baseline surrogate model, with respect to which all methods to incorporate physical prior knowledge will be compared. In Chapter 7 the results generated during this work will be presented. Firstly, the baseline model performance will be analyzed, and subsequently the different physics-informed (PI) methods will be compared to the baseline model performance. In Chapter 8 the conclusions will be drawn from the work presented, and finally in Chapter 9 discussion and suggestions for future developments in this research topic will be proposed.

2

Literature review

2.1. Background on neural networks

In this chapter, the fundamentals of neural networks will be introduced. The ideas presented will be essential for understanding the research field. A lot of information in this section can be treated as general knowledge. For the main reference of this chapter, the textbook of Geron [2019] was used.

2.1.1. Overview of machine learning

Machine learning is a branch of engineering science that focuses on the use of algorithms that can assimilate data, where the assimilation process is referred to as training, and can later use the accumulated knowledge for further predictions based on this previous experience. Typically, machine learning is divided into 3 main categories, as presented in Figure 2.1. The algorithms are divided based on the type of training being done. Supervised learning is learning during which carefully labeled data is given to the algorithm to assimilate. Labeled data means data that have the desired solution appended to them. On the other end of the spectrum, unsupervised learning algorithms train with unlabeled data, by learning the dependencies between the data points without the help of knowing the right answer at the start. In between, there is a group of algorithms that use partially labeled data during training, called semi-supervised learning algorithms. Each group of algorithms in Figure 2.1 is characterized by the type of problem that this group of algorithms can solve. For an extended explanation of the algorithms and what they can do, see Geron [2019].

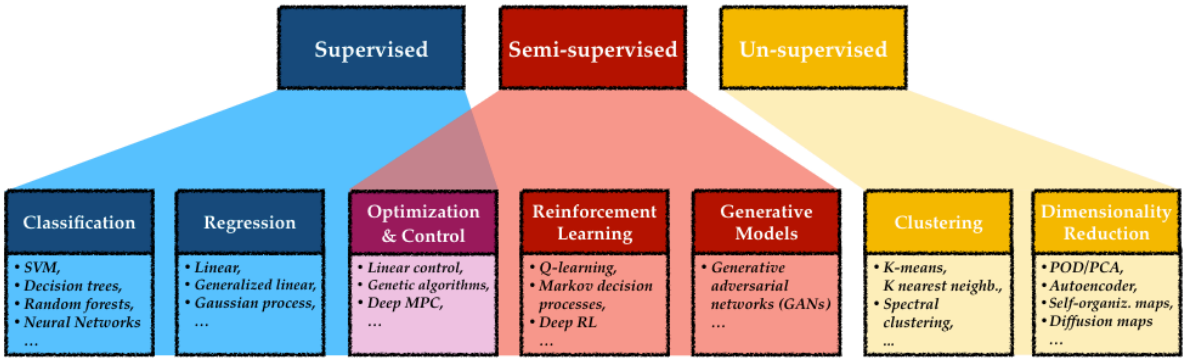


Figure 2.1: Fundamental division of machine learning algorithms. Source: Brunton et al. [2020]

2.1.2. Introduction to neural networks

Although there are many algorithms, one of the most popular classes of ML algorithms is called neural networks (NN). The reason for their widespread use is that they are efficient when handling large data sets. Moreover, NN can tackle problems that belong to various fields and various groups, as presented in Figure 2.1. Based on the survey [Liu et al., 2017], neural networks have been applied to a wide range of problems such as image recognition, computer vision, language processing and music recognition, and were also used in COVID-19 modeling [Shorten et al., 2021], or medical image analysis [Shen et al., 2017]. Additionally, they can be easily scaled to tackle complex problems, outperforming other ML techniques [Geron, 2019]. Neural networks can also be used as function approximators [Hornik et al., 1989]. Using the input data, the model, which approximates a function, can predict the output of the function. The function here can be any mathematical expression. The most basic type of neural network is called a multilayer perceptron or MLP in short. It started with the proposal of a single perceptron by an American psychologist Frank Rosenblatt in 1957 [Geron, 2019]. Combining multiple perceptrons into layers creates an MLP. MLP neural networks are composed of an input layer, hidden layers in the middle, and an output layer in the end. The input layer, as the name suggests, is where the user provides information to the network, based on which the prediction can be made in hidden layers and presented in the correct form by the output layer. In essence, each layer is made out of neurons, with each neuron, representing first a linear transformation operator having a weight and a bias, followed by a non-linear transformation through a selected activation function. A single neuron is depicted in Figure 2.2.

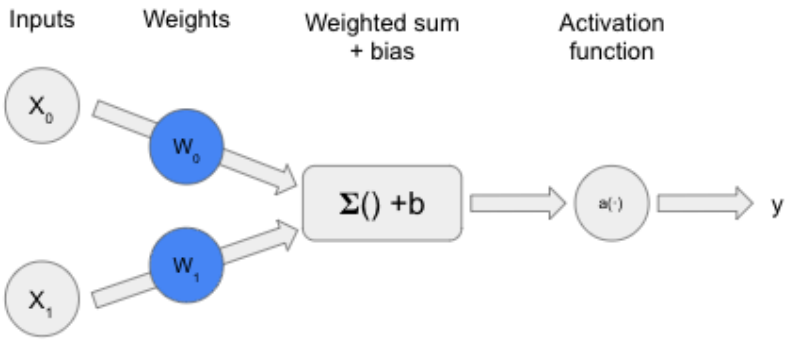


Figure 2.2: A single neuron consists of weighted sum of inputs and a bias term which is later transformed with an activation function and passed as output.

Therefore, the output of a single neuron is:

$$y = \sigma\left(\sum_{i=1}^{n} x_i w_i + b\right) \tag{2.1}$$

The activation function is a function applied at the output of the neuron, and it applies a transformation to the output to introduce non-linearity to the output of the neuron. It is a bio-inspired operation [Geron, 2019], that is supposed to mimic the brain's behavior, allowing neurons to be active, or "stimulated", by different inputs. This allows for the network to learn non-linear behavior in the data, by allowing different neurons in the network to output relatively large (very active neuron) or relatively low (low-acting or inactive neuron) values. Activation functions, responsible for the value of the neuron output, sometimes scale or cap the maximum or minimum value, or both. The choice of an activation function is highly problem-specific, and there are many functions to choose from. Some of the common activation functions are sigmoid, hyperbolic tangent, ReLU, and Leaky ReLU. Their plots are presented in Figure 2.3.

During training, the weights and biases are optimized to fit the training data, using the method called backpropagation [Rumelhart et al., 1986]. Using automatic differentiation [Güne¸ et al., 2018], it

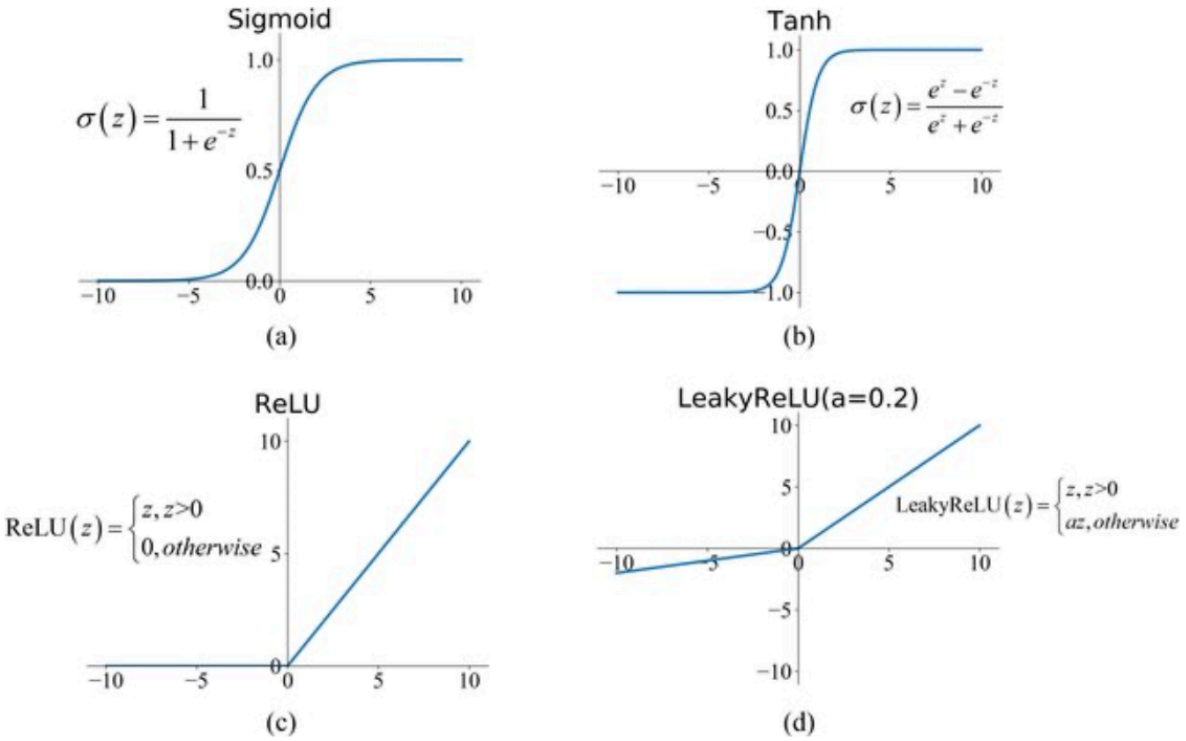


Figure 2.3: Commonly used activation functions, image source Feng et al. [2019]

is possible to calculate the derivatives of network error with respect to all network model parameters (weights and biases), with just two passes: one forward and one backward. Having all those gradients calculated, the gradient descent step can be performed.

A typical training of a neural network consists of a forward pass, loss function evaluation, and a backward pass.

In the forward pass, the input is processed through the input layer and sent to the first hidden layer, where the output is calculated using the stored weight and biases. Then the output of that layer is sent to the next layer to do the same, and so on. Unlike during the prediction step, in the forward pass of backpropagation, the intermediate results and the outputs from each neuron in the hidden layer are stored, since they are needed for the derivative calculations.

Next, the loss function is evaluated based on the output of the last layer of the network. The loss function is the function that measures how well the neural network models the training data set.

Finally, the backward pass is performed. By applying the chain rule of derivatives and using the stored neuron outputs, the derivatives of the loss function with respect to all model parameters are calculated, which are then propagated backward to evaluate the sensitivity of the error to each neuron's weights and biases. Finally, knowing the gradients, the gradient descent step can be performed to optimize and update the model parameters.

A single combined forward and backward passes form an iteration, and a pass through the entire training set forms an epoch. In cases where the number of training samples is very large, batching can be used. The batching operation randomly selects a smaller subset of samples at each iteration to use for the optimization step. This smaller data set of samples used allows to limit the computational effort required for the update of parameter.

Loss functions

A loss function is any function or expression that is evaluated during training to guide the optimizer in adjusting the weights of the neurons. The most commonly used formulations for this purpose are L1 and L2 loss functions, which are also built into common ML frameworks like TensorFlow [Abadi et al., 2015].

The L1 loss function can be written as:

$$\mathcal{L}_{L1} = \sum_{n=1}^{N} |y_{n,truth} - y_{n,pred}| \tag{2.2}$$

The L2 loss function can be written as:

$$\mathcal{L}_{L2} = \sum_{n=1}^{N} (y_{n,truth} - y_{n,pred})^2 \tag{2.3}$$

Where N is the number of samples, y_{pred} is the prediction of the network for a given sample and y_{truth} is the ground truth value for a given sample.

These are just the most commonly used functions, but there are many more to choose from. Loss functions can affect the performance of a NN in many ways. For example, Janocha and Czarnecki [2017] in their work applied to classification problems state that the choice of loss functions not only has an impact on the model training but also can affect how well the model can be trained, as well as how the learning of the model can be generalized to other problems. Loss functions can be crafted specifically with the particular problem in mind. In case of this work, where the NNs will be used to predict CFD-like physical fields around or inside geometries, loss functions incorporating both pure data loss, as well as, physics equations-based loss and boundary condition loss could be used. This topic will be explored in further detail in Section 2.4.

2.1.3. Optimization algorithms

There are many optimization algorithms that are used for the training of neural networks, like Stochastic Gradient Descent, ADAM [Kingma and Ba, 2014], or L-BFGS [Liu and Nocedal, 1989]. There are two main types of optimizers used for NN training: first- and second-order method optimizers. First-order methods (ADAM, SGD), as the name suggests, use only first-order derivatives to perform an optimization step. Second-order methods (L-BFGS) use both first- and second-order derivatives for this task.

ADAM optimizer

The ADAM optimizer is an extremely popular optimizer used in ML. To understand the ADAM optimizer, another concept that needs to be introduced is the concept of momentum first proposed by Polyak [1964]. SGD uses a gradient calculated on a randomly chosen batch of training set. This means that, if the batch is small, the direction of the descent can oscillate a lot, as it varies from batch to batch. Algorithms with momentum try to solve this problem, by leveraging past gradient information along with the current gradient calculation to improve the stability and convergence rate.

In the momentum algorithm:

$$\begin{aligned} \mathbf{m} &\rightarrow \beta \mathbf{m} - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \\ \theta &\rightarrow \theta + \mathbf{m} \end{aligned} \tag{2.4}$$

At each iteration, the algorithm subtracts the local gradient from the momentum vector, and it updates the weights by adding the new momentum vector to them. The momentum vector is calculated as an exponentially decaying average of past gradients. The parameter β is called *momentum* and is a multiplier of the momentum vector to prevent it from growing too large. Usually, this is set to values of around 0.9.

The ADAM optimizer [Kingma and Ba, 2014] uses adaptive estimates of first- and second-order momentums. Exponentially decaying moving averages of the gradient of the cost function (m) and square of gradient of the cost function (s) are used for the update of the step, which are used to estimate the first- and second-order momentums.

$$\begin{aligned}
&\mathbf{m} \rightarrow \beta_1 \mathbf{m} - (1-\beta_1)\,\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \\
&\mathbf{s} \rightarrow \beta_2 \mathbf{s} + (1-\beta_2)\,\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \otimes \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \\
&\widehat{\mathbf{m}} \rightarrow \frac{\mathbf{m}}{1-\beta_1^t} \\
&\widehat{\mathbf{s}} \rightarrow \frac{\mathbf{s}}{1-\beta_2^t} \\
&\boldsymbol{\theta} \rightarrow \boldsymbol{\theta} + \eta \mathbf{m} \oslash \sqrt{\mathbf{s} + \epsilon}
\end{aligned} \tag{2.5}$$

β_1 and β_2 are model hyperparameters that weight the contribution of the past terms in the update step. Due to the initialization used in this algorithm, where s and r are initialized as zero, the algorithm is biased towards zero values as well. To overcome this, the algorithm introduces bias correction terms $\widehat{\mathbf{s}}$ and $\widehat{\mathbf{m}}$. The final hyperparameter introduced is ϵ, which is a smoothing term and is usually initialized to 10^{-7}.

2.1.4. Overfitting

The training of neural networks is an iterative process, with each iteration updating the parameters of the network to optimize the network output with reference to the ground truth provided. To monitor the progression of training, tools like TensorBoard [Abadi et al., 2015] are used to read the training logs, where for each number of iterations, the state of the loss function is stored along with other metrics if indicated. This is an analogous tool to residual plots in CFD, with a key difference being that in machine learning there is a very critical phenomenon occurring called overfitting. One of the most common metrics for evaluating data fit is R^2, which is calculated as follows:

$$R^2 = 1 - \frac{\sum_i (y^i_{truth} - y^i_{pred})^2}{\sum_i (y^i_{truth} - \bar{y}_{truth})^2} \tag{2.6}$$

Almost always when training a neural network, or indeed any machine learning model, the data set is divided into 2 to 3 parts: training, test, and validation data sets. The training data set is used only for the iterative training process. The validation data set is used for every n iteration during training to test the prediction performance of the model. Finally, the test set is used after the model has finished training to check again the inference performance of the model with the data that the model has never worked with before.

Overfitting occurs when the model starts to fit too well to the training data sets, losing prediction accuracy on the validation data set. This can be easily spotted by tracking training and validation sets losses with the progress of training, as overfitting is characterized by an increasing loss on the validation data set, while the performance of the model on the training data set is still improving or stagnating.

2.1.5. Convolutional neural networks

In this section, the convolutional neural network (CNN) architecture will be introduced.

The main difference between the MLP network introduced earlier and a CNN is the type of neurons used in the layers. CNNs are widely used in pattern recognition and classification of images.

CNNs are made up of three types of layers [O'shea and Nash, 2015]: convolutional layers, pooling layers, and fully connected layers like those seen in MLPs. The stacking of these three different types of layers forms a simple CNN as presented in Figure 2.4.

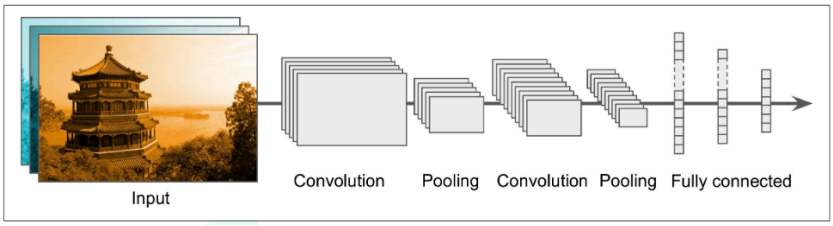


Figure 2.4: Simple CNN architecture. Source: [Geron, 2019]

Convolutional layer

Convolutional layer is formed by a learnable kernel with an activation function. These kernels have three parameters: width, height, and depth. Kernels, as well as images, can be thought of as matrices. For images, each pixel is a matrix entry, where each entry is a numerical value that indicates the brightness of the pixel. Kernels are matrices of learnable weights, where width and height parameterize the size of the kernel and signify how large of a portion of an image the kernel aggregates the features from. The depth of the kernel is normally dependent on the number of colors a pixel has. In a typical screen or camera, a pixel would consist of RGB (red, green, and blue) colors.

The kernel scans the picture f and obtains the pixel values and calculates the scalar product between the learnable weight of each kernel h and the captured pixel values. Then the scalar products in the kernel are summed. After a filter passes the entire image, the resultant set of filter and pixel multiplications form a convolved feature. This process is illustrated in Figure 2.5. Subsequently, the bias term is added to the convolved feature and the result passes through the activation function, forming a feature map.

Each entry in the convolved feature G is calculated as follows:

$$G_{n_H,n_W} = (f * h)_{n_H,n_W} = \sum_j \sum_k h_{j,k} f_{n_H-j,n_W-k} \tag{2.7}$$

A kernel consists of a matrix of learnable weights and a single learnable bias.

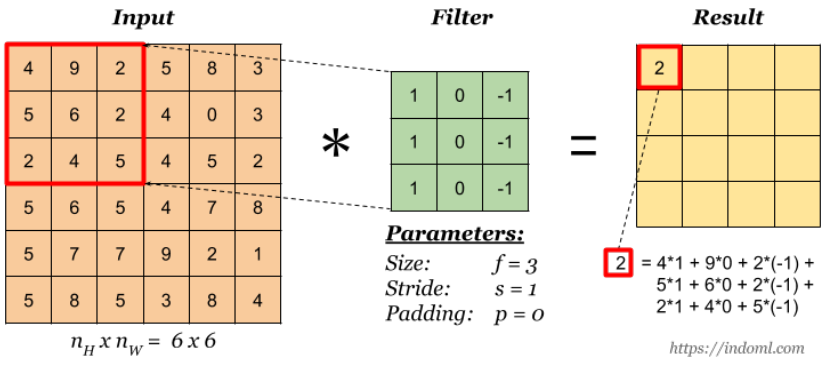


Figure 2.5: Convolution operation. Source: [Prijono, 2018]

Pooling layer

The main function of the pooling layer is the reduction in the dimensionality of the image representation. This is done by applying a fixed-size grid and reading values from its receptive fields that combine the activation map features falling under the grid. Max-pooling (see Figure 2.6) is a method that is often applied, that for each grid application pulls the maximum value for the grid and passes it to a new activation map.

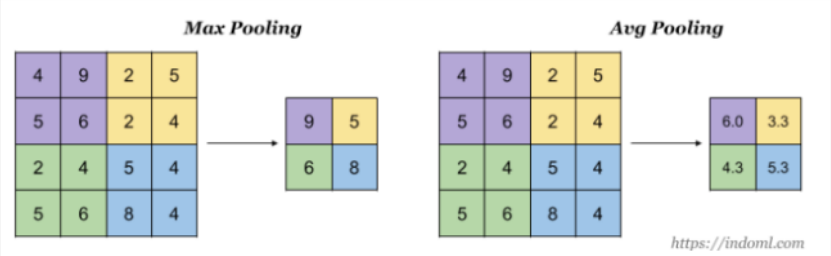


Figure 2.6: Pooling operation. Source: [Prijono, 2018]

Fully-connected layers

Fully connected layers flatten the last activation map from upstream, and using that feature vector, they capture complex relationships between the global features in the image, outputting a one-dimensional feature vector. Fully connected layers are also used to output the information from the network in the desired form. For example, in image classification tasks, the last layer would be fully connected with activation function like softmax, which is used to normalize the output to form a probability. This would give the probability that a given image belongs to a given class.

2.2. Geometric deep learning

This section is a first of many that will introduce more task-specific aspects of NNs. In this section geometric deep learning will be introduced which is one of the key ideas used in the NN models trained for this project. Geometric deep learning was pioneered by the work of Bronstein et al. [2016] who coined the term and laid out the formal foundations of the field. In his paper, Bronstein et al. [2016] presents different deep learning methods, applied before mainly to euclidean domains like 2D planes (images), now extended to non-euclidean data structures, like 3D graphs and meshes. A key distinction is made between two main tasks to be solved in geometrical deep learning, the first one being characterising the data structure and the second one being analyzing the functions defined on the data structures in the non-euclidean domain. Bronstein et al. [2016] focuses mainly on the second problem, which presents different classes of approaches for generalizing convolution-like operations to graphs. Spectral methods, spectrum-free methods, charting-based methods, and combinations of the above are presented in this paper.

Spectral methods are based on the assumption that convolution can be generalized to a graph by looking at linear operators that commute with Laplacian matrix, which is a matrix representation of a graph. Such methods operate on a spectrum of graph weights, which are formed by the eigenvectors of the graph Laplacian. Such methods suffer from a lack of transferability to different domains, as each domain consists of its own fixed spectrum of eigenvectors of the graph Laplacian and the spectral filter coefficients are dependent on those eigenvectors. Spectrum-free methods rely on the fact that a polynomial of the Laplacian acts as a polynomial on the eigenvalues. Thus, it is possible to represent a filter using polynomial expansion rather than relying on the frequency domain of the spectral multipliers. The most notable example of such convolution is presented as part of the ChebNet which relies on Chebyshev polynomials. Unfortunately, those methods also suffer from a lack of transferability to different domains. Charting-based methods, on the other hand, generalize the convolution to non-euclidean domains and utilize pooling of features by means of pointwise calculation of statistics

like mean and covariance. An example of such an approach is proposed by Masci et al. [2015]. The proposed geodesic convolution neural networks are based on local geodesic polar coordinates. The weighing functions can be obtained as a product of Gaussians. MoNet by Monti et al. [2017] extends the work of Masci et al. [2015] by defining the geometric convolutions that allow for a more generic spatial domain framework for geometric deep learning on non-euclidean structures. Implementation of parametric and learnable kernels with Gaussians replaces the fixed kernels, allowing for additional degrees of freedom compared to the other works presented above, where only filters were learnable. A major advantage of the MoNet approach is that it takes full advantage of the mesh-specific geometric structure, compared to approaches that treat the mesh as a graph where normals and coordinates are features of each node. In addition, it requires no additional processing steps that would introduce interpolation errors or limit the designer's choice with regard to freedom of meshing. For these reasons, a GNN based on MoNet convolutions will be used in this work, which are defined as follows:

Let w be a set of weighing functions $w = (w_1, .., w_J)$ such that $w_j : R^2 \rightarrow R$ and $u : x, y \rightarrow \rho(x, y), \theta(x, y)$ is the local polar geodesic distance between the vertices x and y. Given a mesh with vertices V and vector field $f : V \rightarrow R$ on this mesh, the patch operator is defined as:

$$\forall x \in V\,, j \in \{1, ..., J\},\ D_j(x)f = \sum_{y \in N(x)} w_j(u(x, y))f(y) \tag{2.8}$$

The output of the geometric convolution layer with learnable kernel weights $g = (g_1, ..., g_J)$ is given by:

$$f * g = \sum_{j=1}^{J} g_j D_j(x)f \tag{2.9}$$

In MoNet, the weighing functions used are parametric Gaussian kernels.

$$w_j(u) = exp(-\frac{1}{2}(u - \mu_j)^T \sum_{j}^{-1}(u - \mu_j)) \tag{2.10}$$

with learnable mean μ_j and diagonal covariance matrix $\sum_j$

2.3. Neural network based surrogate modelling for CFD

Numerical simulations, such as CFD, used to capture flow behavior, produce a huge amount of numerical data. The emerging field of machine learning thrives in environments where large quantities of data are present. Due to efficient operation and relative ease of use, many ML techniques are being applied to improve CFD simulations or leverage CFD simulations to make models of the flows. What makes ML techniques even more interesting is their ability to capture patterns from the data. Similarly to tasks like image classification, where trained NN classifies images based on the foreground object presented in the image, NN fed with FVM simulations can recognize different patterns in the data within the geometrical domain. Fluid mechanics is full of such visual patterns, many of them repeating across different scales [Taira et al., 2017].

Distinctions can be made about the field of ML for CFD, and those are based on where in the process the ML component is being used. Two distinct groups can be formed here: approaches incorporating the ML solutions into the CFD solver itself, improving or replacing some components of the solver. The other approach would be to use the data generated by the CFD solver to extract information or assimilate the data in some way, allowing for posterior usage. One of the main approaches in this field is to use the CFD-generated data to build surrogate models. In this section, a concise review of CFD-data-based NN surrogate modelling approaches will be made.

A way to use machine learning for CFD is to use the ground truth data generated by the solver, to train a neural network, essentially replacing the solver. In this scenario, the neural network, for a

specific range of cases on which it was trained, could emulate the solver and predict CFD-like results for new geometries. One of the first research groups to propose this idea was Guo et al. [2016], where the authors proposed a framework for 2D and 3D flow prediction using a CNN, for steady flow. The simulation data was fed to the CNN by applying voxelization to the CFD simulations, which means that the fields from the mesh were projected onto a Cartesian grid. The sign distance function (SDF) was used for geometry representation. A single set of stacked convolutional layers was used to encode the geometrical features from the SDF. Then, for each field component separately, a decoding branch is created to predict the field value. The network was trained on low Reynolds number ($Re = 20$) CFD simulations, generated using LBM. The work showed a comparison between SDF and binary geometry representation, showing a much better accuracy when SDF was used. It was argued that this was caused by the more global information contained within the SDF, whereas binary representation is much more locally restricted. The performance achieved on the validation set for the experiment was not fully analyzed, although the visual field comparison on the validation test samples presented appeared accurate. For 3D geometries, the average velocity field error for the validation data set was below 3% compared to the LBM simulations.

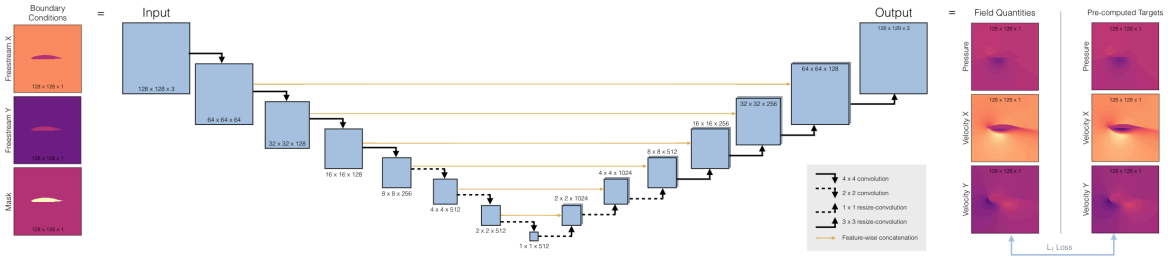


Figure 2.7: U-Net architecture as used by Thuerey et al. [2020]. Source: [Thuerey et al., 2020].

Thuerey et al. [2020] used a U-Net architecture shown in Figure 2.7, based on the CNN architecture, to construct a surrogate model trained on 2D RANS simulations of airfoils with a much wider range of Reynolds numbers $[0.5, 5]$ million. The authors present the results of how the number of trainable parameters of the models as well as the amount of training data impact the performance of the models. Smaller models provided lower error on small data set and vice versa - the more data was available, the better the larger models performed in relation to smaller models. Furthermore, the generalizability of the models was assessed by modifying the angle of attack of the data set to be outside the scope of the original data sets. Then various composition data sets were fed to the model, showing a decreased performance of the model, the less data from the original data set was present. Although the modifications to the samples outside of distribution were low, the errors were still relatively acceptable, below 3% over the three channels (x, y velocities and pressure).

In the first papers Guo et al. [2016] and Thuerey et al. [2020] presented a common research approach to input data into a CNN-derived model. In these approaches, CFD data calculated on an unstructured grid are projected onto a Cartesian regular grid as seen in Figure 2.8, which later allows the data to be easily inputted into a CNN framework. This approach comes with drawbacks. Resampling causes a decrease in the order of accuracy of the CFD data due to the unavoidable extrapolation and interpolation required in the process. Additionally, projection of the data to a Cartesian grid causes loss of information in highly sensitive areas of the flow, like in the boundary layer area, where the CFD mesh will have much more cells to capture the high gradients of the solution. Resampling loses this information, unless a prohibitively large Cartesian grid is used of the order of cell size of the boundary layer cells in the CFD mesh, but this is absolutely unfeasible from the computational effort standpoint. These methods work in a research environment with relatively simple geometries, but in an industrial setting using an unstructured grid with a variable grid from sample to sample in the training set is unavoidable [Kashefi et al., 2021].

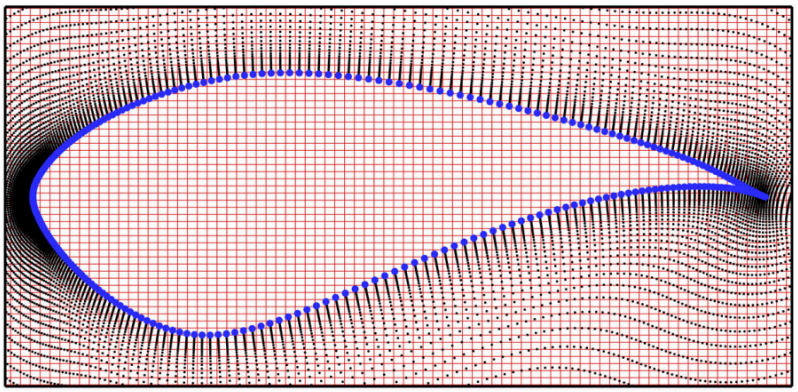


Figure 2.8: Overlay of Cartesian grid and structured C meshes. The first is the input structure of CNN and the other is the typical simulation mesh. Interpolation between the two will introduce large errors. Source:[Bhatnagar et al., 2019].

Another variation on the topic was presented in the paper by Kashefi et al. [2021], where instead of using a CNN-based model, PointNet architecture [Qi et al., 2016] was used to work directly on a point cloud with fields. This architecture was originally developed in 2016 for 3D computer vision tasks, such as point cloud classification. This is a very convenient solution, as the point cloud can be exported directly from the CFD mesh. This is a large step forward in the usability direction, allowing for usage of unstructured meshes. In this work, the point cloud represents both the field data and implicitly represents the geometry by the void in the point cloud. The model was trained on 2D CFD results of various simple geometrical shapes at low Reynolds number conditions (20 to 84). Overall, the model produced good results on the validation test set with the lowest errors for pressure and slightly higher errors for velocity components. The max errors appeared on the walls of the geometries, where no slip conditions were assigned.

In this work, some effort was put into analyzing the trained model from the aspect of explainability, that is, analyzing which points in the domain are seen by the networks as critical points in the prediction of global features. Then, for critical and non-critical points the momentum and continuity residuals were computed by running a single training iteration, to evaluate the gradients of solution with respect to input point locations. The results of this study show higher residual values for critical points. This was explained by the fact that the critical points are much more affected by changes in location of the points due to their large contribution to the global features, hence having higher residual values. The work on PointNets was then extended to work within the PINN framework [Kashefi and Mukerji, 2022]. Further details on this topic will be presented in Section 2.4

2.4. Incorporation of physics into neural network-based surrogate models

In this section, different approaches to physical prior knowledge incorporation, specifically into machine-learning-based surrogate models, will be presented. This is often seen as a potential way to improve generalizability and transfer learning of machine learning models. Incorporation of physical prior knowledge was already indirectly done by training the model on physical data, such as CFD results, but there are ways to directly introduce such prior knowledge into the models itself.

2.4.1. NN-solver coupling

One of the ways of including physical prior knowledge into machine learning models is to couple the NN with a physical solver. In the paper by De Avila Belbute-Peres et al. [2020].

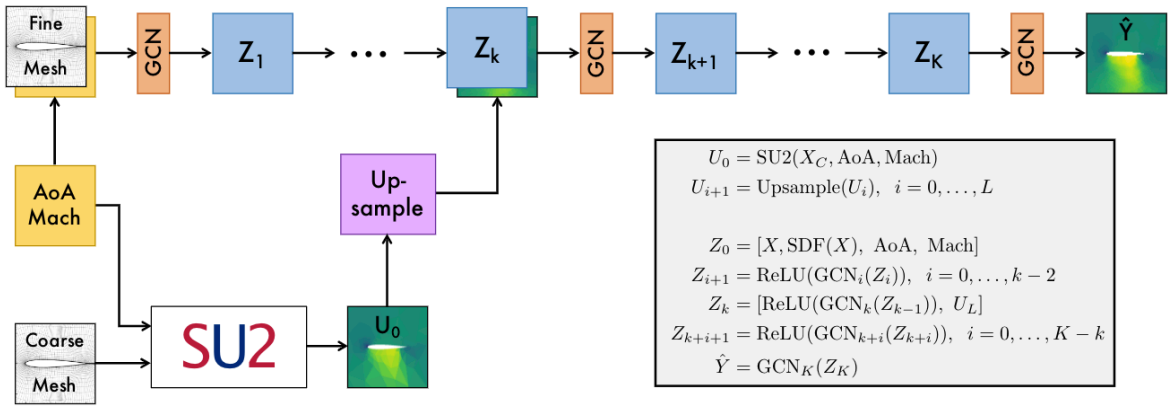
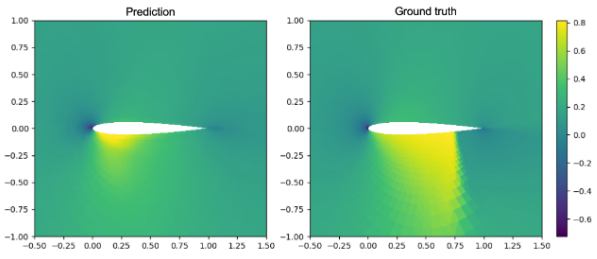


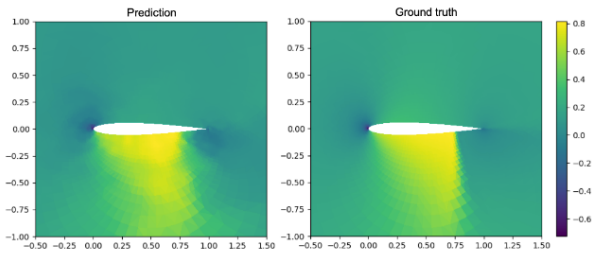
Figure 2.9: Diagram of CFD-GCN architecture. Source: [De Avila Belbute-Peres et al., 2020].

A differentiable CFD solver (SU2 [Palacios et al., 2013]) was embedded in a graph convolutional NN (GCN) (see Figure 2.9). The input angle of attack and Mach number are fed into both the solver branch and the GCN branch of the model. In the solver branch, a simulation is run on a learnable coarse mesh grid. The GCN branch, apart from the case-specific scalar values, is fed a refined version of the mesh, which is processed by graph convolutional layers. Then the output of that graph convolutional layer is concatenated to the result of the coarse CFD simulations, which first pass through an upsampling step. Finally, the CFD results and the fine mesh features are processed inline by the GCN to produce a final high-fidelity solution. In the learning process, the coarse mesh is optimized for the best final performance thanks to the adjoint method for reverse mode differentiation available in SU2.

The interpolation and extrapolation capabilities of the model were tested by predicting results for α-M parameter pairs that were either in or outside of the training data set. The extrapolation test setup was particularly interesting due to the construction of the train/test set. In the train set, cases with no shock waves were used, and in the test set, only the conditions that cause shock discontinuities were used. As part of the analysis, an ablation study was performed, testing the GCN branch without the CFD solver, coarse CFD solver by itself, and the entire model, but with non-learnable mesh. Generalization is where the addition of the CFD solver into the GCN dramatically improves performance. In this part of the study, the GCN alone performs very poorly and even coarse simulation outperforms it for this given training time. The field results are presented in Figure 2.10. The results show that the CFD-GCN model is able to capture shock relatively accurately, unlike the GCN model, which fails to generalize and predict flow structures similar to what it was trained on. This shows that this type of CFD solver-in-the-loop implementation can be viewed as a form of physics-informed learning. As this method uses a CFD solver in the loop, it is important to take into account the inference time. For this simple case of 2D airfoil with a fine mesh having only 6648 nodes and the coarse mesh having 354 nodes, the inference time took around $2s$ vs. $0.1s$ for the GCN model and $137s$ for the CFD simulations on fine mesh.



(a) X-velocity component comparison between GCN model and CFD. Source: [De Avila Belbute-Peres et al., 2020].



(b) X-velocity component comparison between CFD-GCN model and CFD. Source: [De Avila Belbute-Peres et al., 2020].

Figure 2.10: Field predictions comparison for generalization experiment. Samples for the test set.

2.4.2. Physics-informed neural networks - PINNs

Physics-informed neural networks, first proposed by Raissi et al. [2019] aim to use neural network models to solve partial differential equations, by harnessing the power of the underlying equations and the ability of neural networks to model complex non-linear functions [Hornik et al., 1989]. His way to approach this was to introduce additional terms in the loss function that were evaluating the underlying physical equations on the domain of interest of the model, as well as terms taking into account the boundary conditions. This addition utilized the automatic differentiation commonly present in neural network models to obtain the necessary gradients for the equation-based loss calculation. This simple addition was aimed at decreasing the model tendency to overfit in low-data regimes, as well as to improve its generalizability and overall performance in low-data scenarios. This is mainly accomplished by the additional physics-based loss terms like L_{PDE} (see Figure 2.11) that during training penalize the solutions that do not adhere to the underlying physics and have high loss, which can be seen as a way of applying physics-based regularization to the network.

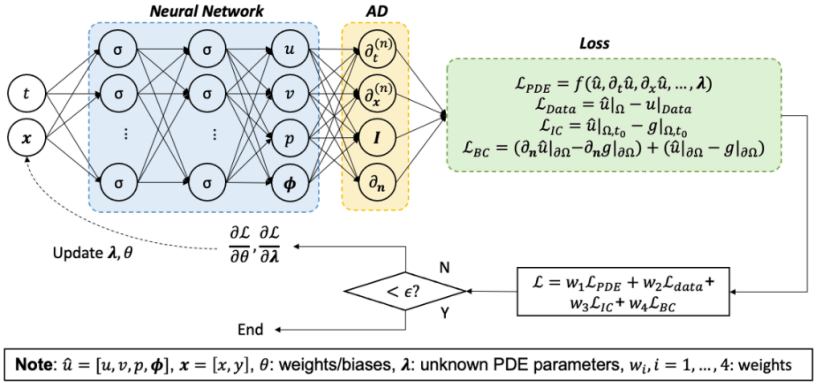


Figure 2.11: Schematic of PINNs structure. Source: [Cai et al., 2021a].

In its most common implementation, the backbone of a PINN can be a feedforward MLP neural network. Then utilizing capabilities of the packages like TensorFlow, the gradients of the output with respect to inputs can be obtained, and based on those the PDE loss, BC and IC losses can be calculated. To evaluate them, a set of collocation points is also required on the boundary and inside the domain. Those can be randomly sampled within the regions of interest or can be used to target a more specific area of the domain. Apart from the new loss terms, the usual loss coming from the data itself is still used.

Jin et al. [2021] presented a paper in which MLP-based PINNs (VP-NSFnets) were used to solve incompressible, unsteady flows in simple cases, using a data-free approach. In the data-free approach, good results were obtained for low Reynolds number flows. The models were also used to solve a turbulent channel flow at $Re = 1000$, showing the ability of the model to sustain turbulence, although with high pressure errors, increasing with timestamp time.

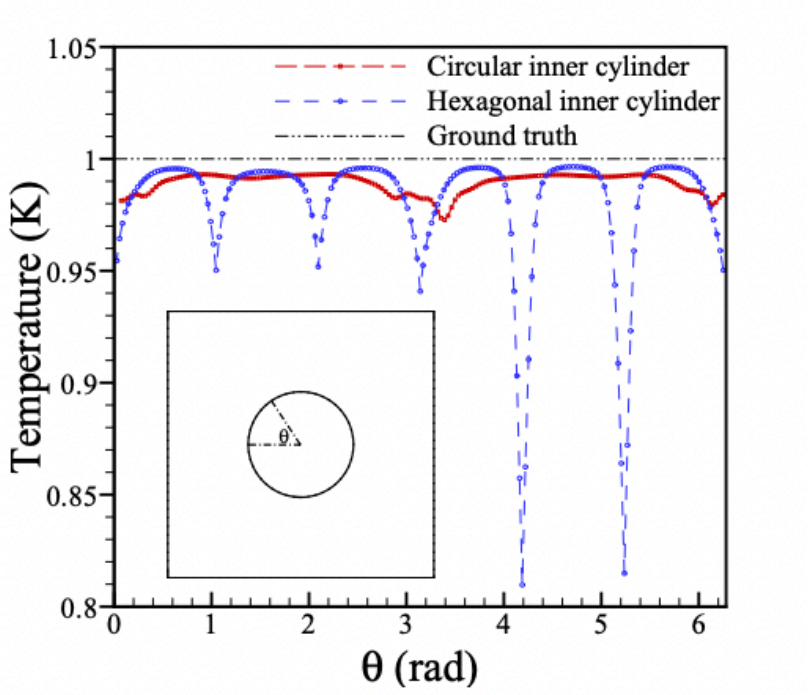


Figure 2.12: PointNet-based PINN, generalization experiment surface temperature prediction on the test set. Source: [Kashefi and Mukerji, 2022].

Kashefi and Mukerji [2022] presented work using a PointNet as a base for a PINN surrogate model, where predictions were obtained for simplified flow cases for various simple geometries at the same flow conditions. This architecture has the advantage of being able to directly use the simulation point cloud as collocation points or observation points. The models were used in both data-free and ill-posed approaches. In the ill-posed experiment, the 2D thermally driven convection problem was solved with incomplete BC observation data. The model was shown to be highly dependent on pressure data observations, as the model without it predicted a pressure field with average error increasing more than 70 times. Generalization performance was tested on unseen geometries, showing good agreement for circular geometries, but much poorer performance on hexagonal sample, with high errors at the corners of the geometry, as seen in Figure 2.12.

The majority of work in this area of research uses different implementations of the PINN framework in low-speed, low-Reynolds number settings. The work of Mao et al. [2019] is one of very few examples of PINN applied to high-speed flows. In this work, Euler equations were used in the loss for both inverse and forward problems. For forward problems, two approaches were tested: either with specified BC and IC or without specified BC and IC but with scattered observations of field values in the domain. The results presented show that the model is capable of accurately capturing shocks, with the distribution of collocation points in the domain being the main factor that affects the resolution and the number of collocation points being a secondary factor. Mao et al. [2019] identified a key problem with this work: in order to be able to place the collocation points accurately in the domain, one first has to know the approximate position of the shock. Since this work was published, several works have been written on conditional domain sampling for collocation points such as Lu et al. [2020] or Nabian et al. [2021] that tackle this issue. Lu et al. [2020] proposed to address this problem by adding more collocation points in the domain where the loss magnitude is greater than a specified threshold. In the case of Nabian et al. [2021], the method of importance sampling [Press et al., 2007] was used for PINN training. This method relies on selecting training points based on the proposal distribution that is proportional to a piecewise constant approximation of the loss function.

In many of the scenarios presented, either no simulation data was used for the training, or analytical test cases were used, or low Reynolds number simulations were used with no turbulence modeling

necessary. Very few works attempt to assimilate data generated at higher Re using the PINN framework. Eivazi et al. [2022] uses simulation data from various turbulence models (LES and DES) as training data for a series of test cases (without mixing the turbulence models) and predicts a RANS solution without any assumption on turbulence. The training simulation data are used only to specify the values of the fields at the boundaries of the domain. Among other cases, LES simulation data were used to solve the flow over the trailing edge of a NACA4412 airfoil, with $Re_c = 200000$, with the results obtained showing a mean velocity error below 3%. It would be interesting to investigate how such a loss formulation would behave on a model used for surrogate modeling rather than solving a single case solution.

CNN-based PINNS

MLP-based PINNs are relatively simple to implement and can produce accurate results, but lack the ability to utilize the geometrical information about the data. Additionally, due to pointwise learning, they require a lot of collocation points for the evaluation of the PDE loss to arrive at a converged solution. This introduces a large computational overhead. An approach to deal with some of the short-comings of MLP-based PINNs is to use CNN architecture as a backbone of the model. Ma et al. [2021] presented a PINN based on the U-Net architecture. U-Nets are encoder-decoder networks based on CNNs, that allow for efficient spatial data processing, making it possible to both extract local and global spatial features from the structured images or meshes. In this instance, a binary geometry representation is used to process the geometry. The Navier-Stokes equations (N-S) based loss is used on the inner domain, BC loss is used on the boundary, and the data loss can be added if observations are present. Applying the PDE physics-driven loss function to a CNN introduces a complication related to the evaluation of the gradients, as unlike in MLP, the simple pointwise gradient evaluation in backpropagation cannot be used. In this case, convolutional filters were constructed based on finite differences to evaluate the differential terms in the $\mathcal{L}_{PDE}$. One of the experiments of particular interest was a model trained on a steady flow around the cylinder. For this model, the geometry was fixed as well as the inflow conditions, with the only variable being the Reynolds number. It was varied between 0.8 and 20.2 for the training. This range of Reynolds numbers was selected, as in this range the flow structures created in the wake of the cylinder vary considerably, allowing for a relatively challenging test for the architecture. According to the literature [Zdravkovich and Bearman, 1998], in this span of Re for flows below $Re = 5$ a creeping laminar flow should be present and for flows above that threshold, the flow should transform into one with a pair of trapped symmetric vortices behind the cylinder. The model was first tested in interpolation tasks with good agreement with the finite-volume method (FVM) ground truth. Subsequently, the model was tasked with extrapolating to higher Reynolds numbers. The accuracy of the model was measured by comparing the length of the trapped symmetric vortices behind the cylinder between the FVM simulations and the predictions of the NN model. The results of this study are presented in Figure 2.13.

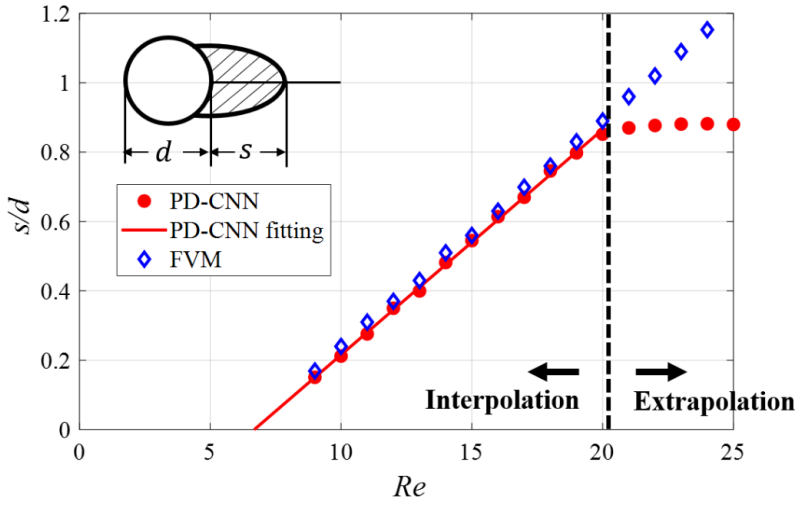


Figure 2.13: Size of vertices pair against Re. Comparison of performance of CNN models in interpolation and extrapolation. Compared to FVM ground truth data. Source: Ma et al. [2021].

An approach to learning from irregular meshes was introduced by Gao et al. [2021a], using a CNN as the backbone of the PINN. The approach encodes an elliptical transformation into the CNN to project irregular meshes and governing PDEs into a regular domain, allowing for CNN capabilities without issues associated with mesh projection. This model was called PhyGeoNet. The elliptical projection is deterministic and pre-computed, resulting in no data loss and low computational effort. This approach also encodes boundary conditions into the padding of the convolutional filter, simplifying the loss function, and making the learning process faster. This architecture is shown to work well on irregular 2D geometries for heat and N-S equations, without the need for labeled data, even for variable geometries. However, it is limited to 2D irregular domains and can only handle a small subset of geometries with more than five C^0 continuous edges.

GNN-based PINNs

A subsequent article from the same research group [Gao et al., 2021b] proposed an architecture based on graph neural networks (GNNs) to address the limitations of PhyGeoNet. The model operates on unstructured grids, with mesh nodes acting as graph nodes and mesh edges acting as graph edges. The input is the spatial position of the node, and the output is the field quantity at the node. The model uses graph convolution operations and message passing to update the feature vectors of the graph. The spectral-based method of the Chebyshev graph convolutions [Defferrard et al., 2017] is used. Piecewise polynomial basis functions are used to reduce the dimensionality of the search space. The conservation laws are discretized using the continuous Galerkin method, and the variational formulation of the PDE residuals is used for the physics-informed loss. The model can be used for both forward and inverse problems, and its application to fluid mechanics was demonstrated in lid-driven cavity flow and idealized stenosis in medical field. The predictions obtained had errors below 1% and 2%, respectively, relative to the ground truth simulations.

2.4.3. Data transformations

A more straightforward way to improve the physical understanding of the models is to take advantage of physical knowledge in the data pre-processing pipeline. In particular, applying appropriate data scaling and normalization can lead to great training improvements, as shown by Cao et al. [2016].

More generally, physics-based data enrichment, which has shown promise in the work by Kissas et al. [2020], is essentially using physics-based non-dimensionalization of the data inside the PINN architecture used to recover full-flow data from scattered human MRI data. In this work, both physics-based non-dimensionalization and scaling to zero mean and unit variance were used. The non-dimensionalization proposed was:

$$L = \sqrt{\frac{1}{N}\sum_{j=1}^{D}\left(A_0^j\right)}, \quad U = 10,$$

where $j = 1, \ldots, D$. At this point, we define the quantities:

$$\hat{u} = \frac{u}{U}, \quad \hat{A} = \frac{A}{A^o}, \quad \hat{p} = \frac{p}{p_0}, \quad x_* = \frac{x}{L}, \quad t_* = \frac{t}{T},$$

where $p_0 = \rho U^2, T = \frac{L}{I}$ and $A^o = L^2$. In this work, the impact of scaling and non-dimensionalization was examined versus a case with no data processing used. The results show that the model with no scaling and no non-dimensionalization fails to learn the desired velocity through y-bifurcation, most probably due to the input data being of highly different scales. In such a case, lack of scaling can have a detrimental effect on training of the network, as the network will be biased towards variables of higher magnitude. On the other hand, the models where both scaling and non-dimensionalization were used predict the velocity evolution with high accuracy when compared to synthetic test data. The effects of scaling to zero mean and unit variance are well documented [Glorot and Bengio, 2010] , where the approach mitigates the problem of vanishing gradients, but the effect of the physics-based non-dimensionalization was not examined in isolation.

Thuerey et al. [2020] in his work, where the U-Net architecture was used to construct surrogate models trained on RANS simulations, showed the impact of non-dimensionalization in isolation. Three models were compared: model A where no non-dimensionalization was done, model B where the pressure and velocities were non-dimensionalized by free stream velocity and free stream velocity squared, respectively, and finally model C where additionally to non-dimensionalization with reference values, pressure null space was removed. It is important to note that for all models, the last step of preprocessing is the scaling of the data between the $[-1, 1]$ range. The trained models were evaluated on 400 randomly selected samples from the UIUC airfoil database. The results show that the average absolute error with respect to the ground truth pressure and velocity for model A is very large, more than 200 times higher than that for model B. Model C achieved a four-fold decrease in error compared to model B. A graphical representation of the differences between the model results is presented in Figure 2.14. The positive effect of such data preprocessing was associated with the simplification of the solution space, as well as simplification of relationships between the input and output quantities.

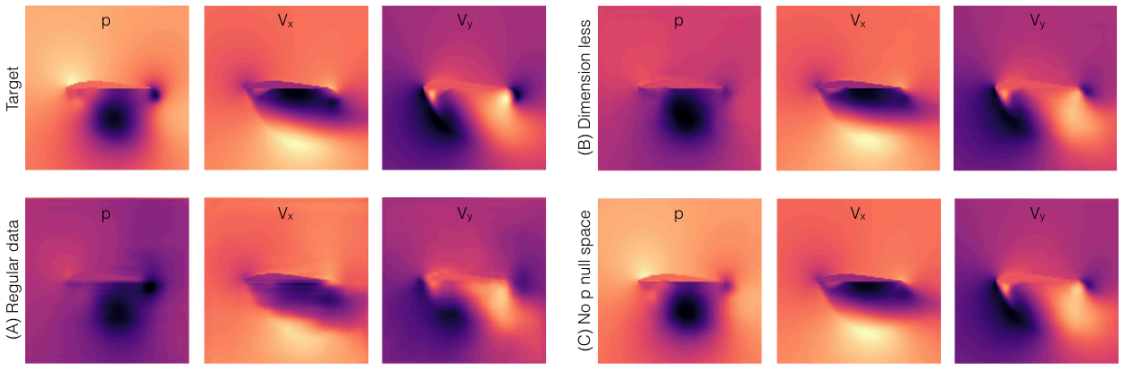


Figure 2.14: comparison of the different non-dimensionalization strategies on airfoil prediction of pressure and velocity components. Source: Thuerey et al. [2020].

2.5. Transfer learning and neural networks

In order to train machine learning models, data is often essential. In real-life applications, there are very few examples where such data is easy to obtain. Data generation requires a lot of work to create, especially when it comes to labeled data. One of the most simple ways to generate labeled data is to use humans to do the labeling. Even nowadays humans are used in data labeling processes [Von Ahn et al., 2008], for example, using the CAPTCHA system and its derivatives, which was acquired by Google and serves as a tool for human verification. Automated data labeling strategies are available [Güne¸ et al., 2018], but this only solves the problem of labeling the data. Often, obtaining the data itself is a major issue. This is where transfer learning is often used. This set of techniques allows one to transfer parts of the information from a data set or a trained model, where there is abundant information, to an area where the information coverage is more sparse. Provided there is some transferable information or relations between the data, utilizing transfer learning can reduce the necessary data requirement in the new target domain [Pan and Yang, 2010].

This work is focused on fluid mechanics applications, but the field of transfer learning for fluid mechanics applications is a relatively unexplored territory. Therefore, in this chapter, works from other fields, such as image processing, will be used to fill the knowledge gaps. Some of the model architectures used in NN for fluid mechanics are related to or even are directly taken from the work done in the image processing field, so one would hope that the experience gained in those related fields can be transferred to application in fluid mechanics.

Pan and Yang [2010] wrote one of the most fundamental surveys on transfer learning (TL). In this work, mainly clustering, regression, and classification tasks were considered. To keep the work consistent, the notation and nomenclature will be based on that foundational work.

2.5.1. Nomenclature and type of transfer learning

Domain $\mathcal{D}$ consists of feature space $\mathcal{X}$ and marginal probability distribution $P(X)$, where $X = \{x_1, ..., x_n\}$ where n corresponds to the sample number. In a given $\mathcal{D}$, the task consists of a label space $\mathcal{Y}$ and a predictive function $f(\cdot)$, where task is $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$. f can be used to predict the label y_i for a given x_i.

The source domain $\mathcal{D}_\mathcal{S}$ is the domain that will be used as a source of transferable knowledge to the target domain $\mathcal{D}_\mathcal{T}$.

There are two main distinctions in transfer learning. Transfer learning settings describe the data landscape from the perspective of labeled data availability. Transfer learning approaches define the strategies that can be used. These approaches are very often setting-dependent.

Four main TL settings can be distinguished. Firstly, there is traditional machine learning that does not utilize transfer learning. In this setting, the source and the target domains, as well as the source and the target tasks, are the same. Secondly, there is inductive transfer learning, which uses the same source and target domains, but only related source and target tasks. Then there is unsupervised transfer learning, for which both the domain and the tasks only share some relation. Finally, there is transductive transfer learning which uses related domains but the same tasks between source and target.

There are several approaches to TL, and they can be selectively applied to different TL settings. Instance-transfer aims to transfer some data from the source domain, after transformation, such as reweighting, to the target domain. Feature-representation-transfer approach is based on finding a suitable learned feature representation of the source domain to enhance knowledge transfer to the target domain. Parameter-transfer approach is based on transferring model hyperparameters from the source domain to the target domain. Relational-knowledge-transfer is based on finding relations between the source and the target domains data and utilizing those to transfer-learn.

For this review, transferring knowledge of parameters and of instances will be presented in more detail, as those approaches are seen as most applicable to our problem.

2.5.2. Transferring knowledge of instances

In his work Pan and Yang [2010], introduces several approaches of instance-transfer to inductive TL. One of those is an algorithm called TrAdaBoost developed by Dai et al. [2007]. This algorithm is used in a data setting where labeled data is available in both the source and target domains, with both domains having the same labels but different distributions. Assuming that there is some overlap in the domains, the algorithm aims to iteratively reweigh the source data to promote the helpful data over the unhelpful data. This is done by using a base classifier that is trained on the reweighed source and target data but is evaluated only on target data.

In the work of Wang et al. [2018b], an approach was presented to deal with the transfer of knowledge between tasks with overlapping domains. The proposed approach assumes that there is a source domain with a large amount of labeled data and a target domain with a small amount of labeled data. In this work, first, a model is trained on the source domain. The trained model is then used to evaluate samples from the target data set. This is done using the data dropout method first proposed by [Wang et al., 2018a], where the target data are evaluated on the pre-trained model from the source domain. The influence on the loss of each sample in the target data set is calculated using the influence metric derived by [Wei Koh and Liang, 2017], which calculates the influence of the loss based on a sample from validation and test sets from the target domain. If the influence of the given training samples is greater than zero, the sample is removed from the training set. Subsequently, such an optimized training data set can be used to retrain the pre-trained model. This method was tested on an extensive array of image classification tasks on various data sets, with good results showing the effectiveness of the method, even in settings of non-uniform data distribution (class distribution) in the test set.

2.5.3. Transfer Knowledge of Parameters

In this approach, it is assumed that for the target and source domains, the task parameters should be relatable and could be transferred from the target to the source to improve performance. In the paper by Yosinski et al. [2014], the approach of transferring network layers, trained on a source domain, to a network that is going to be trained on the target domain is closely analyzed. The authors trained two networks (A and B), each on a different half of the ImageNet data set. They then transferred a varying number of layers from network A to network B, and tested the performance of the resulting network (AnB) on the target data set. The results showed that AnB, where the n number of first layers of the network was fixed after 1, 2 and 3 layers, had almost the same accuracy as the base B model (see Figure 2.15). This suggests that the early layers of the network were responsible for general features of the classes that were transferable to the new data set with different classes. They also found that when 4, 5 or 6 first layers were transferred, performance suffered, after which it recovered back to baseline B levels. Additionally, they also repeated the same experiment with both AnB and BnB, but this time with all layers learning. This is referred to as AnB^+ and BnB^+. The results show that when all layers are allowed to learn, the performance of the transferred network improves beyond the performance of the baseline model.

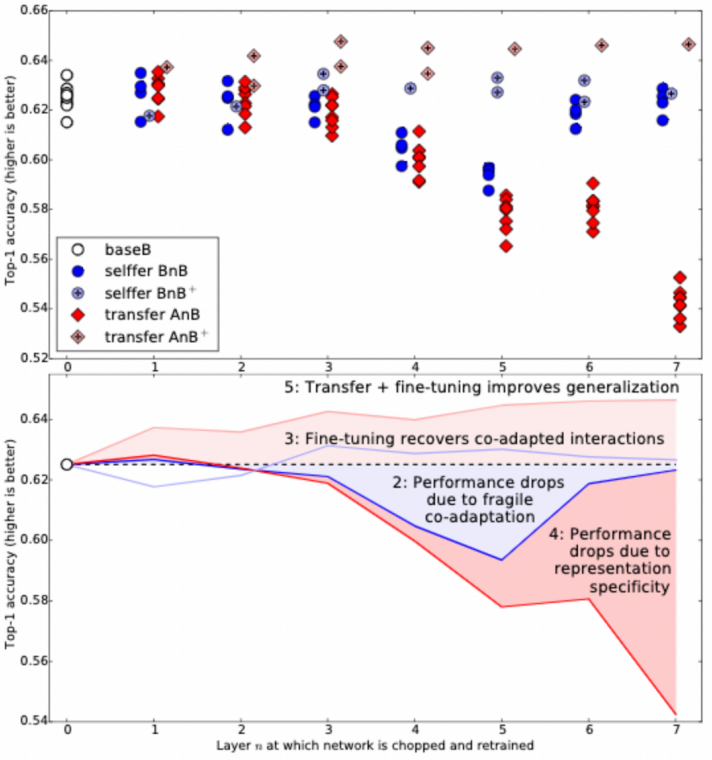


Figure 2.15: Results showing performance curves for transfer knowledge of parameters study, between the target and the source networks. Different number of layers are transferred from the source to the target and the performance is evaluated to assess the impact of such transfer. Source: [Yosinski et al., 2014].

Chen et al. [2021] presented work that examined parameter transferability for PINNs. In this work, the models were trained in the data-free regime, with training supervised only on the physics-informed loss. Several variations of MLP-based PINNs were tested: the original PINN framework [Raissi et al., 2019], PINN with learning rate annealing [Raissi et al., 2020] and GP-PINN [Wang et al., 2020]. The transfer learning was tested on two PDEs separately: first, solving the Helmholtz equation with different forcing terms, and second, solving the N-S equations at different Reynolds numbers. In each case, a benchmark case was set, and then transfer learning was tested applying the first layers n of the trained benchmark model to the different test cases. In the case of the N-S equations, the benchmark was set at $Re = 100$, and the target tasks were set at $Re = 10, 200, 300, 400, 500$ respectively. In both cases, the domain was 1×1 square. In the case of N-S equations, the setup represented the lid-driven cavity flow. The results show that the average L2 error on the fields decreased for all cases, and all transfer levels improved the predictions. For the lid-driven cavity flow, the best results were obtained for GP-PINN.

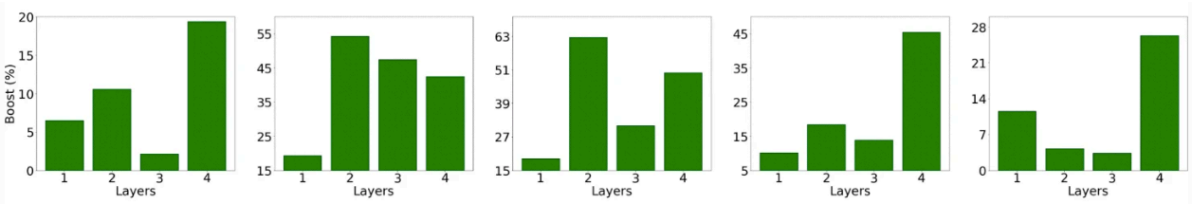


Figure 2.16: Parameter transfer by layer, boost (relative improvement) from source to subtask. Left to right Re=[10,200,300,400,500] Source:[Chen et al., 2021].

The transfer learning capability of the VP-NSFnets [Jin et al., 2021], introduced in Section 2.4.2, was tested using the Kovasznay flow. The model trained on the data at $Re = 40$ was used to transfer to a new task, set at $Re = 60$. The model weights of all layers of the trained network at $Re = 40$ were used

to initialize the layers of the network with the target of $Re = 60$. Subsequently, the target network was fine-tuned with new BCs. The result was a decrease in runtime by 80%, compared to a network trained from scratch for the same Reynolds number. The accuracy obtained was also better for a converged fine-tuned model vs. model trained from scratch. The transfer from $Re = 40$ to $Re = 80$ was carried out in the same way, and the conclusions were also valid for this experiment, although slightly lower accuracy was achieved for the fine-tuned model, although still superior to the network trained from scratch for the same target conditions. This would mean that not only did transfer learning help reduce computational cost, but also had a positive impact on accuracy itself.

2.6. Conclusion

This literature review tried to provide a ground-up view of physics-informed surrogate modeling techniques along with a transfer learning background. In the process of the review, several gaps in the current knowledge were identified. The thesis work that follows, will aim to tackle those research gaps and, hopefully, bring value to the field. Surrogate modeling used so far in research lacked powerful models with the geometrical processing power, which is seen as one of the main limitations of the methods presented. Geometrical deep learning offers an opportunity to combine the latest research in geometry processing with research in physics-informed machine learning. The application of state-of-the-art PINN frameworks to GNN models appears to be a clear research direction. It was shown that PINNs in the data-free regime fail to produce accurate generalizable results for industrial cases, but the hope is that observation data coming from CFD results can improve the performance of such models. Physics-based data augmentation by non-dimensionalization, in a number of works presented, has also shown a promise to improve the training of surrogate models and combining such methods with geometric deep learning, and performing detailed performance analysis that is currently lacking in the field, presents itself as a feasible research direction.

Another promising research direction was inspired by the work of De Avila Belbute-Peres et al. [2020], where GCN was linked with a low-resolution CFD solver to produce high-resolution Euler CFD results for 2D airfoils. This exact method does not seem to be suitable for industrial applications, in mind for the thesis, due to suspected large computational overheads at inference time needed for more realistic geometry 3D simulations. Nevertheless, the method is inspirational, as the low-resolution Euler CFD simulations that was used as the input field, can be substituted with an even simpler approximation of the flow, like, for example, potential flow. This would essentially force the neural network not only to upsample the simulation result, but also to correct it to match the target simulation physics, in the case of this work, most probably RANS turbulent simulation. This seems like a research avenue worth exploring.

Apart from Chen et al. [2021], little research has been done on transfer learning applied to physics-informed machine learning, and this avenue of research appears to be another domain worth exploring. What is missing in the field is a clear way to assess the similarity of samples in the target and source domains. Such similarity metric perhaps could be correlated with the positive transfer of information between target and source domains to allow for a solution for instance-based transfer learning applicable to surrogate modeling.

3

Research Question & Objectives

3.1. Research Question(s)

Based on the conclusions of the literature review, the main research question proposed for the thesis is:

> "How different ways of including physical priors into NN surrogate models for fluid flow prediction affect the accuracy of inference, transfer learning, and generalizability capabilities?"

3.1.1. Sub-questions

Based on the research question, several sub-questions were formulated to arrive at the final answer to the overarching question from the above section. To be able to compare physics-informed approaches, a baseline will have to be established, which leads to the first sub-question:

> "Based on the selected test case data set, what is the performance (at inference time, transfer learning, and generalizability) of an optimized model that does NOT leverage physical prior knowledge?"

Subsequently, having implemented the proposed approaches to improving the models by leveraging physical knowledge, the following question has to be answered:

> "What is the performance of the physics-informed model at inference time and in the transfer learning task, and what is the impact of the methods on the generalizability of the model?"

Finally, if time allows, combining the different approaches together will be attempted, which leads to the final sub-question:

> "Can the different methods be combined, and if so, what is their cumulative effect?"

3.2. Research Objective

Firstly, to be able to compare the performance of surrogate model predictions, a library of CFD simulations will have to be created. This part is influenced by the creators of the ML software used for this

thesis, as they have a commercial interest in making the model function on a particular practical test case, in this case an automotive parametric HVAC geometry. The creation of the library will involve CFD study of the existing parameterised geometry to discover its sensitivity to different parameters. If necessary, the parameterization can be changed to alter the simulation data obtained to reach the desired production-like standards. Subsequently, the baseline uninformed model will be trained and evaluated for its inference, generalizability, and transfer learning capabilities.

Subsequently, different ways of including physical prior knowledge will be implemented in the ML software using which PI models will be trained, and the performance in the same tasks will be evaluated again. This leads to the main research objectives for this master thesis project:

> "Evaluate the inference, generalizability and transfer learning performance of different methods to embed physical prior knowledge into neural network-based surrogate models and compare the results to the baseline model based on the HVAC CFD simulation library."

Finally, if possible, different methods will be combined and again evaluated and analyzed on how they impact the performance of the surrogate models.

4

Methodology

In this chapter, the methodology used to address the research questions will be explained. Firstly, the target use case will be presented. Subsequently, the data generation process will be explained. Then, data pre-processing steps will be outlined. Finally, the design of experiments will be presented. Using those experiments, the model performance will be assessed so that to answer the research questions posed in the previous chapter.

4.1. HVAC use case

The work presented here addresses the topic of ML-based surrogate modeling. In order to build a model, data are needed: either observational or simulation data of a selected geometry/application. For this work, a use case based on a parametrized automotive HVAC geometry was chosen. The geometry is presented in Figure 4.1.

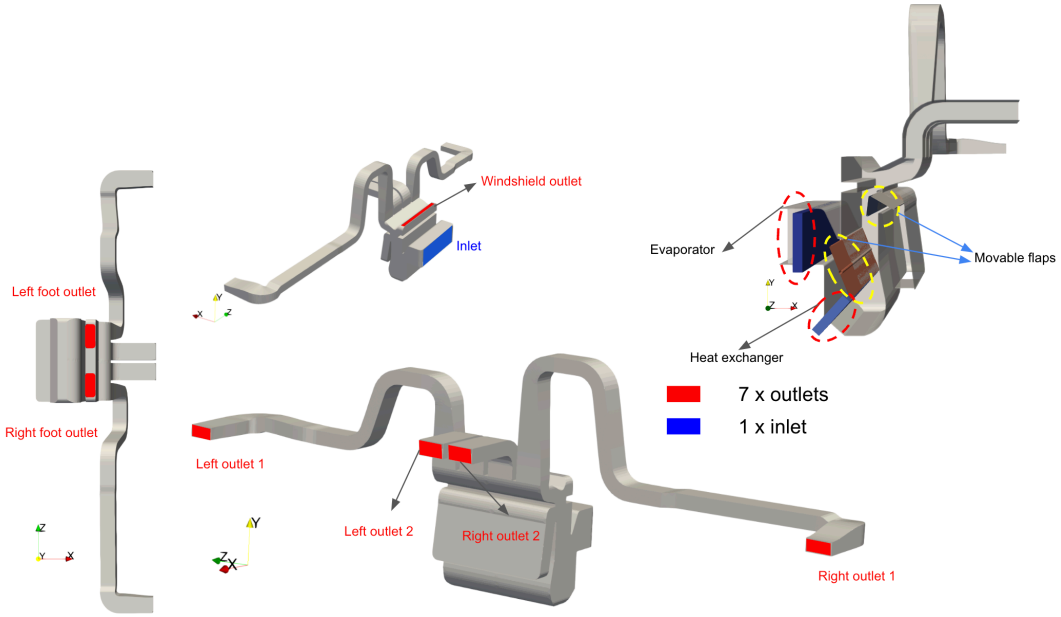


Figure 4.1: Base HVAC geometry.

The geometry consists of the central section and the two large side ducts. The central section houses the inlet, evaporator, heat exchanger, flaps used to direct air flow, and five outlets. The five outlets simulate the configuration of a passenger car, where the central ducts point at the passenger's face, while the ducts at the bottom point toward the feet of the passengers in the front row. The outlet at the top of the central section points towards the wind shield. The exchanger and evaporator are used as sink and source terms to add and remove heat from the system. There are five flaps in the HVAC that can close off certain outlets or just redirect the flow and alter the mass flow distribution between the outlets. The side ducts, coming out of the central section, direct the air toward the outside edges of the passenger cabin. The mock-up of the HVAC geometry, placed within the car outline, is presented in Figure 4.2.

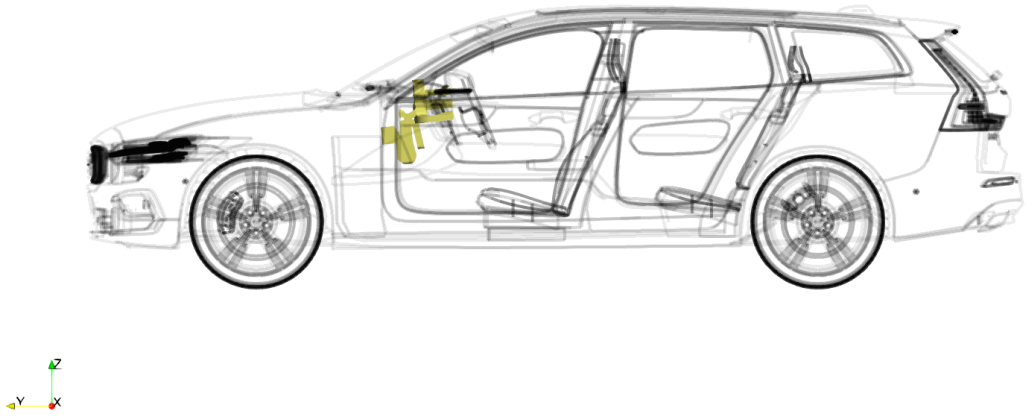
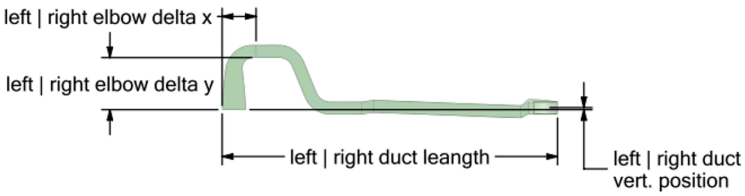


Figure 4.2: Base HVAC geometry in yellow placed in a reference car. Car geometry source:Rail [2021].

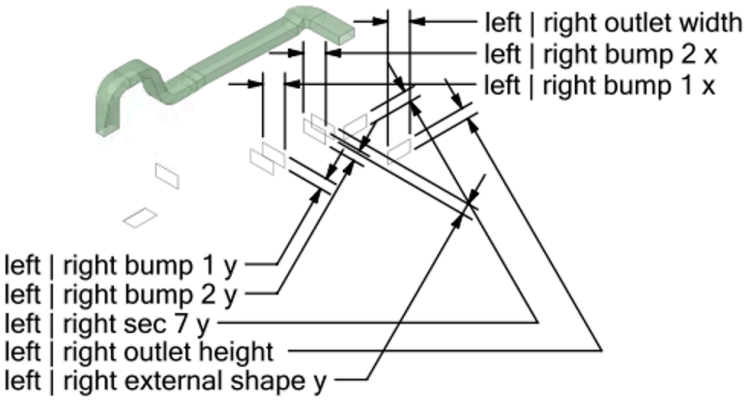
The base geometry was then translated to Ansys SpaceClaim and parameterized. The outside shape of the central section, called the HVAC box, is fixed, while only the flap angles are variable. The shape of the side ducts is parametrized with 15 parameters each. Finally, there is a scale parameter that allows for global size scaling. The geometry of the side ducts works as a series of lofts guided by a path linking 2D rectangular sections. The parameterization of the side ducts governs the 2D section locations, where the sections are presented in Figure 4.3. The duct geometry is formed by a loft over those sections. The full list of parameters, as well as parameter ranges, are presented in Table 4.1.

Table 4.1: HVAC geometrical parameters and their ranges.

Parameter name	Range low	Range high	Parameter name	Range low	Range high
Duct parameters					
Left elbow delta x [mm]	-40	20	**Right elbow delta x [mm]**	-20	40
Left elbow delta y [mm]	-5	200	**Right elbow delta y [mm]**	-5	200
Left duct length [mm]	500	800	**Right duct length [mm]**	500	800
Left duct vert. position [mm]	-20	80	**Right duct vert. position [mm]**	-20	80
Left bump 1 y [mm]	-10	10	**Right bump 1 y [mm]**	-10	10
Left bump 1 x [mm]	-10	10	**Right bump 1 x [mm]**	-10	10
Left bump 2 y [mm]	-10	10	**Right bump 2 y [mm]**	-10	10
Left bump 2 x [mm]	-10	10	**Right bump 2 x [mm]**	-10	10
Left external shape y [mm]	0	10	**Right external shape y [mm]**	0	10
Left sec. 7 y [mm]	-20	20	**Right sec. 7 y [mm]**	-20	20
Left outlet width [mm]	-20	20	**Right outlet width [mm]**	-20	20
Left outlet height [mm]	-20	20	**Right outlet height [mm]**	-20	20
Left global height [mm]	-10	10	**Right global height [mm]**	-10	10
Left global width [mm]	-10	10	**Right global width [mm]**	-10	10
Door angles					
Left foot door angle [°]	15	60	**Right foot door angle [°]**	15	60
Left windshield door angle [°]	1	155	**Right windshield door angle [°]**	1	155
Global parameters					
Door heater angle [°]	22	90	**Scaling factor**	0.9	1.1



(a) Frontal view.



(b) Isometric view.

Figure 4.3: HVAC duct parameterization.

4.2. Data generation

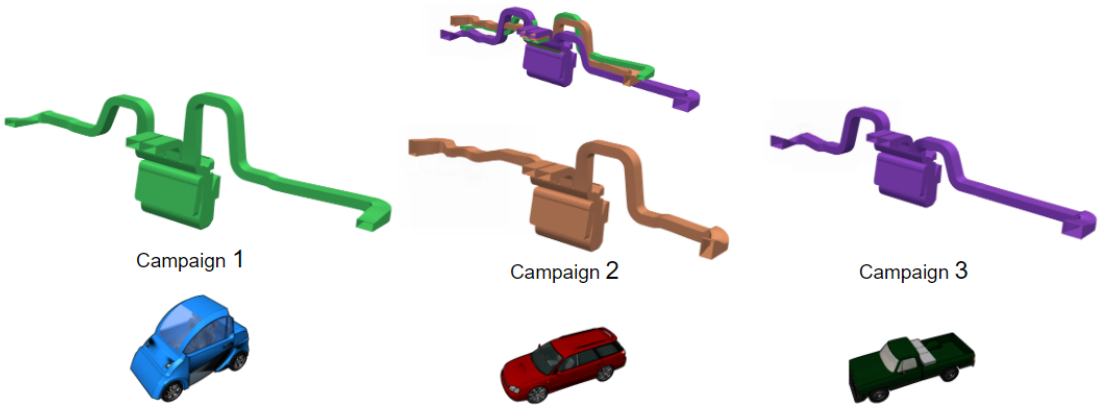


Figure 4.4: HVAC visualization of the multi-campaign data set.

The DOE for generating the training data was performed as follows. Four parameter windows were first defined, which can be seen to correspond to HVAC geometries for four different vehicles. This is represented by Figure 4.4, where you can see campaign 1 being as if it was designed for a smaller car like a city car, while campaign 3 was designed for a larger car, like a pick-up truck, and so on. To achieve this, the geometry parameterization was leveraged, allowing to reproduce such a data set. Appendix C presents parameter ranges and fixed parameters for each campaign. The design process for the geometries was largely empirical, based on the idea that during a car design process, the HVAC system is designed to fit some constraints, like the outlet location, overall width, and length of the car. The HVAC geometry can then be freely adapted to different car models. Each of these four design areas will be referred to as ”campaigns”. Within each campaign, a number of parametric variations were performed within the range of the defined parameter window.

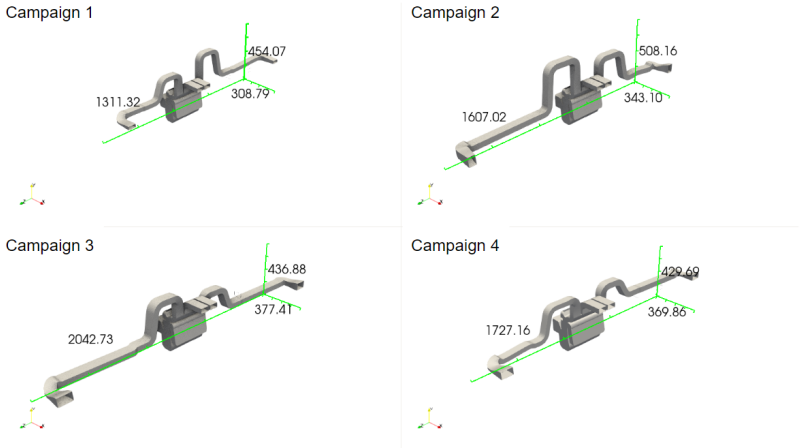


Figure 4.5: Sample geometries from all campaigns created, with added reference global dimensions in $[mm]$.

To ensure a non-sparse and uniform coverage of the design space, Latin hypercube sampling was used to randomly select the parameters for each sample within the data set. The flap angles for this project, as well as heat exchanger and evaporator properties, were fixed. This was done, as it was thought that adding such large non-geometrical sources of flow variability will drown out the signal coming from the flow changes due to the geometrical variability. It is important to note that flaps were set to angles that allow flow to pass through all the outlets. To add realism to the data and increase the

difficulty of the task for the ML model, each campaign was assigned three different inlet velocities, for which each geometry within the campaign will be simulated for. This means that effectively one HVAC geometry is the parent to three simulated samples, as in Figure 4.6. In total, 630 samples were created, divided into four campaigns as detailed in Table 4.2. The details of each campaign are presented in Appendix C.

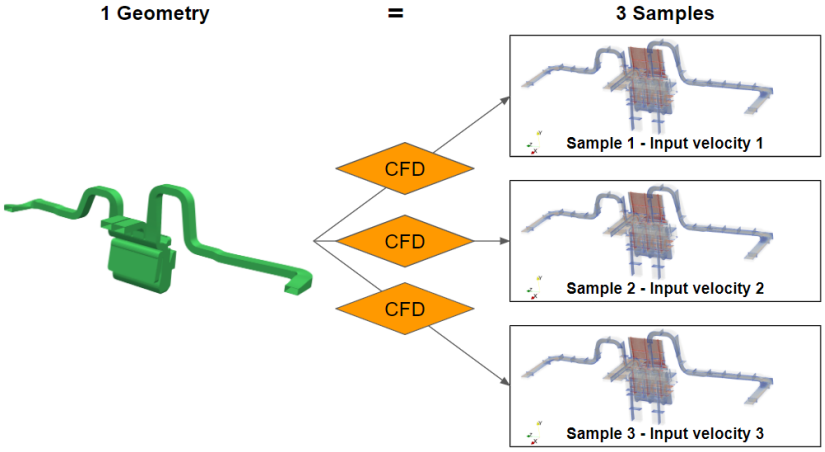


Figure 4.6: Each geometry is simulated thrice to form 3 samples each at different inlet velocity.

Table 4.2: Campaign overview.

	Campaign 1	**Campaign 2**	**Campaign 3**	**Campaign 4**
Number of samples	3x70	3x40	3x40	3x40
Inlet velocities $\frac{m}{s}$	1.00, 1.50, 2.00	1.50, 2.00, 2.50	2.00, 2.50, 3.00	1.08, 1.58, 2.08

Campaign 4 is set to be the target data set to assess the performance of the model in various data regimes. Campaign 1 is the largest campaign that will be both used as a data set for transfer learning, and to assess the single-campaign performance scaling with number of samples. Finally, campaigns 2 and 3 are auxiliary campaigns to aid transfer learning. As visible in Figure 4.5 the campaigns vary between each other in global dimensions, as well as in the shapes of the side duct outlets. Then within a campaign, minor details change, such as the local shape of the ducts, with the global dimensions remaining untouched to maintain the signature campaign resemblance.

The simulations were done entirely within the Ansys suite of tools. Firstly, each sample's geometry and simulation domain was modeled in Ansys SpaceClaim. Subsequently, meshing and simulations were performed in Ansys Fluent. The specific meshing and simulation setup was an inherited part of the project. The entire process was governed by Ansys Workbench, where the data set parameterization is stored and each sample parameters are sent to the meshing and simulation software, one by one, for simulation.

The created mesh was an unstructured polyhedral mesh with added prism layers. The overall mesh size was around 2.5 million cells, with an average y^+ of less than 1. The simulations use RANS, as well as the k-ω SST turbulence model. Due to scarce resources and time frame limitations runtime was crucial. Because of this, hybrid initialization was used (potential flow) to speed-up the simulations. The heat exchanger as well as the evaporator were modeled as porous media. Furthermore, relatively relaxed convergence stopping criteria ($1e-3$ for continuity, x,y,z velocity, k and omega, and $1e-6$ for energy) were used, together with the maximum iteration step criterion, as in Figure 4.7.

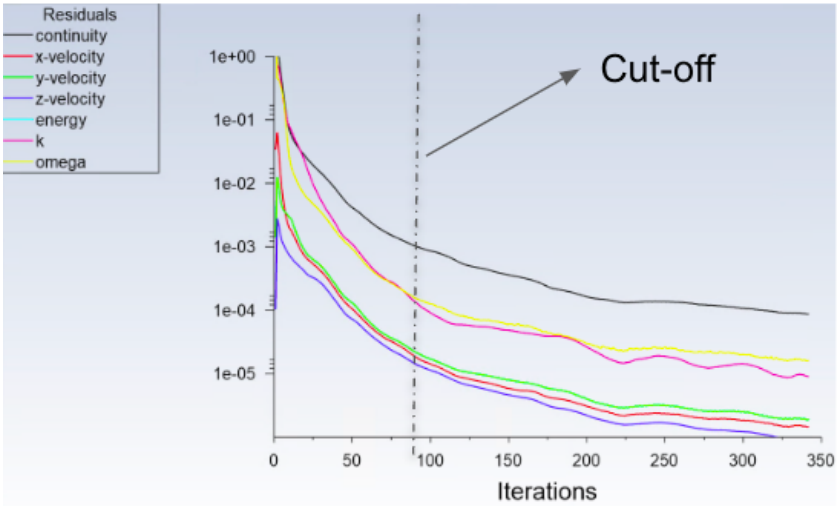


Figure 4.7: Example unstopped simulation run, with iteration stopping criterion plotted.

The convergence study is presented in Appendix A.

For the purpose of this work, the following values of interest were extracted from the CFD simulations. Firstly, pressure and x, y, and z velocity fields were extracted from the simulations. Furthermore, seven mass flows were saved for each outlet, along with an area-averaged pressure drop. The list of saved result variables is much longer than this, but only the mentioned quantities of interest were used for the training of the surrogate models in this work. The spatial location of the mass flows, along with reference names of the geometrical sections used in this work, are presented in Figure 4.8. Each mass flow corresponds to the outlet section of the HVAC geometry.

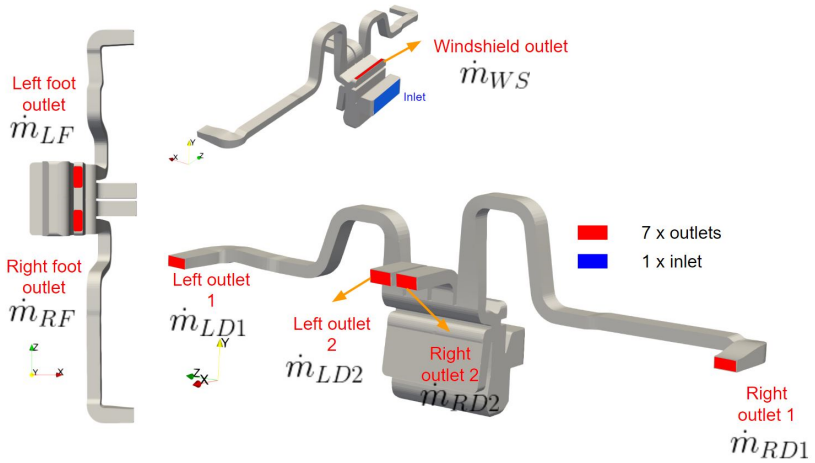


Figure 4.8: Mass flow locations.

4.3. Data pre-processing

To be able to train the model, a significant amount of data pre-processing had to be done.

The initial step of data pre-processing was data assessment looking for any corrupted samples. This was done by calculating various statistics over each simulated sample and analyzing, campaign by campaign, the samples. Provided that a sample was simulated correctly, the values of interest were not large outliers and that the sample had a correctly generated mesh and geometry, it was assumed that a given sample was a good sample. Minor outliers or samples slightly out of distribution were not

removed, as it was assumed that the designed data set was close to what a real-life customer would like to model and know the results for, hence the design space was not curtailed artificially by putting too strict boundaries.

Before building a model, one first has to decide if a training should utilize only surface-based field values, or the entire 3D flow field. For the purpose of this research project, the entire 3D flow field was used, as it allows for more detailed analysis of the model performance, due to the ability to recalculate the scalar values, based on the field predictions. Besides, one of the planned approaches to incorporate physics knowledge, physics-informed loss, is based on 3D derivatives of the flow field, which can only be obtained with volumetric predictions.

The results are read in Fluent format. To better capture the variability in the side ducts, which are the main source of variability in the data, the simulation mesh is downsampled to a target density that corresponds to that of the side ducts. This allows the model to put enough emphasis on the ducts compared to, e.g. the HVAC box, which has a higher point density in the simulation mesh due to the higher number of intricate geometrical features. In fact, it was observed that omitting this downsampling step led to low-quality predictions in the side ducts due to the model being biased to the main HVAC box.

An additional remeshing step was performed in order to ensure a uniform density and that normals were correctly assigned. Figure 4.9 (top) presents the mesh after exporting it from Ansys SpaceClaim as an *.stl*. It was found that during the geometry extraction process both the normals for the same faces were flipped and the meshing had to be redone due to it being highly irregular. Figure 4.9 (bottom) presents the HVAC geometry after remeshing, revealing a uniform mesh.

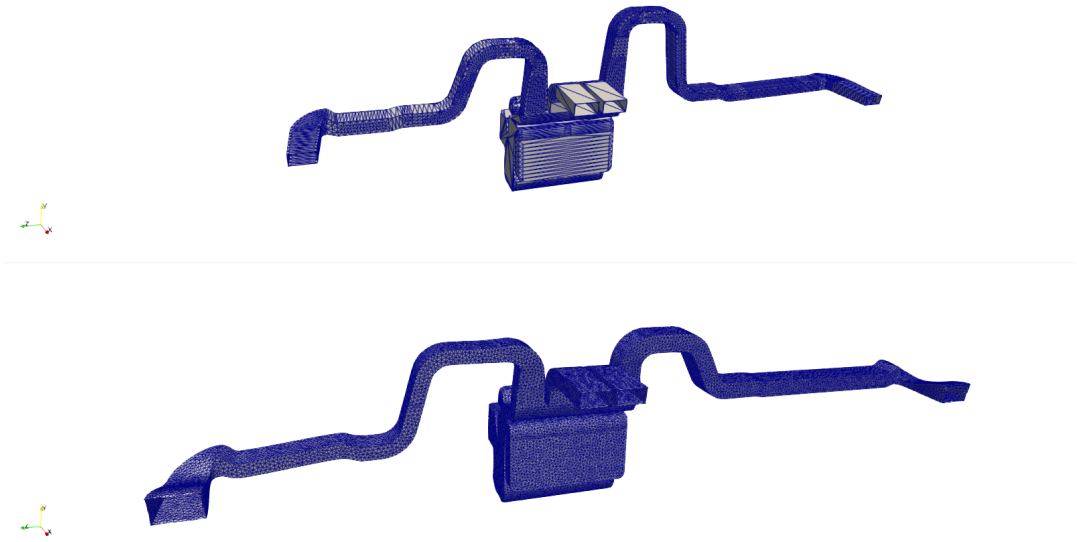


Figure 4.9: Effect of remeshing. Bad mesh at the top vs. good mesh at the bottom.

4.4. Design of Experiment

In order to answer the research questions presented, a rigorous and repeatable series of tests has to be conducted to assess the performance of the models in various data settings. For this purpose, three experiments were created (see Table 4.3), each of them was set in different data regimes.

1. Experiment 1 tested the performance of the model on campaign 4 in isolation. This was to obtain the benchmark value of the best performance of a trained model in a data-rich environment, as

well as to assess the inference performance in this data-rich setting.

2. Experiment 3 was a transfer learning experiment that assessed the TL and generalizability performance of the models on transfer from a single campaign (1) to campaign 4.
3. Experiment 5[1] was a transfer learning experiment that assessed generalizability and TL performance when transferring from campaigns 1,2 and 3 to campaign 4.

Experiments 3 and 5 were both trained on the same task as Experiment 1: predicting the validation set consisting of the same data set of three inlet velocities times ten geometries from campaign 4 (thirty samples in total). The hope was that using additional campaigns in the training set would allow models in Experiments 3 and 5 to reach comparable performance to models in Experiment 1 with fewer samples from campaign 4 needed in the training set.

Each experiment was run several times, varying the size of the training data set. For Experiment 1 the runs were performed with increasing sizes of the training data set. In experiments 3 and 5, in all runs, the transfer learning data sets were fully used and only the size of the target data set in the training set changed. This was to measure the impact of additional target data set samples in the training set on the model performance.

Table 4.3: DOE for model performance assessment, *camp* is a shortening for word campaign.

Experiment name	Experiment run	Training set	Validation set
Experiment 1 (*exp1*)	3x5	3x5 camp4	3x10 camp4
	3x10	3x10 camp4	3x10 camp4
	3x20	3x20 camp4	3x10 camp4
	3x30	3x30 camp4	3x10 camp4
Experiment 3 (*exp3*)	3x0	3x60 camp1	3x10 camp4
	3x1	3x60 camp1 + 3x1 camp4	3x10 camp4
	3x5	3x60 camp1 + 3x5 camp4	3x10 camp4
	3x10	3x60 camp1 + 3x10 camp4	3x10 camp4
Experiment 5 (*exp5*)	3x0	3x60 camp1 + 3x30 camp2 + 3x30 camp3	3x10 camp4
	3x1	3x60 camp1 + 3x30 camp2 + 3x30 camp3 + 3x1 camp4	3x10 camp4
	3x2	3x60 camp1 + 3x30 camp2 + 3x30 camp3 + 3x2 camp4	3x10 camp4
	3x5	3x60 camp1 + 3x30 camp2 + 3x30 camp3 + 3x5 camp4	3x10 camp4
	3x10	3x60 camp1 + 3x30 camp2 + 3x30 camp3 + 3x10 camp4	3x10 camp4

[1] Experiments 4 was originally planned to test the generalization performance with zero-shot predictions but as it never yielded any meaningful results it was removed from the scope of this thesis, but the legacy numbering of the experiments was maintained.

5

Baseline model

This chapter will present the baseline model which was be used throughout the rest of this work.

5.1. General settings

For the purpose of this work, the GNN model was used. The basic architecture, which is recalled in Figure 5.1, predicts the field surface values on the input geometry, as well as scalar quantities. For such an architecture, the model inputs are the geometry surface mesh and any associated input scalars. The input scalar value can be any significant scalar that characterizes the sample, like inflow free stream velocity, particular geometry parametrization variable, or other quantities that characterise the flow. This is passed to the Geometry processing block that utilizes the geodesic convolutions [Monti et al., 2017] to process the geometry. One of the outputs of the Geometry processing block is a low-dimensional global embedding. The global embeddings are fed to a ScalarNet whose role is to predict output scalars, while the outputs of the geometry processing block are fed to a FieldNet whose role is to predict fields. The training process is governed by a loss function, typically L1 or L2 loss functions.

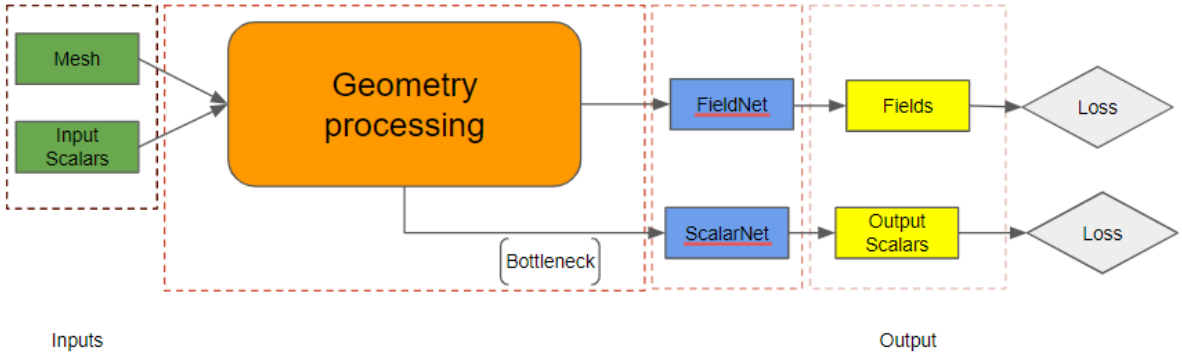


Figure 5.1: Stock GNN model.

Figure 5.1 shows the high-level architecture of the baseline model that will be used throughout this work. The inputs and outputs of the model are presented in Figure 5.2. The inputs to the model are the surface mesh of the HVAC geometry and the optional input scalars. The model outputs are the seven mass flows and pressure drop scalar predictions, velocity and pressure volumetric field predictions, and optionally the spatial gradients of the predicted fields.

All subsequent models used will utilize this baseline architecture, unless otherwise specified. Separate loss functions are used for fields and each of the scalars. Then the loss function used for training is as follows:

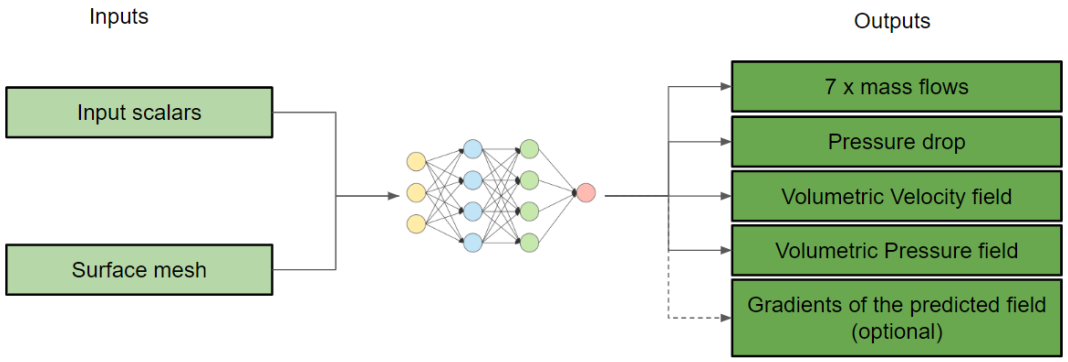


Figure 5.2: Final baseline model inputs and outputs.

$$L_{Total} = L_{fields} \times w_{L_{fields}} + \Sigma_i^n (L_{scalar\,branch} \times w_{scalar\,branch}) \tag{5.1}$$

A hyperparameter search led us to using $W_{L_{fields}} = 1$ and $W_{L_{scalar_branch}} = 0.1$, which we will use throughout this work.

6

Physics-informed modifications for baseline model improvement

In this chapter, details of the methods for incorporating prior physical knowledge will be introduced. In the following chapters, the results of the experiments with those different methods will be presented.

Three main methods were shortlisted as the most promising approaches. Firstly, using a simplified solution as an additional input field to the model. Secondly, using physics-based scaling of the training and validation data. Lastly, using losses based on the governing equations.

6.1. Simplified solver solution as an additional input

The explanation of the method can be seen in Appendix B

6.2. Physics-based scaling

The second method selected for this experiment was physics-based scaling of the data. This was proven to work well in previous work in simpler models, as discussed in Section 2.4.3.

For the purpose of this study, two methods were tested. Firstly, velocity **V** and mass flows $\dot{m}$ were scaled by the inlet velocity V_{inlet}, while pressure p and pressure drop scalar p_{drop} were scaled by the inlet velocity squared. This is a standard scaling used in many fluid mechanics applications, and this is the scaling that was used in both Thuerey et al. [2020] and Kissas et al. [2020]. It is important to note that this is not non-dimensionalization, as pressure is not scaled by the density of the fluid.

$$\mathbf{V}^\star = \frac{\mathbf{V}}{V_{inlet}} \quad \dot{m}^\star = \frac{\dot{m}}{V_{inlet}} \quad p^\star = \frac{p}{V_{inlet}^2} \quad p^\star_{drop} = \frac{p_{drop}}{V_{inlet}^2} \tag{6.1}$$

The other method used was linear scaling of all quantities with inlet velocity:

$$\mathbf{V}^\star = \frac{\mathbf{V}}{V_{inlet}} \quad \dot{m}^\star = \frac{\dot{m}}{V_{inlet}} \quad p^\star = \frac{p}{V_{inlet}} \quad p^\star_{drop} = \frac{p_{drop}}{V_{inlet}} \tag{6.2}$$

This scaling is usually not applied, unless a creeping flow ($Re << 1$) is assumed, in a scenario where viscous effects are dominant. In this case, the Reynolds number, based on the outlet, is considerably higher, roughly in the range of $\approx [1000, 15000]$[1]. But nevertheless, in this internal flow case the

[1] Calculation based on the conditions in the outlet of the side duct

viscous effects are considerable, which was the reason behind trying this approach.

After using this physics-based scaling, the standard centering and scaling to unit variance is used.

The scaling was implemented as a data set transform performed before the data was given to the model for training. Hence, the raw predictions of the model were also scaled. To obtain the predictions in the non-scaled state, the data transformation was applied in reverse to the predictions, and only afterwards the results were analysed.

6.3. Physics-informed losses

This approach is based on work done by Raissi et al. [2017], and all subsequent publications in the field. It is based on the incorporation of a loss based on governing equations into the model. Unusually, in the data-free regime, PINNs are used to solve single case unsteady flow using the full N-S equations combined with losses on the boundary conditions of the domain. In this case, a similar approach is used on steady flow in a surrogate model setting with the training data obtained using the RANS equations with the $k-\omega\ SST$ model.

RANS momentum equation for incompressible flow in vector form is the following:

$$\frac{\partial \overline{\mathbf{V}}}{\partial t} + (\overline{\mathbf{V}} \cdot \nabla)\overline{\mathbf{V}} = -\frac{1}{\rho}\nabla p + \frac{1}{\rho}\nabla \cdot \tau + \mathbf{f} \tag{6.3}$$

RANS continuity equation for incompressible flow in vector form is the following:

$$\nabla \cdot \overline{\mathbf{V}} = 0 \tag{6.4}$$

Where $\overline{\mathbf{V}}$ is the Reynolds-averaged velocity vector, ρ is the fluid density, p is the pressure, τ is the Reynolds stress tensor, **f** is the body force vector,t is time.

For the physics-informed loss to help with the training, it should be aimed at solving the same system of equations as the ones modelled using CFD. This raises the problem that if the full RANS system of equations is to be used with the $k-\omega\ SST$ turbulence model, implementing the loss based on the equations becomes highly complex due to both the high computational demands for calculating higher-order gradients for the momentum equation to obtain the loss, as well as implementing the calculation for the additional terms coming from the specific turbulence model used.

For these reasons, a decision was made to include only the continuity equation in the physics-informed loss. The usage of lower-order derivatives combined with equations only based on the time-averaged flow velocities simplifies the loss implementation and keeps the computational expenses manageable.

Moreover, the loss components based on the boundary conditions were not used, due to increased complexity of implementation for complex geometries such as the ones used for this study. Instead, the data loss component can be thought of as fixing the boundary conditions in an indirect way during the learning process.

This results in the following new loss function:

$$L_{Total} = L_{fields} w_{L_{fields}} + \Sigma_i^n (L_{scalar\,branch} w_{scalar\,branch}) + w_{L_{\mathrm{PDE}}} L_{\mathrm{PDE}} \tag{6.5}$$

Where:

$$
\begin{aligned}
L_{\text{fields}} &= \frac{1}{N}\sum_{i=1}^{N}\left(f_{gt,i} - f_{pred,i}\right)^2 \\
L_{\text{PDE}} &= \frac{1}{N}\sum_{j=1}^{N}\left(\frac{\partial \overline{u}_{pred,j}}{\partial x} + \frac{\partial \overline{v}_{pred,j}}{\partial y} + \frac{\partial \overline{w}_{pred,j}}{\partial z}\right)^2
\end{aligned}
\tag{6.6}
$$

The partial derivatives for the L_{PDE} were calculated on the prediction point cloud using automatic differentiation. A very important aspect of the loss computation is the weighing of each component, as many authors [Lu et al., 2020, Nabian et al., 2021, Abbasi and Andersen, 2022, Wang et al., 2020] uncovered the convergence sensitivity of physics-informed losses to the weights of the loss component. For the purpose of this work, several different weights were tried, presented in Table 6.1.

Table 6.1: Loss weights tried with this method.

N.	$w_{L_{fields}}$	$w_{L_{scalars}}$	$w_{L_{PDE}}$
1	0	0	1E-5
2	0	0	1E-8
3	1	0	0.1
4	1	0	0.01
5	1	0	[1E-8,1E-7,1E-6, 1E-5]

In Table 6.1 the different weights used for the loss terms are presented. This was to assess the impact of the L_{PDE} as well as control the impact of the data in the training. In cases 1 and 2 the model was trained in a data-free regime that relied solely on PDE loss to govern the optimization of the model parameters. In subsequent cases 3,4 and 5, both model and data losses are used to optimize the model parameters.

For the purpose of this work, physics-informed losses were used as a correction method (see Figure 6.1), as suggested in some of the works by Karniadakis [2020], Zhu et al. [2022]. By correction, it is meant that first a baseline model is trained, in this case the setup as for Experiment 1, so a single-campaign (4) training set with 3×30 samples. The training is run until completion, and the predictions on the validation set are made. Subsequently, the model architecture and weights are transferred to another model, which, instead of using only data losses, uses the new loss, which includes L_{PDE}. Analysis of predictions of the baseline model is performed and outlier or under-performing sample is found, and that sample is then used as a singular training sample for the new model, which iterates on this one sample for up to 2000 iterations. Intermediate checkpoints of the model are saved to be able to analyze the progress of the samples throughout the training process. Subsequently, the predictions of the single samples are made and analysed compared to the prediction of the baseline model and the ground-truth CFD data. During training, baseline model predictions for the corrected sample are used as supervision data.

The correction approach to using physics-informed losses is another way of trying to minimize the convergence sensitivity of this method. If an optimized model is used as a base model, the chances are that it is already close to a minimum in the loss landscape, and applying this correction loss, will only allow to identify the minimum better. This is done under the assumption that the selected sample is a valid sample that represents the data set well and that the PDE baseline model is well trained for the data set.

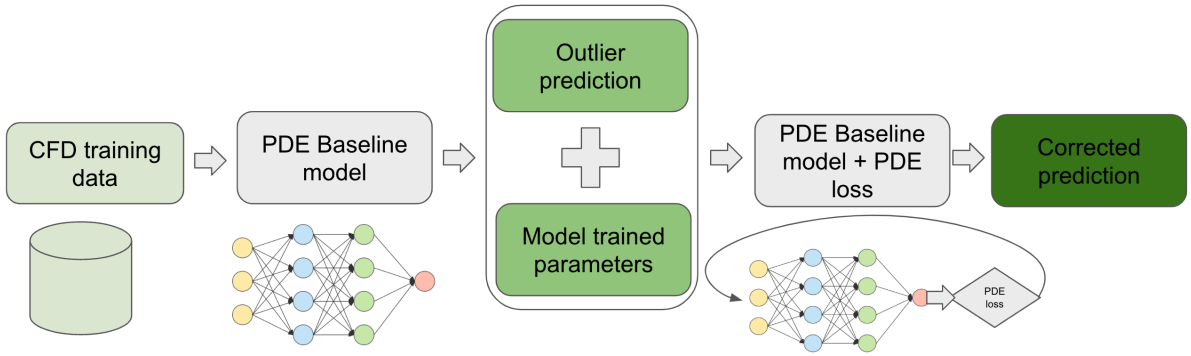


Figure 6.1: Physics-informed loss prediction correction workflow.

Several aspects of this PDE baseline model differ from the baseline model described in Chapter 5. The activation functions are switched from ReLu to softmax to allow for differentiability of the outputs with respect to the coordinates. Additionally, no input field is used in this model, as this is an additional input with respect to which gradients for the L_{PDE} cannot be computed. Finally, gradient tracking is enabled so that partial derivatives of outputs with respect to the point cloud coordinates can be computed. This makes the results of this experiment unsuitable for direct comparison with the other experiments.

The next chapter (Chapter 7) will present the results of the baseline model, followed by results and analysis of all the methods presented in this chapter.

7

Results

In this section, the results obtained using models with the different methods used to incorporate physical prior knowledge will be presented, starting with a detailed analysis of the baseline model and followed by physics-informed method analysis. It is important to note that due to the large number of models trained during this work[1], detailed analysis of the prediction accuracy of each model is not possible, therefore only global metrics for the models will be shown to present the general performance, and only when details are necessary for complete understanding, will the details be presented.

To remind the reader, the baseline model was first run and then the models with different physics-informed methods were run for all the experiments presented in Table 4.3. Due to a large number of models trained, keeping track of them may be hard. To solve this the following model naming convention will be used from now on, <*method*>_exp<*n*> and if a particular run configuration will be mentioned, the configuration will be added at the end of the code name to form the following name: <*method*>_exp<*n*>_<*number of target data sets samples in the training set*>. Four methods are presented in this chapter (for a detailed explanation, refer to Chapter 6):

1. Adding a simplified solution (potential flow in the case of this work) as an additional input to the model (method type designation *pot*).
2. Linear scaling of the data before training (method type designation *nd_lin*).
3. Mixed scaling of the data before training (method type designation *nd_mix*).
4. Adding a simplified solution as an additional input to the model, combined with linear scaling of both the input fields and the training data (method type designation *nd_pot*).

Finally, the baseline models have the method type designation of *bs*. To give an example a baseline model run of Experiment 5 with 3×5 samples from campaign 4 in the training set will be from now on referred to as: *bs_exp5_3x5*

7.1. Model performance assessment

In order to systematically assess the model performance, a consistent set of metrics gathered from each model run must be used to allow for a like-for-like comparison between the results. The presentation of those metrics and the post-processing needed to achieve them will be the topic of this section.

[1] For the final version of this report, 85 models were trained and analyzed, each model having 6 scalar outputs to analyze and 4 fields.

7.1.1. Statistical metrics used

The main metrics used to assess accuracy in this study are the following: L1 error, the coefficient of determination also known as R^2, and the correlation coefficient R.

L1 error is the most straightforward metric. It is also known as absolute error, and is defined as follows:

$$L1 = |y_{gt} - y_{pred}| \tag{7.1}$$

Where y_{gt} is the ground truth value and y_{pred} is the predicted value. Often, this error is used to measure both field and scalar prediction accuracy. For fields, firstly, the L1 error would be calculated in a pointwise fashion, between the CFD simulation data, which act as ground truth data and the prediction point cloud. Subsequently, the L1 error can be averaged per sample and then per validation data set to yield a measure of field prediction accuracy for a given run of an experiment.

To assess a prediction of the scalars, one of the metrics used is called the correlation coefficient, or R in short. In this work, a *numpy* implementation of the Pearson product-moment correlation coefficient was used. It is calculated as follows:

$$R = \frac{\sum \left(y^i_{gt} - \bar{y_{gt}}\right)\left(y^i_{pred} - \bar{y_{pred}}\right)}{\sqrt{\sum \left(y^i_{gt} - \bar{y_{gt}}\right)^2 \sum \left(y^i_{pred} - \bar{y_{pred}}\right)^2}} \tag{7.2}$$

This is a measure of the linear correlation between two variables. It ranges from -1 to 1, where -1 indicates a perfect negative correlation, 0 indicates no correlation, and 1 indicates a perfect positive correlation. This metric will be used to assess the correlation of the scalar prediction with the ground-truth values. This metric does not take into account the absolute value of the predicted value, but rather captures if the trend represented by the predictions matches the trend in the ground truth data. This is an important metric as, although it does not assess the absolute precision of the predictions, it does allow one to assess whether the tool predicts the right trends in the data. Often, engineers working in industry are not interested in the absolute precision of the tool, but rather its ability to capture relative differences between the designs. High correlation coefficients mean that the model captures those trends and, although absolute precision may be low, capturing the trends in the data is often enough to determine relative differences between different designs and to decide on their desirability or not.

The coefficient of determination, or R^2, is a statistical metric that represents the proportion of variance in the dependent variable that is predictable from the independent variable(s) in a regression model. Under the assumption that the training and validation data have the same variance, R^2 should be in the range between 0 and 1, with higher values indicating a stronger relationship between the variables.

$$R^2 = 1 - \frac{\sum_i (y^i_{gt} - y^i_{pred})^2}{\sum_i (y^i_{gt} - \bar{y}_{gt})^2} \tag{7.3}$$

It is calculated as the ratio of the explained variance to the total variance, where the explained variance is the variance of the predicted values and the total variance is the variance of the actual values. This metric will be used to assess both the scalar and field predictions. In addition to this, additional physics-based metrics were used to evaluate the quality of field predictions, as described next.

7.1.2. Scalars computed on predicted fields

In addition to purely statistical metrics such as the R^2 score, additional metrics were used in order to assess the quality of field predictions from a physical perspective. Note that visual inspections of each predicted field in each experiment run in this work is not possible due to the sheer number of runs performed and predictions made, hence the need for a quantitative metric. For this purpose, CFD-like scalar values are recalculated using the field predictions. This allows for like-for-like comparisons of the quantities of interest calculated from the prediction fields to the ones obtained from the CFD simulations.

It is important to note that in the architecture used in this work, scalars and fields are predicted using two distinct branches. Therefore, the scalar predictions made by the model and the scalar calculations made based on the predicted fields may not yield exactly the same values.

In ANSYS Fluent, the mass flow is calculated as follows [ANSYS Inc, 2013]:

$$\int \rho \vec{v} \cdot d\vec{A} = \sum_{i=1}^{n} \rho_i \vec{v}_i \cdot \vec{A}_i \tag{7.4}$$

This is a sum of the mass flows through each facet that belongs to the selected outlet plane.

The pressure drop is calculated as the area-averaged pressure of the HVAC inlet. For the CFD simulations, the null pressure boundary condition is used at all HVAC outlets, so there is no need to subtract the pressure at the outlets from the inlet pressure.

$$P_{drop} = \frac{\sum_{i=1}^{n} P_i A_i}{\sum_{i=1}^{n} A_i} \tag{7.5}$$

To calculate those scalar values, access to the facet area is necessary. Due to the model input being the point cloud, this method relies on the existence of the mesh associated with the point cloud, which is a considerable limitation.

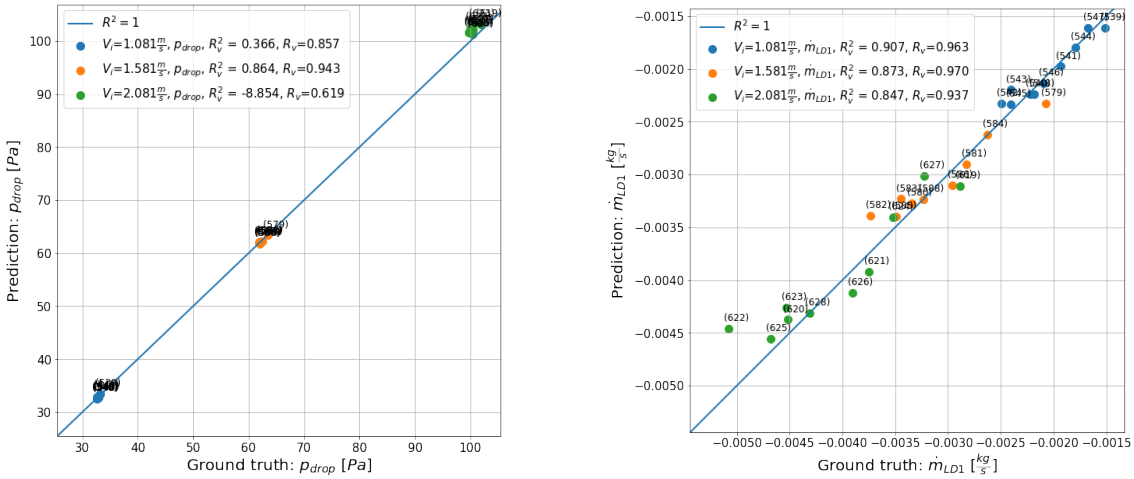
Having the scalars computed from fields allows for using the same statistical metrics to be used on them as used for scalars outputted by the model directly. This allows for a more reliable and quantifiable insight into the model field predictions. Another way to utilize this calculation is to build a loss function based on it. This was not done in this work, but could be worth exploring in a future work.

7.1.3. Metric calculation methodology

At this point, important clarification is needed on the topic of performance metrics. The typical scatter plot for a single model run for a given training set configuration for the baseline model is presented in Figure 7.1b. To measure the accuracy of the prediction, various statistics are calculated.

- If a value for a singular scalar or field error is reported, it is, in fact, an average over three velocity clusters, which you can see highlighted in green, orange, and blue in Figure 7.1b. 3 velocity clusters exist, as each geometry is predicted for three different inlet velocities. This averaging over clusters is done instead of global R^2 for the entire validation set, as this gives a more accurate understanding of how the model captures the trends within each cluster. The global R^2 calculated on the validation set without taking into account the velocity clusters would be artificially inflated by the variation of the minimum to maximum values in the validation set. In a design scenario, an engineer wants to know the accurate performance of a given design at each velocity set point. Hence, such a methodology for the computation of the statistics is used. For this purpose, first, for each scalar predicted by the model, the metrics are computed separately for each inlet velocity cluster. Those single-scalar metrics calculated per cluster are marked by the subscript **v**, for example, R_v or R_v^2. Subsequently, to agglomerate the metric for a single scalar over the three inlet velocity clusters, the per-cluster metrics are averaged over three clusters. Those metrics are marked by subscript **c**, for example R_c.
- When the total average statistic is quoted for a training, it is the statistic computed per cluster for each variable that was later averaged over a set of variables. This is especially important for the R and R^2 computations. In the case of correlation for scalars, the average was calculated over five mass flows and the pressure drop. For R^2 the average was computed over five mass flows only. This is done consistently throughout the work. The omission of the pressure drop in the R^2 calculation is due to the fact that the model was never able to accurately capture the magnitude of the pressure drop, while it was able to capture trends (hence, inclusion in the calculation R). This probably happened due to dominance of inter-cluster variability over the very low intra-cluster

variability, as can be seen in Figure 7.1a. This relative low inter-cluster variability leads the model to focus its attention on other scalars that are more uniformly distributed and have larger errors during training. Limited attention of the model is spent on learning the intra-cluster variability of the pressure drop scalars, leading to a model that is not able to accurately infer the magnitude of the scalar due to the variability within clusters. To signify global run metrics like R and R^2, the **bar** superscript will be used, so, for example, $\overline{R}$ and $\overline{R^2}$.



(a) Example of raw results for p_{drop} for *bs_exp1_3x30*.

(b) Example of raw results for $\dot{m}_{LD1}$ for *bs_exp1_3x30*.

Figure 7.1: Examples of predicted scalar scatter plots.

Figure 7.1b presents the scatter plot of ground truth CFD values vs. the predictions from ScalarNet, for the mass flow of the left duct 1. For this particular run of the baseline model, with a training set of thirty geometries simulated at three different inlet velocities, the values are well captured by the model with high R_v and R_v^2 as can be seen in the legend. It can also be seen that R_v^2 decreases slightly with increasing inlet velocity. This is a trend that is repeated among other experiments and makes physical sense, as a higher inlet velocity would mean a higher Re as well as a higher chance of turbulent flow in the ducts, which can be much more complex to capture due to its high local variability in the flow patterns.

7.2. Baseline model results

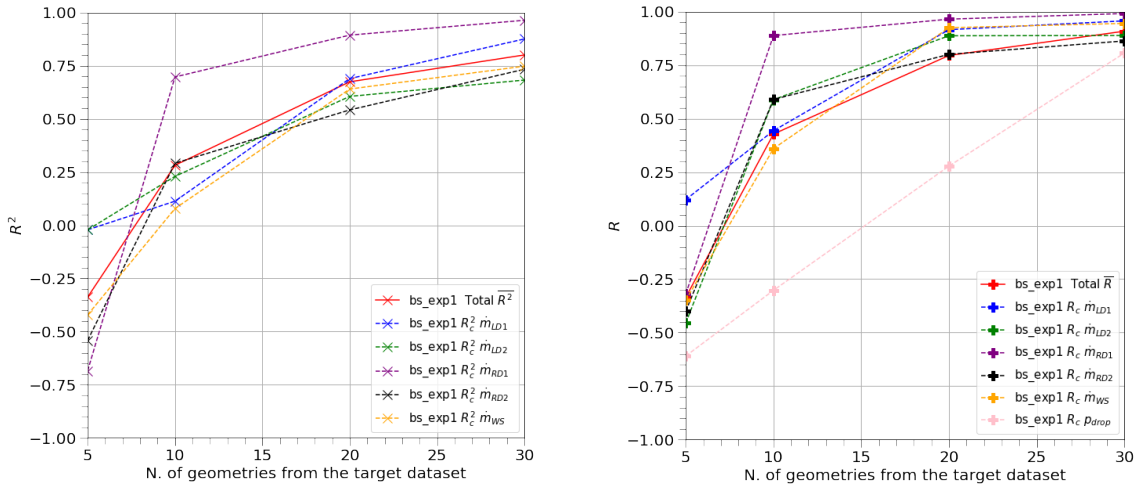
The results of this model will establish the benchmark to which all subsequent methods will be compared. Starting with Experiment 1 (*bs_exp1*) where inference performance is evaluated with a single campaign (4) and moving to transfer learning and generalization assessment looking at Experiments 3 (*bs_exp3*) and 5 (*bs_exp5*), where, respectively, campaign 1 and campaigns 1,2 and 3 are used as base campaigns, and a small amount of data is added to the training set from campaign 4 to assess the transferability of the knowledge between campaigns. All models are then evaluated based on the prediction of a fixed validation set that is part of campaign 4.

This section tackles the first sub-question from the Chapter 3:

> "Based on the selected test case data set, what is the performance (at inference time, transfer learning, and generalizability) of an optimized model that does NOT leverage physical prior knowledge?"

7.2.1. Baseline model - inference performance

An analysis of *bs_exp1* will be carried out, looking at how the performance scales with number of samples in the training set, as well as how it reaches a performance plateau.



(a) R^2. **(b)** R.

Figure 7.2: *bs_exp1*: R^2 and R of the predicted scalars from ScalarNet of the test set for various runs of *bs_exp1*.

In Figure 7.2 the evolution of the R_c^2 and R_c for *bs_exp1* with the size of the training data set is plotted. A clear increasing and converging trend is visible on almost all scalars, showing that with increasing training data set size, prediction accuracy improves, which was to be expected. The main outlier is the correlation metric on the p_{drop} (see Figure 7.2b), which improves almost linearly with an increase in the training data set size. Looking at the Figure 7.2 and the absolute errors presented in Table 7.1 and Table 7.2, it is visible that by the training set size of 20 geometries (3×20 samples) the performance on the validation data set shows a high correlation of more than 0.75 and a positive R_c^2 of above 0.5 for all scalars considered. At this data set size, the errors for all scalars halve, relative to the model performance with 5 geometries (3×5 samples) in the training set. Many of the scalars have very low L1 errors of about 1%. The largest errors for the model with the largest data set (3×30 samples) were obtained for the $\dot{m}_{LD1}$ and $\dot{m}_{RD1}$. This was to be expected, as those mass flows show the highest variance within the inlet velocity clusters, and large values of performance metrics like R_c^2 and R_c measure the accuracy relative to each scalar's variance. This means that for these particular scalars, the relative accuracy is high, but precision in a global sense is lower when compared to other predicted scalars.

Table 7.1: *bs_exp1*: Absolute and percentage mean L1 error of predictions vs. CFD ground truth for scalar quantities.

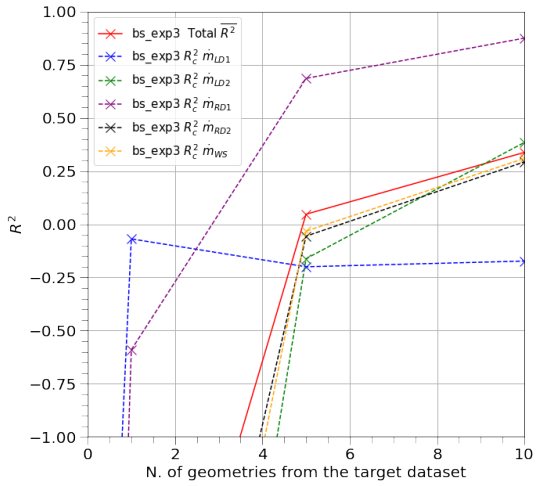
N. of geometries in training set	$\dot{m}_{LD1}$		$\dot{m}_{LD2}$		$\dot{m}_{RD1}$		$\dot{m}_{RD2}$		$\dot{m}_{WS}$		p_{drop}	
	$\overline{L1}$ $[kg/s]$	$\overline{L1}$ [%]	$\overline{L1}$ $[kg/s]$	$\overline{L1}$ $[kg/s]$	$\overline{L1}$ $[kg/s]$	$\overline{L1}$ [%]	$\overline{L1}$ $[kg/s]$	$\overline{L1}$ [%]	$\overline{L1}$ $[kg/s]$	$\overline{L1}$ [%]	$\overline{L1}$ $[Pa]$	$\overline{L1}$ [%]
5	0.0004105	14.44%	0.0001173	2.49%	0.0005546	18.07%	0.0001234	2.63%	0.0004632	2.23%	0.9926281	1.38%
10	0.0003436	12.70%	0.0000793	1.67%	0.0002847	8.73%	0.0000800	1.69%	0.0003411	1.60%	0.7956736	1.09%
20	0.0002142	7.59%	0.0000620	1.24%	0.0001068	3.98%	0.0000665	1.38%	0.0002400	1.09%	0.6608915	0.85%
30	0.0001397	4.41%	0.0000576	1.17%	0.0000862	2.81%	0.0000529	1.07%	0.0002146	0.98%	0.6683922	0.78%

Table 7.2: *bs_exp1*: Absolute L1 error of predictions vs. CFD ground truth for field quantities.

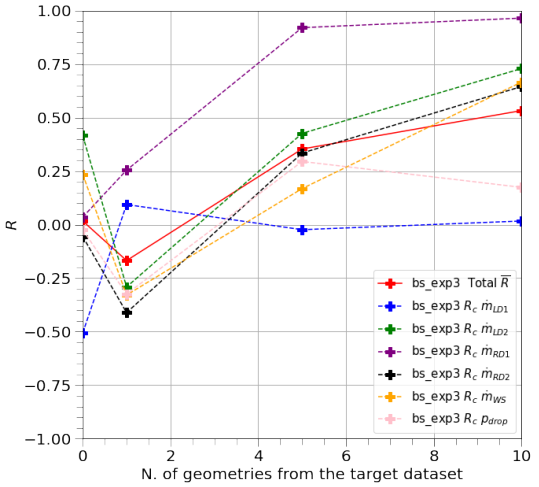
N. of geometries in training set	$\overline{L1}$ error			
	Pressure $[Pa]$	**X Velocity** $[m/s]$	**Y Velocity** $[m/s]$	**Z Velocity** $[m/s]$
5	1.0306267	0.098893605	0.12224015	0.098756544
10	0.8665144	0.083754085	0.10680834	0.08623453
20	0.7356513	0.07560715	0.095826864	0.0743815
30	0.7056592	0.06964034	0.09004241	0.06922776

7.2.2. Baseline model - transfer learning performance

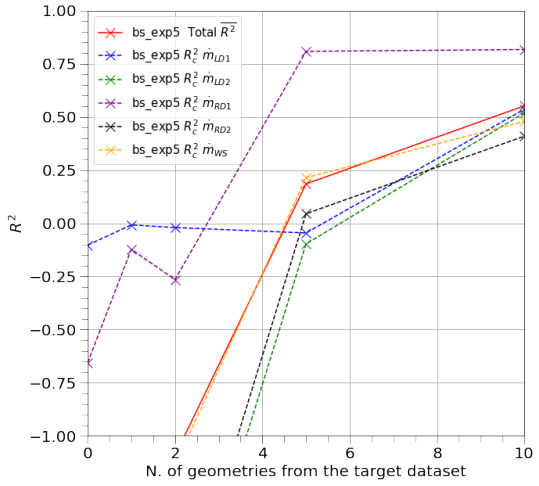
In this section, the results of Experiments 3 (*bs_exp3*) and 5 (*bs_exp5*) will be analyzed in a transfer learning setting. Experiment 3 uses full campaign 1 as the base training set, to which an increasing number of samples from campaign 4 are added. Experiment 5 uses combined campaigns 1, 2, and 3 as the base training data set, with an increasing number of geometries added from campaign 4. After each training, the model is used to predict the validation data set, which is a fixed subset of campaign 4. It is important to reiterate that no geometry is shared between the training and validation set. Therefore, each training/validation set consists of *n* geometries, each simulated for three inlet velocities.



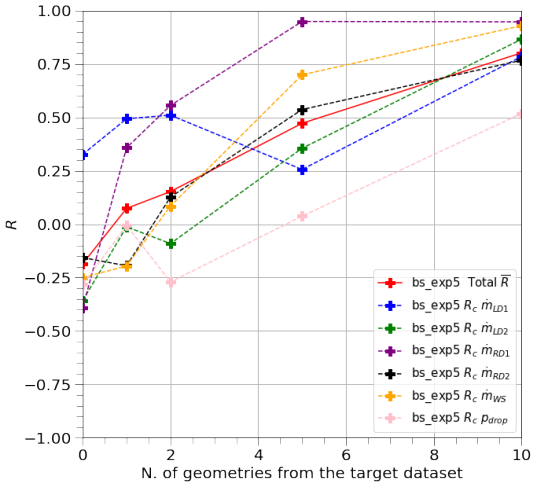
(a) R^2 Experiment 3 (*bs_exp3*).



(b) R Experiment 3 (*bs_exp3*).



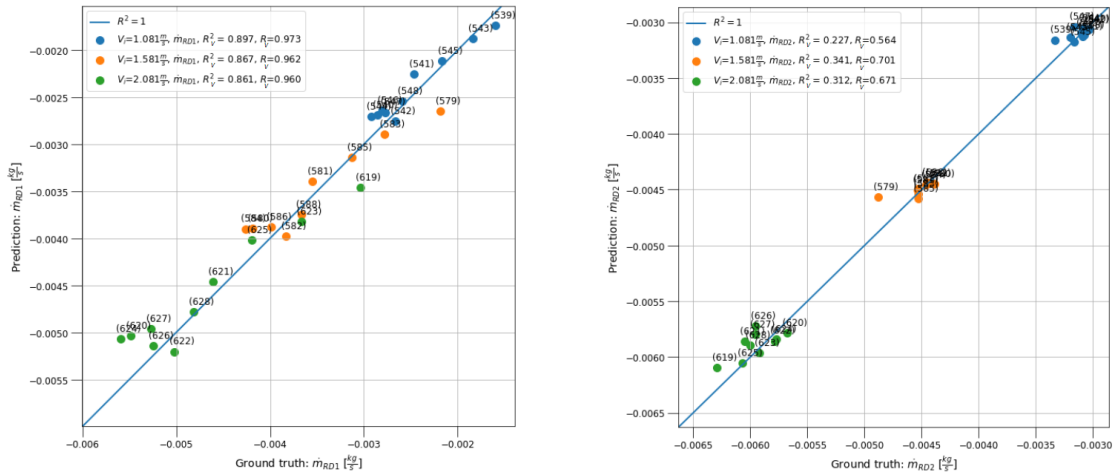
(c) R^2 Experiment 5 (*bs_exp5*).



(d) R Experiment 5 (*bs_exp5*).

Figure 7.3: R^2 and R of the ScalarNet predicted scalars of the test set for various runs of Experiments 3 and 5.

The results in Figure 7.3 show that $\overline{R}^2$ and $\overline{R}$ improve for *bs_exp3* and *bs_exp5* with increasing number of geometries from campaign 4 in the training set. Furthermore, there is a clear performance benefit in adding more base campaigns to the training set, since the models of *bs_exp5* (which has three base campaigns: 1,2 and 3) outperform the models of *bs_exp3* (which has only one base campaign: 1) consistently throughout the range of training data set configurations, with $\overline{R^2}$ at 10 geometries run of 0.55 for *bs_exp5* vs. 0.34 for *bs_exp3*. Likewise, the $\overline{R}$ obtained for this run is much higher for *bs_exp5* which is at 0.80 versus 0.53 for *bs_exp3*.



(a) $\dot{m}_{RD1}$. **(b)** $\dot{m}_{RD2}$.

Figure 7.4: *bs_exp3_3x10*: CFD ground truth vs. model predictions for $\dot{m}_{RD1}$ and $\dot{m}_{RD2}$.

In both *bs_exp3* and *bs_exp5*, $\dot{m}_{RD1}$ is the scalar with the highest prediction accuracy out of all the scalars, as seen in Figure 7.3. This is due to several factors. First, the right duct 1, at which end the mass flow $\dot{m}_{RD1}$ is calculated, has quite a large geometrical intra-campaign variability, making the local geometry very influential on the mass flow at the duct. This allows the geometry processing part of the network to easily capture the geometrical dependencies into the bottleneck based on which the scalar predictions are made. The other scalars are harder for the model to capture due to their smaller variability caused by changes between the geometries. This variability difference can be seen in Figure 7.4, where the prediction vs. ground truth are plotted for two scalars, with cluster colored based on the inlet velocity. Looking at the scatter plots one can see that $\dot{m}_{RD2}$ (Figure 7.4b) values are less affected by geometry and more by the inlet velocity, compared to $\dot{m}_{RD1}$ (Figure 7.4a). This is visible in the cluster distribution differences between the two scalars plotted. $\dot{m}_{RD2}$ clusters are very tightly packed together, showing a large variability due to the inlet velocity but a small variability due to geometrical differences. On the other hand, clusters of $\dot{m}_{RD1}$ are much more spread out, showing a large variability due both to inlet velocity and geometrical variability. As the local geometry of the other outlets (not left duct 1 or right duct 1 in Figure 4.8) associated with scalars, like $\dot{m}_{RD2}$, remains constant, the factors that affect mass flows are the secondary effects resulting from the changes in the side duct geometries (left duct 1 and right duct 1), as well as changes due to the different inlet velocities. As the effect on those mass flows of inlet velocity is much bigger than the effects caused by the geometrical changes, the model easily captures the inlet velocity dependency, but then struggles to predict the inter-cluster variability caused by the geometrical changes.

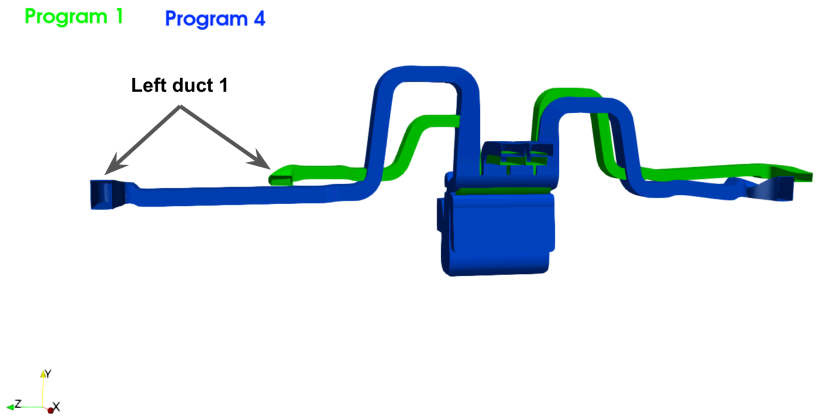


Figure 7.5: A geometry from campaign 1 compared to a geometry from campaign 4, showing the position difference between left ducts in the two geometries.

Moreover, in *bs_exp3* the $\dot{m}_{LD1}$ is an outlier throughout the experiment, as visible in Figure 7.3, where for this scalar the models perform particularly badly. The reasons behind this can be seen in Figure 7.5, where the figure shows a sample from campaign 1 in green and campaign 4 in blue. The geometries are fed into the model and are centered around the middle of the HVAC box, as shown in Figure 7.5. The relative distance between the outlet of campaign 1 and campaign 4 makes it harder for the model to transfer the knowledge from abundant campaign 1 to scarce campaign 4 for that duct. Therefore, this mass flow is captured so poorly by the *bs_exp3* models compared to the others, which are much closer together in the coordinate space.

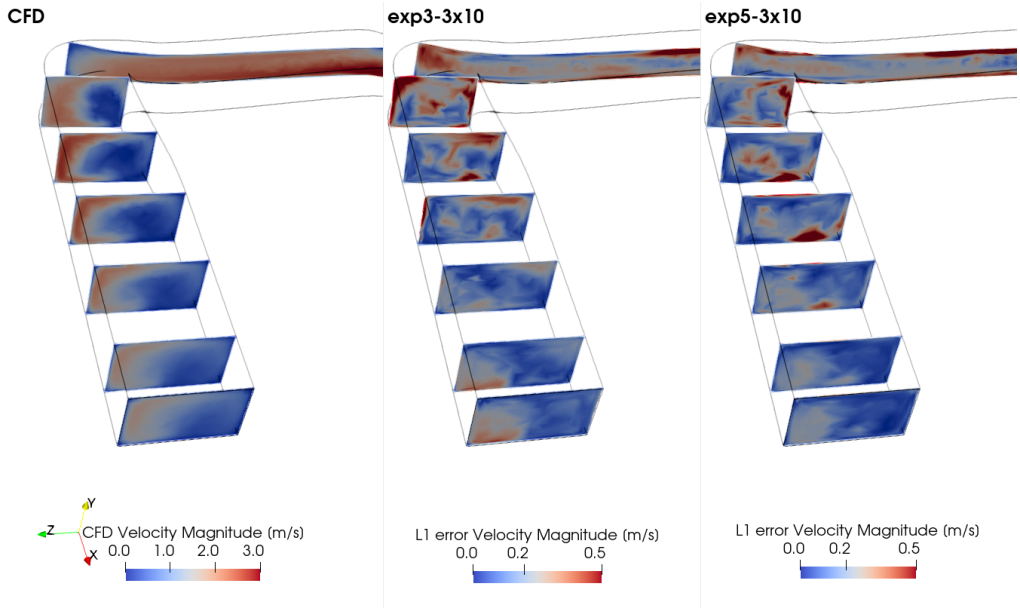


Figure 7.6: CFD and L1 errors for dp-585 sample from *bs_exp3_3x10* and*bs_exp5_3x10*.

This phenomenon is also visible when comparing the velocity field prediction errors between CFD, *bs_exp3* and *bs_exp5*, as on Figure 7.6. L1 errors are considerably higher in the left duct 1 for the *bs_exp3_3x10* model than for the same run of *bs_exp5*. In *bs_exp3* prediction, large areas of high error can be observed, especially around the last duct turn. In addition, the errors are higher all the way to the outlet plain of the duct. Interestingly, the error magnitude for the duct section upstream of the final turn looks very similar for both predictions, whereas all the large differences are downstream of the turn.

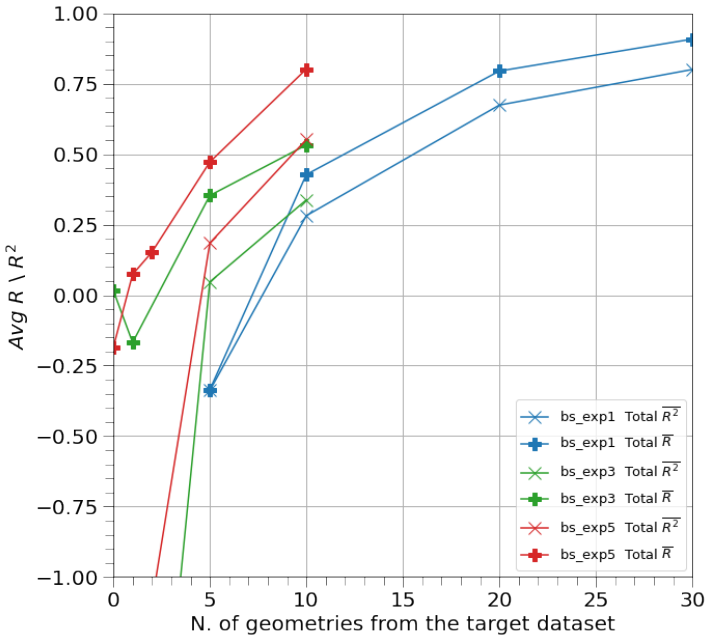
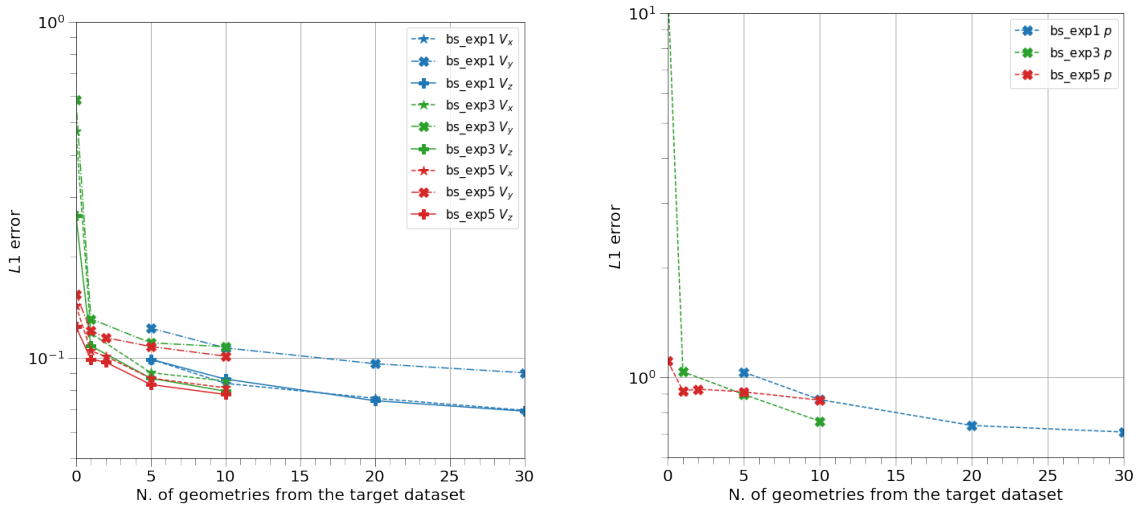


Figure 7.7: $\bar{R}$ and $\bar{R}^2$ comparison between *bs_exp1*, *bs_exp3* and *bs_exp5* for scalars from ScalarNet.

The purpose of transfer learning in this work is to improve the performance on the target data set by adding a small number of target data samples to the training set. To assess whether transfer learning had a positive impact on model performance, the results of *bs_exp1* are compared with the results of *bs_exp3* and *bs_exp5* in Figure 7.7. This compares the performance of the model in a single target campaign with the model using transfer learning.

Looking at Figure 7.7, both TL experiments *bs_exp3* and *bs_exp5* compare favorably with *bs_exp1* for the same number of geometries in the training data set. Both *bs_exp3* and *bs_exp5* with 5 and 10 geometries in the training set have higher $\bar{R}^2$ and $\bar{R}$ than the respective runs of *bs_exp1*. For *bs_exp5_3x10*, the correlation is on par with the correlation obtained in *bs_exp1* with twice the number of geometries from the target campaign used. Figure 7.8 presents the L1 error on velocity and pressure fields on the validation data set of *bs_exp1*, *bs_exp3* and *bs_exp5*. For velocity in Figure 7.8a the results show a clear converging error trend for all experiments. Furthermore, the trends obtained from scalar predictions hold for the field predictions, with *bs_exp5* having lower errors than *bs_exp3* across all training runs for all the velocity field components. Moreover, for the same number for geometries from campaign 4, both TL experiments have lower errors than *bs_exp1*. For pressure field in Figure 7.8b both TL experiments have equal or lower errors than *bs_exp1*, but for this field prediction, *bs_exp3* reaches lower errors than *bs_exp5* at larger data set sizes.



(a) Velocity. **(b)** Pressure.

Figure 7.8: L1 error on fields for Experiments 1,3 and 5 for baseline model.

7.2.3. Generalization performance

In this section, the ability of the model to generalize will be assessed. This is the ability to utilize the learning on one task and use the learnt knowledge to apply it to another task without additional data. In this work, this was tested by training two models, one of *bs_exp3_3x0* and one of *bs_exp5_3x0*, where no target data geometries were added to the training set.

Table 7.3: Correlation and mean percentage L1 error for two zero-shot runs, one for *bs_exp3_3x0* and *bs_exp5_3x0* each.

Cluster Inlet velocity V_{inlet}	$\dot{m}_{LD1}$		$\dot{m}_{LD2}$		$\dot{m}_{RD1}$		$\dot{m}_{RD2}$		$\dot{m}_{WS}$		p_{drop}	
$[m/s]$	R	$\overline{L1}$ [%]	R	$\overline{L1}$ [%]	R	$\overline{L1}$ [%]	R	$\overline{L1}$ [%]	R	$\overline{L1}$ [%]	R	$\overline{L1}$ [%]
bs_exp3_3x0												
1.08	-0.5863	28.74%	0.2116	29.99%	0.0775	43.93%	-0.0915	31.39%	0.1690	30.88%	-0.5771	6.47%
1.58	-0.4235	31.83%	0.5938	31.59%	-0.0090	42.34%	-0.0915	31.42%	0.0718	30.71%	0.0733	2.48%
2.08	-0.5085	33.95%	0.4452	33.74%	0.0361	45.43%	0.0166	34.59%	0.4639	31.59%	0.4307	8.52%
Average over clusters	-0.5061	31.51%	0.4169	31.77%	0.0349	43.90%	-0.0554	32.47%	0.2349	31.06%	-0.0244	5.82%
bs_exp5_3x0												
1.08	0.0458	15.85%	-0.2958	10.30%	-0.2676	19.71%	0.0017	12.49%	0.1572	12.49%	-0.1625	7.36%
1.58	0.3470	14.37%	-0.3408	14.62%	-0.5274	21.85%	-0.3017	13.64%	-0.2704	11.38%	-0.1132	4.88%
2.08	0.5867	15.50%	-0.4369	10.59%	-0.3765	19.62%	-0.1705	9.70%	-0.6386	9.00%	-0.5722	1.65%
Average over clusters	0.3265	15.24%	-0.3578	11.84%	-0.3905	20.39%	-0.1568	11.94%	-0.2506	10.96%	-0.2826	4.63%

Analyzing the values in Table 7.3, the errors for both generalization experiments are much higher than even for *bs_exp1_3x5* geometries configuration (Table 7.1). However, the errors from *bs_exp5_3x0* are less than half of *bs_exp3_3x0* errors. This can be attributed to the geometrical relatedness of the different campaigns used for the training, as in Figure 7.5. However, even with high errors, some scalars have a relatively high correlation. In *bs_exp5* the $\dot{m}_{LD1}$ for the highest velocity cluster reaches a correlation of 0.59, while in *bs_exp3* high positive correlation is predicted for $\dot{m}_{LD2}$ and $\dot{m}_{WS}$. Furthermore, the predicted p_{drop} for the low-velocity cluster has a correlation of -0.60 while the high-velocity cluster has a positive correlation of 0.43. These results are particularly surprising considering that in *bs_exp3_3x0* only campaign 1 was used, which consists of geometries simulated at $[1.0, 1.5, 2.0]\frac{m}{s}$, so in theory the high-velocity cluster for campaign 4 $(V_{inlet} = 2.08\frac{m}{s})$ is outside of the training data distribution.

Table 7.4: L1 error on the predicted fields for generalization runs: *bs_exp3_3x0* and *bs_exp5_3x0*.

Cluster Inlet velocity V_{inlet} $[m/s]$	$L1_c$ **error**			
	Pressure $[Pa]$	**X-velocity** $[m/s]$	**Y-velocity** $[m/s]$	**Z-velocity** $[m/s]$
bs_exp3_3x0				
1.08	6.0269	0.3125	0.3904	0.1698
1.58	10.9955	0.4693	0.5830	0.2635
2.08	18.9192	0.6328	0.7820	0.3609
Average over clusters	11.9805	0.4715	0.5851	0.2647
bs_exp5_3x0				
1.08	0.8033	0.1022	0.1069	0.0824
1.58	1.0012	0.1407	0.1515	0.1236
2.08	1.5094	0.1854	0.2051	0.1647
Average over clusters	1.104638	0.142729	0.154495	0.123563

Table 7.4 presents the L1 errors on the predicted fields. Here, the errors are presented separately for each velocity cluster in the validation set. Both models exhibit the same behaviour: lower velocity clusters have lower errors, whereas higher velocity clusters have higher errors. *bs_exp5_3x0* L1 errors are considerably lower when compared to *bs_exp3_3x0* results, but still larger than the *bs_exp1_3x5* (Figure 7.8). To investigate the predicted fields more closely, fields were plotted for one of the samples in Figure 7.9 and Figure 7.10.

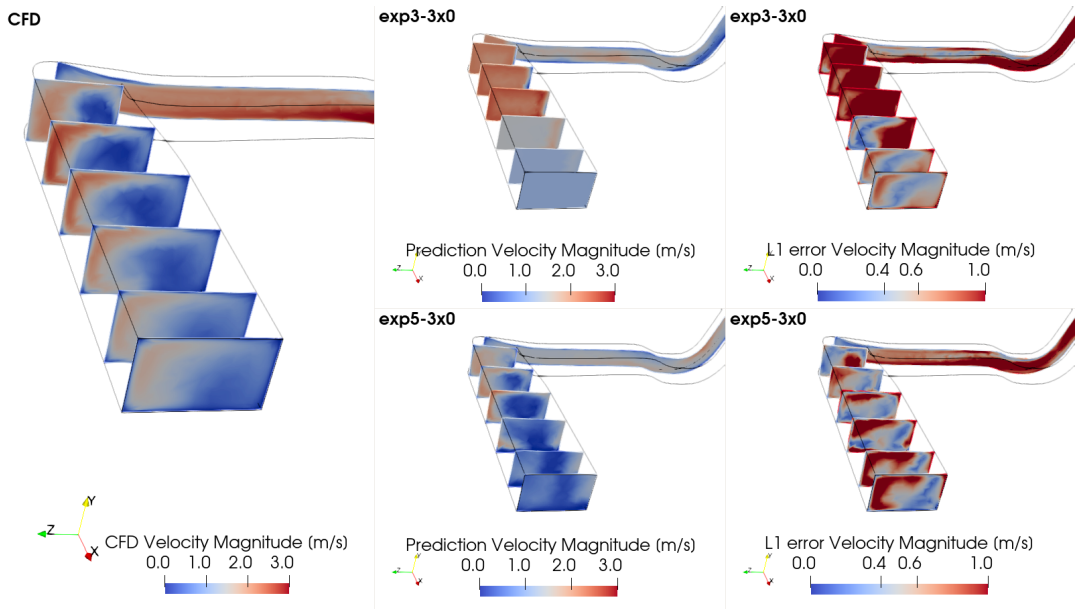


Figure 7.9: Baseline model generalization velocity comparison between *bs_exp3_3x0* and *bs_exp5_3x0* dp-585 sample.

In Figure 7.9 and Figure 7.10, the prediction of the velocity and pressure fields is compared with the ground truth for one of the side ducts of the sample *dp-585*. In the velocity graphs in Figure 7.9, the predictions of the *bs_exp3_3x0* model are smeared, while roughly showing the correct magnitude. There is a significant improvement visible in the prediction of *bs_exp5_3x0*, where the model was able to capture some of the prevailing high-velocity zones downstream of the duct turn. The L1 error in the velocity magnitude also shows much smaller errors in the prediction of the *bs_exp5_3x0* model. In general, those predictions are far from capturing the fields precisely, as the errors in the prediction are noisy. Moreover, they do not allow one to make an informed design decision based on the prediction, but at least they are able to recover an average pattern based on knowledge learned from the training set.

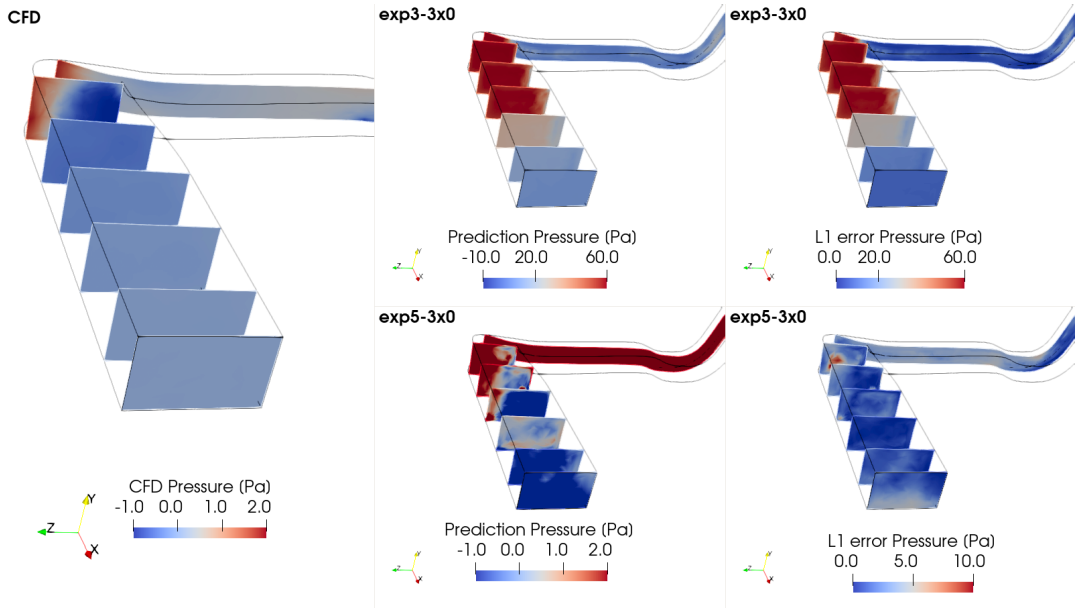


Figure 7.10: Baseline model generalization pressure comparison between *bs_exp3_3x0* and *bs_exp5_3x0* dp-585 sample.

Looking at the pressure comparison in Figure 7.10, the prediction of *bs_exp3_3x0* is highly erroneous, where the predicted pressure is more than an order of magnitude different from the CFD result. *bs_exp5_3x0* model prediction has much lower errors, which shows the benefit of the additional campaigns in the training set. However, the model struggles to predict pressure accurately, even for the *bs_exp5_3x0* prediction. Errors are of one order of magnitude larger than the actual prediction. For example, at the outlet the model predicts large areas of negative pressure, while in CFD the boundary condition of *0 Pa* is imposed there. This is a consistent feature for all the samples in the campaigns, and the model is unable to transfer this knowledge to this new campaign.

7.2.4. Baseline model performance - summary

In this section, the baseline model performance was established both in a single-campaign inference setting (*bs_exp1*) and in transfer learning inference (*bs_exp3* and *bs_exp5*). In addition, the generalization of the multi-campaign models was analyzed. Overall, the single-campaign training performance increased with increasing size of the training data set. In the transfer learning task, comparing models that transfer from a single campaign with models that transfer from multiple campaigns, as expected, the TL models reached better performance for the same number of samples from the target data set in the training set. Moreover, the single-campaign transfer is very sensitive to the choice of the campaign to be used as the base campaign. If the base campaign is not similar or has features not existing in the target campaign, the task of inferring correctly the impact of those geometrical features can be difficult for the model. This was also very apparent in generalization task performance analysis, where adding more base campaigns for TL significantly improved target data field prediction accuracy. Statistically, the larger the number of base campaigns, the higher the chance that the features in those will resemble the features of the target campaign, making it easier for the model to transfer the knowledge to the new campaign. On a general note, the importance of relative differences in scalar distributions was highlighted as an important factor to take into account, as their clustering and variance within clusters can make the task less or more challenging to learn.

7.3. Physics-informed surrogate models

In this section, the results of experiments with the methods introduced in Chapter 6, will be presented. This will aim to answer the two remaining Research Sub-questions from Chapter 3, namely:

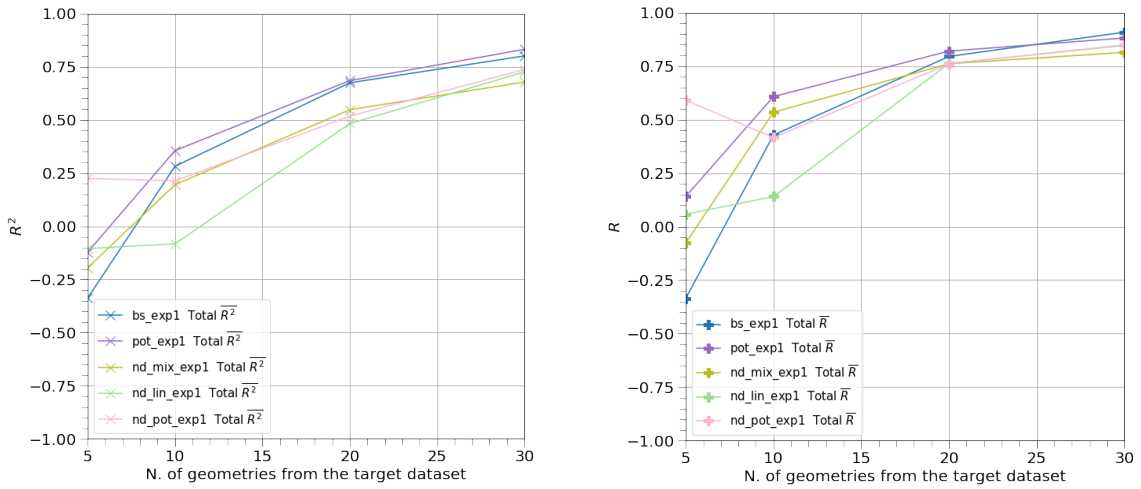
> "What is the performance of the physics-informed model at inference time and in the transfer learning task, and what is the impact of the methods on the generalizability of the model?"

and,

> "Can the different methods be combined, and if yes, what is their cumulative effect?"

To answer these questions, the entire DOE, presented in Table 4.3, will be run after having applied the selected methods. Moreover, a simplified solution addition method and linear scaling were later combined and used together to assess the compound effect of the approaches. Due to the approach that was taken to physics-informed losses, the results of this experiment will be presented in a separate section, which can be seen as a method to apply on top of any of the predictions made by the models presented in this work.

7.3.1. Physics-informed methods assessment in improving inference performance



(a) R^2 Experiment 1.

(b) R Experiment 1.

Figure 7.11: R^2 and R of predicted scalars from ScalarNet of the test set for baseline experiment, a simplified solution as an additional input experiment, linear and mixed scaling, and finally the combination of linear scaling and a simplified solution addition experiment.

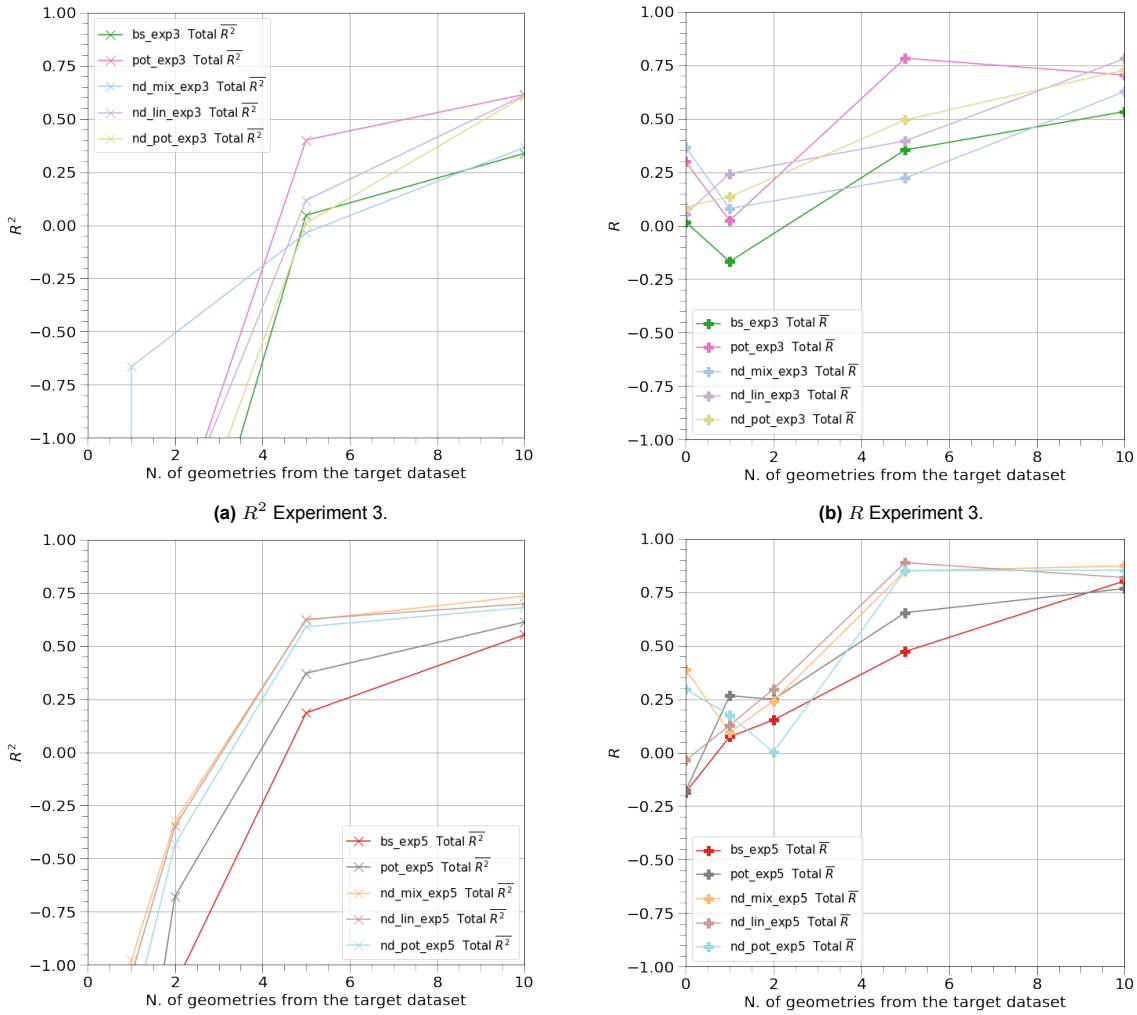
In this section, the impact on performance of physics-informed (PI) methods on the single-campaign model training run of Experiment 1 will be analyzed. The general performance of the models is presented in Figure 7.11a and Figure 7.11b. Here, a comparison is made with reference to the *bs_exp1*. In general, all methods show a converging trend with an increase in the size of the training set. With training set sizes of 20 and 30 geometries, the *bs_exp1* model performance on average is almost as good as any of the methods tried. The simplified solution addition method (*pot_exp1*) is the closest in performance to the *bs_exp1* models in a rich data environment. In runs with a smaller training set sizes, the addition of a simplified solution appears to improve the model performance with respect to *bs_exp1*. Moreover, on average, with the lowest data set size of 5 geometries, the combined simplified solution addition and scaling method (*pot_nd_exp1*) is an outlier, as it shows by far the best performance of all the methods tried at this training set size. This improvement in performance coming from

the addition of a simplified solution at lower training set sizes is encouraging, as it points towards the ability of the model to utilize this input in low data regimes. This is particularly encouraging for further experimentation in the TL setting.

The worst performance was measured for the models with linear scaling applied (*nd_lin_exp1*). This observation leads to an interesting question. The difference between linear and mixed scaling is the treatment of pressure-related quantities. For the calculation of $\overline{R^2}$ the pressure drop scalar is not taken into account, which means that the velocity fields and the mass flows training data were treated the same way between the training with linear (*nd_lin_exp1*) and mixed scaling (*nd_mix_exp1*). However, there is a very large performance difference between the linear and mixed scaling in Figure 7.11a in the low data regime. This could signify that the different pressure scaling between the two training runs has a significant impact on the velocity-related quantities. Looking at the correlations in Figure 7.11b, the trends from the R^2 plots are maintained, with *pot_exp1* showing superior performance with reference to the baseline at a smaller training data set size, while *nd_lin_exp1* performs poorly in low data regimes, but recovers at the training set size of 20 geometries. *pot_nd_exp1* performance curve has roughly the same shape as the curve for *nd_lin_exp1*, although it is shifted to a higher performance starting point at low data regimes, perhaps due to the added base flow prediction coming from the simplified solution addition.

7.3.2. Physics-informed methods assessment in improving transfer learning performance

In this section, the proposed methods will be assessed in a TL setting using the design of Experiment 3 and Experiment 5- transferring from a single base campaign to the target campaign and transferring from three base campaigns to one target campaign, respectively. In both cases, the target campaign is campaign 4.



(a) R^2 Experiment 3.

(b) R Experiment 3.

(c) R^2 Experiment 5.

(d) R Experiment 5.

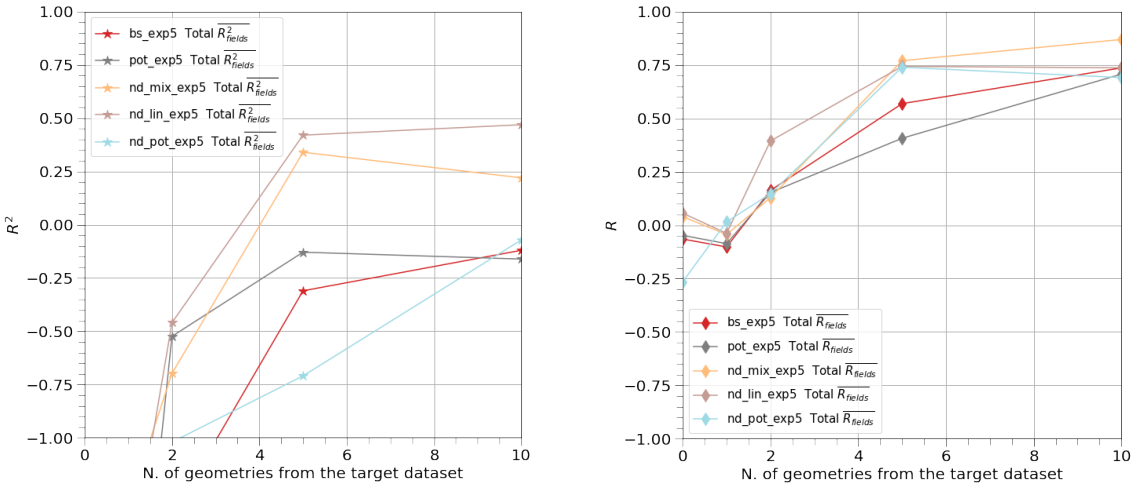
Figure 7.12: R^2 and R of predicted scalars from ScalarNet of the test set for baseline experiment (*bs*), a simplified solution addition (*pot*), linear (*nd_lin*) and mixed scaling (*nd_mix* and finally fusion of linear scaling and a simplified solution addition experiment(*pot_nd*.

Figure 7.12 presents the general performance curves for the different methods applied to Experiments 3 and 5. For Experiment 3, all methods consistently outperform the *bs_exp3* model, with the exception of the *nd_mix_exp3_3x5*. In single-campaign transfer learning *pot_exp3* consistently outperforms other approaches, reaching considerably better $\overline{R^2}$ than the *bs_exp3* models. It performs particularly well in the 3×5 training run, where it reaches $\overline{R^2}$ of 0.40 vs. 0.10 for the baseline model. *pot_exp3_3x10* as well as the *nd_lin_exp3_3x10* and *nd_pot_exp3_3x10* models reach very similar high performance ($\overline{R^2} \approx 0.6$ and $\overline{R} \approx 0.75$), considerably outperforming the *bs_exp3_3x10* model run.

Looking at Experiment 5, overall all methods outperform the *bs_exp5*, but their relative performance to each other has changed relative to Experiment 3. *nd_mix_exp5*, *nd_lin_exp5* and *pot_nd_exp5*

reach very similar performance at almost all data set sizes, constantly outperforming the *bs_exp5* models. Furthermore, *pot_exp5* performs the least favourably of all the methods presented, but it nonetheless outperforms the *bs_exp5* models throughout the range of training set sizes. The largest $\overline{R^2}$ performance difference between the *bs_exp5* and the PI methods occurs at a *3x5* data set size where *nd_mix_exp5* and *nd_lin_exp5* reach almost identical performance of 0.62 vs. 0.2 for the *bs_exp5_3x5* model. Overall, this consistent performance between two types of scaling in the R^2 plots indicates that at least in this Experiment 5 the interaction between the pressure quantities and velocity-derived quantities did not significantly affect the overall model scalar prediction performance.

To assess the quality of the fields, the scalars were recomputed using the predicted fields, and the results for Experiment 5 are presented in Figure 7.13. The immediate observation when looking at the plot of Figure 7.13a is the substantially lower overall R^2 scores for all the training runs compared to scalars predicted from ScalarNet, plotted in Figure 7.12c. With no runs of *bs_exp5* models, as well as the *pot_exp5* and *pot_nd_exp5* reaching values of R^2 of above 0. Furthermore, both *nd_lin_exp5* and *nd_mix_exp5* reach positive values, with linear scaling reaching R^2 of 0.48 at the largest training set size. This poor performance for some methods would indicate that the model did not correctly capture the field variability of the validation set at the outlet planes. Looking at the correlation calculated based on the scalars from fields in Figure 7.13b, the performance of the scalars is more similar to the performance of the ScalarNet scalars, with all models reaching high correlation (0.70 and higher) with increasing training set size. *pot_nd_exp5*, *nd_mix_exp5* and *nd_lin_exp5* outperform the *bs_exp5* model at 3×5 data set size, while at the highest added data set size of 3×10 *nd_mix_exp5* reaches a correlation of 0.85 vs. 0.75 for the *bs_exp5_3x10* model. These results combined with the findings from Figure 7.13a suggest that the models can capture the outlet fields well, in terms of their variance with respect to different geometrical changes, but struggle to accurately capture the magnitude of the fields at the outlets, which is highlighted by the low R^2.



(a) R^2 Experiment 5.

(b) R Experiment 5.

Figure 7.13: R^2 and R of the calculated scalars based on the predicted fields of the test set for the baseline experiment (*bs*), the simplified solution addition experiment (*pot*), linear (*nd_lin*) and mixed scaling (*nd_mix* and finally the fusion of the linear scaling and the simplified solution addition experiment (*pot_nd*.

Detailed plots of L1 errors for both Experiment 3 and Experiment 5 fields for different methods can be found in Appendix D.

Effects of adding a simplified solution

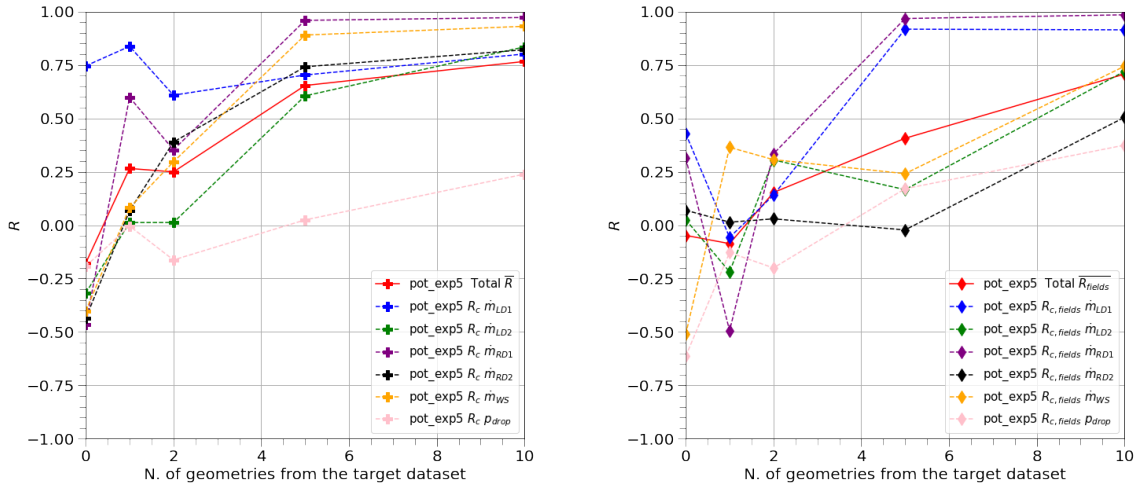
Using the velocity field obtained from the simplified solution simulation (potential flow solution in the case of this work), one can calculate the mass flow for each duct. This was done to analyze how the performance of the model compares with the performance of simplified solution-derived mass flows.

The calculated results for campaign 4 are presented in Table 7.5.

Table 7.5: R^2 and R metrics on mass flows from the validation set of campaign 4 calculated using the simplified solution (potential flow (PF)) velocity solution obtained from ANSYS Fluent.

Cluster Inlet velocity V_{inlet} **[m/s]**	$\dot{m}_{LD1}$		$\dot{m}_{LD2}$		$\dot{m}_{RD1}$		$\dot{m}_{RD2}$		$\dot{m}_{WS}$	
	R^2	R	R^2	R	R^2	R	R^2	R	R^2	R
1.080	-17.794	0.453	-95.143	0.366	-16.582	-0.507	-122.191	0.267	-825.135	-0.440
1.580	-17.075	0.941	-86.193	0.820	-12.584	0.890	-101.442	0.667	-2894.681	0.421
2.080	-15.677	0.893	-98.351	0.733	-15.515	0.853	-93.813	0.647	-924.864	0.329
Average	-16.848756	0.761981	-93.228958	0.639852	-14.893660	0.411717	-105.815373	0.526838	-84.002264	0.540166

The mass flows derived from the velocity field which was obtained from the simplified solution are presented in Table 7.5. As expected, they have very low, non-physical R^2. This is because the velocity magnitude is lower in the ducts for the simplified solution (PF) vs. viscous CFD simulations. Moreover, the simplifications in the PF theory (lack of viscous effects and flow being irrotational) lead to much more uniform flow patterns at the outlet. Nevertheless, looking at the scalar correlation obtained from this simplified solution velocity field, there is a significant positive correlation between the simplified solution-derived values and the CFD. This allows to compare the correlations between the GNN models, trained during this work, to the values obtained using the simplified solution. To illustrate this, correlation evolution with the size of the data set per scalar are presented in Figure 7.14, where both plots for scalars from the ScalarNet and scalars calculated based on the predicted fields are plotted for *pot_exp5*.



(a) R from ScalarNet output.

(b) R calculated from predicted field.

Figure 7.14: R for *pot_exp5*.

When comparing the average R_c for the simplified solution (Table 7.5) with the ScalarNet scalars (Figure 7.14a), it is visible that the model needs a threshold number of training geometries from the target data, before its predictions are better than the correlations obtained using the simplified solution. For most scalars, R is higher than the simplified solution-derived scalars at the 3×5 samples from the target campaign. An outlier is $\dot{m}_{LD1}$ which oscillates around the value obtained using the simplified solution velocity field for the entire range of data set sizes. Looking at scalars from fields, which in general perform slightly worse than scalars from ScalarNet, as seen in previous sections, the majority of the scalars outperform simplified solution-derived scalars at the biggest data set size. It is important to note that for all data set sizes, trained models outperform the simplified solution in terms of R^2.

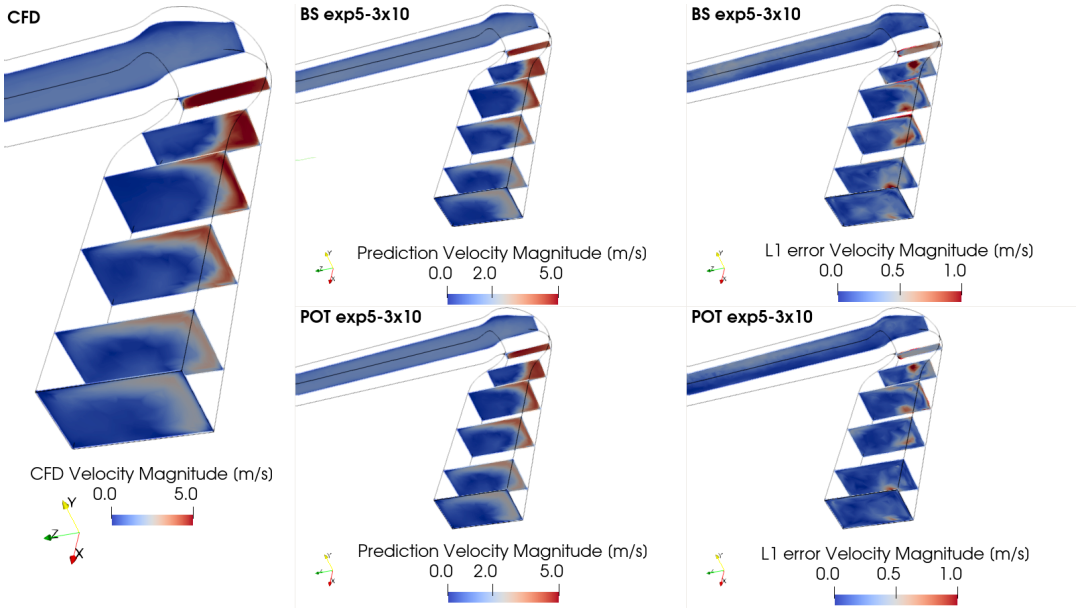


Figure 7.15: Comparison between CFD, *bs_exp5_3x10* model prediction and *pot_exp5_3x10* model.

The velocity fields for *bs_exp5_3x10* as input were plotted in Figure 7.15. This was done to investigate the field predictions and error patterns to see if they changed due to the addition of the input field. A particular sample from the validation set was chosen for this visualization, as the sample plotted has a particular geometrical feature, a converging-diverging section causing large acceleration in the flow. This feature is captured by the simplified solution , and the hypothesis that the simplified solution helps with predicting features like this was investigated. Looking at the plot, at first glance both predictions present similar flow patterns and magnitude as the CFD ground truth, with both predictions slightly under-predicting the velocity in the throat of the converging section. Looking at the L1 error plots, it is apparent that the error patterns between the two predictions are very similar, although the prediction from the model with the simplified solution has significantly lower errors, especially in the throat section of the duct. This would suggest that, indeed, the addition of the simplified solution may be helping to predict this feature more accurately.

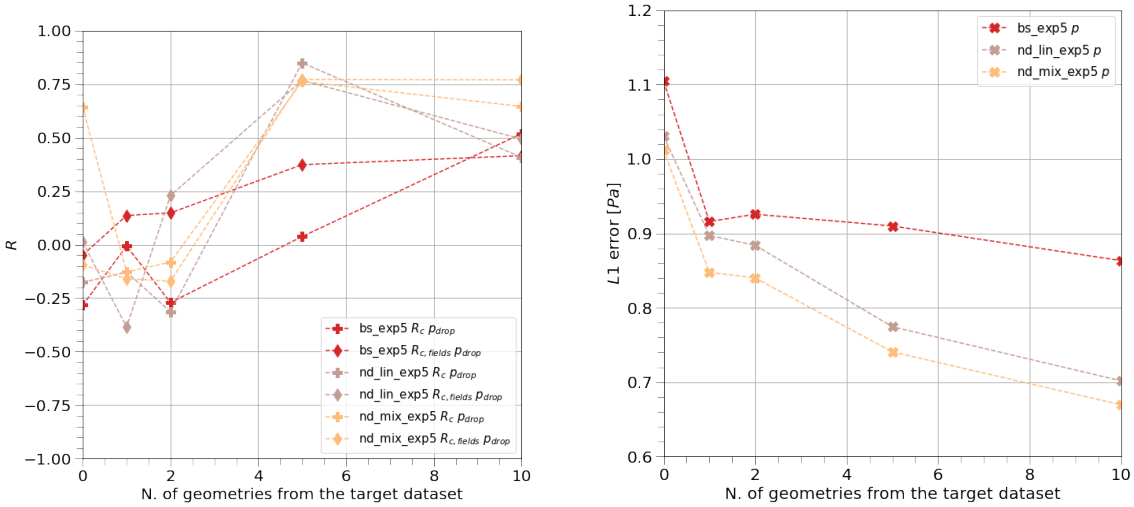
Comparing scaling effects

Previously, in Section 7.3.2, the positive impact of scaling on velocity-derived quantities was presented based on the R^2 plots. Nevertheless, the difference between the scaling methods themselves is in the way the pressure quantities are treated. For the purpose of this comparison, Figure 7.16a compares the pressure drop R_c both from the ScalarNet scalars, and the scalars calculated from the predicted field. The results show that the R_c calculated based on the pressure drop from the fields is lower for the same training run, than the R_c for the pressure drop from the ScalarNet.

Moreover, the addition of data seems to have a stabilizing effect on the difference between the p_{drop} computed from fields and p_{drop} from the ScalarNet, where the larger the training set, the lower the relative difference. The best R_c (0.83) is reached by the p_{drop} from ScalarNet in the *nd_lin_exp5_3x5* run. This is surprising, as one would expect the best model performance to occur at the highest data set size (*3x10*). However, *nd_mix_exp5* R_c on p_{drop} from ScalarNet was very close at the same data set size. Moreover, unlike for linear scaling, the higher correlation was maintained at higher training set size when mixed scaling was used. When looking at both scalars from fields and from ScalarNet with mixed scaling, the R_c obtained was considerably higher than for *bs_exp5_3x10* and *nd_lin_exp5_3x10*.

L1 pressure field error evolution with data set size is plotted in Figure 7.16b for Experiment 5. As expected, the pressure field average absolute error decreases with the increase in the number of samples from campaign 4 in the training set. Moreover, both scaling methods present a steeper rate of

decrease, and drop to lower error values than the *bs_exp5*. The error curves for both scaling methods look almost identical in shape, but there is a substantial difference in magnitude, where mixed scaling consistently reaches lower error values than linear scaling.



(a) Comparison of R and R_{fields}.

(b) Comparison or L1 errors on pressure field.

Figure 7.16: Comparison of pressure drop and pressure field performance for the baseline model versus scaling techniques for Experiment 5.

As the values of $R_c^2 > 0$ on p_{drop} were never reached, the magnitude of this quantity was not yet analyzed. Looking at Table 7.6, the L1 errors for p_{drop} from Experiment 5 were compared between the baseline and different scaling methods. Across all three methods compared, the errors drop with the increased contribution of campaign 4 in the training set. In the zero-shot experiment (*3x0*), the baseline model predicts by far the most erroneous value with the L1 percentage error of 4.63%. For the same data set size, linear scaling reaches the lowest value of 1.7%. The lowest overall error was reached by the largest data set size for a model with mixed scaling, where the error drops to 0.58% vs.0.72% for linear scaling p_{drop}.

Table 7.6: Comparison of the L1 pressure drop error for the different scaling methods used.

N. of geometries in training set	p_{drop} **L1 error**					
	bs_exp5		***nd_lin_exp5***		***nd_mix_exp5***	
	L1 [Pa]	L1 [%]	L1 [Pa]	L1 [%]	L1 [Pa]	L1 [%]
0	2.370	4.63%	0.756	1.70%	1.179	2.87%
1	0.792	1.08%	0.645	0.94%	0.671	1.12%
2	0.743	1.00%	0.654	0.91%	0.664	1.05%
5	0.880	1.22%	0.659	0.78%	0.720	0.91%
10	0.628	0.77%	0.570	0.72%	0.474	0.58%

7.3.3. Physics-informed methods assessment in improving generalization performance

In this section, the impact of the proposed PI methods on the generalization performance of the models is assessed. This was be done by comparing the zero-shot model training runs with different methods against the baseline model performance for Experiment 5. Table 7.7 presents the correlations and L1 errors for all zero-shot runs for Experiment 5 using different PI methods. In general, the model with mixed scaling (*nd_mix_exp5_3x0*) has considerably better R (0.38) compared to models with other PI methods $(-0.17, -0.03, 0.3)$ and the baseline model (-0.18). Moreover, the model with mixed scaling deviates from the trend presented by other models. Both the baseline and the rest of the zero-shot PI

models predict the data correlation of $\dot{m}_{LD1}$ well (to a different degree, but this scalar is a positive outlier in all these runs). The *bs_exp5_3x0* model as well as the *nd_lin_exp5_3x0* and *pot_exp5_3x0* capture no correlation between all the other scalars. *pot_nd_exp5_3x0* captures the $\dot{m}_{LD1}$ correlation well, as well as shows some signal of learning on $\dot{m}_{RD2}$ and $\dot{m}_{WS}$. The *nd_mix_exp5_3x0* scalar correlation patterns change significantly, as the model reaches a significant correlation on all scalars, apart from $\dot{m}_{LD1}$ and $\dot{m}_{WS}$.

Table 7.7: Correlation and mean percentage L1 error for zero-shot runs, for different physics-informed methods on Experiment 5.

Cluster Inlet velocity $V_{inlet}\ [m/s]$	$\dot{m}_{LD1}$		$\dot{m}_{LD2}$		$\dot{m}_{RD1}$		$\dot{m}_{RD2}$		$\dot{m}_{WS}$		p_{drop}		**Average**	
	R	L1 [%]	R	L1 [%]	R	L1 [%]	R	L1 [%]	R	L1 [%]	R	L1 [%]	**R**	**L1 [%]**
Baseline Model Exp 5 3x0														
Average over clusters	0.3265	15.24%	-0.3578	11.84%	-0.3905	20.39%	-0.1568	11.94%	-0.2506	10.96%	-0.2826	4.63%	**-0.1853**	**12.50%**
Model with simplified solution Exp 5 3x0														
Average over clusters	0.7457	13.70%	-0.3182	11.66%	-0.4653	20.79%	-0.4366	14.30%	-0.4017	11.86%	-0.1925	1.83%	**-0.1781**	**12.36%**
Model with linear scaling Exp 5 3x0														
Average over clusters	0.7297	13.92%	-0.2704	11.00%	0.0227	20.13%	-0.3590	10.72%	-0.1608	8.29%	-0.1761	1.70%	**-0.0357**	**10.96%**
Model with mixed scaling Exp 5 3x0														
Average over clusters	0.0998	14.33%	0.4247	13.02%	0.3811	20.71%	0.6071	12.95%	0.1709	11.50%	0.6421	2.87%	**0.3876**	**12.56%**
Model with linear scaling and simplified solution Exp 5 3x0														
Average over clusters	0.7354	13.87%	0.0385	10.31%	0.1404	19.69%	0.6121	11.00%	0.4298	7.48%	-0.1808	0.60%	**0.2959**	**10.49%**

Figure 7.17 compares the velocity field for the *bs_exp5_3x0* model with the same model configuration with mixed scaling. In addition, *nd_mix_exp5_3x1* was added for comparison. The *nd_mix_exp5_3x0* model reaches relatively high correlations on the scalars, and adding one sample to the training data from campaign 4 produces much worse results, as can be observed in Figure 7.12d. When comparing the fields, this trend is not reflected in the accuracy of the predictions, as the *nd_mix_exp5_3x0* has much larger error on the velocity field than *nd_mix_exp5_3x1*. The zero shot prediction with mixed scaling has large errors close to the bottom wall upstream from the final bend. Moreover, the velocity patterns downstream of the bend do not resemble the training data patterns, as there are high-velocity zones on both sides of the duct. When comparing the zero-shot with mixed scaling to zero-shot prediction from the baseline model, the baseline model shows lower errors, especially downstream of the final bend. Moreover, the flow patterns resemble the final flow more closely.

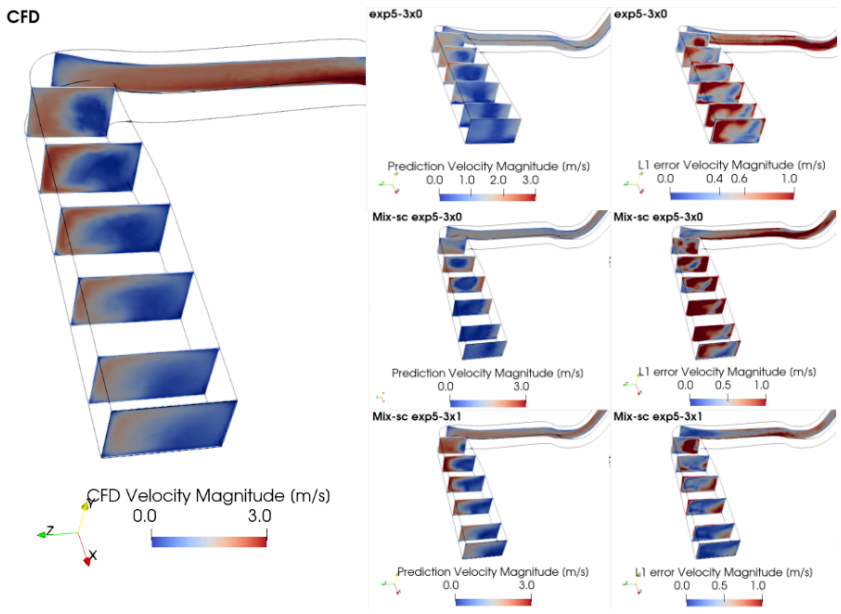


Figure 7.17: Zero shot prediction of dp-585 from Experiment 5 3x0 and 3x1 runs with mixed scaling.

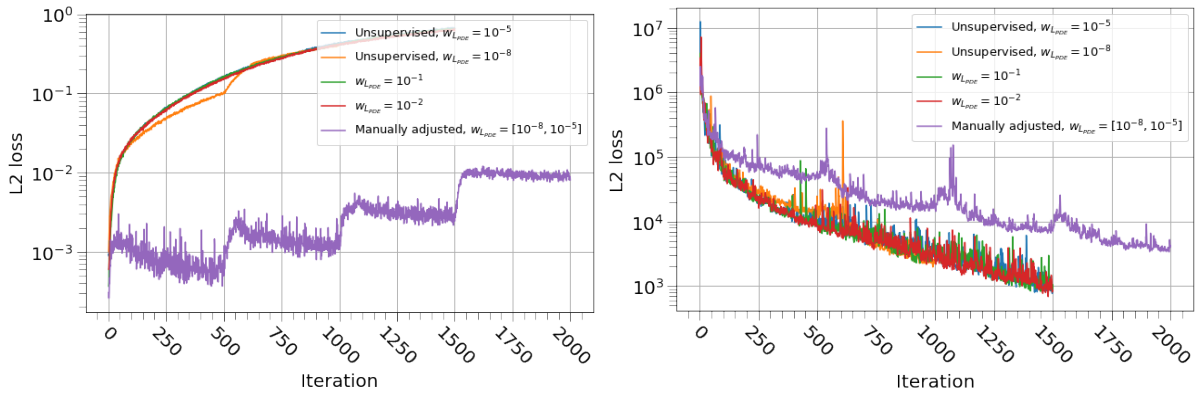
7.3.4. Physics-informed losses

In this section, the results of experimentation with physics-informed losses are presented. In this work, this approach is treated as a posterior correction of the surrogate model predictions, rather than surrogate modeling. In this section, one of the trained models was used to predict the validation data set. Subsequently, an outlier from the predictions was chosen on the basis of average field error values. Then, the trained model formed the base model to which a PDE loss is added, and the model is retrained again with the new loss and the outlying sample as the only sample in the training set. Different weightings of the PDE loss were tested. This was done to establish the impact of the PDE loss on the prediction, in this setting, where unlike other works on the topic (see Cai et al. [2021a]), no boundary condition loss is used. Rather an implicit information about boundary and initial condition is given in the form of data. In this case the prediction of the outlying sample of the base model before retraining with PDE loss. For all models trained in this section, all layers of the network were fixed, apart from the FieldNet. This was done, as the FieldNet is the main component of the network responsible for the field values predictions and improving those is the target use case of the method. For this reason, in this section only field prediction will be assessed. The expected result of this experiment was that the PDE loss, since it is solving part of the same equations the ground truth CFD is solving, would correct the outlying predictions, to closer represent the CFD ground truth.

For the various loss-weighing configurations used, the model was saved at intermediate steps to evaluate the evolution of the predictions with the progress of the training run. The following models were trained:

- Unsupervised 1 - no loss on the data with $w_{l_{PDE}} = 1$
- Unsupervised 2 - no loss on the data with $w_{l_{PDE}} = 1E-08$
- Supervised 1 - supervision on the data with $w_{l_{PDE}} = 1E-01$
- Supervised 2 - supervision on the data with $w_{l_{PDE}} = 1E-02$
- Supervised 3 - supervision on the data with gradually increasing $w_{l_{PDE}}$, starting from $w_{l_{PDE}} = 1E-8$ and increasing by order of magnitude for roughly each 500 iterations until completion at $w_{l_{PDE}} = 1E-5$ at 2000 iterations.

The training curves are presented in the Figure 7.18.

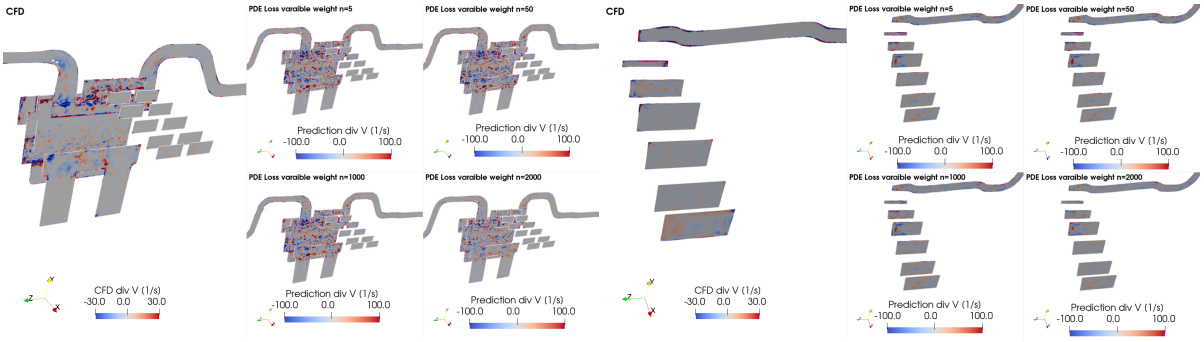


(a) Field L2 loss.

(b) Divergence L2 loss.

Figure 7.18: Loss plots for various weightings used to the physics loss training.

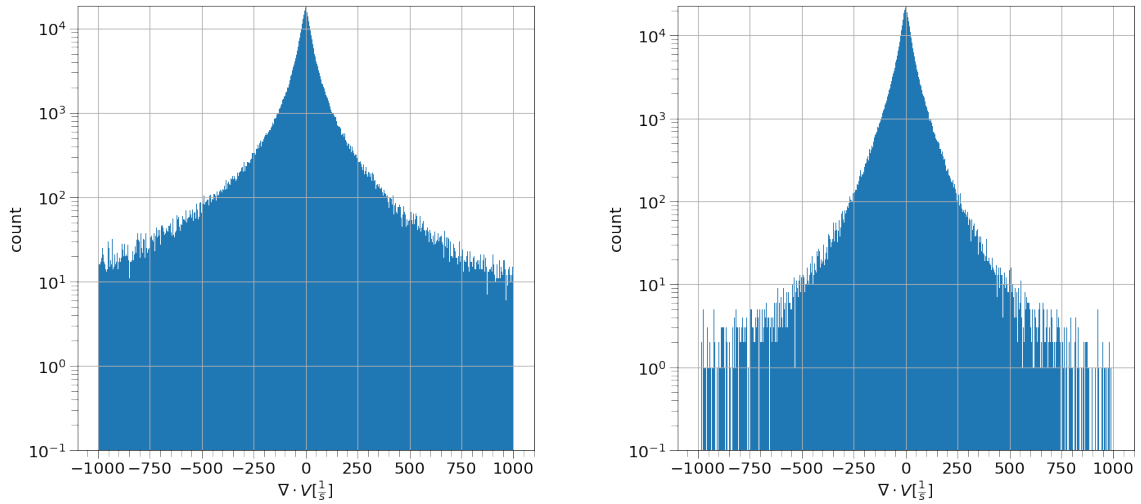
First, to validate the workings of the loss, the CFD gradients were used to calculate the divergence of the velocity field, and then the same was done for the predictions of the Supervised 3 training. Intermediate models were saved after $n = 5, 50, 1000, 2000$ iterations. The results are presented in Figure 7.19.



(a) Main HVAC box. **(b)** Left duct.

Figure 7.19: CFD divergence vs. divergence as predicted at various intermediate steps of variable weight model training.

The results show that the divergence of the velocity field in both the CFD and the prediction of the model is much higher in the main HVAC box than in the ducts. Moreover, the divergence predicted by the model is much higher than the divergence calculated by the CFD. Furthermore, the patterns between the CFD and the model predictions repeat themselves, especially in the HVAC box. Moreover, it can be noticed that, with the progression of the training, the divergence of the predicted velocity field decreases, especially in the very high divergence zones in the HVAC box. To further illustrate this, the histograms of the divergence of the velocity field for the predicted sample are presented in Figure 7.20.



(a) $n = 5$ iterations. **(b)** $n = 1500$ iterations.

Figure 7.20: Divergence of the velocity distribution for predictions of the Supervised 3 training.

The results show a clear change between the initial prediction of the model, after 5 iterations in Figure 7.20a and the predictions after 1500 iterations in Figure 7.20b. The distribution narrows, with a larger number of field points closer to having zero divergence. What is not visible in the plots is the fact that the extreme values are the most affected by training. These values are most often found near the walls of the HVAC box, as can be seen in Figure 7.19. Unfortunately, looking back at Figure 7.19 the decrease in divergence occurs mainly in areas where the divergence is large, so the main HVAC box, whilst the quantities of interest for this particular work are in and around the inlet and outlets of the geometry.

Table 7.8: Physics-informed loss experiment results including intermediate results. Relative improvement on L1 error for scalars calculated from fields and fields themselves.

Intermediate prediction			Scalars from fields						Fields			
Name/mode	w_{LPDE}	N_{iter}	$\dot{m}_{LD1}$	$\dot{m}_{LD2}$	$\dot{m}_{RD1}$	$\dot{m}_{RD2}$	$\dot{m}_{WS}$	$_{drop}$	Pressure	X-velocity	Y-velocity	Z-velocity
Supervised 2	1E-2	5	19183.13%	10.05%	68.19%	**-17.70%**	**-33.54%**	**-51.49%**	**-3.78%**	**-1.18%**	0.51%	0.04%
Supervised 2	1E-2	50	58524.56%	174.67%	**-13.12%**	211.94%	199.34%	1.98%	19.62%	50.91%	34.26%	4.42%
Supervised 2	1E-2	150	89266.37%	268.28%	**-46.33%**	347.39%	477.00%	19.60%	24.57%	96.49%	89.77%	8.99%
Supervised 2	1E-2	1500	204397.65%	1306.59%	575.92%	1981.40%	2798.98%	175.50%	82.48%	560.34%	477.78%	126.84%
Supervised 3	1E-8	5	11941.38%	36.13%	5.68%	15.73%	28.80%	75.05%	1.11%	0.95%	0.68%	0.17%
Supervised 3	1E-8	50	6307.43%	16.69%	15.91%	**-11.18%**	**-0.78%**	**-5.69%**	1.82%	2.38%	1.71%	0.23%
Supervised 3	1E-8	150	401.59%	24.48%	2.56%	1.01%	5.03%	0.28%	1.00%	1.53%	0.88%	0.11%
Supervised 3	1E-7	500	3541.99%	19.85%	2.73%	**-4.77%**	**-4.26%**	0.98%	0.43%	1.00%	0.61%	**-0.04%**
Supervised 3	1E-6	1000	3379.21%	30.42%	6.88%	1.99%	14.11%	2.39%	0.68%	2.40%	1.10%	0.31%
Supervised 3	1E-5	1500	11338.15%	33.74%	**-5.88%**	0.44%	27.27%	5.21%	2.82%	4.09%	2.68%	0.42%
Supervised 3	1E-5	2000	2077.80%	46.26%	**-14.95%**	6.45%	7.40%	19.65%	8.02%	17.77%	13.13%	1.15%
Unsupervised 1	1E-5	5	14514.10%	14.56%	77.55%	**-16.85%**	14.68%	43.97%	76.06%	**-0.40%**	**-1.02%**	0.21%
Unsupervised 1	1E-5	50	46799.00%	157.39%	**-62.20%**	178.30%	196.50%	97.51%	109.99%	48.31%	32.58%	4.11%
Unsupervised 2	1E-8	5	49653.34%	**-26.76%**	194.39%	**-71.88%**	**-53.31%**	237.55%	114.54%	**-0.78%**	1.70%	1.43%
Unsupervised 2	1E-8	50	18420.46%	89.28%	140.89%	81.29%	154.77%	673.71%	226.41%	38.78%	38.54%	7.01%
Unsupervised 2	1E-8	150	25991.69%	179.79%	188.31%	235.22%	410.22%	724.14%	222.09%	82.39%	72.90%	5.84%
Unsupervised 2	1E-8	1000	88791.00%	1059.44%	173.63%	1618.05%	2264.04%	1364.27%	261.71%	445.54%	407.95%	75.60%

To track the results, the relative percent error improvement on scalars from fields and L1 errors of the fields were calculated. The results of the various experiments are presented in the Table 7.8 (complete results available in the Appendix E), with **bold** metrics meaning a relative improvement over the baseline model prediction. Overall, the results show no instance where all quantities of interest are improved by applying the correction. Moreover, all unsupervised models diverge very quickly from the ground truth. The same phenomenon occurs when the model is supervised on data, but is much slower and greatly depends on the choice of the weight of the physics-informed loss. In the case of Supervised 3 training, the model is able to correct the field divergence, while not diverging greatly from the ground truth. But even in this scenario, the improvements gained are small and very inconsistent. In all other models, there are intermediate predictions that show an improved performance, usually in the first 50 iterations. The best intermediate prediction was produced by the Supervised 2 model , with 0.01 weight on the PI loss, as it was able to decrease the L1 errors on $\dot{m}_{RD2}$, $\dot{m}_{WS}$, p_{drop}, pressure field and x-velocity field. In this prediction, none of the other fields diverged too much, but the L1 error on the $\dot{m}_{LD1}$ increased by more than 19000%. To further investigate this, the fields were plotted in Figure 7.21.

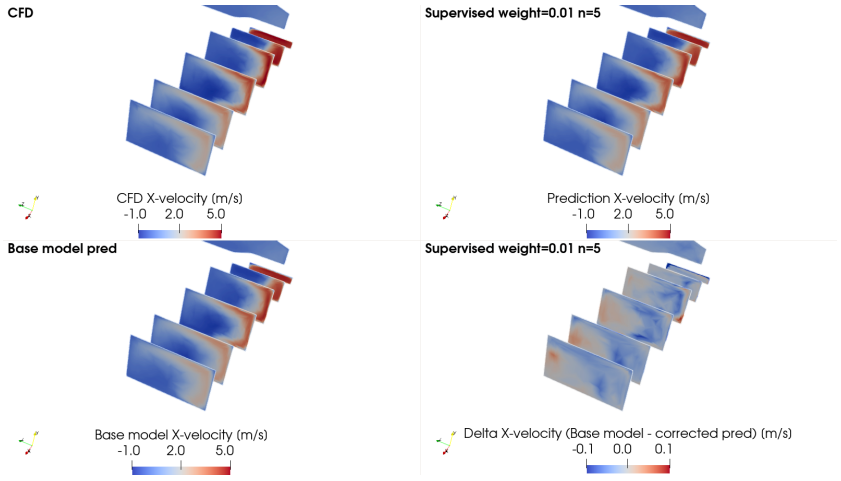


Figure 7.21: Velocity field, comparison between CFD, base model prediction and prediction correction model in Supervised 2.

The correction effect is not visible directly on the field, even though the x-velocity L1 errors decreased on average by 1.18%. Looking at the difference in delta between the base model prediction and the prediction of the Supervised 2 model, there are some subtle differences, where the high-velocity zones decrease in magnitude and the low-velocity zones increase in magnitude for the corrected sample. These are perhaps signs of the diffusive character of the loss.

If large weights are applied to the PI loss, the model rapidly diverges from the original solution, probably because of the difference in the loss magnitude between the field loss and the divergence loss. To balance the losses, a sufficiently small weight has to be applied to the PI loss to equalise their magnitudes to the same order of magnitude. If the model is left to train with a high loss on the PI loss, the model has a diffusive effect on the predictions, as can be seen in Figure 7.22.

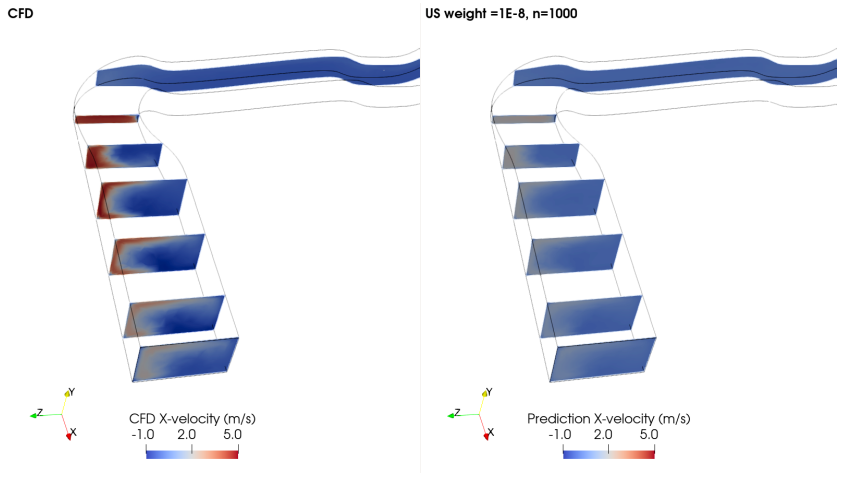


Figure 7.22: X-velocity field comparison between CFD and PI loss model prediction, unsupervised training, $w_{L_{PDE}} = 1E-8$, $n = 1000$.

Figure 7.22 shows, how the unsupervised model deteriorates the prediction of the velocity. The prediction is smoothed out and almost completely loses any definition of the distinctive velocity patterns visible in CFD and which are relatively well captured by the base model. The high-velocity zones still appear in the prediction, but they are of significantly lower magnitude than the base prediction and the CFD. This shows the importance of fixing the boundary conditions. However, even with an implicit boundary condition fixed by an accordingly selected field loss weights, the model did not provide significant constant benefits in the predictions. This could be attributed to many possible reasons. Firstly, only the continuity equation was used in the loss. Perhaps, solving this single equation is not enough to gain any consistent, explainable benefit in the correction process. Moreover, this method relies on the base model prediction to be of sufficient quality, to accurately convey the boundary condition information to the PI correction model. Perhaps the prediction is not of sufficient quality to deliver that information, and either an additional loss on the boundaries of the domain is needed, or perhaps a fixed constraints of values on the boundary of the domain are necessary.

7.3.5. Physics-informed surrogate models - summary

In general, improved model performance was observed when using physics-informed surrogate models, especially in transfer learning and generalization tasks. In a single-campaign inference task *(exp1)*, adding a simplified solution as an additional input to the model improved the model performance in low data regimes, when the training data set was smaller or had fewer samples from the target campaign. In transfer learning performance, when only one campaign was available to transfer (*exp3*), the addition of a simplified solution provided the largest performance gain. Moreover, the addition of a simplified solution was observed to help predict the impact of certain geometrical features in the flow, but overall field errors were often greater when a simplified solution was added, perhaps due to the dissimilarity of the flow field to the bulk flow in CFD training data.

In the setting of transferring from multiple campaigns to the target campaign, the best performance in field prediction was observed for the simplified solution addition method. The mixed scaling method also outperformed linear scaling when looking at prediction of the pressure drop and pressure field. Moreover, mixed scaling was shown to outperform all other methods when tasked with generalizing

the learned knowledge to a new target campaign. Combining two methods together of the simplified solution addition and linear scaling of training data did not provide a compounding improvement, but performed worse than any of the methods separately in all tasks presented.

Finally, an outlier prediction correction method was investigated in the form of a divergence equation-based loss. It was shown that in both supervised and unsupervised settings, the prediction correction method did not improve the prediction, moreover, it deteriorated its accuracy by diffusing the predicted fields. This is suspected to be the fault of implicit boundary conditions information in the form of base model predictions. Furthermore, signs of the challenging task of loss weighing were observed, which could indicate the need to implement loss weighing to make this type of training more robust.

8

Conclusion

In this work, different methods for improving the NN-based surrogate model inference, TL, and generalization performance were tested on a real-world 3D HVAC use case. This is a unique piece of work in the field, as it is not only one of the few works that attempt this type of surrogate modeling on a 3D domain but also uses a real-world use case based on a set of in-house-made CFD RANS simulations. The initial part of this work focused on establishing a suitable model architecture and training pipeline. Subsequently, the baseline model was trained in a series of different data settings to establish its performance in inference, transfer learning, and generalization in an uninformed state. Subsequently, the same experiments were repeated with a series of methods to establish the effect of the method on the model.

All in all, four main methods were tested: addition of simplified solution as input to the model, linear and mixed scaling of the training data, and governing-equation-based (PDE) losses. In addition, a combined method of adding a simplified solution as input and linear training data scaling was tested.

Overall, the addition of a simplified solution and both scaling methods provided some benefits, especially in low-data regimes, in both single-campaign and multi-campaign training scenarios. In a single-campaign setting, adding a simplified solution as an additional input had a minor positive effect on the accuracy of the predictions of the models at lower training data set sizes. In a setting where only one historical campaign to transfer from (Experiment 3), adding the simplified solution showed the non-negligible performance boost. Finally, in a setting where three historical campaigns are available for transfer learning (exp5), all methods presented provided a benefit over the baseline models, with mixed scaling method showing the largest benefits, as it not only improved both fields and scalar predictions, but it has also improved the correlations achieved in zero-shot predictions. Finally, in none of the experiments did the addition of simplified solution combined with the linear scaling method significantly outperform the best of either of the methods used alone.

Subsequently, the PDE loss based on continuity equations was tested as a tool for the prediction outlier correction. The base model without the loss was used as a donor of the model parameters for the model with the PDE loss, which was retrained with the added PDE loss. The re-trained model was tested in both a supervised and unsupervised setting, where supervision was carried out on the outlier sample predictions from the parameter-donor model. The method was validated by comparing the predicted velocity field divergence to the velocity field divergence from the CFD data. The correction experiments with the data showed some signs of the method having a positive effect, by lowering some L1 errors with respect to the base model predictions, but the improvements were not achieved constantly throughout all scalars within each run.

9

Discussion and future work recommendations

9.1. Discussion

Although the work provided a meaningful input that allowed to answer the research questions, this was a long and structured research process, where errors or misjudged decisions propagate throughout the work. Often, there is not enough time or resources to correct them, as you notice the mistakes so far in the process that backtracking is not feasible. Learning from hindsight, if the same research questions were to be answered anew, a number of changes could be made to this work to improve its quality.

For this work, the model was trained on seven mass flows, but only five out of seven were used in evaluation of the models. This was because the model was never able to capture the variability of those two mass flows that ended up being excluded from model evaluation. To examine this more closely, a posterior convergence study was done on the CFD results, showing that the CFD results for those badly captured mass flows were never stable for the mesh refinement level used for this work. Moreover, the convergence study revealed that although all other scalars used for the training did indeed reach a stable steady state, the residuals did not converge in all cases to the values set as the convergence criteria. This would indicate that the CFD setup could be improved to allow for better convergence. Perhaps, improving the mesh quality or increasing the domain size further downstream of the outlier ducts could allow for better convergence. Moreover, extending the runs could also help to reach a better convergence. It is important to remember that if simulations are run for longer with larger domains, the computational cost will also increase. The main reason for which the CFD setup in this work was designed the way it was, was to balance out the quality of the simulations with the computational cost. Nevertheless, the current training data created during this assignment are still usable if one considers those limitations. Perhaps in future training runs, the models should be limited to only values for which a steady-state, stable solution was obtained. Noisy, badly converged training data could have a negative impact on the training process of the models. Moreover, in the future, if further attempts are made to produce training CFD-based data, it is important to conduct the convergence study before any bigger sets of simulations are run, and while running the simulations, a closer care has to be taken in monitoring the convergence of the simulations. In this work, the CFD setup did not inform the user if a given CFD simulation converged or not, unless the simulation was done by hand step-by-step.

The stochasticity of the model was not taken into account in this work. It is often the case that a model trained multiple times in the same setting with the same data will produce different results due to inherited stochasticity in the training process. This stochasticity is present in couple of forms. For example, in this work, the batch size was set to one, which means that at each iteration, one

sample from the training set was used for the iteration. The order in which the samples are drawn is random. Based on this randomly drawn sample, the optimizer gradients are calculated and used to set the gradient descent direction in the optimization landscape. To evaluate the impact of stochasticity in training on the results, a selected number of training runs for different experiments can be re-run a number of times to evaluate the variations in the final predictions. This could allow one to establish a variability-/randomness-dependent component of the prediction/measurement error of the models, which would be helpful in evaluating the predictions. Moreover, such a model can be later used to form ensemble predictions, perhaps further improving the model predictions, as each model may be learning the same data in a different way, and combining the predictions may yield more accurate results.

In training runs with PDE loss, all experiments were performed using only one outlying sample from the base model. This was done to balance the time and resources available for this work. Ideally, the same experiments could be repeated for at least two more samples to account for the impact that different base prediction can have on the training with the PDE loss. Furthermore, the outlier was selected using the largest average L1 error on the fields. Perhaps instead it would be interesting to select the samples for correction, based on the largest divergence of the velocity field, to see if for such samples the positive corrective impact of the PDE loss would be greater.

9.2. Future work recommendations

The field of physics-informed machine learning is a new field of research with many interesting potential research directions to be taken. Based on the experience gained through performing the presented work, the following aspects could be interesting avenues to look at when considering further work on physic-informed surrogate modeling for CFD simulations.

Firstly, the impact of the flow physics on the performance of different physics-informed methods has to be more closely analyzed. In this work, a very specific case study was used, with internal flow with high viscous force impact, with Reynolds number between *[1500 and 15000]*. The conclusions on the method performance may only apply to similar flows. Experimenting with the presented methods in different use cases would allow one to have a better understanding of the best practices with regard to which methods to use for what use case depending on the particular flow physics.

In this work, PDE loss did not provide any consistent benefit in the outlier correction setting. There are several possible reasons why this could have been the case. Firstly, implicit boundary condition information, coming from the training supervision on the baseline model outlier prediction data, could be a downside of the method. Perhaps the boundary condition information was not captured correctly in this sample, or the information was not easily captured by the model, leading to ineffective boundary condition fixing and, moreover, diffusive impact on the loss on the predicted corrected fields. Experimenting with either an additional loss to penalize non-adherence to boundary conditions or some form of hard constraint on the boundary conditions could be an interesting direction of research that could potentially improve the method performance. Moreover, using only part of the Navier-Stokes equations in the loss could be another reason for such diffusive behavior of the loss. Experimentation with PDE loss consisting of the full Navier-Stokes equations could be an interesting direction of research on this type of application of the PDE loss.

Those are just the main possible future work directions, and hopefully the process of reading this work can inspire further directions of potential research, allowing for this research area to further develop.

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, and Google Research. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. 2015. URL `www.tensorflow.org`.

Jassem Abbasi and Pål Østebø Andersen. Physical Activation Functions (PAFs): An Approach for More Efficient Induction of Physics into Physics-Informed Neural Networks (PINNs). 5 2022. doi: 10.48550/arxiv.2205.14630. URL `https://arxiv.org/abs/2205.14630v1`.

ANSYS Inc. ANSYS Fluent Theory Guide, 11 2013.

Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of Aerodynamic Flow Fields Using Convolutional Neural Networks. *arXiv*, 2019.

Michael M. Bronstein, Joan Bruna, Yann Lecun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 11 2016. ISSN 10535888. doi: 10.1109/msp.2017.2693418. URL `https://arxiv.org/abs/1611.08097v2`.

Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine Learning for Fluid Mechanics. *arXiv preprint*, 2020. doi: https://doi.org/10.48550/arXiv.1905.11075. URL `www.annualreviews.org`.

Shengze Cai, Zhiping Mao, Zhicheng Wang, · Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, pages ppb–ppb, 2021a. doi: 10.1007/s10409-021-0xxxx-x. URL `https://doi.org/10.1007/s10409-021-0xxxx-x`.

Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-Informed Neural Networks for Heat Transfer Problems. 2021b. doi: 10.1115/1.4050542. URL `http://asmedigitalcollection.asme.org/heattransfer/article-pdf/143/6/060801/6688635/ht_143_06_060801.pdf`.

Xi Hang Cao, Ivan Stojkovic, and Zoran Obradovic. A robust data scaling algorithm to improve classification accuracies in biomedical data. *BMC Bioinformatics*, 17(1):359, 9 2016. ISSN 14712105. doi: 10.1186/S12859-016-1236-X. URL `/pmc/articles/PMC5016890//pmc/articles/PMC5016890/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC5016890/`.

Xinhai Chen, Chunye Gong, Qian Wan, Liang Deng, Yunbo Wan, Yang Liu, Bo Chen, and Jie Liu. Transfer learning for deep neural network-based partial differential equations solving. *Advances in Aerodynamics*, 3(1):1–14, 12 2021. ISSN 25246992. doi: 10.1186/S42774-021-00094-7/FIGURES/8. URL `https://aia.springeropen.com/articles/10.1186/s42774-021-00094-7`.

Wenyuan Dai, Qiang Yang, Gui Rong Xue, and Yong Yu. Boosting for transfer learning. *ACM International Conference Proceeding Series*, 227:193–200, 2007. doi: 10.1145/1273496.1273521.

Filipe De Avila Belbute-Peres, Thomas D Economon, and † J Zico Kolter. Combining Differentiable PDE Solvers and Graph Neural Networks for Fluid Flow Prediction. 2020.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. 2017. URL https://github.com/mdeff/cnn_graph.

Hamidreza Eivazi, Mojtaba Tahani, Philipp Schlatter, and Ricardo Vinuesa. PHYSICS-INFORMED NEURAL NETWORKS FOR SOLVING REYNOLDS-AVERAGED NAVIER-STOKES EQUATIONS. *Computer Methods in Applied Mechanics and Engineering*, 390, 2022. doi: https://doi.org/10.1016/j.cma.2021.114502. URL https://www.sciencedirect.com/science/article/abs/pii/S0045782521007076.

Junxi Feng, Xiaohai He, Qizhi Teng, Chao Ren, Honggang Chen, and Yang Li. Reconstruction of porous media from extremely limited information using conditional generative adversarial networks. *Physical Review E*, 100(3), 9 2019. ISSN 24700053. doi: 10.1103/PHYSREVE.100.033308.

Han Gao, Luning Sun, and Jian Xun Wang. PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. *Journal of Computational Physics*, 428:110079, 3 2021a. ISSN 10902716. doi: 10.1016/J.JCP.2020.110079. URL www.elsevier.com/locate/jcp.

Han Gao, Matthew J Zahr, and Jian-Xun Wang. Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems. 2021b.

Aurelien Geron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly, 2nd edition, 2019.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. 2010. URL http://www.iro.umontreal.

Atılım Güne¸, Güne¸s Baydin, Barak A Pearlmutter, and Jeffrey Mark Siskind. Automatic Differentiation in Machine Learning: a Survey. 2018. URL https://chainer.org/.

Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional Neural Networks for Steady Flow Approximation. In *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 481–490, 2016. ISBN 9781450342322. doi: 10.1145/2939672.2939738. URL http://dx.doi.org/10.1145/2939672.2939738.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8.

Katarzyna Janocha and Wojciech Marian Czarnecki. On Loss Functions for Deep Neural Networks in Classification. *Schedae Informaticae*, 25:49–59, 2 2017. ISSN 20838476. doi: 10.48550/arxiv.1702.05659. URL https://arxiv.org/abs/1702.05659v1.

Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 426:109951, 2 2021. ISSN 10902716. doi: 10.1016/J.JCP.2020.109951. URL www.elsevier.com/locate/jcp.

George Em Karniadakis. DeepOnet: Learning nonlinear operators based on the universal approximation theorem of operators., 9 2020. URL https://www.youtube.com/watch?v=1bS0q0RkoH0.

Ali Kashefi and Tapan Mukerji. Physics-Informed PointNet: A Deep Learning Solver for Steady-State Incompressible Flows and Thermal Fields on Multiple Sets of Irregular Geometries. *arXiv preprint*, 2022. doi: https://doi.org/10.48550/arXiv.2202.05476. URL https://arxiv.org/abs/2202.05476.

Ali Kashefi, Davis Rempe, and Leonidas J Guibas. A Point-Cloud Deep Learning Framework for Prediction of Fluid Flow Fields on Irregular Geometries. *Physics of Fluids*, 33(2), 2021. doi: https://doi.org/10.1063/5.0033376.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 12 2014. doi: 10.48550/arxiv.1412.6980. URL https://arxiv.org/abs/1412.6980v9.

Georgios Kissas, Yibo Yang, Eileen Hwuang, Walter R. Witschey, John A. Detre, and Paris Perdikaris. Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 358, 1 2020. ISSN 00457825. doi: 10.1016/J.CMA.2019.112623. URL www.sciencedirect.comwww.elsevier.com/locate/cma.

Jichao Li, Xiaosong Du, and Joaquim R R A Martins. Machine Learning in Aerodynamic Shape Optimization. *arXiv*, 2022. doi: https://doi.org/10.48550/arXiv.2202.07141. URL https://arxiv.org/pdf/2202.07141.pdf.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming 1989 45:1*, 45(1):503–528, 8 1989. ISSN 1436-4646. doi: 10.1007/BF01589116. URL https://link.springer.com/article/10.1007/BF01589116.

Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A Survey of Deep Neural Network Architectures and Their Applications. *Neurocomputing*, 234, 2017.

L U Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DEEPXDE: A DEEP LEARNING LIBRARY FOR SOLVING DIFFERENTIAL EQUATIONS. *arXiv*, 2020. URL https://arxiv.org/abs/1907.04502.

Hao Ma, Yuxuan Zhang, Nils Thuerey, Xiangyu Hu, and Oskar J Haidn. Physics-driven Learning of the Steady Navier-Stokes Equations using Deep Convolutional Neural Networks. *arXiv preprint*, 2021. doi: https://doi.org/10.48550/arXiv.2106.09301Focustolearnmore. URL https://arxiv.org/pdf/2106.09301.pdf.

Mike MacKenzie. Artificial Intelligence & AI & Machine Learning, 2018. URL https://www.flickr.com/photos/mikemacmarketing/30212411048.

Zhiping Mao, Ameya D Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. 2019. doi: 10.1016/j.cma.2019.112789. URL www.sciencedirect.comwww.elsevier.com/locate/cma.

Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on Riemannian manifolds. 1 2015. URL http://arxiv.org/abs/1501.06297.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodoì, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017. doi: https://doi.org/10.48550/arXiv.1611.08402. URL https://arxiv.org/abs/1611.08402v3.

Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civil and Infrastructure Engineering*, 36(8):962–977, 4 2021. doi: 10.1111/mice.12685. URL http://arxiv.org/abs/2104.12325http://dx.doi.org/10.1111/mice.12685.

Keiron O'shea and Ryan Nash. An Introduction to Convolutional Neural Networks. 2015. URL https://arxiv.org/abs/1511.08458.

Francisco Palacios, Michael R. Colonno, Aniket C. Aranake, Alejandro Campos, Sean R. Copeland, Thomas D. Economon, Amrita K. Lonkar, Trent W. Lukaczyk, Thomas W.R. Taylor, and Juan J. Alonso. Stanford University Unstructured (SU2): An open-source integrated computational environment for multi-physics simulation and design. *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition 2013*, 2013. doi: 10.2514/6.2013-287.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. ISSN 10414347. doi: 10.1109/TKDE.2009.191.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1 1964. ISSN 0041-5553. doi: 10.1016/0041-5553(64)90137-5.

William H Press, Saul a Teukolsky, William T Vetterling, and Brian P Flannery. Numerical Recipes 3rd Edition: The Art of Scientific Computing. *Sample page from NUMBERICAL RECIPES IN C*, 1:1262, 2007. ISSN 00361445. URL https://books.google.com/books/about/Numerical_Recipes_3rd_Edition.html?hl=de&id=1aAOdzK3FegC.

Benny Prijono. Student Notes: Convolutional Neural Networks (CNN) Introduction – Belajar Pembelajaran Mesin Indonesia, 3 2018. URL https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:77–85, 12 2016. doi: 10.48550/arxiv.1612.00593. URL https://arxiv.org/abs/1612.00593v2.

Rail. Volvo V60 | 3D CAD Model Library | GrabCAD, 8 2021. URL https://grabcad.com/library/volvo-v60-1.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2 2019. ISSN 10902716. doi: 10.1016/J.JCP.2018.10.045. URL www.elsevier.com/locate/jcp.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv preprint*, 2017. URL https://arxiv.org/abs/1711.10561.

Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science (New York, N.Y.)*, 367(6481):1026, 2 2020. ISSN 10959203. doi: 10.1126/SCIENCE.AAW4741. URL /pmc/articles/PMC7219083//pmc/articles/PMC7219083/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC7219083/.

D E Rumelhart, James L McClelland, and C Asanuma. Parallel Distributed Processing: Foundations. *Parallel distributed processing: explorations in the microstructure of cognition*, pages 45 – 76, 1986. URL https://books.google.co.uk/books?id=ktLhoQEACAAJ.

Dinggang Shen, Guorong Wu, and Heung Il Suk. Deep Learning in Medical Image Analysis. *Annual review of biomedical engineering*, 19:221, 6 2017. ISSN 15454274. doi: 10.1146/ANNUREV-BIOENG-071516-044442. URL /pmc/articles/PMC5479722//pmc/articles/PMC5479722/?report=abstracthttps://www-ncbi-nlm-nih-gov.tudelft.idm.oclc.org/pmc/articles/PMC5479722/.

Connor Shorten, Taghi M. Khoshgoftaar, and Borko Furht. Deep Learning applications for COVID-19. *Journal of Big Data 2021 8:1*, 8(1):1–54, 1 2021. ISSN 2196-1115. doi: 10.1186/S40537-020-00392-9. URL https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00392-9.

Kunihiko Taira, Steven L. Brunton, Scott T.M. Dawson, Clarence W. Rowley, Tim Colonius, Beverley J. McKeon, Oliver T. Schmidt, Stanislav Gordeyev, Vassilios Theofilis, and Lawrence S. Ukeiley. Modal Analysis of Fluid Flows: An Overview. *AIAA Journal*, 55(12):4013–4041, 2 2017. ISSN 1533385X. doi: 10.48550/arxiv.1702.01453. URL https://arxiv.org/abs/1702.01453v2.

N Thuerey, K Weißenow, L Prantl, and Xiangyu Hu. Deep Learning Methods for Reynolds-Averaged Navier-Stokes Simulations of Airfoil Flows. *arXiv preprint*, 2020. URL https://github.com/thunil/.

Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. reCAPTCHA: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 9 2008. ISSN 00368075. doi: 10.1126/SCIENCE.1160379.

Sifan Wang, Yujun Teng, and Paris Perdikaris. UNDERSTANDING AND MITIGATING GRADIENT PATHOLOGIES IN PHYSICS-INFORMED NEURAL NETWORKS A PREPRINT. 2020. URL https://github.com/PredictiveIntelligenceLab/GradientPathologiesPINNs.

Tianyang Wang, Jun Huan, Baidu Research, and Bo Li. Data Dropout: Optimizing Training Data for Convolutional Neural Networks. *ArXiv*, 2018a.

Tianyang Wang, Jun Huan, Baidu Research, and Michelle Zhu. Instance-based Deep Transfer Learning. *arXiv*, 2018b. URL `https://arxiv.org/pdf/1809.02776.pdf`.

Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. 2017.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems*, 27(NIPS 14), 2014.

M. M. Zdravkovich and P. W. Bearman. Flow Around Circular Cylinders—Volume 1: Fundamentals. *Journal of Fluids Engineering*, 120(1):216–216, 3 1998. ISSN 0098-2202. doi: 10.1115/1.2819655.

Min Zhu, Handi Zhang, Anran Jiao, George Em Karniadakis, and Lu Lu. Reliable extrapolation of deep neural operators informed by physics or sparse observations. *arXiv*, 12 2022.

A

Appendix CFD convergence study

The models used in this work learn the CFD data and can only be as good at predicting the new results as the training CFD data itself. This is why checking the quality of the training data is important. To do so, a convergence study was performed by simulating a selected geometry from the validation set of campaign 4 at five mesh refinement levels. The goal of this was to see if the solution is independent of the mesh settings, and to see if, and if yes, at which point the values of interest converge to a stable value. This was done in the last phase of the project, rather than at the beginning, as designing the mesh and the CFD setup was not an objective of this work, but rather an inherited part of the project. Poor performance on the $\dot{m}_{LF}$ and $\dot{m}_{RF}$ scalars was the motivation for this study. The details of the mesh refinement levels are presented in Table A.1. Refinement level 4 presents the mesh used in the study for data generation. After each simulation, the average y^+ was extracted for the mesh, along with the evolution of the quantities of interest with the progress of the simulation.

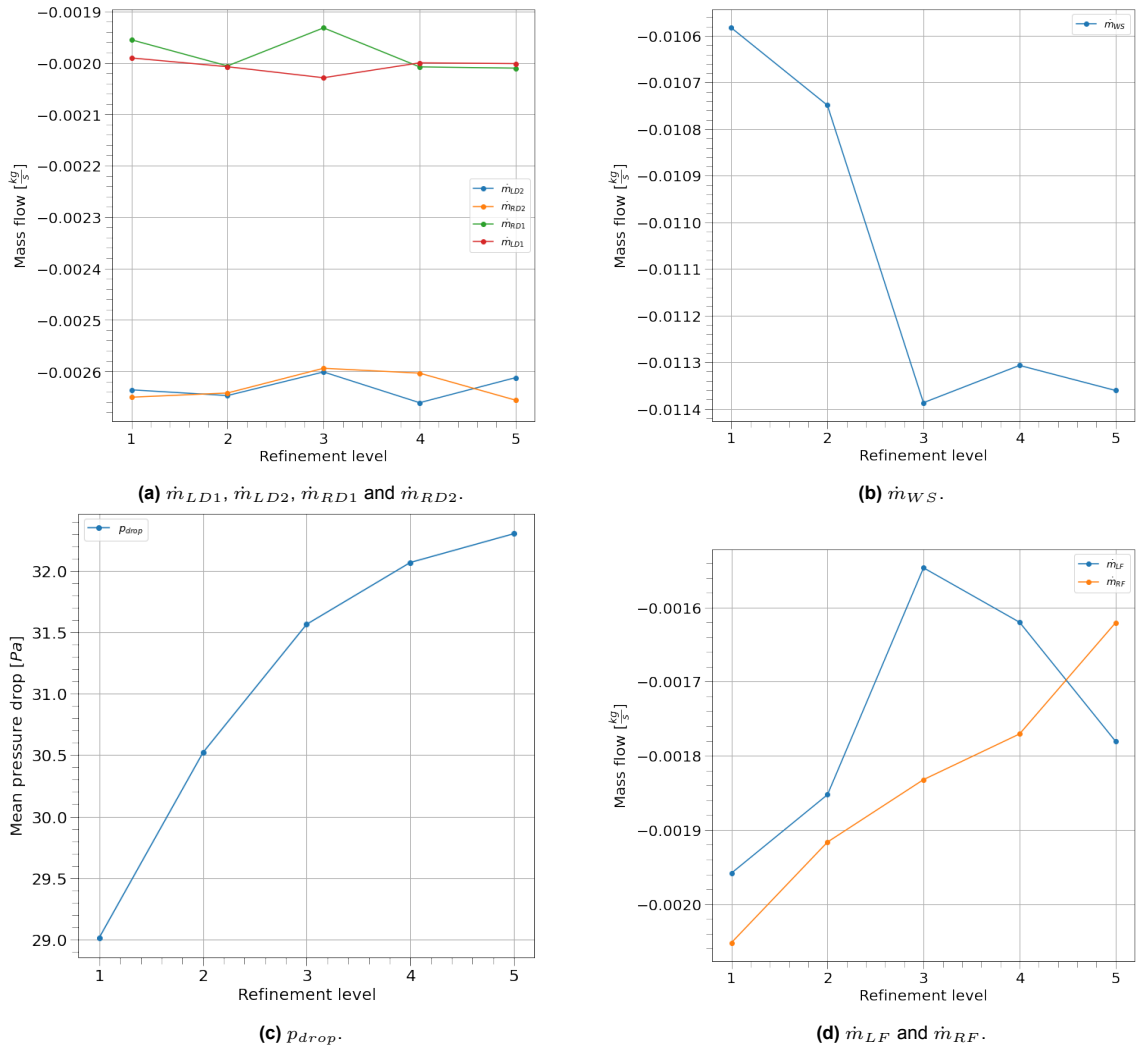
Table A.1: Mesh refinement levels for the convergence study - 4 was the mesh setting used for this work.

Refinement Level	Number of cells	Average y^+
1	240902	3.90
2	532793	2.11
3	1484677	1.05
4	2609012	0.66
5	3627306	0.49

The plots of the quantities of interest vs. the mesh refinement levels were plotted in Figure A.1. For $\dot{m}_{LD1}$, $\dot{m}_{LD2}$, $\dot{m}_{RD1}$ and $\dot{m}_{RD2}$ the values of the mass flows are stable, varying little between each simulation. This was deemed acceptable, as for those particular values, the solution is independent of the mesh for the refinement levels tried. For $\dot{m}_{ws}$ a converging trend was observed with the mesh refinement level, and by the mesh refinement level 3 the values obtained were stable and independent of the mesh. The converging trend for p_{drop} was also observed, but a fully stable and independent value was not obtained for no consecutive mesh refinement levels. Nevertheless, a very steady converging trend, close to a plateau, was observed, deeming the refinement level 4 usable due to the trend shown in the data. Finally, for $\dot{m}_{LF}$ and $\dot{m}_{RF}$ no converging trend was observed. The $\dot{m}_{RF}$ value increases almost linearly with the mesh refinement level, while $\dot{m}_{LF}$ shows signs of instability. These values cannot be treated as mesh independent. In order to reach this state, the CFD setup would have to be improved.

In summary,the mesh refinement level 4, which was the mesh used for data generation, for this particular sample showed that the quantities of interest $\dot{m}_{LD1}$, $\dot{m}_{LD2}$, $\dot{m}_{RD1}$, $\dot{m}_{RD2}$, $\dot{m}_{ws}$ have reached mesh independence by refinement level 4. For p_{drop} mesh independence is not fully reached, but it

is close, and can be conditionally accepted for the purpose of this study, taking into consideration the limited resources and time available for this work. Finally, $\dot{m}_{LF}$ and $\dot{m}_{ws}$ did not show mesh independence.



(a) $\dot{m}_{LD1}$, $\dot{m}_{LD2}$, $\dot{m}_{RD1}$ and $\dot{m}_{RD2}$.

(b) $\dot{m}_{WS}$.

(c) p_{drop}.

(d) $\dot{m}_{LF}$ and $\dot{m}_{RF}$.

Figure A.1: Tracking of the quantities of interest for different levels of mesh refinement.

B

Simplified solver solution as an additional input - method explanation

For the purpose of this work potential flow solver was used as a simplified solver.

Potential flow exists under the assumption that the flow is irrotational and inviscid.

$$\nabla \times \mathbf{V} = \omega = 0 \tag{B.1}$$

The velocity potential ϕ can be defined as:

$$\mathbf{V} = \nabla\phi \; or \; u = \frac{\partial\phi}{\partial x} \; v = \frac{\partial\phi}{\partial y} \; w = \frac{\partial\phi}{\partial z} \tag{B.2}$$

$$\nabla \cdot \mathbf{V} = 0 \tag{B.3}$$

Applying the continuity equation (Equation B.3) to Equation B.2, reduces to Laplace's equation for the velocity potential:

$$\nabla^2\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2} = 0 \tag{B.4}$$

Potential flow can be a valid approximation of fluid flow, for example, for subsonic flows outside of boundary layers, where viscosity can be neglected. The flow inside an HVAC is not one of those flows, with large wetted area internal flow, where viscous forces have a large effect on the flow. The potential flow solution for a flow inside a HVAC is much different from a viscous CFD solution as shown in Figure B.1.

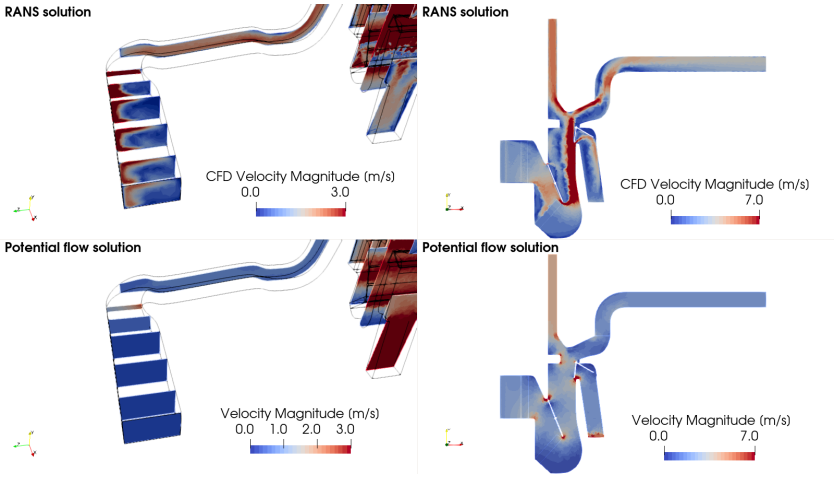


Figure B.1: Potential flow vs. viscous CFD solution for HVAC geometry.

As visible in Figure B.1, the flows vary largely, between the viscous and rotational steady-state RANS CFD simulation and the potential flow simulation. The flow is much more uniform in the potential flow simulation with no vorticity and viscous effects affecting the flow, whereas the CFD simulation flow is much more varied. This is especially visible in the HVAC box area, where the flow turns around the first set of doors that direct the flow through the heat exchanger.

Despite this difference, the potential flow solution can still be a source of valuable information. Firstly, it is highly affected by geometrical variations in the cross section of the ducts. Additionally, although the flow within the middle of the ducts is almost never inviscid, it is relatively close to the values in the CFD simulations. A more complex solution could have been used as the input field, like the laminar flow solution, but the key reason for using potential flow is its ease of calculation and very fast simulation runtime. A more complex solution could provide a better final result, but the additional runtime would make the entire process of creating a surrogate model much more computationally expensive. This would make this ML-based approach to getting flow predictions much less desirable when compared to classical FVM methods.

For the purpose of this project, potential solutions were computed for all samples in the data set using ANSYS Fluent. The solution of Equation B.4, is solved where the velocity components are obtained by taking the gradient of the velocity potential. The flowing boundary conditions are implemented [ANSYS Inc, 2013].

The wall non-penetrability boundary condition fixes the velocity normal to the wall to be 0.

$$\left.\frac{\partial \phi}{\partial n}\right|_{wall} = 0 \tag{B.5}$$

Inlet boundary and far-field boundaries are based on the velocity specified by the user.

$$\left.\frac{\partial \phi}{\partial n}\right|_{inlet} = V_{\perp} \tag{B.6}$$

The outlet boundary is specified as having zero velocity potential.

$$\phi = 0 \tag{B.7}$$

This process yields x,y and z-velocity components which are then passed as an additional input field into the model.

On average, the simulation time for the potential flow took 10 seconds and the solution-solving process took an additional minute. This time does not include the geometry creation and meshing

process. The entire process, as used for this work, took around 10 minutes per sample. This means that for 630 geometries, the solution process would take 6300 minutes, which is 105 hours, which is less than 4.5 days. Important to note is that the efficiency of the solution process was not taken into account, and probably the solution time can be decreased further. For example, the solution was calculated on the same mesh quality as the viscous CFD. Decreasing the mesh size would allow for faster computation of the solution without lowering its quality.

C

Campaign details

In order to create campaigns with the parameterized HVAC geometry, the geometrical parameters have to be carefully varied. For a given campaign, the geometrical parameters of the HVAC geometry were divided into 3 groups:

- Fixed parameters - these are the fixed parameters for a given campaign; these typically are the parameters that affect the shape of the HVAC in the global scale - modify it overall boundaries and dimensions, and would be fixed in the design process of the system due to component interfaces.
- Semi-variable parameters - these are the parameters that vary within a limited range chosen from the full parameter range of the given parameter.
- Variable parameters - these are the parameters that can be varied fully in the parameter space, no matter what campaign is used. These are topically minor geometrical changes.

Independent of the campaign, the fixed, semi-variable and variable parameters were kept the same, while the values or ranges within the parameters changed. The division of parameters from Table 4.1 is presented in Table C.1.

Table C.1: HVAC geometrical parameter type division.

Fixed parameters	**Semi-variable parameters**	**Variable parameters**
Right duct length	Right elbow delta x	Right bump 1 x
Left duct length	Left elbow delta x	Left bump 1 x
Right duct vert. position	Right elbow delta y	Right bump 1 y
Left duct vert. position	Left elbow delta y	Left bump 1 y
Scaling factor	Right global height	Right bump 2 y
Right outlet width	Right global width	Left bump 2 y
Right outlet height	Left global height	Right external shape y
Left outlet width	Left global width	Left external shape y
Left outlet height		Right sec. 7 y
		Left sec. 7 y

In the following sections, the fixed and semi-variable parameter values/ranges are reported for each of the campaigns. The variable parameters vary in its full range for each of the campaigns. The range is specified in Table 4.1.

C.1. Campaign 1 parameter ranges

Table C.2: Fixed and semi-variable parameter selection/subrange for campaign 1.

Parameter name	Range min	Range max
Fixed parameters		
Right duct length [mm]	700.00	
Left duct length [mm]	550.00	
Right duct vert. position [mm]	-20.00	
Left duct vert. position [mm]	-20.00	
Scaling factor	0.90	
Right outlet width [mm]	-5.00	
Right outlet height [mm]	-5.00	
Left outlet width [mm]	-5.00	
Left outlet height [mm]	-5.00	
Semi-variable parameters		
Right elbow delta x [mm]	9.78	20.21
Left elbow delta x [mm]	-17.86	-12.10
Right elbow delta y [mm]	161.15	199.02
Left elbow delta y [mm]	81.27	119.02
Right global height [mm]	-2.28	2.31
Right global width [mm]	-9.31	-4.68
Left global height [mm]	-2.29	2.32
Left global width [mm]	-9.28	-4.68

C.2. Campaign 2 parameter ranges

Table C.3: Fixed and semi-variable parameter selection/subrange for campaign 2.

Parameter name	Range min	Range max
Fixed parameters		
Right duct length [mm]	550.00	
Left duct length [mm]	800.00	
Right duct vert. position [mm]	40.00	
Left duct vert. position [mm]	40.00	
Scaling factor	1	
Right outlet width [mm]	-15.00	
Right outlet height [mm]	30.00	
Left outlet width [mm]	-15.00	
Left outlet height [mm]	30.00	
Semi-variable parameters		
Right elbow delta x [mm]	17.83	22.17
Left elbow delta x [mm]	-28.90	-21.10
Right elbow delta y [mm]	85.78	113.22
Left elbow delta y [mm]	165.78	194.22
Right global height [mm]	5.26	8.73
Right global width [mm]	-1.73	1.73
Left global height [mm]	5.26	8.73
Left global width [mm]	-1.73	1.73

C.3. Campaign 3 parameter ranges

Table C.4: Fixed and semi-variable parameter selection/subrange for campaign 3.

Parameter name	Range min	Range max
Fixed parameters		
Right duct length [mm]	800.00	
Left duct length [mm]	800.00	
Right duct vert. position [mm]	80.00	
Left duct vert. position [mm]	80.00	
Scaling factor	1.1	
Right outlet width [mm]	20.00	
Right outlet height [mm]	20.00	
Left outlet width [mm]	20.00	
Left outlet height [mm]	20.00	
Semi-variable parameters		
Right elbow delta x [mm]	-7.33	-2.67
Left elbow delta x [mm]	0.80	9.20
Right elbow delta y [mm]	-5.00	25.31
Left elbow delta y [mm]	34.69	65.30
Right global height [mm]	5.13	8.87
Right global width [mm]	5.13	8.87
Left global height [mm]	5.13	8.87
Left global width [mm]	5.13	8.87

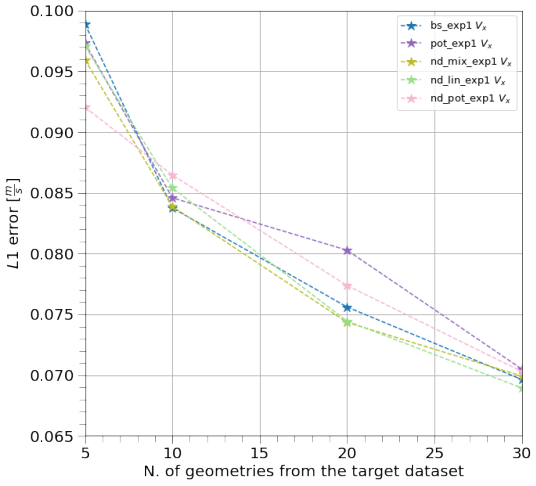
C.4. Campaign 4 parameter ranges

Table C.5: Fixed and semi-variable parameter selection/subrange for campaign 4.

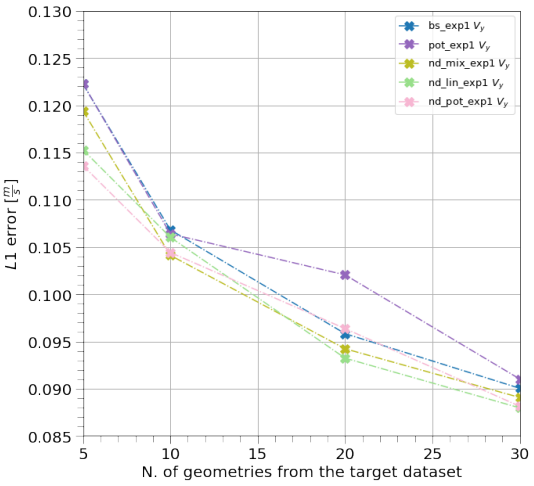
Parameter name	Range min	Range max
Fixed parameters		
Right duct length [mm]	750.36	
Left duct length [mm]	562.76	
Right duct vert. position [mm]	-4.92	
Left duct vert. position [mm]	51.71	
Scaling factor	1.1	
Right outlet width [mm]	11.56	
Right outlet height [mm]	10.59	
Left outlet width [mm]	10.60	
Left outlet height [mm]	11.91	
Semi-variable parameters		
Right elbow delta x [mm]	-17.27	-13.68
Left elbow delta x [mm]	-35.30	-28.82
Right elbow delta y [mm]	54.75	84.24
Left elbow delta y [mm]	30.63	60.12
Right global height [mm]	4.90	7.78
Right global width [mm]	-4.37	-1.50
Left global height [mm]	-4.76	-1.89
Left global width [mm]	-9.44	6.56

D

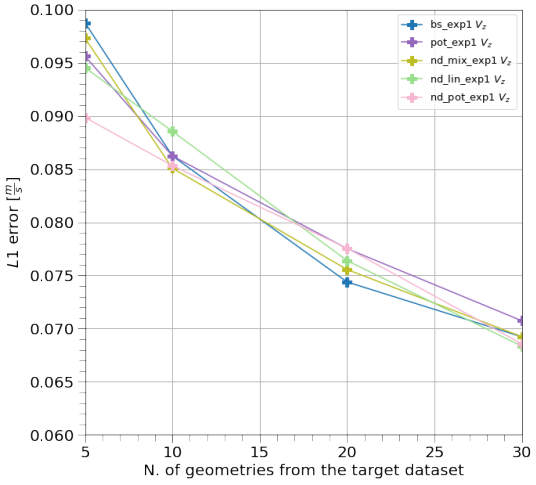
Physics-informed methods comparison of L1 errors on fields.



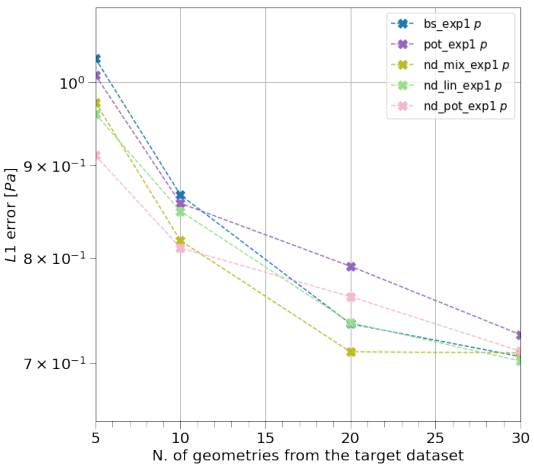
(a) X-velocity.



(b) Y-velocity.

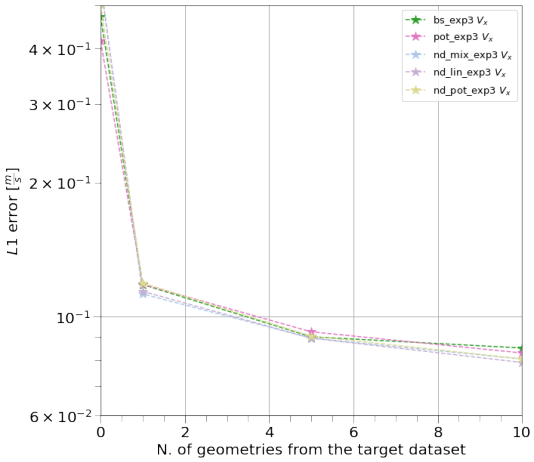


(c) Z-velocity.

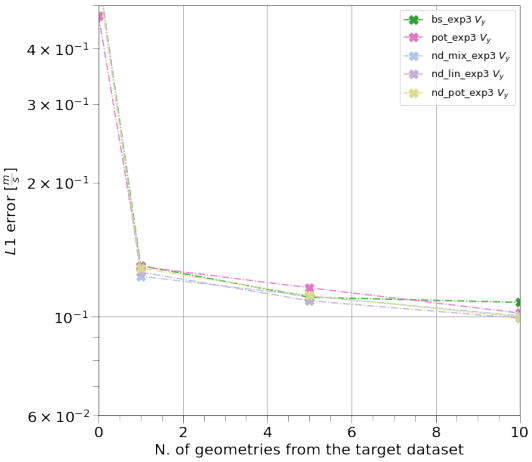


(d) Pressure.

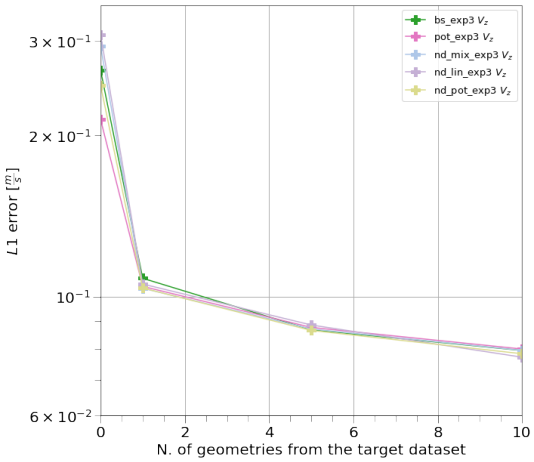
Figure D.1: PI methods comparison of L1 errors on fields for Experiment 1.



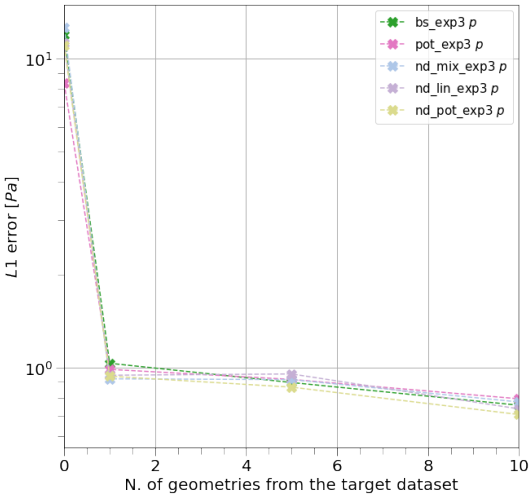
(a) X-velocity.



(b) Y-velocity.

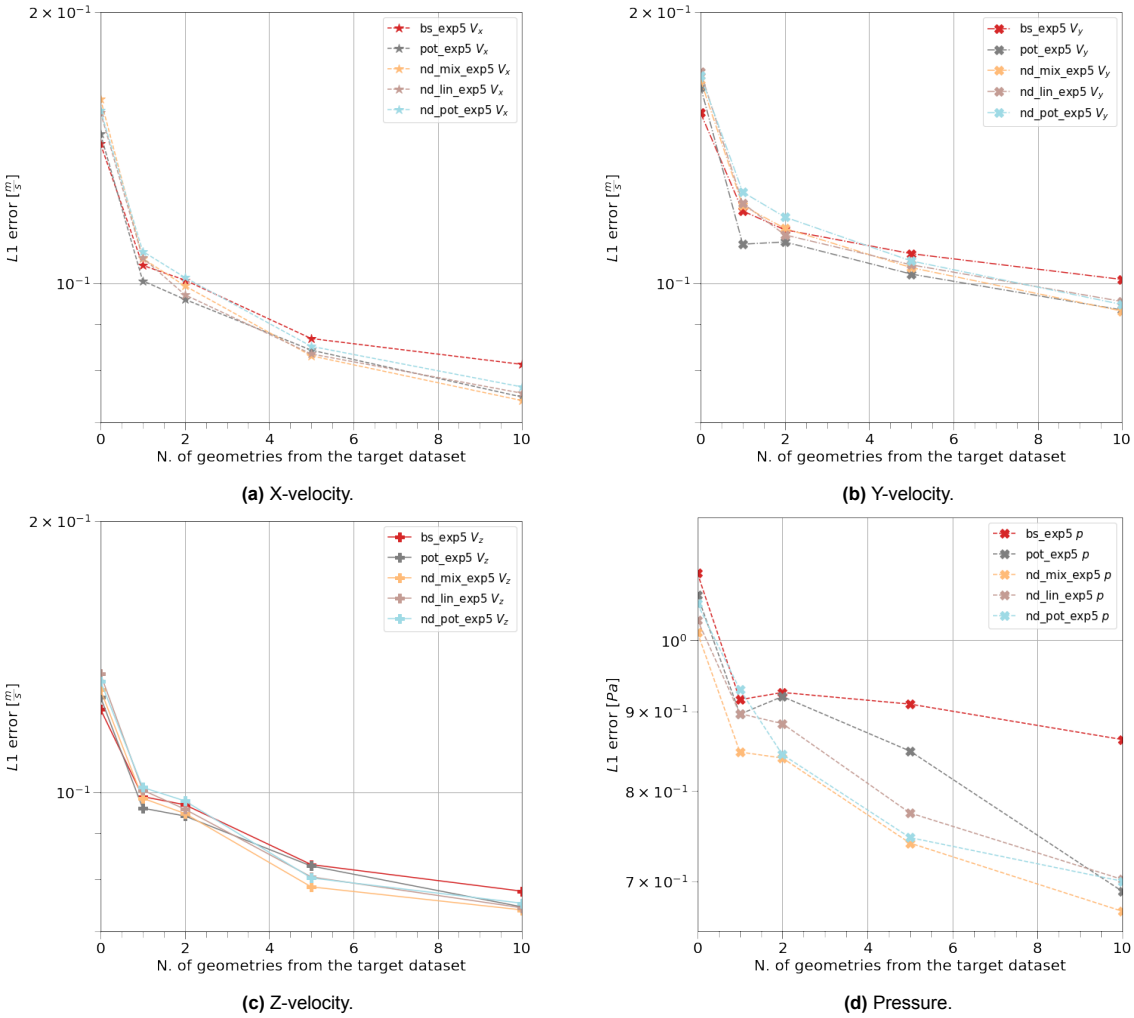


(c) Z-velocity.



(d) Pressure.

Figure D.2: PI methods comparison of L1 errors on fields for Experiment 3.



(a) X-velocity.

(b) Y-velocity.

(c) Z-velocity.

(d) Pressure.

Figure D.3: PI methods comparison of L1 errors on fields for Experiment 5.

E

Full PDE loss correction results

Table E.1: Physics-informed loss, experiment results including intermediate results. Relative improvement on L1 error for scalars calculated from fields and fields themselves.

Intermediate prediction			Scalars from fields						Fields			
Name/mode	w_{LPDE}	N_{iter}	$\dot{m}_{LD1}$	$\dot{m}_{LD2}$	$\dot{m}_{RD1}$	$\dot{m}_{RD2}$	$\dot{m}_{WS}$	$_{drop}$	Pressure	X-velocity	Y-velocity	Z-velocity
Supervised 1	1E-1	5	4444.41%	38.87%	28.38%	29.54%	**-21.43%**	50.23%	0.00%	0.82%	**-2.06%**	0.09%
Supervised 1	1E-1	50	104181.00%	211.88%	37.51%	265.07%	250.26%	**-10.57%**	17.87%	54.49%	26.96%	3.92%
Supervised 1	1E-1	150	118111.54%	316.11%	12.94%	424.65%	519.48%	**-13.41%**	22.50%	100.63%	79.05%	7.21%
Supervised 1	1E-1	1500	223822.21%	1372.23%	641.89%	2086.42%	2883.01%	238.60%	80.95%	569.74%	463.69%	130.12%
Supervised 2	1E-2	5	19183.13%	10.05%	68.19%	**-17.70%**	**-33.54%**	**-51.49%**	**-3.78%**	**-1.18%**	0.51%	0.04%
Supervised 2	1E-2	50	58524.56%	174.67%	**-13.12%**	211.94%	199.34%	1.98%	19.62%	50.91%	34.26%	4.42%
Supervised 2	1E-2	150	89266.37%	268.28%	**-46.33%**	347.39%	477.00%	19.60%	24.57%	96.49%	89.77%	8.99%
Supervised 2	1E-2	1500	204397.65%	1306.59%	575.92%	1981.40%	2798.98%	175.50%	82.48%	560.34%	477.78%	126.84%
Supervised 3	1E-8	5	11941.38%	36.13%	5.68%	15.73%	28.80%	75.05%	1.11%	0.95%	0.68%	0.17%
Supervised 3	1E-8	50	6307.43%	16.69%	15.91%	**-11.18%**	**-0.78%**	**-5.69%**	1.82%	2.38%	1.71%	0.23%
Supervised 3	1E-8	150	401.59%	24.48%	2.56%	1.01%	5.03%	0.28%	1.00%	1.53%	0.88%	0.11%
Supervised 3	1E-7	500	3541.99%	19.85%	2.73%	**-4.77%**	**-4.26%**	0.98%	0.43%	1.00%	0.61%	**-0.04%**
Supervised 3	1E-6	1000	3379.21%	30.42%	6.88%	1.99%	14.11%	2.39%	0.68%	2.40%	1.10%	0.31%
Supervised 3	1E-5	1500	11338.15%	33.74%	**-5.88%**	0.44%	27.27%	5.21%	2.82%	4.09%	2.68%	0.42%
Supervised 3	1E-5	2000	2077.80%	46.26%	**-14.95%**	6.45%	7.40%	19.65%	8.02%	17.77%	13.13%	1.15%
Unsupervised 1	1E-5	5	14514.10%	14.56%	77.55%	**-16.85%**	14.68%	43.97%	76.06%	**-0.40%**	**-1.02%**	0.21%
Unsupervised 1	1E-5	50	46799.00%	157.39%	**-62.20%**	178.30%	196.50%	97.51%	109.99%	48.31%	32.58%	4.11%
Unsupervised 1	1E-5	150	70342.30%	250.82%	**-60.38%**	315.96%	458.71%	95.41%	108.45%	92.73%	92.45%	6.71%
Unsupervised 1	1E-5	1500	329116.43%	1405.45%	888.47%	2149.76%	2703.82%	87.53%	137.12%	548.44%	544.77%	130.73%
Unsupervised 2	1E-8	5	49653.34%	**-26.76%**	194.39%	**-71.88%**	**-53.31%**	237.55%	114.54%	**-0.78%**	1.70%	1.43%
Unsupervised 2	1E-8	50	18420.46%	89.28%	140.89%	81.29%	154.77%	673.71%	226.41%	38.78%	38.54%	7.01%
Unsupervised 2	1E-8	150	25991.69%	179.79%	188.31%	235.22%	410.22%	724.14%	222.09%	82.39%	72.90%	5.84%
Unsupervised 2	1E-8	1000	88791.00%	1059.44%	173.63%	1618.05%	2264.04%	1364.27%	261.71%	445.54%	407.95%	75.60%