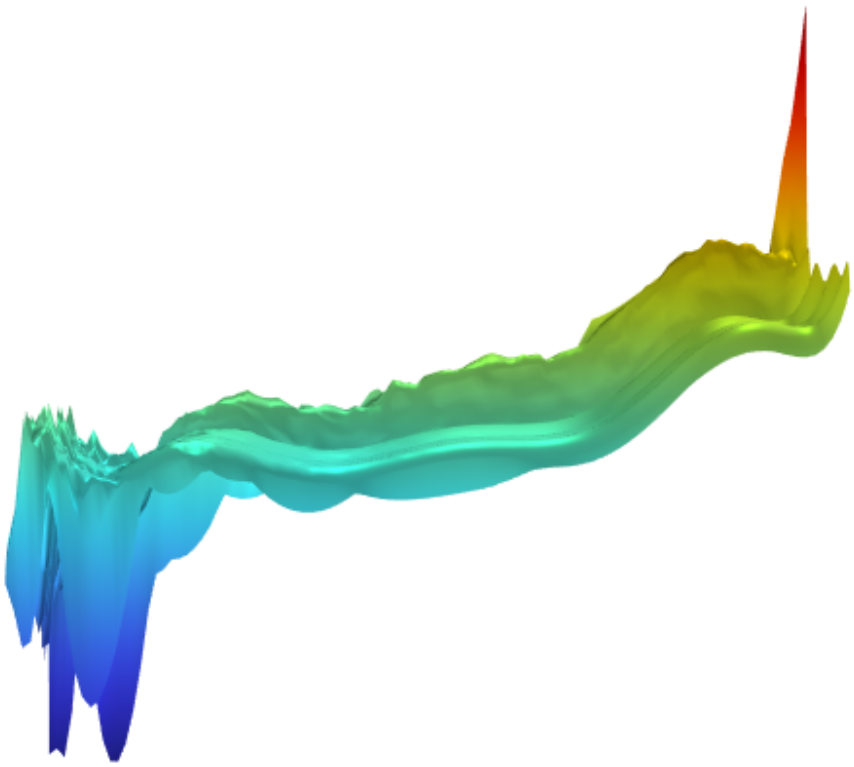
CHALMERS
UNIVERSITY OF TECHNOLOGY



Feasibility of Deep Neural Network Surrogate Models for Simulations of High Voltage Equipment Design

Master's Thesis in Sustainable Electric Power Engineering and Electromobility

JOSEFIN ALBERS
FELIX WOLKE

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's Thesis 2026

Feasibility of Deep Neural Network Surrogate Models for Simulations of High Voltage Equipment Design

JOSEFIN ALBERS
FELIX WOLKE

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Electric Power Engineering
Chalmers University of Technology
Gothenburg, Sweden 2026

Feasibility of Deep Neural Network Surrogate Models for Simulations of High Voltage Equipment Design
JOSEFIN ALBERS, FELIX WOLKE

Supervisors: Olof Hjortstam (Hitachi Energy Research), Yuriy Serdyuk (Department of Electrical Engineering), Thomas Hammarström (Department of Electrical Engineering)
Examiner: Yuriy Serdyuk (Department of Electrical Engineering)

Master's Thesis 2026
Department of Electrical Engineering
Division of Electric Power Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Error surface, DNN solution in comparison with reference dataset

Typeset in LaTeX
Gothenburg, Sweden 2026

Feasibility of Deep Neural Network Surrogate Models for Simulations of High Voltage Equipment Design

JOSEFIN ALBERS, FELIX WOLKE
Department of Electrical Engineering
Chalmers University of Technology

Abstract

With the recent advances in Machine Learning, there is a natural interest in investigating potential application areas where it could contribute to more efficient solutions. This thesis investigates the feasibility of the Deep Neural Network (DNN) approach implemented in the COMSOL Multiphysics software, for simulations of high voltage components by comparing respective results with traditional finite element solutions. The project encompasses a best practice study for DNN model parameter optimization, by considering electrostatic problems for two cases, utilizing different representative geometries. In addition, the result from one of the study cases is further applied for establishing best practices for streamer breakdown detection problems.

The DNN approach was found feasible for applications with relatively low-complexity geometries and simple physics. However, there are areas of study where it did not outperform traditional finite element solutions. Thus, when simulating complex geometries or physics with DNN, the improvement in efficiency compared to using finite element methods diminishes. From the obtained results, it is apparent that further studies in the area are necessary to explore advantages and limitations of the method.

Keywords: DNN, ML, AI, FEM, PCE, GPM, streamer, electrostatics, surrogate

Acknowledgements

We want to thank Olof Hjortstam (supervisor, Hitachi Energy Research), Yuriy Serdyuk (supervisor and examiner, Chalmers) and Thomas Hammarström (supervisor, Chalmers) for their consulting and questioning, continuously ensuring the quality of this thesis. It provided important guidance when entering this relatively new and non-studied field.

Josefin Albers and Felix Wolke, Gothenburg, May 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AI	Artificial Intelligence
DNN	Deep Neural Network
ELU	Exponential Linear Unit
FEM	Finite Element Method
GPM	Gaussian Process Modeling
HV	High Voltage
ML	Machine Learning
NN	Neural Network
PCE	Polynomial Chaos Expansion
PDE	Partial Differential Equation
ReLU	Rectified Linear Unit
SM	Surrogate Model

Contents

List of Figures

List of Tables

1
Introduction

1.1 Background

With the recent proceedings in deep learning, there is a natural interest in finding areas where this technology has the potential to make a change. Today, HV design usually relies on computationally expensive FEM simulations, and implementing ML for these simulations might make the process more efficient, maybe even discovering unsafe conditions missed by human operators.

1.2 Earlier works

While the studies done on ML in conjunction with COMSOL multiphysics may be sparse, studies have been done in the area of physics informed neural networks (PINN). An example area of study is insulator diagnostics, where deep learning in conjunction with imaging has been used to diagnose electric field and temperature distributions [1]. PINN has also shown potential in being able to estimate physical parameters, such as charge density distributions, electrical ion mobility and electrode diameter in a 1-dimensional geometry by being trained on HV measurement data [2].

The idea behind PINN is to set constraints for a NN based on the laws of physics, making them more ideal for solving physics-based problems. While these constraints are not seen in COMSOL multiphysics, the NN can still be used to map multivariate input data to output data, solving problems in its own way.

1.3 Objective

The objective of this thesis is to study ML Surrogate models in COMSOL Multiphysics, establishing a best practice for working with electrostatic problems, based on a study using two simple two-dimensional geometries with circular electrodes surrounded by a grounded rectangular box.

Additionally, the established best practice should be tried on a streamer detection study, and lastly, conclusions should be drawn from the result around possibilities and boundaries for the use of ML in HV simulations.

1.4 Boundaries

- The project uses the built-in ML functionality of COMSOL Multiphysics, not relying on external simulation tools.
- The models are trained using sparse datasets, only obtained from FEM simulations.
- Complex geometries are not studied as part of this thesis, as the focus is an initial study of the use of ML models in COMSOL.
- The parameter optimization for the best practice study, does not take into consideration the interaction effect between different ML parameters.

2

Theory

2.1 Machine Learning

Machine Learning (ML) is a subtopic of *Artificial Intelligence* (AI). Unlike other types of AI, it is based on *learning*, in contrast to rule-based varieties. This essentially means that the framework does not make use of fixed rules or functions, but rather on model training in conjunction with other methods such as *neurons*, for decision-making.

2.1.1 Datasets

A *dataset* is a set of data to be processed through the ML framework. Datasets are commonly split into two or three subsets, in order to enable training. This is commonly known as a *train-test-split*.

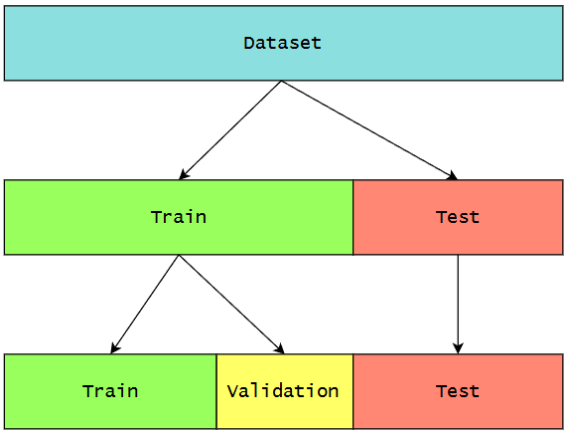
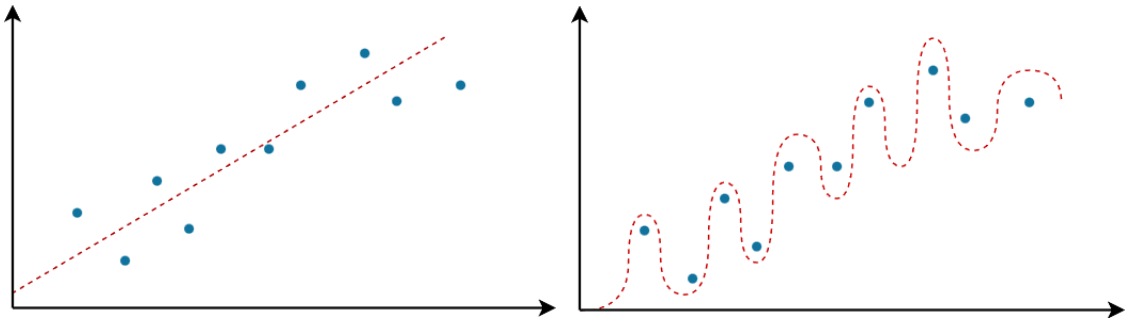


Figure 2.1: The commonly used train-test-split

During initial training phases of ML models, the test data is to be kept unseen from the model. If the three-subset model with a validation set is used, this subset may be introduced to the ML model during the training phase, in order to tune *hyperparameters*, which are crucial elements for dictating how the model works.

2.1.2 Overfitting

Overfitting describes an undesirable case where a model has adapted too well to the training data, to the extent where it makes inaccurate predictions of the test data.



(a) Acceptable linear fit **(b)** Overfitting

Figure 2.2: Example of overfitting in regression

Figure 2.2 shows a human-interpretable example of overfitting. Assuming the blue points depict the training set, adding additional datapoints from the test set to case (a) would still yield an acceptable fit, while in case (b), the regression has over-adapted to each of the existing datapoints from the training set, resulting in excessive errors when new (test) points are taken into consideration.

There are a few common methods of avoiding overfitting in ML, *regularization*, *dropout* and *early stopping* [3].

2.1.2.1 Regularization

Regularization is a way of counteracting overfitting by penalizing large weights in the model. This effectively prevents the model from fitting too well to the training data. *Weight decay*, as used in this thesis, is a form of regularization.

2.1.2.2 Dropout

In some circumstances, the model becomes too reliant on specific neuron pathways. This can be mitigated by making use of *dropout*, a method in which neurons are randomly switched off during training, forcing the model to make use of alternative pathways [3].

2.1.2.3 Early stopping

If the performance of the model against the validation dataset starts decreasing during training, the training can be stopped at the point where this becomes noticeable. This is called *early stopping* [4].

2.1.3 Losses

In ML theory, the *losses* describe the accuracy of model predictions with respect to the actual data.

2.1.3.1 Loss functions

Loss functions are used in order to quantify training and validation losses. There are several metrics used to quantify these losses, a common one (used by COMSOL Multiphysics) being *Root Mean Squared Error* (RMSE).

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n} \frac{(f_i - \hat{f}_i)^2}{n}} \tag{2.1}$$

Apart from quantifying the losses, the functions additionally serve as objective functions when training, and are often minimized to produce the best performing model.

2.1.3.2 Convergence plots

A convergence plot can be drawn by applying a loss function when training a model. Convergence plots can be used to diagnose overfitting, as the validation losses will show a clear upward trend while the training losses continue decreasing, due to the model over-adapting (overfitting) to the training data.

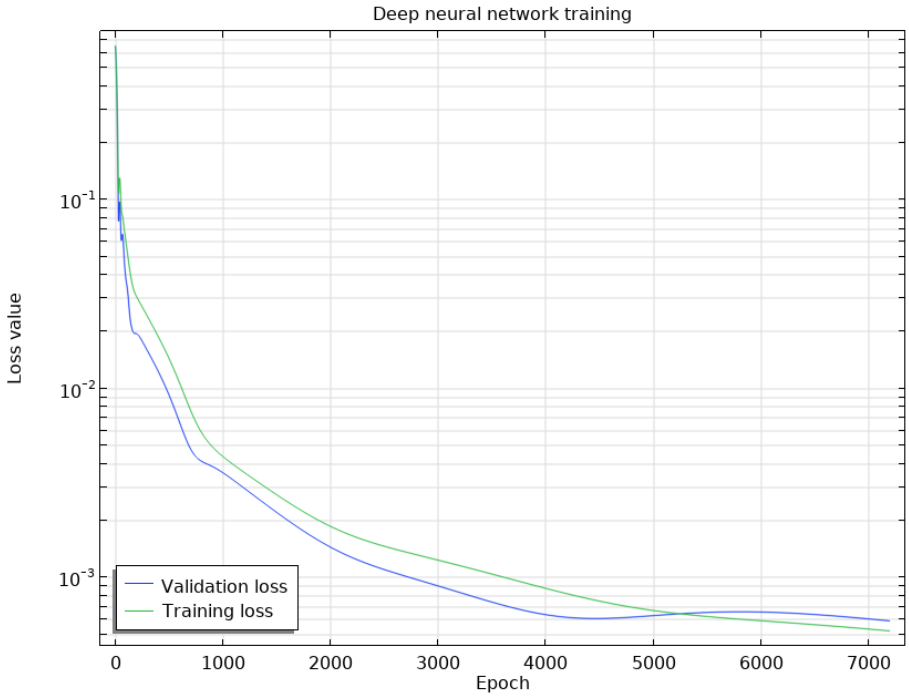


Figure 2.3: Example of a convergence plot of a neural network, showing validation and training losses with respect to the number of epochs trained

2.1.4 Neural Networks

A *Neural Network* (NN) is a ML framework used for decision-making by simulating neurons. It consists of several layers, typically one input layer, one output layer, and one or several hidden layers between them.

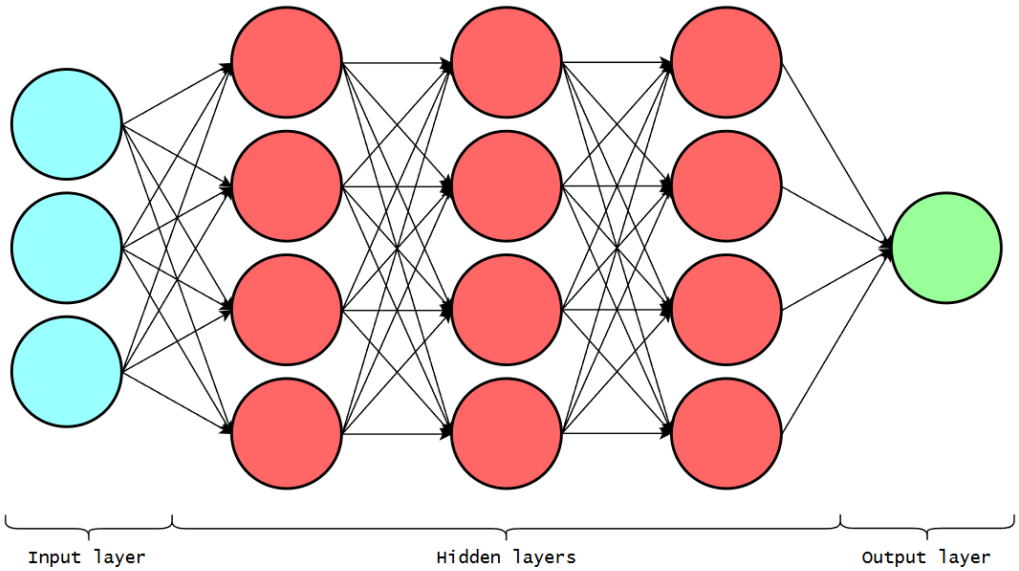


Figure 2.4: Example of a simple five-layer feed forward NN featuring one input layer consisting of three inputs, three fully connected hidden layers with four neurons each, and a single-output layer

2.1.4.1 Structure

There is no singular structure of a NN, as it can be configured in ways appropriate for specific tasks. A feed-forward NN as used in this project, though, consists of a number of *neurons* in parallel, which are collectively called *layers*. These layers are connected serially, forming the path from input to output.

The first layer in this series is called the input layer, as it is the layer that receives the input data. In the same sense, the last layer is called the output layer. The layers in-between are typically referred to as *hidden layers*, as there is no human interaction with them.

In Figure 2.4, the layers are *fully connected*, meaning that each neuron in a layer connects to *all* neurons in the preceding layer.

NNs can have a rectangular shape where all layers have an equal amount of neurons, but the layers can also have different numbers of neurons, thereby constructing NNs of different shapes. They can, as an example, be shaped as a funnel, where the number of neurons decreases with each layer, or like a bottleneck (also called an autoencoder) where one or more of the hidden layers have a smaller number of neurons compared to the rest of the layers.

2.1.4.2 Deep Neural Networks

A *deep neural network* refers to a NN with two or more hidden layers. While there is no well-defined boundary that classifies a NN as a deep neural network, it is common to draw the line at two or more hidden layers.

2.1.4.3 Training

To be able to make meaningful predictions, a NN needs to be trained, or in other words, iterated over n epochs, while updating its weights subject to minimization of the loss function.

2.1.4.4 Activation functions

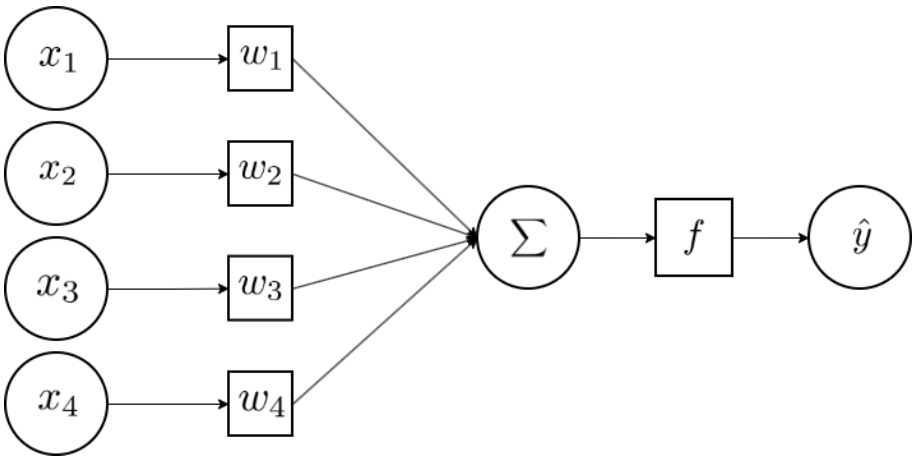


Figure 2.5: Schematic of a perceptron, the ML equivalent of a human neuron

Activation functions can be seen as filters used to pass the summed and weighted inputs to the output, introducing a non-linearity, increasing the ability of the neurons to adapt to training data. Different activation functions have different characteristics.

Table 2.1: Table of common activation functions

Activation function	Function value
Linear	x
ReLU	$\max(0, x)$
ELU	$\begin{cases} x, & x \geq 0 \\ \alpha\left(e^x - 1\right), & x < 0 \end{cases}$
tanh	$\tanh(x)$
Sigmoid	$\frac{1}{1+e^{-x}}$

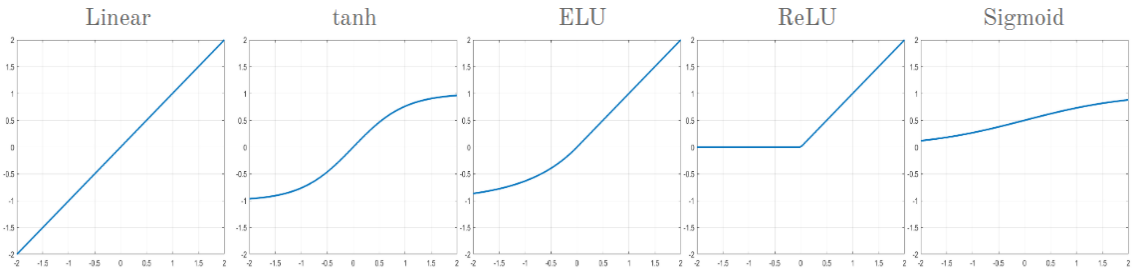


Figure 2.6: Graphs of common activation functions

2.1.5 Gaussian Process

GPM is a method that uses a mean function and a covariance function in conjunction with observations (inputs), yielding an outcome in the form of a predictive function of the observations. It is useful for interpolation of the values given as observations [5].

2.1.6 Polynomial Chaos Expansion

PCE is a method of prediction that works by approximating a relationship to a known polynomial function. It is a good option in cases with nonlinearity, where linearization may provide a bad result, and works best for smooth and continuous functions [6].

2.2 Finite Element Method

The *Finite Element Method* (FEM) is a widely used numerical method for solving complex physical problems. These can rarely be solved analytically due to the complexity from *Partial Differential Equations* (PDE:s) used to describe physical laws. *Discretization* can be used to simplify the problems, which in FEM is done by dividing up the geometry into smaller *finite elements*. Then, by approximating for example the electric field of each finite element, a model can be obtained that accurately describes the field distribution of the whole geometry [7].

2.2.1 Mesh

Discretization is done by introducing *meshing*, which for a 2D model consists of a number of triangles or quadrilaterals, spanning the whole geometry. These polygons represent the finite elements of the domain, and have a *node* in each vertex. Higher-order elements may contain more nodes, located along the edges or inside the element.

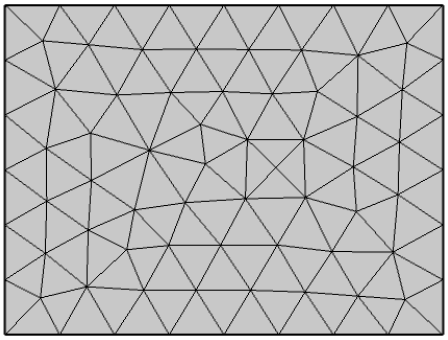


Figure 2.7: Triangular mesh on an example 2D geometry

2.2.2 Approximation of the elements

To approximate the values of the elements, the desired equations are solved for each node. Then using shape functions, the desired value can be interpolated over the element. By using shape functions of different orders, such as a linear shape function $U = ax + by + c$, or a quadratic shape function $U = a_1x^2 + b_1y^2 + a_2x + b_2y + c$, the accuracy of the interpolation can be controlled.

2.3 Streamer discharge

A streamer is a type of electrical discharge where an electron avalanche has ionized the medium around an electrode, allowing charge to travel through the ionized region, forming a tree-shaped discharge. The discharge does not bridge a gap between two electrodes, but rather dissipates into the surrounding media.

In order for a streamer to form, the Raether-Meek criterion, commonly called the streamer criterion, needs to be fulfilled. This is accomplished when the integral of the critical region of the critical path (between two electrodes) evaluates to $\gtrsim 18.3$, or in other words, $\gtrsim \ln(10^8)$.

$$\int_0^{x_c} \alpha_{\text{eff}}\, dx \gtrsim \ln(10^8) \tag{2.2}$$

In Equation 2.2, α_{eff} denotes the effective ionization coefficient. It is defined as $\alpha_{\text{eff}} = \alpha - \eta$, where α (the ionization coefficient) quantifies the rate of secondary electron emission, and the attachment coefficient η quantifies the rate of electron capture by other molecules.

3
Methods

The project has two main parts. The first part is a best practice study, where parameters and settings were explored and optimized. Here, three different ML methods were tested and compared. The second part is a test of the best performing ML method using the resulting parameters of the best practice study on a streamer detection simulation.

3.1 Best practice study setup

This section describes how the best practice study was set up in COMSOL multiphysics. *Introduction to Surrogate Modeling*, from the COMSOL documentation, was used to set up the method of this thesis [8].

3.1.1 Geometries

The ML methods were evaluated on two different two-dimensional geometries.

3.1.1.1 Geometry 1

Geometry 1 is a simple two-dimensional representation of a cylindrical high voltage (HV) conductor with applied voltage V_0, in a grounded quadratic geometry.

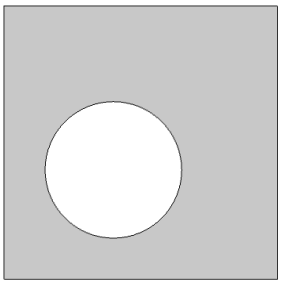


Figure 3.1: Geometry 1

Table 3.1: Parameters for geometry 1

Parameters	Values
V_0 [kV]	100
l_{square} [m]	1
r [m]	Varies
l_x [m]	0.15
l_y [m]	Varies
ϵ_{gas} [-]	1

3.1.1.2 Geometry 2

Geometry 2 is identical to geometry 1, except for the addition of a resistive layer to the bottom of the quadratic geometry. l_y still denotes the distance to ground in the y-axis, rather than to the top boundary of the resistive layer. The external dimensions remain unchanged.

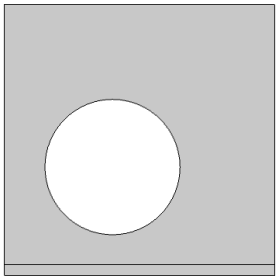


Figure 3.2: Geometry 2

Table 3.2: Additional parameters for geometry 2

Parameters	Values
d_{solid} [m]	0.04
ϵ_{solid} [-]	5

3.1.2 Physics

The study uses the electrostatic physics module, which essentially solves Gauss' law (Equation 3.1) by substitution with Equation 3.2 (derivative of Poisson's equation for electrostatics) on the meshed geometry.

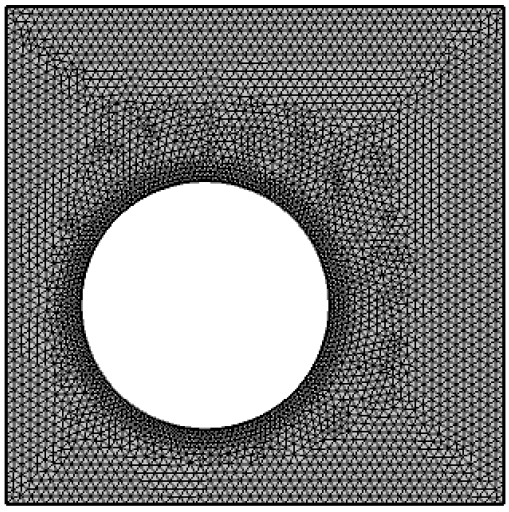
$$\nabla \cdot \mathbf{D} = \rho_v \tag{3.1}$$

$$\mathbf{E} = -\nabla V \tag{3.2}$$

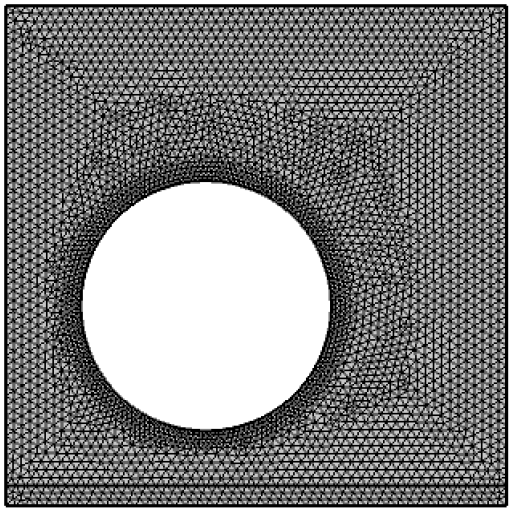
The quantity of interest is $E_{\max}$, the maximum electric field norm along the conductor boundary in the geometrical domain.

3.1.3 Mesh

The mesh preset *extra fine* was used for the entire geometry, apart from the conductor boundary, where the preset *extremely fine* was used.



(a) Mesh of geometry 1



(b) Mesh of geometry 2

Figure 3.3: Meshing of the geomtries

3.1.4 Surrogate Model Training

To generate the training (and validation) datasets for the ML methods, the *Surrogate Model Training* functionality of COMSOL was used.

3.1.4.1 Dataset quality

In order for the ML model to produce a good result, the input data needs to display a clear enough pattern representation of the true solution. To ensure this, the input data was sampled using uniform distributions across the solution domain, or in other words, each sampled datapoint had an equal probability of being anywhere in these distribution regions.

To ensure good data sampling in regions of the domain where the output parameter had a high gradient, with respect to the input parameter, the sampling was split up into two regions, one for the majority of the solution space, and one for the high-gradient regions, by running the surrogate model training data generation twice. The second, improving run of the surrogate model training was done using *Improve and build surrogate model* functionality in COMSOL. 50% of the total number of datapoints was used for each run, and the mathematical distribution of points inside these reinforcement regions was still chosen to be uniform. This resulted in a dataset with a more concentrated sampling in the high-gradient regions.

3.1.4.2 Training dataset parameter intervals

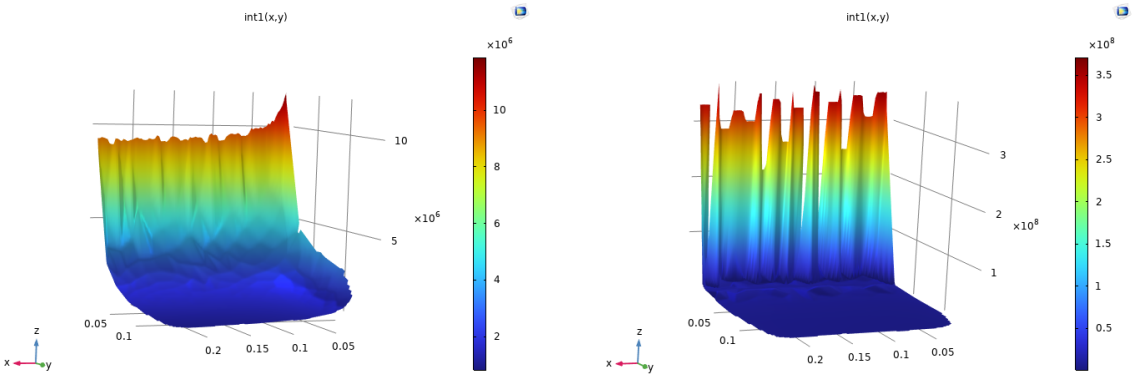
The input parameters for the study are conductor radius r, and distance to ground in the y-axis, ly for both geometries. Their corresponding parameter intervals were set as shown in Table 3.3.

Table 3.3: Interval for input parameters

Input parameter	Geometry 1	Geometry 2
r [m]	0.01 - 0.25	0.01 - 0.25
l_y (first run) [m]	0.0105 - 0.15	0.0002505 - 0.15
l_y (second/improving run) [m]	0.01 - 0.0105	0.00025 - 0.0002505

The output parameter was defined as the maximum electric field norm $E_{\max}$ at the boundary of the conductor.

Figure 3.4, shows the linear interpolation of the 400 point datasets. The x-axis is the radius of the conductor, y-axis is the distance to ground and the z-axis is the maximum electric field.



(a) Linear interpolation of the 400-point dataset for geometry 1

(b) Linear interpolation of the 400-point dataset for geometry 2

Figure 3.4: Linear interpolation of 400-point dataset of each geometry

3.1.5 ML methods

Three different ML methods integrated into COMSOL were tested.

The *Deep Neural Network* functionality was used to train the DNN models, using the datasets generated by the Surrogate Model Training. This was done by specifying a table containing a dataset generated by Surrogate Model Training, along with hyperparameters used for the DNN training.

Similarly to what was done for DNN training, the *Gaussian Process* and *Polynomial Chaos Expansion* functionalities also made use of the same pre-generated training datasets. They differ from DNN, as they have fewer hyperparameters, and the training in itself is substantially quicker. By default, no validation data is used for GPM or PCE training.

3.2 Quantification of results

The following quantification method was used both for the best practice study and for the later streamer detection study. The error, the computational time and the reliability of the model were used to draw conclusions around the optimal parameter settings.

3.2.1 Prediction error of the ML models

A method of calculating the error of the ML models was determined, to facilitate comparisons between different parameter settings. The error calculation was preformed by calculating the volumetric L2 norm between the ML model and a reference dataset consisting of a high number of datapoints, and using a very fine mesh, in order to reach a high accuracy. The reference dataset does not equal reality, but represents using a FEM solution of high resolution.

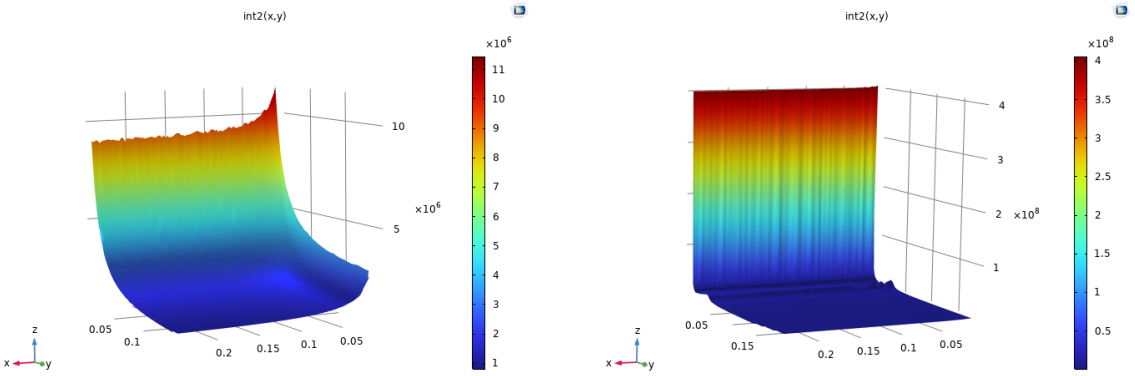
3.2.1.1 Separate reference dataset

In order to define a reference model for each geometry, a separate large reference dataset was generated using *Surrogate Model Training*, sampled at 4000 points in the solution domain and using the same maximum electric field boundary probe on the conductor as the output and r and l_y as the input parameter. The input parameter intervall are specified in Table 3.4.

Table 3.4: Interval for input parameters for reference dataset

Input parameter	Geometry 1	Geometry 2
r [m]	0.01 - 0.25	0.01 - 0.25
l_y (first run) [m]	0.0105 - 0.15	0.00025000000000005 - 0.15
l_y (second/improving run) [m]	0.01 - 0.0105	0.00025 - 0.00025000000000005

Figure 3.5, shows the linear interpolation of each of the two reference datasets, where the x-axis is the radius, y-axis is the distance to ground and the z-axis is the maximum electric field along the conductor boundary.



(a) Linear interpolation of reference dataset for geometry 1

(b) Linear interpolation of reference dataset for geometry 2

Figure 3.5: Linear interpolation of reference dataset for each geometry

The reference dataset for geometry 1 consisted of 4000 datapoints with a first run of 2000 points on the majority of the domain and 2000 improving points concentrated toward high-field regions.

The reference dataset for geometry 2 consisted of 3559 datapoints instead of the intended 4000, due to an error. The first run consisted of 2559 datapoints and the improving run consisted of 1000 datapoints. The lower number of improving points steam from needing a shorter interval in the improving run to capture the much higher gradient of the high field area. Given the relatively large number of datapoints and the fact that only the low-field region was affected, regenerating the dataset because of the error was deemed not necessary.

The mesh of both the geometries was set to extremely fine, in order for the datasets to be generated with high accuracy. To allow comparison with the simulated ML models, the reference models was linearly interpolated into a 2-dimensional surface.

3.2.1.2 Error calculation

In order to quantify the errors of the ML models, the continuous L2 norm (as seen in Equation 3.3) was used. $\hat{f}$ denotes the predicted function (the ML solution) or the interpolation of the training dataset, and f denotes the true function which in this work is the reference model (linear interpolation of the reference dataset). The interpolation was used for both the reference dataset and training dataset when preforming the calculations, since it was used for comparisons of datasets of varying sizes.

$$\|f(x,y) - \hat{f}(x,y)\|_2 = \sqrt{\iint_D (f(x,y) - \hat{f}(x,y))^2 \, dxdy} \tag{3.3}$$

In COMSOL, a surface integral over a Grid 2D dataset with the same parameter bound as presented in Table 3.3 (lower bound from second run and higher bound from first run of l_y) was integrated across the solution domain, using the expression $(f(l_y, r) - \hat{f}(l_y, r))^2$. The square-root calculation was then performed externally. Due to an error, the calculations done for geometry 1 on device 1 uses slightly different parameter bounds of 0.01026-0.249517 for r and 0.010104-0.149683 for l_y, rather than the parameter bounds described in Table 3.3. Since no error comparisons were done between different tests and devices, it was determined to have a negligible impact on the result.

The resulting error from the L2 norm calculation doesn't indicate a good or bad model in itself, since it merely describes the volumetric L2 error between the reference dataset and the ML model as a scalar value. The error calculation was therefore also performed on interpolations of the training datasets, to allow a fixed point of comparison, meaning that conclusions can be drawn about the validity of the ML models, as well as its accuracy compared to interpolation between FEM results.

3.2.2 Computational time

When training a model in COMSOL Multiphysics, the computational time is printed after the training is completed. This can be used as a metric when judging the usability of the ML models. It should, however, be noted that the training time can vary between devices due to different computational power and other processes running at the same time, making the training time have limited precisions as a comparison tool.

The number of epochs could also be a suitable metric since it in most cases is proportional to the training time, with the added benefit that it is constant between devices. There are, however, some cases where DNN parameters change the training time per epoch, making it necessary to use identical parameters when using this metric for inter-device comparisons.

In combination with the training time differences explained earlier, this resulted in each individual parameter test being completed on the same device, while multiple devices were used across the different tests. The aim was to find general trends for each test instead of using the resulting values for comparisons between tests and devices. Which device out of the two used during this project was therefore noted for each test.

3.3 Best practice study

This chapter of the project aims to find a best practice for working with HV problems similar to those of geometry 1 and 2, by conduction of an optimization test on the parameters in the ML functions.

3.3.1 Deep Neural Network

As the parameter optimization for the DNN is a high-order multivariate problem, the optimal solution considering all parameters at the same time is hard to obtain. The interaction effect of the different parameters were therefore intentionally not considered. Instead, a methodical process was used, where each parameter was optimized using set values for the other parameters, as can be seen in Table 3.14.

Table 3.5: Default values for DNN parameters during optimization

Neurons × layers	Learning rate	Weight decay	Batch size	Activation function
10 × 4	10^{-3}	0	370	tanh

Some parameters and settings were not optimized, and were therefore set to their default values, which can be seen in Table 3.6.

Table 3.6: Set values for DNN parameters that were not optimized

Optimizer	Loss function	Validation data	Validation data fraction
Adam	Root-mean-square error	Random sample of data values	0.1

The same 400-point dataset was used for all tests, except during the optimization of the batch size, where two additional (200- and 600-point) datasets were used, and during the training dataset study.

Iterations of the same model configuration with different random seeds were tested in order to eliminate measurement noise, and to obtain a clearer picture of which configuration performs best, resulting in a best practice value or interval for each parameter. The random seeds used were five four-digit numbers (2854, 2751, 4827, 9351 and 7284), pseudorandomly generated using a calculator, and 0, the default random seed in COMSOL.

3.3.1.1 Number of epochs

The number of epochs were decided with respect to early stopping, which means that each DNN trial has its own number of epochs in order to combat overfitting. This was done by training the model two times for each configuration. The first training was conducted to determine the number of epochs that results in a validation loss minimum, based on the convergence plot. The second training was then performed using the optimal number of epochs determined in the first training. A maximum number of epochs was set to 500 000 to limit extensive simulations.

3.3.1.2 Number of neurons and layers

Different numbers of neurons and layers were tested. The main approach was to optimize a rectangular structured network, consisting of m neurons times n layers. First, configurations with $m = 4, 6, 8, \ldots, 16$ and $n = 1, 2, 3, 4, 5$ were tested to explore the performance of the model, using random seed 0. Then higher-performing variations were run with five additional random seeds (2854, 2751, 4827, 9351 and 7284), in order to produce a mean error per variation. In this extended evaluation, configurations with $m = 4, 6, 8, \ldots, 16$ and $n = 3, 4, 5$ were considered.

A smaller test of some common DNN architectures was also conducted. The structures tested were funnel-shaped networks, bottlenecked *autoencoder* networks, and the inverses of both. The different shapes were tested with different number of neurons, which can be seen in Table 3.7, and the configurations were tested using three different random seeds (0, 2854, and 2751). Device 1 was used for all tests considering optimization of the number of neurons and layers.

Table 3.7: Tested shape configurations

Structure	Number of neurons	Structure	Number of neurons
Funnel	64-32-16 32-16-8 16-8-4 8-6-4	Inverse Funnel	16-32-64 8-16-32 4-8-16 4-6-8
Autoencoder/Bottleneck	64-32-64 32-16-32 16-8-16 8-6-8 6-4-6	Inverse Bottleneck	32-64-32 16-32-16 8-16-8 6-8-6 4-6-4

3.3.1.3 Learning rate

The learning rate was initially tested between 10^{-1} and 10^{-7}, incrementing the exponent by 1 at each step, for the random seed 0. Based on the performance observed in this first test, the interval was narrowed, and two additional random seeds (2854 and 2751) where used to more accurately evaluate the effect of the learning rate between 10^{-1} and 10^{-6}. Device 2 was used for the duration of the whole test.

3.3.1.4 Weight decay

Different values of weight decay was used, either its default value of 0, or ranging from 10^{-1} to 10^{-7}. In this case, 0 signifies that no weight decay is implemented. This was done for the random seeds 0, 2854 and 2751. As done with the learning rate, a second test using additional random seeds (4827, 9351 and 7284), on a narrowed interval (10^{-5} to 10^{-7}, as well as no weight decay), was then used to more accurately evaluate the effect on model performance. Device 1 was used for the duration of the whole test.

3.3.1.5 Batch size

The batch size (the number of datapoints the DNN trains on at a time during an epoch) was tested on three different datasets with 200, 400 and 600 datapoints. Batch sizes used were defined as percentages of the training dataset size, ranging from 100% to 12.5%, with steps of 2.5 percentage points between 100% and 87.5%, and steps of 12.5 percentage points between 87.5% and 12.5%. This was done for three random seeds (0, 2854, 2751). Device 1 was used for the duration of the whole test.

3.3.1.6 Activation function

The different activation functions, namely Linear, ReLU, ELU, Sigmoid, and tanh, were tried and evaluated for best performance. The activation function undergoing testing was applied to all layers, and each configuration was tested using three different random seeds (0, 2854 and 2751). Device 2 was used for the duration of the whole test.

3.3.1.7 Training dataset size

To test the optimal number of training points for each geometry, a test was conducted using training datasets which consisted of 10, 50, 100, 200, 400, 600, 800 or 1000 points together with the default parameter settings. For geometry 2, two additional dataset sizes of 1200 and 1400 points were also tested. For these tests, new datasets were generated for all sizes. For each iteration, a batch size of 90% of the dataset was used. Each configuration was run using 3 different random seeds (0, 2854 and 2751), and all simulations were conducted on device 2.

3.3.2 Gaussian Process

For Gaussian Process, the varied parameters were covariance, mean, optimization method, number of restart points, maximum number of surrogate evaluations, maximum number of optimization iterations, and maximum matrix size. Three different random seeds were used, the default value of 1014, and the two random seeds 9750 and 7445. The same 400-point dataset as in the DNN optimization study was used and device 2 was used for the duration of the whole test.

The test progressed by successively testing a number of variations, one parameter at a time. From these variations, the best result (least error) of each parameter variation was selected as the best possible choice. In the end of the simulation run, a new run using all of these best choices was performed.

Table 3.8 shows the default settings used, while table 3.9 shows the different values testes for each ML method parameter.

Table 3.8: Default parameter settings for Gaussian Process

Covariance	Mean	Optim. method	Restart points	Max no. of SM eval.	Max optim. iter.	Max mat. size
Matérn 3/2	Constant	Direct	Auto (8)	10 000	500	2 000

Table 3.9: Tested value of each parameter settings for Gaussian Process

Covariance	Mean	Optim. method	Restart points	Max no. of SM eval.	Max optim. iter.	Max mat. size
Matérn 3/2	Constant	Direct	Auto (8)	10 000	500	2 000
Matérn 5/2	Linear	Monte carlo	10	20 000	1000	4 000
Sqared exp.	Quadratic	-	20	30 000	2000	8 000
Single lay. NN	-	-	-	-	-	-

3.3.3 Polynomial Chaos Expansion

For Polynomial Chaos Expansion, the room for parameter variation was smaller. Here, maximum polynomial degree, Q norm, relative tolerance, and maximum matrix size were varied. The same 400-point dataset as in the DNN optimization study was used. Device 2 was used throughout the test.

The same methodology as was used for finding the best parameter choices in the Gaussian Process study was also used for this study.

Table 3.10 shows the default settings used, while table 3.11 shows the different values testes for each ML method parameter.

Table 3.10: Default parameter settings for Polynomial Chaos Expansion

Max degree	Q-norm	Rel. tol.	Max size
Auto	Auto	10^{-3}	2 000

Table 3.11: Tested value of each parameter settings for Polynomial Chaos Expansion

Max pol. degree	Q norm	Rel. tolerance	Max. mat. size
Auto	Auto	10^{-3}	2 000
30	0.3	10^{-4}	4 000
50	0.5	10^{-5}	8 000
70	0.7	-	-

3.4 Streamer detection study setup

This section describes how the streamer detection studies were set up in COMSOL Multiphysics.

3.4.1 Geometry and mesh

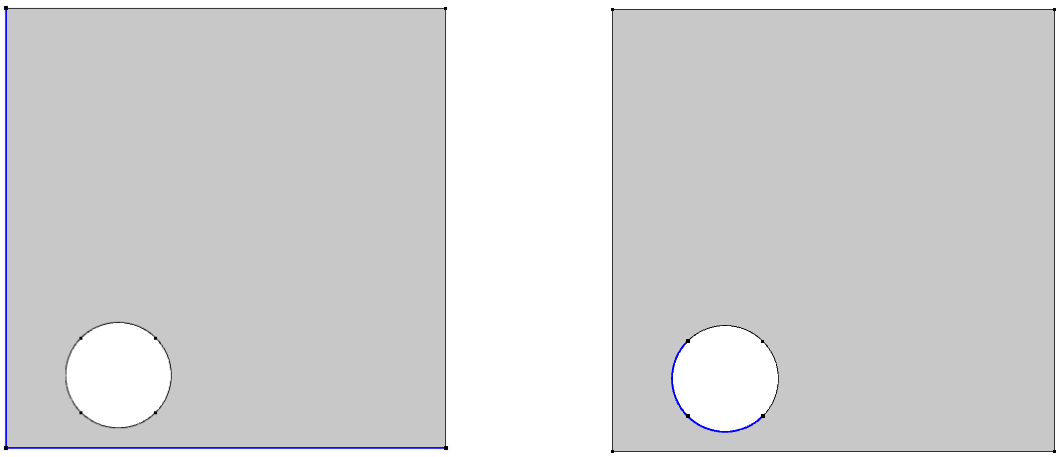
For the streamer detection study, the geometry and mesh settings used for geometry 1 were reused, as described in section 3.1.1.1 and 3.1.3.

3.4.2 Physics

The study makes use of the electrostatic and particle tracing physics modules. The electrostatic module is used in conjunction with a stationary solver to calculate the electric field across the geometric domain, and the particle tracing is used in conjunction with a time dependent solver to determine the maximum value of the streamer integral across the geometric domain (see Equation 2.2) by tracing particles from the cathode to the anode, where a particle counter is implemented.

A streamer integral value $\gtrsim \ln(10^8)$ doesn't hold any scientific importance for this study, but rather indicates that conditions for streamer formation are present.

The cathode was selected as the two outer boundaries closest to the conductor, and the particle counter was selected as a semicircular part of the conductor boundary, rotated to be as close to the corner connecting the cathode boundaries as possible. This was done to reduce the computational time of the time dependent study, knowing that a streamer inception will not happen outside of these regions as the two possible critical paths are covered.



(a) Cathode **(b)** Particle counter

Figure 3.6: Boundaries for cathode and particle counter selection

3.4.3 Surrogate Model Training

The input parameters and their corresponding parameter interval were set as shown in Table 3.12.

Table 3.12: Interval for input parameters

Input parameter	Range
V_0 [kV]	60 - 100
l_y (first run) [m]	0.0105 - 0.04
l_y (second/improving run) [m]	0.01 - 0.0105

As the output, two parameters of interest were solved for, the first being the maximum value of the streamer integral across the domain, and the second one being the Schwaiger factor (degree of uniformity) $\eta = E_{\text{avg}}/E_{\text{max}}$.

3.4.4 Reference datasets

The radius of the conductor was not set as an input parameter in this study, but was instead set as values of 0.01 m, 0.12 m and 0.25 m, eliminating the need for four densely sampled dimensions, and resulting in three different reference datasets. Both the first run and the improving run consisted of 2000 datapoints, resulting in a total of 4000 datapoints per reference set. The input parameters and their corresponding parameter intervals were set as shown in Table 3.13.

Table 3.13: Interval for input parameters

Input parameter	Range
U [kV]	60 - 100
l_y (first run) [m]	0.01000005 - 0.04
l_y (second/improving run) [m]	0.01 - 0.01000005

The same outputs as for the training datasets were set (the maximum value of the streamer integral across the domain and the Schwaiger factor).

3.4.5 DNN parameters

Table 3.14, present the DNN parameters used as the default settings during the streamer detection study. These parameters come from the result of the best practice study.

Table 3.14: Default values for DNN parameters during streamer study

Neurons × layers	Learning rate	Weight decay	Batch size	Activation function
4 × 5	10^{-3}	0	90%	tanh

3.5 Streamer detection study

3.5.1 Neurons, layers and dataset size study

A study exploring the optimal number of neurons and layers, and the optimal training dataset size, was done for the streamer detection case. These studies were performed simultaneously by testing a set interval of the number of layers and neurons for each dataset size. 4-5 layers with 4, 6 and 8 neurons, respectively, was chosen from the result of the best practice test for DNN structure, presented in section 4.1.1.1. These different DNN structures were tested on dataset sizes of 10, 50, 200 and 400 datapoints.

This was done in order to study DNN's performance with respect to the sparsity of the training dataset, while finding the optimal DNN structure for this case.

3.5.2 Degree of uniformity

To study the effects of field uniformity on DNN performance, a Schwaiger factor (degree of uniformity) study was conducted. As the model considers distance to ground as an input, the field uniformity will not be constant across a single solution space even though there exists a separate one for each choice of conductor radius. Thus, a mean Schwaiger factor is calculated for every such solution, and the error of the model is studied with respect to it.

The DNN size used during this study was the best performing choice in the dataset size study, which was presented in Section 4.2.1 (5 layers, 4 neurons each). The 200-point training dataset was used.

4

Results and discussion

4.1 Best practice study

In this section, the result of the best practice studies are presented and discussed. The errors of the training dataset interpolations can be found in Table 4.1, and were used to quantify the validity of the DNN models. As described in Section 3.2.1.2, devices 1 and 2 had a small discrepancy between their integration domains, leading to a difference in the error of the training data interpolation.

Table 4.1: 400-point training dataset error

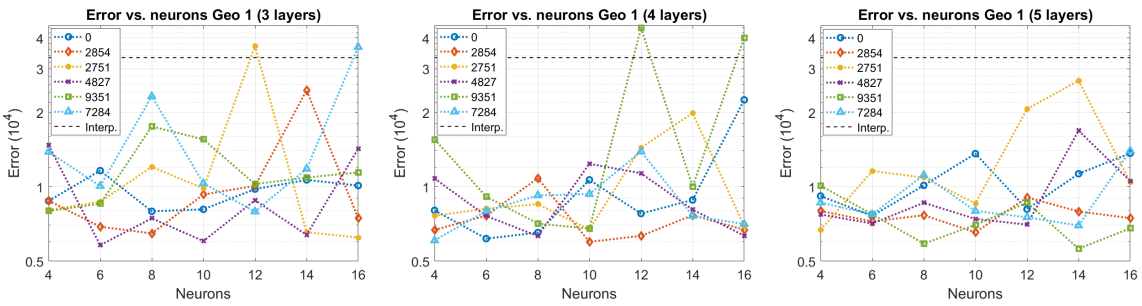
Geometry	Training data error
Geometry 1	33 303 (Device 1) and 33 703 (Device 2)
Geometry 2	6 205 562

4.1.1 Deep Neural Network

In this section, the result of the DNN parameter evaluation is presented. The figures and plots presented in this section are compiled results. The raw data for each test can be found in Appendices A-G.

4.1.1.1 Rectangular DNN

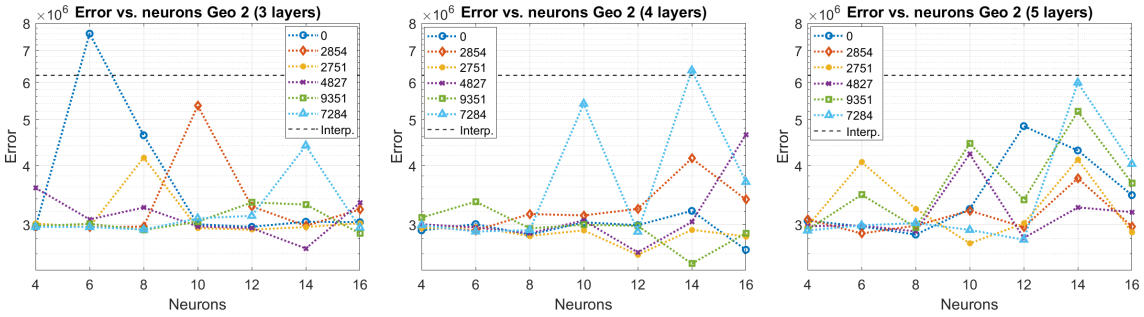
In Figures 4.1-4.2, the result of the optimization tests is presented with a different plot for each number of layers. Each plotted line in the figures is the result of a different random seed, and the black dashed line shows the interpolation error of the respective training dataset.



(a) 3 layers **(b)** 4 layers **(c)** 5 layers

Figure 4.1: Error plots for geometry 1 using 3 layers (a), 4 layers (b), and 5 layers (c)

When using 3 layers in the DNN, there is a substantial variance in the error between the different random seeds. This indicates that the DNN complexity doesn't match the problem making the model validity quite volatile. When instead using 4 layers there is still a lot of variance between the different random seeds when using 4 or 12-16 neurons. This could suggest that using 4 neurons has to low complexity for this problem, and using 12-16 neurons has too high complexity. 6-10 neurons has less variance, indicating that the complexity of the network matches the problem. The variance when using 5 layers in the DNN is lower in general, because of the higher complexity. By using a network with 5 layers and $\lesssim 10$ neurons, the variance can be minimized, meaning a reliable model with a around a third of the error in comparison with the training dataset interpolation error.



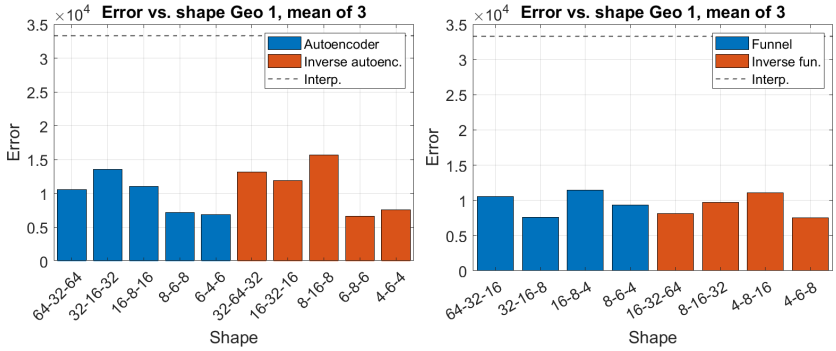
(a) 3 layers **(b)** 4 layers **(c)** 5 layers

Figure 4.2: Error plots for geometry 2 using 3 layers (a), 4 layers (b), and 5 layers (c)

The result, when using a 3-layered DNN on the dataset from geometry 2, shows higher error variance when using a smaller number of neurons, which indicates that the network complexity is too low for the problem. When using a DNN with 4-5 layers and ≤ 8 neurons, the variance is small, indicating that the network is complex enough for the problem. The error of the DNN for these DNN dimensions are around half of the error of the training data interpolation.

4.1.1.2 DNN shape study

Figures 4.3-4.4 and Tables 4.2-4.3 shows the result of the DNN shape studies, for geometry 1.



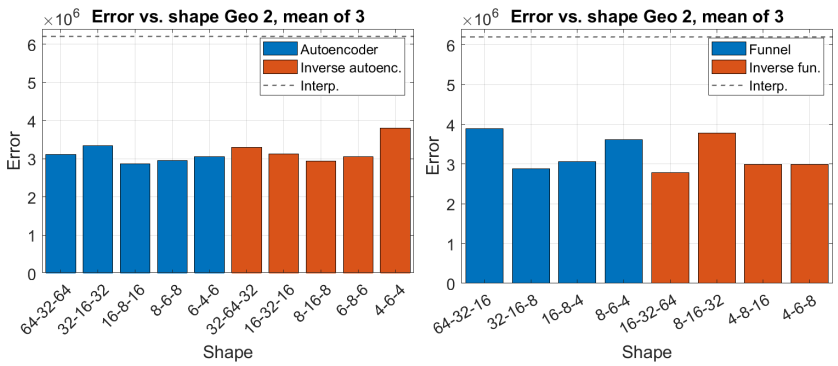
(a) Autoencoder-type **(b)** Funnel-type

Figure 4.3: Shape study, geometry 1 (Mean of 3)

Table 4.2: Result of shape study for geometry 1 (Mean of 3)

Shape	Neurons	Error	Training time [s]	Epochs	Time per epoch [ms]
Funnel	64-32-16	10 536	22	31 333	0.70
	32-16-8	7 567	28	36 667	0.71
	16-8-4	11 478	53	154 333	0.35
	8-6-4	9 335	171	499 333	0.34
Autoencoder	64-32-64	10 596	34	37 333	0.93
	32-16-32	13 572	94	166 667	0.56
	16-8-16	11 064	34	8 5333	0.40
	8-6-8	7 174	122	341 667	0.36
	6-4-6	6 869	171	497 333	0.34
Inverse Funnel	16-32-64	8 130	20	29 667	0.68
	8-16-32	9 701	55	122 667	0.45
	4-8-16	11 056	54	161 000	0.33
	4-6-8	7 490	133	395 000	0.33
Inverse Autoencoder	32-64-32	13 171	44	51 667	0.79
	16-32-16	11 937	34	62 000	0.55
	8-16-8	15 709	29	79 333	0.37
	6-8-6	6 612	35	102 333	0.35
	4-6-4	7 608	126	392 333	0.32

For the autoencoder and its inverse used on geometry 1, the result shows a general trend of smaller errors, together with an increase in training time, when using fewer neurons. For the two funnel shapes, the error stays somewhat consistent while the training time increases when using fewer neurons. The errors for both of these types of shapes are in the same magnitude as the errors of the rectangular DNN models presented in Section 4.1.1.1, but the autoencoder (or inverse autoencoder) with fewer neurons produces the lowest errors, and would therefore be the best choice out of these options.



(a) Autoencoder-type **(b)** Funnel-type

Figure 4.4: Shape study, geometry 2 (Mean of 3)

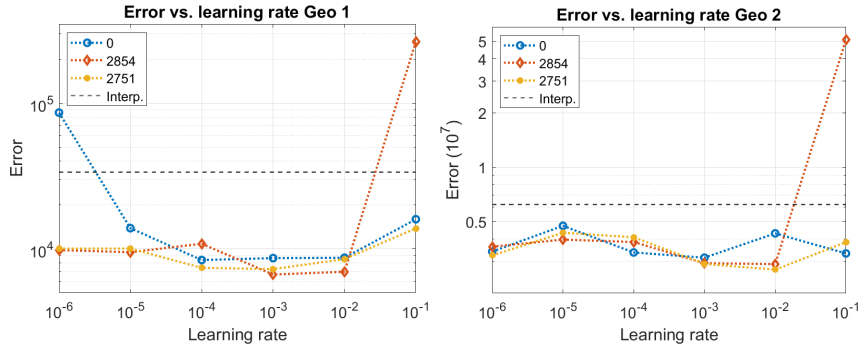
Table 4.3: Result of shape study for geometry 2 (Mean of 3)

Shape	Neurons	Error	Training time [s]	Epochs	Time per epoch [ms]
Funnel	64-32-16	3 891 345	27	40 800	0.65
	32-16-8	2 881 930	52	112 467	0.46
	16-8-4	3 064 402	78	232 667	0.33
	8-6-4	3 609 554	119	337 000	0.34
Autoencoder	64-32-64	3 119 792	45	58 667	0.76
	32-16-32	3 349 667	81	141 000	0.56
	16-8-16	2 873 264	86	223 667	0.37
	8-6-8	2 962 569	129	379 000	0.34
	6-4-6	3 058 421	82	246 000	0.32
Inverse Funnel	16-32-64	2 788 231	80	120 333	0.66
	8-16-32	3 774 509	96	206 667	0.47
	4-8-16	2 989 780	132	389 667	0.34
	4-6-8	2 988 003	134	390 333	0.34
Inverse Autoencoder	32-64-32	3 296 249	31	43 667	0.73
	16-32-16	3 133 033	50	94 667	0.52
	8-16-8	2 943 147	77	208 000	0.36
	6-8-6	3 050 754	115	336 000	0.34
	4-6-4	3 805 048	114	382 667	0.30

For geometry 2, the training time increases with a decrease in number of neurons. Using an autoencoder or its inverse, with 8 or 16 neurons at maximum width provides the lowest errors. This could be explained by geometry 2 being more complex, and therefore needing a more complex DNN architecture. Again, the funnel shapes show no clear pattern. The magnitude of the errors are, again, in line with the errors of the rectangular DNN presented in section 4.1.1.1. The shape of the DNN seems to therefore not affect the model error as much as the number of layers and neurons.

4.1.1.3 Learning rate

The result of the learning rate studies is presented in Tables 4.4-4.5 and Figure 4.5.



(a) Geometry 1 **(b)** Geometry 2

Figure 4.5: Learning rate study for both geometries, excluding 10^{-7}

Table 4.4: Learning rate result for geometry 1 (Mean of 3 except for 10^{-7})

Learning rate	Error	Training time [s]	Epochs	Time per epoch [ms]
10^{-1}	98 370	6	3 533	1.91
10^{-2}	8 058	53	33 891	1.50
10^{-3}	7 526	130	75 167	1.67
10^{-4}	8 883	144	85 833	1.67
10^{-5}	11 143	344	176 667	1.92
10^{-6}	35 389	658	391 333	1.66
10^{-7}	553 561	767	500 000	1.53

When decreasing the learning rate, the number of epochs increase, as does the training time, making the time per epoch close to constant during the whole test. A learning rate of 10^{-7} was eliminated from further consideration early on, due to its very slow convergence. The error reaches a minimum when using a learning rate of 10^{-3}. The error variance is low for all learning rates with the exception of 10^{-1} and 10^{-7}. With this in mind, a learning rate of 10^{-3} seems to be optimal for minimizing the error, but using a learning rate of 10^{-2} could be a better option if low training time is important, since it still yields a low error.

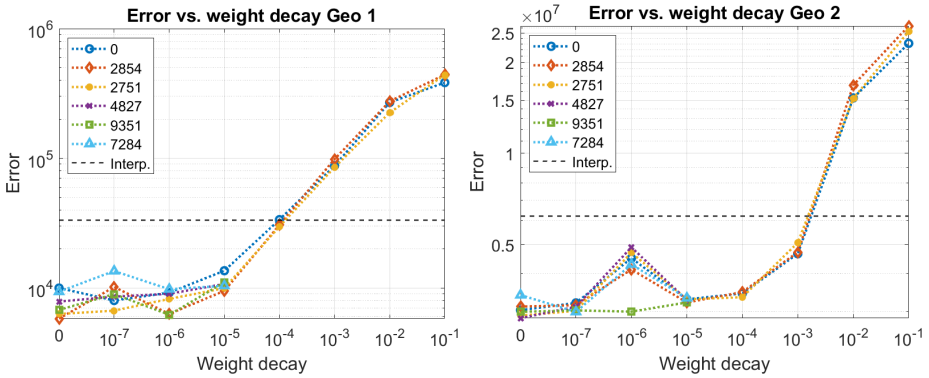
Table 4.5: Learning rate result for geometry 2 (Mean of 3 except for 10^{-7})

Learning rate	Error	Training time [s]	Epochs	Time per epoch [ms]
10^{-1}	19 338 441	4	2 052	2.31
10^{-2}	4 293 251	242	153 144	1.58
10^{-3}	3 148 190	116	70 667	1.69
10^{-4}	3 364 224	316	190 733	1.59
10^{-5}	4 729 060	162	99 000	1.63
10^{-6}	3 403 968	734	438 000	1.68
10^{-7}	29 848 450	853	500 000	1.71

The error when using different learning rates for geometry 2 has a similar pattern as for geometry 1, where a very high or very low learning rate gives a high error, and using 10^{-3} gives the lowest error. The variance is low for all learning rates, except for 10^{-1}. When not considering 10^{-1}, due to its very high error, 10^{-3} also needs the shortest training time, making it the optimal choice. In contrast to the result for geometry 1, the error and training time varies without a clear pattern when using learning rates between 10^{-2} and 10^{-6}.

4.1.1.4 Weight decay

In Tables 4.6-4.7 and Figure 4.6, the result of the weight decay study is presented.



(a) Geometry 1 **(b)** Geometry 2

Figure 4.6: Weight decay study for both geometries

Table 4.6: Weight decay result for geometry 1 (Mean of 3 for 10^{-1} to 10^{-4}, mean of 6 for 10^{-5} to 0)

Weight decay	Error	Training time [s]	Epochs	Time per epoch [ms]
10^{-1}	421 963	< 1	340	0
10^{-2}	256 672	0,33	900	0.21
10^{-3}	90 474	1	2 333	0.44
10^{-4}	31 335	15	31 967	0.50
10^{-5}	10 859	72	137 117	0.52
10^{-6}	8 115	71	154 733	0.47
10^{-7}	9 308	93	205 850	0.46
0	7 669	61	132 967	0.46

The results show a clear trend, where the error decreases with a smaller weight decay, where the smallest error was obtained using no weight decay at all. The training time increases with a smaller weight decay, with an exception for the case without weight decay. Therefore, no weight decay seems to be the best choice, since it produces a low error while having a short training time.

Table 4.7: Weight decay result for geometry 2 (Mean of 3 for 10^{-1} to 10^{-4}, mean of 6 for 10^{-5} to 0)

Weight decay	Error	Training time [s]	Epochs	Time per epoch [ms]
10^{-1}	24 902 044	< 1	337	0
10^{-2}	15 736 691	6	13 700	0.60
10^{-3}	4 814 762	8	18 800	0.45
10^{-4}	3 424 846	28	60 967	0.46
10^{-5}	3 270 067	37	78 867	0.48
10^{-6}	4 255 223	74	155 600	0.48
10^{-7}	3 085 036	74	156 255	0.47
0	3 052 581	90	195 850	0.46

For geometry 2, the same trend of a decreasing error with a smaller weight decay can be seen. However, the result shows a smaller relative decrease in error compared to geometry 1. At the same time, it shows a clear increasing trend for the training time, making the optimal value for the weight decay vary. If a low error is a critical target, no weight decay seems to be optimal, while if a shorter training time is a constraint, a small weight decay (e.g. 10^{-7}) could be a better choice.

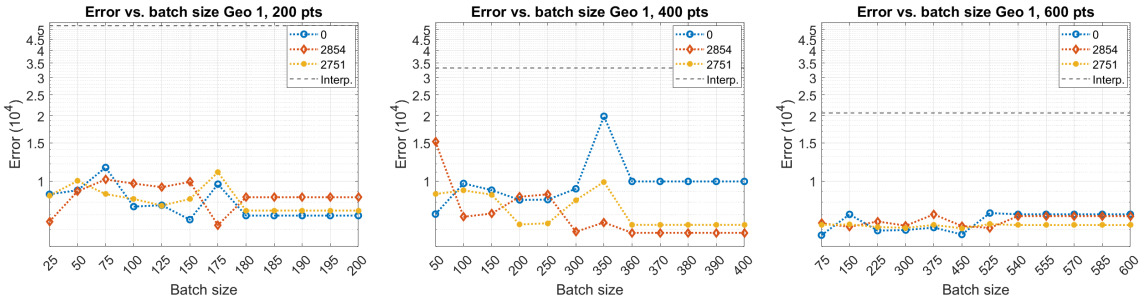
4.1.1.5 Batch size

The batch size tests were done using multiple datasets with different number of datapoints. The interpolation error of each dataset can be seen in Table 4.8. The table also shows the mean error of all the datasets to facilitate comparison, since the errors presented in Tables 4.9 and 4.10 shows the mean error of the different dataset sizes.

Table 4.8: Interpolation error of the training datasets, and the mean error for both geometries

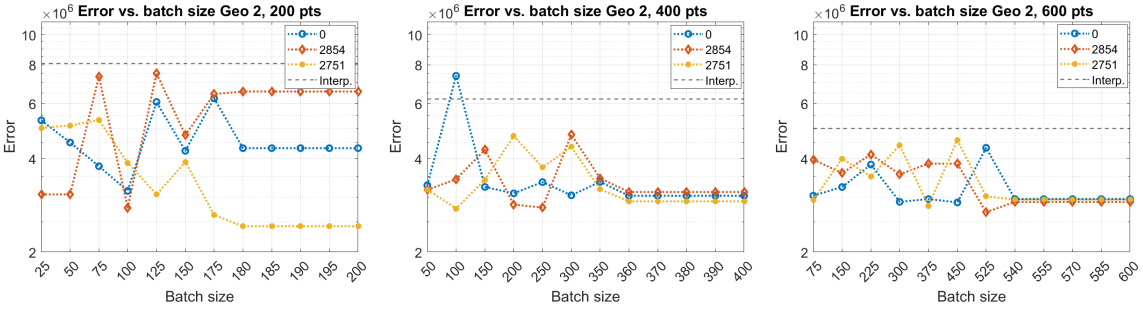
Geometry	Training dataset size	Training interpolation error
Geometry 1	200	52 022
	400	33 303
	600	20 606
	Mean	35 311
Geometry 2	200	8 084 739
	400	6 205 562
	600	4 994 297
	Mean	6 416 915

Tables 4.9-4.10 and Figures 4.7 and 4.8 show the results of the batch size studies.



(a) 200 datapoints **(b)** 400 datapoints **(c)** 600 datapoints

Figure 4.7: Batch size study, geometry 1



(a) 200 datapoints **(b)** 400 datapoints **(c)** 600 datapoints

Figure 4.8: Batch size study, geometry 2

Table 4.9: Batch size result for geometry 1 (Mean of 9, 3 random seeds per dataset size)

Batch size [% of dataset]	Error	Training time [s]	Epochs	Time per epoch [ms]
100	7 214	28	64 111	0.43
97.5	7 214	28	64 111	0.43
95	7 214	28	64 111	0.43
92.5	7 214	28	64 111	0.43
90	7 214	27	64 111	0.43
87.5	9 197	46	68 022	0.68
75	7 336	35	50 211	0.68
62.5	7 507	33	47 344	0.68
50	7 434	33	45 667	0.66
37.5	8 209	37	40 522	0.91
25	8 139	40	36 200	1.10
12.5	8 118	61	29 689	2.08

While the error does not change significantly when varying the batch size, the minimum error is achieved when using a batch size of over 90% of the number of training datapoints. The result does not vary with a batch size above 90%, as the validation data fraction is set to 10%, i.e. 10% of the datapoints are excluded from training, leaving 90% of the original set. The optimal batch size is therefore $\geq$ (1−Validation data fraction).

The time per epoch shows a clear trend where a smaller batch size leads to a longer training time, which can be explained by smaller batch sizes making the DNN need more iterations for updating weights per epoch, to train on all the datapoints. A batch size larger than or equal to 90% is therefore the best scenario in this case, both when it comes to the error and the training time. The error variance does not show a clear trend, but seems to get somewhat larger when using a smaller batch size, again indicating that a large batch size is preferable for the type of problems studied.

Table 4.10: Batch size result for geometry 2 (Mean of 9, 3 random seeds per dataset size)

Batch size [% of dataset]	**Error**	**Training time [s]**	**Epochs**	**Time per epoch [ms]**
100	1 961 865	28	63 389	0.44
97.5	1 961 865	28	63 389	0.44
95	1 961 865	28	63 389	0.44
92.5	1 961 865	28	63 389	0.44
90	1 961 865	28	63 389	0.44
87.5	2 356 171	52	78 100	0.67
75	2 262 193	54	81 422	0.67
62.5	2 307 299	57	85 222	0.68
50	2 057 583	55	82 300	0.67
37.5	2 276 018	66	75 478	0.93
25	2 680 536	75	69 533	1.09
12.5	2 217 944	169	87 767	1.94

The same trend as for geometry 1 can be seen for geometry 2, where the error is somewhat better for a batch size larger than or equal to 90 % of the number of training datapoints. The time per epoch also shows the same pattern, with longer training times for smaller batch sizes, and the error variance gets larger when using a smaller batch sizes, leading to the same optimal batch size of larger than or equal to 90 % of the number of training datapoints or in more general terms, larger than or equal to (1− Validation data fraction).

4.1.1.6 Activation function

The result of the activation function study is presented in Tables 4.11 - 4.12.

Table 4.11: Activation function result for geometry 1 (Mean of 3)

Activation function	Error	Training time [s]	Epochs	Time per epoch [ms]
tanh	7 525	130	75 167	1.67
Linear	445 012	< 1	143	0
ReLU	270 532	2	1850	0.36
ELU	8 024	64	40 000	1.6
Sigmoid	9 210	111	73 333	1.55

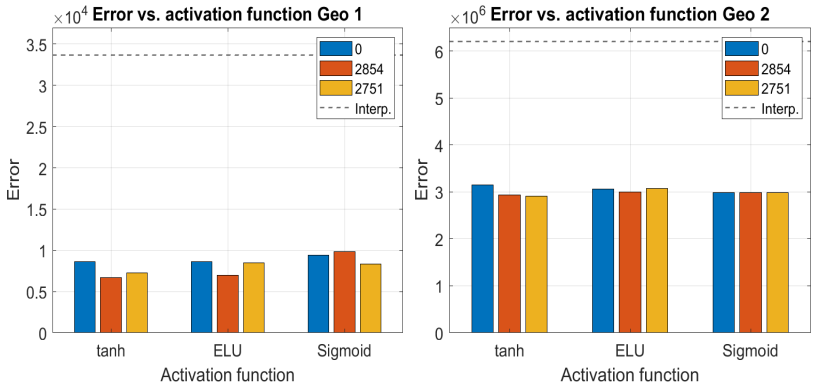
When considering the errors, using the activation functions Linear and ReLU produces models with very high errors while tanh, ELU and Sigmoid produces models with low errors in a similar magnitude.

Table 4.12: Activation function result for geometry 2 (Mean of 3)

Activation function	Error	Training time [s]	Epochs	Time per epoch [ms]
tanh	2 995 610	116	70 667	1.69
Linear	28 550 315	< 1	130	0
ReLU	6 816 011	0.33	434	0.26
ELU	3 044 269	62	38 000	1.62
Sigmoid	2 984 448	88	61 000	1.43

The same conclusion drawn for geometry 1 can be drawn for geometry 2, while using the activation functions Linear and ReLU, where the errors are larger for the DNN model than for the training data interpolation.

To better evaluate the performance of tanh, ELU and Sigmoid, the result of using each one is presented in Figure 4.9, showing individual results for each random seed.



(a) Geometry 1 **(b)** Geometry 2

Figure 4.9: The three best performing activation functions across both geometries

For geometry 1, all three activation functions have error variance in the same magnitude, making it possible to draw a conclusion from the mean errors presented in Table 4.11. To minimize the error for geometry 1, tanh should be used since it has the lowest error out of the three. However, it has the longest training time, meaning that ELU is possibly a better choice if a short training time is an important factor.

The difference in error between the different activations functions for geometry 2 are small, while the relative variance between different random seeds is smaller compared to for geometry 1, making reaching a conclusion harder. The error difference between the different activation functions is negligible, making training time a deciding factor, ELU would therefore be the best option since it has the shortest training time.

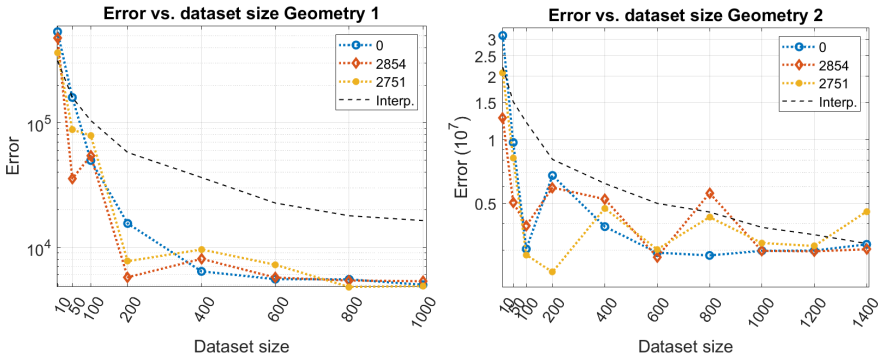
4.1.1.7 Training dataset size

Table 4.13 presents the interpolation errors of the training datasets used for the dataset size study.

Table 4.13: Interpolation error of the training datasets for both geometries

Training dataset size	Interpolation error Geometry 1	Interpolation error Geometry 2
10	314 965	22 085 742
50	159 446	15 149 917
100	104 024	12 100 000
200	57 658	8 071 679
400	36 299	6 208 784
600	22 647	4 997 699
800	17 899	4 545 657
1000	16 380	3 845 777
1200	-	3 551 197
1400	-	3 223 973

The training data interpolation errors are plotted as a black dashed lines in Figure 4.10, which presents the results of the dataset size study.



(a) Geometry 1 **(b)** Geometry 2

Figure 4.10: Dataset size study

The DNN error for geometry 1 has a clear trend where both the error and variance between random seeds decreases when using a larger training dataset size. When using a dataset size equal to or larger than 200 points, the DNN model is much better than just using the training data interpolation, meaning that using DNN provides a better model while using a similar computational load as regular FEM simulations. From around 800 points and up, the error and variance seem to start plateauing, meaning that for a target of minimizing error, a dataset consisting of around 800 datapoints is optimal.

When using less than 1000 datapoints for geometry 2, some error variance can be found. Although 100 and 600 datapoints produce low errors with low variance, no certain conclusion can be drawn because of the lack of a clear pattern. By using 1000 datapoints, the resulting error variance is low while also producing a low error. However, the training data interpolation error at this point is also quite low, which indicates that using FEM simulations could be more reliable (e.g. more proven to produce specific results) than using DNN.

The result presented in Figure 4.10 could indicate that DNN perform better and could therefore be more easily defendable for simpler geometries, as geometry 1 shows greater improvements using a DNN compared to the training data interpolation, while the DNN used on the more complex geometry of geometry 2 has a harder time outperforming the training data.

4.1.1.8 Optimal DNN parameters

Tables 4.14 and Table 4.15, present the found optimal setting of each DNN parameter from the best practice study.

Table 4.14: Best performing DNN parameter values for geometry 1

Neurons × layers	Learning rate	Weight decay	Batch size	Activation function	Trainingdata size
$\leq 10 \times 5$	10^{-3}	0	$\geq 90\%$	tanh	800

Table 4.15: Best performing DNN parameter values for geometry 2

Neurons × layers	Learning rate	Weight decay	Batch size	Activation function	Trainingdata size
$\leq 8 \times$ 4-5	10^{-3}	0	$\geq 90\%$	ELU	1000

4.1.2 Gaussian Process

In Tables 4.16-4.17, the result of the Gaussian Process evaluation is presented. This section presents a compiled result, and the raw data can be found in Appendix H.

Table 4.16: Gaussian Process best result, geometry 1

Error	Covariance	Mean	Optim. method	Restart points
6906	Matérn 5/2	Linear	Direct	10
Max no. of SM eval.	**Max optim. iter.**	**Max mat. size**	**Random seed**	**Training time [s]**
10 000	500	2 000	1014	86

Table 4.17: Gaussian Process best result, geometry 2

Error	Covariance	Mean	Optim. method	Restart points
2 960 625	Matérn 3/2	Linear	Direct	10
Max no. of SM eval.	**Max optim. iter.**	**Max mat. size**	**Random seed**	**Training time [s]**
10 000	500	2 000	1014	87

In comparison with the error results obtained using DNN, Gaussian Process produces errors of the same magnitude. However, the result presented in Appendix H shows that the best performing DNN outperforms the best Gaussian process model when considering the errors, while the error variance is lower for both geometry 1 and geometry 2 when using a Gaussian process model. The training time is similar in length for both models, making neither ML method appear superficially better suited for this application.

4.1.3 Polynomial Chaos Expansion

In Tables 4.18-4.19, the result of the Polynomial Chaos Expansion evaluation is presented. This section presents a compiled result, and the raw data can be found in Appendix I.

Table 4.18: Polynomial Chaos Expansion best result, geometry 1

Error	Max pol. degree	Q norm	Rel. tol.	Max mat. size	Training time [s]
6 840	Auto	Auto	10^{-6}	2 000	5

Table 4.19: Polynomial Chaos Expansion best result, geometry 2

Error	Max pol. degree	Q norm	Rel. tol.	Max mat. size	Training time [s]
2 434 564	Auto	Auto	10^{-4}	2 000	5

The errors using the optimal settings for the Polynomial Chaos Expansion are similar in size in comparison with the errors using an optimized DNN. The result presented in Appendix I however, shows high variance when using a Polynomial Chaos expansion model on geometry 1, where the error are very high in comparison with DNN with the exception of the optimal parameter settings. The result of geometry 2 presents lower variance, with all resulting error being in the same magnitude of the best preforming DNN models. The training time is a lot shorter for Polynomial Chaos Expansion compared to DNN. A conclusion on which ML model is most suitable is therefore hard to reach.

4.1.4 Comparison between the ML models

All three ML methods have different areas where they outperform the others. By using the optimal settings, DNN produces the lowest errors out of all ML methods. Gaussian process has low variance and seems to therefore be able to produce consistently well performing models. Polynomial Chaos Expansion has drastically shorter training time in comparison with the others.

Only one out of the three ML models was chosen to be tested in the application study on a streamer detection case. While all three models showed promising results, DNN was able to produce the lowest errors out of the three models, and was therefore chosen for further study.

4.2 Streamer detection study

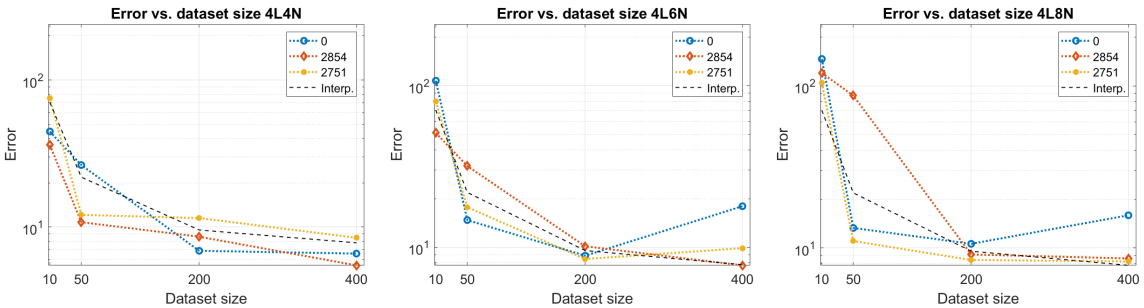
4.2.1 Dataset size study

Table 4.20 shows the dataset interpolation error of each dataset size used.

Table 4.20: Training dataset interpolation error

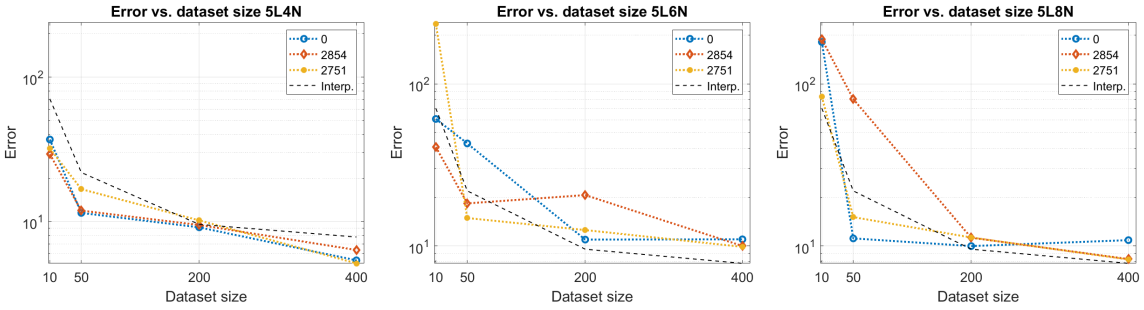
Dataset size	Training interpolation error
10	70.64
50	21.86
200	9.55
400	7.81

These dataset errors are plotted in Figures 4.11 and 4.12, as a black dashed line. Figure 4.11 presents the result of using 4 layers in the DNN, and 4.12 shows the result of using 5 layers.



(a) 4 neurons **(b)** 6 neurons **(c)** 8 neurons

Figure 4.11: Streamer dataset size study, 4 layers



(a) 4 neurons **(b)** 6 neurons **(c)** 8 neurons

Figure 4.12: Streamer dataset size study, 5 layers

For all DNN structures studied, an increase in training dataset size resulted in a general trend of decreasing errors. The error variance, however, does not seem to be universally decreasing with a larger dataset size when using 4 layers. The 5-layered DNNs show a clear trend with lower errors and less variance when more training data is provided. Therefore, 5 layers seems to be a better choice for this case, and by using 4 neurons on each of those layers, the best DNN models can be achieved.

A general conclusion that can be drawn for the result is that, for all combinations of layers, neurons and dataset sizes, the errors of DNN models are either worse, the same, or marginally better than the training dataset interpolation error. This shows that using a DNN does not result in much better models, and in a lot of cases, yields a worse model than simply using interpolations of the FEM solutions.

Although, geometry 1 from the best practice was used together with similar settings, the errors of the DNN compared to the training data are comparably worse. The difference here are a more complex physics being used, indicating that the DNN has a harder time producing a good model when the output is more complex.

4.2.2 Uniformity study

The results of the Schwaiger factor study are presented in Figure 4.13 and Table 4.21.

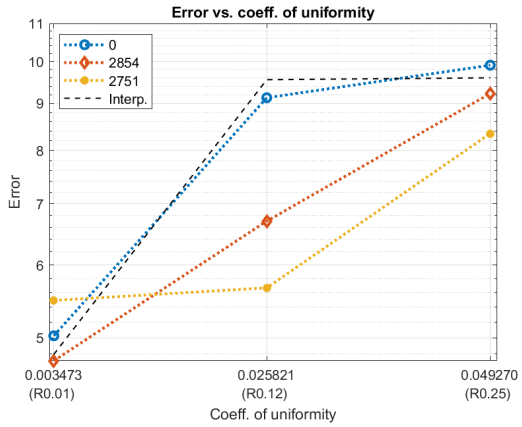


Figure 4.13: Error vs. mean Schwaiger coefficient

Table 4.21: Schwaiger coefficient and training dataset error per radius

Radius [m]	Mean Schwaiger coefficient	Training interpolation error
0.01	0.003473	4.7964
0.12	0.02582	9.5532
0.25	0.04927	9.5976

Although the variance is high, the studies show that the DNN marginally outperforms an interpolation of the training data for the streamer prediction case with a mean Schwaiger factor of $\bar{\eta} = 0.025821$ (conductor radius 0.12 m).

For the two other cases, $\bar{\eta} = 0.003473$ or 0.04927, there are both superior and inferior DNN solutions in comparison to the training data interpolation. This suggests that there is no clear advantage in using DNN for these cases.

The study, in its current form, disproves the hypothesis that there is a simple relation between the field uniformity of a geometry and the DNN error of the streamer inception case. An expansion of the study could possibly shed more light on the hypothetical relation, by averaging of a larger number of simulations with a larger data span.

5
Conclusion

5.1 Best practice

In summary, the best practice study revealed that DNN performs better than simply interpolating the generated training datasets for the simpler (geometry 1) of the two geometries. As for geometry 2, the more complex geometry, there is no clear conclusion to be drawn. While some of the DNN iterations perform better than the interpolation, that is not always the case.

Because of this, the main takeaway from the best practice study is that, for two similar geometries of different complexity, the DNN performs better on the simpler one, where it could be a more viable option than classical FEM simulations.

While PCE and GPM produced similar results, DNN had the single best solutions for the best practice problem.

5.2 Streamer detection

The streamer detection simulations use more complex physics compared to the best practice study, so while the same DNN setting and a simple geometry (geometry 1) was used, the result shows that the DNN has a harder time exceeding the training data interpolation benchmark. In conclusion, the higher complexity of the problem in this study compared to the best practice study, seems to result in the DNN contributing with less benefit and therefore making it less justifiable to use.

With the amount of data given by the Schwaiger factor study, no clear conclusion about DNN performance based on field uniformity could be drawn.

5.3 Future works

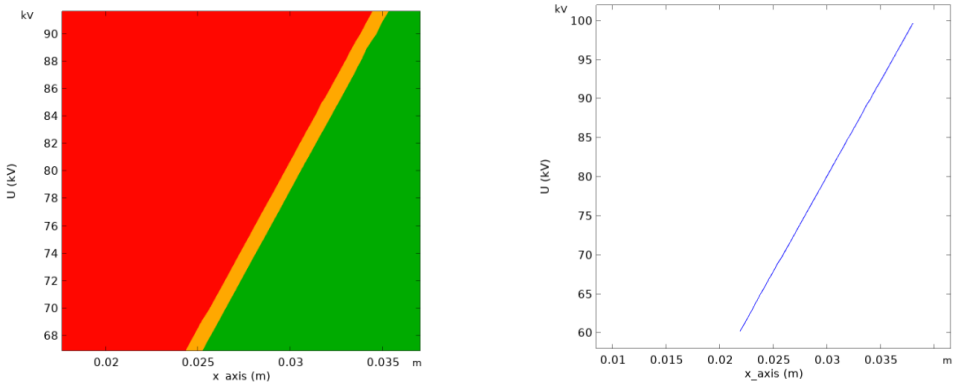
In order to further build on the findings of this thesis, there are some further studies that could be conducted.

In this study, the learning rate is set to a fixed value while running the DNN training. A future study could start with a higher learning rate, and carefully stopping, decreasing the rate and continuing the training in a way that ensures the model still

converges. This could be an alternative way of providing a quick convergence while making sure that the DNN does not miss details in the data.

The PCE and GPM studies were cut short in order to prioritize the DNN studies. Conducting a multivariate study by optimizing PCE or GPM parameters could open doors that have been presumed closed.

As for the streamer study, the DNN could be embedded into a COMSOL application builder program, predicting if a design is safe, uncertain, or strictly unsafe, based on geometrical and physical parameters. This would make for a simpler process than running all simulations through a FEM analysis.



(a) Design safety prediction

(b) Inception voltage vs. critical path length

Figure 5.1: Examples of ML prediction of design safety and streamer inception voltage

The uniformity study was cut short following time constraints. An expansion of the study, sampling different radii as a quantity of interest in surrogate model training rather than running discrete solutions, could mean finding a clear pattern connecting field uniformity to DNN performance.

Regarding data quality, alternative ways of sampling reference and training data could be studied, such as exploring other distributions for enhancement of high-gradient regions. For geometry 2, it is also unclear whether the boundary to the resistive layer was sampled densely enough to be a good physical representation, which could be improved upon in a future study.

As generation of the reference dataset was time consuming, another approach could be tried, where a smaller reference dataset is generated, and the corresponding points are sampled from the continuous ML model solutions. The solution could then be evaluated using a discrete metric, e.g. the discrete L2 norm. While local artifacts might be missed due to the more sparse sampling, it could be seen as a tradeoff between accuracy and reference dataset generation time.

Meshing quality plays a big role in model accuracy. A study on the correlation between meshing quality and DNN performance would be of interest.

Lastly, even though the results of this thesis suggests otherwise, there is still a possibility that ML could help solve problems related to more complex geometries. With this in mind, further studies looking into different geometries of different complexities could present a bigger picture.

Bibliography

[1] İ. Görgöz and M. Cebeci, "Analysis of electric field and temperature distributions of non-uniformly contaminated silicone composite insulators using deep learning," *IEEE Access*, vol. 13, pp. 94 721–94 739, 2025. DOI: 10.1109/ACCESS.2025.3574493.

[2] O. Hjortstam, C. Björnson, F. Ågren, T. Hammarström, Y. V. Serdyuk, and C. Häger, *Usage of physics-informed neural network to extract physical parameters from high voltage experiments*, Conference Paper, 2024.

[3] R. J. J. B. Fila, S. H. Attri, and V. Sharma, "Mitigating overfitting in deep learning: Insights from bayesian regularization," in *2024 IEEE Region 10 Symposium (TENSYMP)*, pp. 1–6, ISBN: 2642-6102. DOI: 10.1109/TENSYMP61132.2024.10752110.

[4] S. Aburass and M. A. Rumman, "Quantifying overfitting: Introducing the overfitting index," in *2024 International Conference on Electrical, Computer and Energy Technologies (ICECET*, 2024, pp. 1–7. DOI: 10.1109/ICECET61485.2024.10698575.

[5] E. Jackson, M. Davy, A. Doucet, and W. J. Fitzgerald, "Bayesian unsupervised signal classification by dirichlet process mixtures of gaussian processes," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, vol. 3, 2007, pp. III-1077-III–1080. DOI: 10.1109/ICASSP.2007.366870.

[6] D. Shen, H. Wu, L. Liu, and D. Gan, "Solving probabilistic power flow with wind generation by polynomial chaos expansion method from the perspective of parametric problems," in *2020 IEEE Power Energy Society General Meeting (PESGM)*, 2020, pp. 1–5. DOI: 10.1109/PESGM41954.2020.9281771.

[7] A. Zare-Bazghaleh and E. Fallah Choolabi, "Incompatibility of finite element method relations with electromagnetic continuity conditions and its impact on energy conversion calculations," *IEEE Transactions on Plasma Science*, vol. 49, no. 6, pp. 2015–2023, 2021. DOI: 10.1109/TPS.2021.3076912.

[8] COMSOL AB. "Introduction to surrogate modeling," Accessed: Jan. 29, 2026. [Online]. Available: https://www.comsol.com/support/learning-center/course/introduction-to-surrogate-modeling-261/introduction-to-surrogate-modeling-94521.

A

Raw data - Neurons and layers

This page is intentionally left blank.

Table A.1: Neurons and layers results for geometry 1

Random seed	Layers	Neurons	Error	Training time [s]	Epochs	Time per epoch [ms]
0	1	4	81 679.25073	2	10 000	0.2
		6	28 481.57299	22	95 000	0.2315789474
		8	33 271.60952	3	14 400	0.2083333333
		10	10 393.26705	101	500 000	0.202
		12	10 137.55395	93	450 000	0.2066666667
		14	8 749.457126	28	130 000	0.2153846154
		16	10 255.73011	16	75 000	0.2133333333
	2	4	19 855.47783	38	150 000	0.2533333333
		6	8 109.500601	26	80 000	0.325
		8	9 419.872611	16	50 000	0.32
		10	6 831.251715	11	35 000	0.3142857143
		12	7 585.578422	8	24 000	0.3333333333
		14	8 025.83329	77	250 000	0.308
		16	7 001.856897	30	100 000	0.3
	3	4	8 769.20749	19	70 000	0.2714285714
		6	11 628.41348	44	140 000	0.3142857143
		8	7 957.512174	69	200 000	0.345
		10	8 105.800392	137	375 000	0.3653333333
		12	9 789.024466	11	30 000	0.3666666667
		14	10 662.55129	18	45 000	0.4
		16	10 114.34625	14	35 000	0.4
	4	4	8 029.445809	126	380 000	0.3315789474
		6	6 171.061497	41	100 000	0.41
		8	6 526.714334	15	34 000	0.4411764706
		10	10 669.1143	16	34 000	0.705882353
		12	7 794.870108	86	190 000	0.4526315789
		14	8 845.337755	84	170 000	0.4941176471
		16	22 479.9911	14	30 000	0.4666666667
	5	4	9 176.164776	75	200 000	0.375
		6	7 628.237018	143	300 000	0.4766666667
		8	10 132.62059	48	100 000	0.48
		10	13 620.93976	14	25 000	0.6
		12	8 119.667481	112	203 000	0.5517241379
		14	11 296.017	21	35 000	0.6
		16	13 669.30869	48	85 000	0.5647058824

Random seed	Layers	Neurons	Error	Training time [s]	Epochs	Time per epoch [ms]
2854	3	4	8 792.10	81	300 000	0.27
		6	6 896.593362	24	70 000	0.3428571429
		8	6 477.653896	30	90 000	0.3333333333
		10	9 309.833511	29	50 000	0.58
		12	10 089.5986	33	65 000	0.5076923077
		14	24 575.19074	49	100 000	0.49
		16	7 456.876021	31	80 000	0.3875
	4	4	6 693.280212	67	200 000	0.335
		6	7 638.71717	29	70 000	0.4142857143
		8	10 832.35893	24	55 000	0.4363636364
		10	5 988.906411	46	100 000	0.46
		12	6 331.192621	37	80 000	0.4625
		14	7 675.350155	30	60 000	0.5
		16	6 695.446214	17	35 000	0.4857142857
	5	4	7 984.171842	38	100 000	0.38
		6	7 239.060713	29	60 000	0.4833333333
		8	7 674.307786	78	160 000	0.4875
		10	6 540.489278	35	65 000	0.5384615385
		12	9 057.648702	80	150 000	0.5333333333
		14	7 945.816509	37	60 000	0.6166666667
		16	7 471.813702	17	30 000	0.5666666667
2751	3	4	8 056.053624	112	400 000	0.28
		6	8 698.5631	25	70 000	0.3571428571
		8	12 038.68764	35	100 000	0.35
		10	9 853.019842	55	150 000	0.3666666667
		12	37 089.08195	23	60 000	0.3833333333
		14	6 553.396066	40	100 000	0.4
		16	6 233.297683	39	100 000	0.39
	4	4	7 635.836562	69	200 000	0.345
		6	8 154.385323	82	190 000	0.4315789474
		8	8 508.466372	29	65 000	0.4461538462
		10	6 718.035427	48	100 000	0.48
		12	14 406.24864	32	65 000	0.4923076923
		14	19 858.75122	16	30 000	0.5333333333
		16	6 931.161519	20	40 000	0.5
	5	4	6 695.072815	80	200 000	0.4
		6	11 591.3761	40	80 000	0.5
		8	10 886.68912	31	60 000	0.5166666667
		10	8 568.605487	43	80 000	0.5375
		12	20 634.24338	43	70 000	0.6142857143
		14	26 898.51297	31	50 000	0.62
		16	10 436.95358	50	85 000	0.5882352941

Random seed	Layers	Neurons	Error	Training time [s]	Epochs	Time per epoch [ms]
4827	3	4	14 740.42062	47	166 000	0.2831325301
		6	5 807.753438	157	449 000	0.3496659243
		8	7 455.803109	102	288 000	0.3541666667
		10	6 031.003233	41	116 000	0.3534482759
		12	8 828.420017	151	396 000	0.3813131313
		14	6 387.800874	61	154 000	0.3961038961
		16	14 259.03223	159	421 000	0.377672209
	4	4	10 800.46295	158	482 000	0.3278008299
		6	7 622.925947	225	498 000	0.4518072289
		8	6 320.126581	87	200 000	0.435
		10	12 388.70453	215	448 000	0.4799107143
		12	11 345.92438	31	64 000	0.484375
		14	8 083.068724	112	219 000	0.5114155251
		16	6 343.500611	110	228 000	0.4824561404
	5	4	7 727.936335	191	494 000	0.3866396761
		6	7 102.253164	219	452 000	0.4845132743
		8	8 621.716766	252	500 000	0.504
		10	7 412.691819	113	215 000	0.5255813953
		12	7 033.917827	57	102 000	0.5588235294
		14	16 875.72221	116	189 000	0.6137566138
		16	10 551.30324	82	138 000	0.5942028986
9351	3	4	7 980.225561	67	237 000	0.2827004219
		6	8 542.013814	167	493 000	0.3387423935
		8	17 554.77143	54	156 000	0.3461538462
		10	15 554.74204	180	495 000	0.3636363636
		12	10 255.24256	60	169 000	0.3550295858
		14	10 903.21054	70	188 000	0.3723404255
		16	11 428.91071	56	97 000	0.5773195876
	4	4	15 548.31181	181	500 000	0.362
		6	9 119.86842	47	113 000	0.4159292035
		8	7 100.985847	48	110 000	0.4363636364
		10	6 770.155094	42	90 000	0.4666666667
		12	43 922.6593	245	500 000	0.49
		14	10 012.4922	37	72 000	0.5138888889
		16	40 052.46559	258	500 000	0.516
	5	4	10 093.5623	207	500 000	0.414
		6	7 746.612163	98	194 000	0.5051546392
		8	5 889.397253	35	65 000	0.5384615385
		10	7 017.90567	27	46 000	0.5869565217
		12	8 628.904913	29	53 000	0.5471698113
		14	5 620.320276	34	55 000	0.6181818182
		16	6 792.127796	43	73 000	0.5890410959

Random seed	Layers	Neurons	Error	Training time [s]	Epochs	Time per epoch [ms]
7284	3	4	13 853.15848	137	496 000	0.2762096774
		6	10 070.74972	22	66 000	0.3333333333
		8	23 248.87094	173	500 000	0.346
		10	10 322.79032	13	37 000	0.3513513514
		12	7 941.599335	26	71 000	0.3661971831
		14	11 825.81921	14	34 000	0.4117647059
		16	36 769.55262	61	158 000	0.3860759494
	4	4	6 072.972913	183	492 000	0.3719512195
		6	7 983.858716	37	90 000	0.4111111111
		8	9 231.792892	97	237 000	0.4092827004
		10	9 346.871134	78	181 000	0.4309392265
		12	13 894.60327	26	58 000	0.4482758621
		14	7 622.466792	23	47 000	0.4893617021
		16	7 073.118124	20	41 000	0.487804878
	5	4	8 628.093648	93	215 000	0.4325581395
		6	7 737.247573	250	500 000	0.5
		8	11 159.30105	143	276 000	0.518115942
		10	7 982.606091	48	89 000	0.5393258427
		12	7 527.482979	30	54 000	0.5555555556
		14	6 973.593048	34	56 000	0.6071428571
		16	14 004.64209	22	39 000	0.5641025641

Table A.2: Neurons and layers results for geometry 2

Random seed	Layers	Neurons	Error	Training time [s]	Epochs	Time per epoch [ms]
0	1	4	3 213 564.999	77	452 000	0.1703539823
		6	3 173 168.763	32	172 000	0.1860465116
		8	3 319 638.535	95	497 000	0.1911468813
		10	3 133 193.259	87	459 000	0.1895424837
		12	3 907 556.781	39	200 000	0.195
		14	3 631 941.629	106	500 000	0.212
		16	7 119 831.459	63	299 000	0.2107023411
	2	4	2 849 491.839	102	484 000	0.2107438017
		6	2 930 494.839	128	500 000	0.256
		8	4 867 031.95	135	500 000	0.27
		10	4 082 768.668	143	500 000	0.286
		12	3 202 342.892	35	117 000	0.2991452991
		14	3 071 302.655	70	234 000	0.2991452991
		16	3 384 671.328	99	334 000	0.2964071856

Random seed	Layers	Neurons	Error	Training time [s]	Epochs	Time per epoch [ms]
0	3	4	2 963 393.325	69	250 000	0.276
		6	7 599 210.485	159	463 000	0.343412527
		8	4 633 681.042	176	500 000	0.352
		10	3 003 065.101	197	500 000	0.394
		12	2 966 462.54	85	222 000	0.3828828829
		14	3 041 414.145	48	117 000	0.4102564103
		16	3 034 254.439	77	190 000	0.4052631579
	4	4	2 916 350.459	195	500 000	0.39
		6	3 005 494.968	35	72 000	0.4861111111
		8	2 855 573.498	71	152 000	0.4671052632
		10	3 032 408.284	24	52 000	0.4615384615
		12	2 990 635.384	95	193 000	0.4922279793
		14	3 206 399.85	23	46 000	0.5
		16	2 652 847.527	126	250 000	0.504
	5	4	3 057 531.684	223	500 000	0.446
		6	2 972 692.382	240	445 000	0.5393258427
		8	2 856 483.853	77	143 000	0.5384615385
		10	3 235 583.41	113	196 000	0.5765306122
		12	4 843 449.184	72	121 000	0.5950413223
		14	4 302 789.793	83	133 000	0.6240601504
		16	3 462 513.538	106	184 000	0.5760869565
2854	3	4	2 976 759.984	84	300 000	0.28
		6	2 962 566.455	101	300 000	0.3366666667
		8	2 970 555.504	150	461 000	0.3253796095
		10	5 354 157.263	74	200 000	0.37
		12	3 275 820.508	54	137 000	0.3941605839
		14	2 981 979.209	37	89 000	0.4157303371
		16	3 228 621.997	46	114 000	0.4035087719
	4	4	2 969 848.481	55	147 000	0.3741496599
		6	2 915 921.81	69	150 000	0.46
		8	3 157 055.59	192	462 000	0.4155844156
		10	3 136 622.387	81	183 000	0.4426229508
		12	3 237 282.811	68	153 000	0.4444444444
		14	4 139 927.536	62	129 000	0.480620155
		16	3 388 805.099	61	129 000	0.4728682171
	5	4	3 078 749.746	60	136 000	0.4411764706
		6	2 871 689.398	81	150 000	0.54
		8	2 983 856.565	68	126 000	0.5396825397
		10	3 210 140.184	71	127 000	0.5590551181
		12	2 962 093.854	10	17 000	0.5882352941
		14	3 758 856.209	93	147 000	0.6326530612
		16	2 972 490.538	64	111 000	0.5765765766

Random seed	Layers	Neurons	Error	Training time [s]	Epochs	Time per epoch [ms]
2751	3	4	3 015 393.838	17	65 000	0.2615384615
		6	2 962 448.312	50	163 000	0.3067484663
		8	4 155 237.659	28	88 000	0.3181818182
		10	2 948 202.842	67	193 000	0.3471502591
		12	2 925 799.036	13	37 000	0.3513513514
		14	2 963 511.431	38	97 000	0.3917525773
		16	3 019 867.547	34	89 000	0.3820224719
	4	4	2 947 083.304	54	141 000	0.3829787234
		6	2 962 347.042	44	98 000	0.4489795918
		8	2 837 992.248	71	150 000	0.4733333333
		10	2 915 956.104	45	96 000	0.46875
		12	2 588 667.611	51	107 000	0.476635514
		14	2 924 397.374	50	104 000	0.4807692308
		16	2 830 476.992	69	149 000	0.4630872483
	5	4	2 976 743.187	62	139 000	0.4460431655
		6	4 066 571.037	86	150 000	0.5733333333
		8	3 240 370.349	68	122 000	0.5573770492
		10	2 736 695.087	85	145 000	0.5862068966
		12	3 016 255.957	21	35 000	0.6
		14	4 118 980.456	78	122 000	0.6393442623
		16	2 889 186.737	73	118 000	0.6186440678
4827	3	4	3 585 944.785	145	500 000	0.9
		6	3 075 532.474	174	496 000	0.3508064516
		8	3 258 834.147	172	494 000	0.3481781377
		10	2 972 675.563	120	338 000	0.3550295858
		12	2 947 592.238	39	111 000	0.3513513514
		14	2 670 505.57	69	179 000	0.3854748603
		16	3 333 016.652	12	33 000	0.3636363636
	4	4	3 003 331.484	67	195 000	0.3435897436
		6	2 959 037.006	13	32 000	0.40625
		8	2 875 899.859	55	129 000	0.4263565891
		10	3 060 702.534	71	157 000	0.4522292994
		12	2 619 790.068	68	151 000	0.4503311258
		14	3 038 354.818	75	155 000	0.4838709677
		16	4 647 580.015	90	197 000	0.4568527919
	5	4	2 975 415.937	80	200 000	0.4
		6	2 985 113.063	98	197 000	0.4974619289
		8	2 900 896.413	95	191 000	0.497382199
		10	4 228 593.147	103	193 000	0.5336787565
		12	2 810 302.475	101	183 000	0.5519125683
		14	3 261 134.772	105	182 000	0.5769230769
		16	3 187 161.747	112	193 000	0.5803108808

Random seed	Layers	Neurons	Error	Training time [s]	Epochs	Time per epoch [ms]
9351	3	4	2 979 781.871	47	170 000	0.2764705882
		6	3 010 797.237	171	500 000	0.342
		8	2 921 386.657	171	500 000	0.342
		10	3 040 772.928	132	373 000	0.3538873995
		12	3 340 658.618	144	400 000	0.36
		14	3 306 962.352	138	361 000	0.3822714681
		16	2 872 159.466	78	200 000	0.39
	4	4	3 102 789.068	86	249 000	0.3453815261
		6	3 353 505.628	85	200 000	0.425
		8	2 945 046.689	113	275 000	0.4109090909
		10	2 995 696.914	76	167 000	0.4550898204
		12	2 981 157.493	52	112 000	0.4642857143
		14	2 479 596.741	62	122 000	0.5081967213
		16	2 877 968.033	74	119 000	0.6218487395
	5	4	2 924 397.374	68	166 000	0.4096385542
		6	3 469 582.107	82	167 000	0.4910179641
		8	2 958 259.623	65	130 000	0.5
		10	4 453 425.648	70	128 000	0.546875
		12	3 383 637.096	122	216 000	0.5648148148
		14	5 203 268.204	85	141 000	0.6028368794
		16	3 674 098.529	112	193 000	0.5803108808
7284	3	4	2966782.769	77	277 000	0.2779783394
		6	2 963 410.198	166	500 000	0.332
		8	2 927 046.293	171	500 000	0.342
		10	3 091 698.562	15	44 000	0.3409090909
		12	3 129 824.276	96	268 000	0.3582089552
		14	4 405 678.154	88	229 000	0.384279476
		16	2 955 621.762	71	193 000	0.3678756477
	4	4	2 992 156.413	92	164 000	0.5609756098
		6	2 899 379.244	132	322 000	0.4099378882
		8	2 919 040.938	93	229 000	0.4061135371
		10	5 404 350.1	65	146 000	0.4452054795
		12	2 896 480.623	129	292 000	0.4417808219
		14	6 354 604.63	72	142 000	0.5070422535
		16	3 691 070.305	94	200 000	0.47
	5	4	2 912 850.837	108	283 000	0.3816254417
		6	2 988 243.631	23	52 000	0.4423076923
		8	3 021 837.19	13	29 000	0.4482758621
		10	2 922 088.294	67	135 000	0.4962962963
		12	2 786 646.73	68	133 000	0.5112781955
		14	5 987 570.459	73	125 000	0.584
		16	4 030 012.407	64	117 000	0.547008547

B

Raw data - Neurons and layers, different shapes

This page is intentionally left blank.

Table B.1: Shape results for geometry 1

Random seed	Shape	Error	Training time [s]	Epochs	Time per epoch [ms]
0	64-32-16	8 995.220953	23	32 000	0.71875
	32-16-8	6 868.842697	11	22 000	0.5
	16-8-4	16 762.45805	65	193 000	0.3367875648
	8-6-4	8 530.357554	180	498 000	0.3614457831
	64-32-64	8 707.755164	23	28 000	0.8214285714
	32-16-32	19 602.80592	141	250 000	0.564
	16-8-16	9 980.731436	32	82 000	0.3902439024
	8-6-8	7 245.964394	11	30 000	0.3666666667
	6-4-6	6 709.396396	166	492 000	0.337398374
	16-32-64	10 816.65383	9	14 000	0.6428571429
	8-16-32	8 756.883007	40	90 000	0.4444444444
	4-8-16	9 917.459352	28	83 000	0.3373493976
	4-6-8	6 675.926902	59	185 000	0.3189189189
	32-64-32	19 797.22203	19	26 000	0.7307692308
	16-32-16	9 958.915604	15	27 000	0.5555555556
	8-16-8	11 523.02044	11	31 000	0.3548387097
	6-8-6	6 727.480955	15	44 000	0.3409090909
	4-6-4	6 675.926902	55	185 000	0.2972972973
2854	64-32-16	6 546.984038	26	37 000	0.7027027027
	32-16-8	7 599.276281	21	43 000	0.488372093
	16-8-4	8 868.539902	64	190 000	0.3368421053
	8-6-4	6 722.573912	180	500 000	0.36
	64-32-64	13 272.52802	47	56 000	0.8392857143
	32-16-32	6 314.269554	96	170 000	0.5647058824
	16-8-16	15 476.4337	39	94 000	0.414893617
	8-6-8	7 556.454195	187	495 000	0.3777777778
	6-4-6	6 538.042521	188	500 000	0.376
	16-32-64	6 022.707033	27	40 000	0.675
	8-16-32	7 500.799957	87	194 000	0.4484536082
	4-8-16	6 450.348828	67	200 000	0.335
	4-6-8	8 633.828815	184	500 000	0.368
	32-64-32	7 045.778878	21	29 000	0.724137931
	16-32-16	8 224.354078	43	80 000	0.5375
	8-16-8	27 515.45021	48	133 000	0.3609022556
	6-8-6	6 845.801633	18	49 000	0.3673469388
	4-6-4	6 869.061071	169	492 000	0.343495935

Random seed	Shape	Error	Training time [s]	Epochs	Time per epoch [ms]
2751	64-32-16	16 066.11341	17	25 000	0.68
	32-16-8	8 232.618053	51	45 000	1.133333333
	16-8-4	8 804.317123	29	80 000	0.3625
	8-6-4	12 751.07839	153	500 000	0.306
	64-32-64	9 808.465731	32	28 000	1.142857143
	32-16-32	14 797.97283	45	80 000	0.5625
	16-8-16	7 734.339015	31	80 000	0.3875
	8-6-8	6 720.044643	167	500 000	0.334
	6-4-6	7 358.872196	159	500 000	0.318
	16-32-64	7 549.569524	25	35 000	0.7142857143
	8-16-32	12 846.01105	39	84 000	0.4642857143
	4-8-16	16 799.70238	66	200 000	0.33
	4-6-8	7 158.910532	155	500 000	0.31
	32-64-32	12 670.04341	91	100 000	0.91
	16-32-16	17 628.66983	43	79 000	0.5443037975
	8-16-8	8 088.695816	29	74 000	0.3918918919
	6-8-6	6 263.784798	73	214 000	0.3411214953
	4-6-4	9 279.924569	153	500 000	0.306

Table B.2: Shape results for geometry 2

Random seed	Shape	Error	Training time [s]	Epochs	Time per epoch [ms]
0	64-32-16	6 259 392.942	29	43 400	0.668202765
	32-16-8	2 923 627.883	11	24 400	0.4508196721
	16-8-4	3 136 032.525	33	100 000	0.33
	8-6-4	2 978 153.791	37	111 000	0.3333333333
	64-32-64	3 510 555.512	29	38 000	0.7631578947
	32-16-32	3 023 739.407	20	38 000	0.5263157895
	16-8-16	2 907 971.802	59	173 000	0.3410404624
	8-6-8	3 039 342.034	57	173 000	0.3294797688
	6-4-6	2 979 228.088	39	126 000	0.3095238095
	16-32-64	3 012 673.232	78	124 000	0.6290322581
	8-16-32	2 811 938.833	78	170 000	0.4588235294
	4-8-16	2 889 039.633	97	300 000	0.3233333333
	4-6-8	2 979 026.687	87	283 000	0.3074204947
	32-64-32	3 056 043.193	58	85 000	0.6823529412
	16-32-16	3 559 072.913	48	95 000	0.5052631579
	8-16-8	3 035 160.622	14	41 000	0.3414634146
	6-8-6	3 109 083.466	36	109 000	0.3302752294
	4-6-4	2 995 146.073	146	500 000	0.292
2854	64-32-16	2 808 131.051	19	30 000	0.6333333333
	32-16-8	2 804 835.111	52	113 000	0.4601769912
	16-8-4	2 994 545.041	134	407 000	0.3292383292
	8-6-4	3 020 910.459	160	500 000	0.32
	64-32-64	3 015 659.132	41	55 000	0.7454545455
	32-16-32	4 297 790.13	51	90 000	0.5666666667
	16-8-16	2 693 807.714	49	136 000	0.3602941176
	8-6-8	2 927 080.457	171	474 000	0.3607594937
	6-4-6	3 214 576.177	173	496 000	0.3487903226
	16-32-64	2 912 164.144	66	100 000	0.66
	8-16-32	5 482 973.646	88	186 000	0.4731182796
	4-8-16	3 028 200.786	173	500 000	0.346
	4-6-8	2 939 574.799	181	500 000	0.362
	32-64-32	3 536 947.837	24	31 000	0.7741935484
	16-32-16	2 947 795.787	41	77 000	0.5324675325
	8-16-8	2 905 959.394	35	97 000	0.3608247423
	6-8-6	3 073 515.902	132	399 000	0.3308270677
	4-6-4	5 428 259.39	73	245 000	0.2979591837

Random seed	Shape	Error	Training time [s]	Epochs	Time per epoch [ms]
2751	64-32-16	2 606 511.078	32	49 000	0.6530612245
	32-16-8	2 917 327.544	92	200 000	0.46
	16-8-4	3 062 629.589	66	191 000	0.3455497382
	8-6-4	4 829 596.256	161	500 000	0.322
	64-32-64	2 833 160.779	64	83 000	0.7710843373
	32-16-32	2 727 471.356	172	295 000	0.5830508475
	16-8-16	3 018 012.591	150	362 000	0.4143646409
	8-6-8	2 921 283.964	160	490 000	0.3265306122
	6-4-6	2 981 459.374	35	116 000	0.3017241379
	16-32-64	2 439 856.553	96	137 000	0.700729927
	8-16-32	3 028 613.544	123	264 000	0.4659090909
	4-8-16	3 052 097.639	127	369 000	0.3441734417
	4-6-8	3 045 406.377	133	388 000	0.3427835052
	32-64-32	3 295 754.845	11	15 000	0.7333333333
	16-32-16	2 892 230.973	60	112 000	0.5357142857
	8-16-8	2 888 321.312	183	486 000	0.3765432099
	6-8-6	2 969 663.281	178	500 000	0.356
	4-6-4	2 991 738.625	122	403 000	0.3027295285

C

Raw data - Learning rate

This page is intentionally left blank.

Table C.1: Learning rate results for geometry 1

Random seed	Learning rate	Error	Training time [s]	Epochs	Time per epoch [ms]
0	10^{-1}	15 978.42295	10	6 300	1.587301587
	10^{-2}	8 692.698085	60	38 206	1.570433963
	10^{-3}	8 645.172063	55	33 500	1.641791045
	10^{-4}	8 386.41759	44	29 500	1.491525424
	10^{-5}	13 885.52376	187	120 000	1.558333333
	10^{-6}	86 295.42282	315	201 000	1.567164179
	10^{-7}	553 561.1981	767	500 000	1.534
2854	10^{-1}	265 307.3689	3	1 505	1.993355482
	10^{-2}	6 968.644057	22	17 000	1.294117647
	10^{-3}	6 668.433099	97	64 000	1.515625
	10^{-4}	10 851.26721	223	146 000	1.55739726
	10^{-5}	9 488.413988	404	170 000	2.376470588
	10^{-6}	9 814.020583	794	473 000	1.6878646934
2751	10^{-1}	13 824.25405	6	2 793	2.148227712
	10^{-2}	8 512.167762	76	46 467	1.635569329
	10^{-3}	7 264.433908	238	128 000	1.859375
	10^{-4}	7 410.330627	164	82 000	2
	10^{-5}	10 056.83847	440	240 000	1.833333333
	10^{-6}	10 056.83847	866	500 000	1.732

Table C.2: Learning rate results for geometry 2

Random seed	Learning rate	Error	Training time [s]	Epochs	Time per epoch [ms]
0	10^{-1}	3 324 154.028	4	3 000	1.333333333
	10^{-2}	4 293 250.517	310	191 893	1.615483629
	10^{-3}	3 148 189.956	62	34 000	1.8235229412
	10^{-4}	3 364 223.536	57	34 000	1.676470588
	10^{-5}	4 729 059.103	147	91 000	1.615384615
	10^{-6}	3 403 968.272	792	500 000	1.584
	10^{-7}	29 848 450.55	853	500 000	1.706
2854	10^{-1}	50 855 678.15	2	544	3.676470588
	10^{-2}	2 902 188.829	358	230 315	1.554392897
	10^{-3}	2 932 422.207	149	98 000	1.520408163
	10^{-4}	3 845 646.89	836	500 000	1.672
	10^{-5}	3 965 602.098	181	109 000	1.660550459
	10^{-6}	3 621 877.966	868	500 000	1.736
2751	10^{-1}	3 835 492.146	5	2 612	1.91424196
	10^{-2}	2 702 054.774	59	37 224	1.584998925
	10^{-3}	2 906 217.473	137	80 000	1.7125
	10^{-4}	4 083 503.398	54	38 200	1.413612565
	10^{-5}	4 329 896.073	158	97 000	1.628865979
	10^{-6}	3 239 290.046	541	314 000	1.722929936

D

Raw data - Weight decay

This page is intentionally left blank.

Table D.1: Weight decay results for geometry 1

Random seed	Weight decay	Error	Training time [s]	Epochs	Time per epoch [ms]
0	10^{-1}	383 927.0764	< 1	405	0
	10^{-2}	269 603.4124	1	1 600	0.625
	10^{-3}	87 666.41318	1	2 700	0.3703703704
	10^{-4}	33 615.47263	4	7 100	0.5633802817
	10^{-5}	13 570.9248	7	13 600	0.5147058824
	10^{-6}	9 189.885745	39	72 900	0.5349794239
	10^{-7}	7 996.74934	70	150 000	0.4666666667
	0	9 969.353038	16	33 700	0.4747774481
2854	10^{-1}	441 655.9747	< 1	221	0
	10^{-2}	275 221.7288	< 1	800	0
	10^{-3}	98 459.12858	1	1 800	0.5555555556
	10^{-4}	30 624.01019	22	43 800	0.502283105
	10^{-5}	9 488.361292	24	48 200	0.4979253112
	10^{-6}	6 301.74579	9	19 600	0.4591836735
	10^{-7}	10 162.1848	69	150 000	0.46
	0	5 796.033126	48	99 400	0.4828973843
2751	10^{-1}	440 306.7113	< 1	393	0
	10^{-2}	225 191.03	< 1	300	0
	10^{-3}	85 295.95536	1	2 500	0.4
	10^{-4}	29 766.92796	20	45 000	0.4444444444
	10^{-5}	9 951.482302	69	148 100	0.465901418
	10^{-6}	8 221.982729	52	117 900	0.4410517388
	10^{-7}	6 674.953183	168	380 800	0.4411764706
	0	6 287.447813	64	143 500	0.4459930314
4827	10^{-5}	10 731.72866	10	19 100	0.5235602094
	10^{-6}	9 001.944234	110	250 000	0.44
	10^{-7}	8 567.671796	110	250 000	0.44
	0	7 844.424772	112	250 000	0.448
9351	10^{-5}	11 025.42516	135	300 000	0.45
	10^{-6}	6 208.542502	78	168 000	0.4642857143
	10^{-7}	8 953.26756	54	112 200	0.4812834225
	0	6 769.268794	42	89 800	0.4677060134
7284	10^{-5}	10 386.52974	188	293 700	0.6401089547
	10^{-6}	9 768.776791	138	300 000	0.46
	10^{-7}	13 493.33169	89	192 100	0.4633003644
	0	9 346.871134	82	181 400	0.4520396913

Table D.2: Weight decay results for geometry 2

Random seed	Weight decay	Error	Training time [s]	Epochs	Time per epoch [ms]
0	10^{-1}	23 093 072.55	< 1	400	0
	10^{-2}	15 238 766.35	15	35 700	0.4201680672
	10^{-3}	4 660 150.212	7	16 100	0.4347826087
	10^{-4}	3 446 302.366	14	31 300	0.447284345
	10^{-5}	3 293 933.818	35	79 100	0.4424778761
	10^{-6}	4 504 997.225	107	237 300	0.4509060261
	10^{-7}	3 204 372.013	23	53 100	0.4331450094
	0	3 033 891.89	23	51 600	0.4457364341
2854	10^{-1}	26 356 972.51	< 1	240	0
	10^{-2}	16 788 091.02	2	4 300	0.4651162791
	10^{-3}	4 715 506.335	16	36 200	0.4419889503
	10^{-4}	3 479 655.155	37	80 900	0.457354759
	10^{-5}	3 216 364.407	10	20 800	0.4807692308
	10^{-6}	4 135 698.248	97	200 000	0.485
	10^{-7}	3 149 682.524	92	191 530	0.4803425051
	0	3 117 948.043	80	183 200	0.4366812227
2751	10^{-1}	25 256 088.37	< 1	370	0
	10^{-2}	15 183 214.42	1	1 100	0.9090909091
	10^{-3}	5 068 629.006	2	4 100	0.487804878
	10^{-4}	3 348 581.789	33	70 700	0.4667609618
	10^{-5}	3 292 719.241	51	107 100	0.4761904762
	10^{-6}	4 699 148.859	11	23 400	0.4700854701
	10^{-7}	3 018 592.387	94	199 100	0.4721245605
	0	2 910 790.271	45	95 700	0.4702194357
4827	10^{-5}	3 274 751.899	83	183 400	0.4525627045
	10^{-6}	4 895 916.666	72	147 800	0.4871447903
	10^{-7}	3 114 578.623	53	111 500	0.4753363229
	0	2 854 365.078	139	294 900	0.4713462191
9351	10^{-5}	3 216 675.302	24	45 800	0.5240174672
	10^{-6}	2 999 533.297	133	282 500	0.4707964602
	10^{-7}	3 028 613.544	131	279 700	0.468358956
	0	2 995 696.914	77	167 300	0.460251046
7284	10^{-5}	3 325 958.508	18	37 000	0.4864864865
	10^{-6}	4 296 044.693	21	42 600	0.4929577465
	10^{-7}	2 994 378.066	49	102 600	0.477582846
	0	3 402 792.97	173	382 400	0.4524058577

E

Raw data - Batch size

This page is intentionally left blank.

Table E.1: Batch size results for geometry 1 using different random seeds

Random seed	Datapoints	Batch size	Error	Training time [s]	Epochs	Time per epoch [ms]
0	200	200	6 936.713919	17	49 900	0.3406813627
		195	6 936.713919	17	49 900	0.3406813627
		190	6 936.713919	18	49 900	0.3607214429
		185	6 936.713919	18	49 900	0.3607214429
		180	6 936.713919	17	49 900	0.3406813627
		175	9 683.491106	29	49 900	0.5811623246
		150	6 644.170377	28	49 800	0.562248996
		125	7 744.67559	28	49 200	0.5691056911
		100	7 631.448093	26	44 900	0.579064588
		75	11 559.41175	35	43 300	0.8083140878
		50	9 075.516514	49	48 600	1.008230453
		25	8 697.528385	45	23 500	1.914893617
	400	400	9 969.353038	15	33 700	0.4451038576
		390	9 969.353038	15	33 700	0.4451038576
		380	9 969.353038	15	33 700	0.4451038576
		370	9 969.353038	15	33 700	0.4451038576
		360	9 969.353038	15	33 700	0.4451038576
		350	19 906.02924	70	97 000	0.7216494845
		300	9 220.086767	40	55 800	0.7168458781
		250	8 219.367372	27	40 900	0.6601466993
		200	8 190.177043	23	36 300	0.6336088154
		150	9 088.564243	27	30 400	0.8881578947
		100	9 757.253712	41	36 600	1.120218579
		50	7 047.765603	22	10 400	2.115384615
	600	600	7 033.633485	36	74 000	0.4864864865
		585	7 033.633485	36	74 000	0.4864864865
		570	7 033.633485	36	74 000	0.4864864865
		555	7 033.633485	36	74 000	0.4864864865
		540	7 033.633485	36	74 000	0.4864864865
		525	7 125.587695	25	33 300	0.750750750
		450	5 677.235243	42	56 200	0.7473309609
		375	6 109.909983	67	89 200	0.7511210762
		300	5 949.453756	77	97 800	0.7873210634
		225	5 914.304693	28	26 000	1.076923077
		150	7 027.090436	10	8 900	1.123595506
		75	5 634.80257	29	14 100	2.056737589

Random seed	Datapoints	Batch size	Error	Training time [s]	Epochs	Time per epoch [ms]
2854	200	200	8 428.404357	14	44 000	0.3181818182
		195	8 428.404357	14	44 000	0.3181818182
		190	8 428.404357	14	44 000	0.3181818182
		185	8 428.404357	14	44 000	0.3181818182
		180	8 428.404357	14	44 000	0.3181818182
		175	6 272.001276	48	86 900	0.5523590334
		150	9 924.061668	12	21 900	0.5479452055
		125	9 381.524396	9	17 000	0.5294117647
		100	9 744.382997	9	17 000	0.5294117647
		75	10 181.35551	11	14 000	0.7857142857
		50	8 978.028737	11	10 700	1.028037383
		25	6 510.529932	58	29 500	1.966101695
	400	400	5 776.504133	47	99 500	0.472361809
		390	5 776.504133	46	99 500	0.4623115578
		380	5 776.504133	46	99 500	0.4623115578
		370	5 776.504133	45	99 500	0.4522613065
		360	5 776.504133	44	99 500	0.4422110553
		350	6 452.363908	63	89 200	0.7062780269
		300	5 840.54792	65	91 200	0.7127192982
		250	8 690.512068	22	31 000	0.7096774194
		200	8 474.491135	17	27 500	0.6181818182
		150	7 089.569804	80	93 200	0.8583690987
		100	6 847.992407	103	92 600	1.112311015
		50	15 189.79921	171	83 400	2.050359712
	600	600	6 900.579686	9	18 000	0.5
		585	6 900.579686	9	18 000	0.5
		570	6 900.579686	9	18 000	0.5
		555	6 900.579686	9	18 000	0.5
		540	6 900.579686	9	18 000	0.5
		525	6 086.131776	71	95 100	0.7465825447
		450	6 209.428315	41	54 900	0.7468123862
		375	7 027.019283	59	78 100	0.7554417414
		300	6 231.452479	73	90 600	0.8057395143
		225	6 519.355796	58	54 900	1.056466302
		150	6 153.373059	72	61 800	1.165048544
		75	6 396.092557	132	61 200	2.156862745

Random seed	Datapoints	Batch size	Error	Training time [s]	Epochs	Time per epoch [ms]
2751	200	200	7 315.873154	26	74 600	0.3485254692
		195	7 315.873154	26	74 600	0.3485254692
		190	7 315.873154	26	74 600	0.3485254692
		185	7 315.873154	26	74 600	0.3485254692
		180	7 315.873154	26	74 600	0.3485254692
		175	10 987.26536	23	39 300	0.5852417303
		150	8 270.671073	4	6 400	0.625
		125	7 719.261623	4	6 300	0.6349206349
		100	8 274.297553	3	5 700	0.5263157895
		75	8 714.585475	3	4 000	0.75
		50	10 032.94573	3	2 800	1.071428571
		25	8 573.680657	17	8 200	2.073170732
	400	400	6 287.447813	64	143 500	0.4459930314
		390	6 287.447813	64	143 500	0.4459930314
		380	6 287.447813	64	143 500	0.4459930314
		370	6 287.447813	64	143 500	0.4459930314
		360	6 287.447813	64	143 500	0.4459930314
		350	9 922.247729	74	103 600	0.7142857143
		300	8 173.126697	68	96 300	0.7061266874
		250	6 393.27772	68	99 800	0.6813627255
		200	6 313.952803	29	44 200	0.6561085973
		150	8 667.087169	76	85 400	0.8899297424
		100	9 061.898256	62	54 500	1.137614679
		50	8 721.467766	67	32 100	2.087227414
	600	600	6 274.551777	21	39 800	0.527638191
		585	6 274.551777	21	39 800	0.527638191
		570	6 274.551777	21	39 800	0.527638191
		555	6 274.551777	21	39 800	0.527638191
		540	6 274.551777	21	39 800	0.527638191
		525	6 334.666526	14	17 900	0.782122905
		450	6 061.517962	15	19 400	0.7731958763
		375	6 279.012661	12	14 600	0.8219178082
		300	6 093.603203	39	47 000	0.829787234
		225	6 144.021484	15	13 500	1.111111111
		150	6 314.269554	11	9 300	1.182795699
		75	6 292.932544	11	4 800	2.291666667

Table E.2: Batch size results for geometry 2 using different random seeds

Random seed	Datapoints	Batch size	Error	Training time [s]	Epochs	Time per epoch [ms]
0	200	200	4 314 626.287	22	61 900	0.3554119548
		195	4 314 626.287	21	61 900	0.3392568659
		190	4 314 626.287	21	61 900	0.3392568659
		185	4 314 626.287	21	61 900	0.3392568659
		180	4 314 626.287	21	61 900	0.3392568659
		175	6 238 188.84	35	60 600	0.5775577558
		150	4 229 420.764	55	95 300	0.5771248688
		125	6 071 984.848	23	37 800	0.6084656085
		100	3 145 504.729	31	52 800	0.5871212121
		75	3 774 519.837	34	38 900	0.8740359897
		50	4 499 333.284	99	89 700	1.10367893
		25	5 301 131.955	162	80 300	2.01743462
	400	400	3 033 891.89	22	51 600	0.4263565891
		390	3 033 891.89	23	51 600	0.4457364341
		380	3 033 891.89	22	51 600	0.4263565891
		370	3 033 891.89	25	51 600	0.484496124
		360	3 033 891.89	26	51 600	0.503875969
		350	3 362 736.98	62	93 800	0.6609808102
		300	3 044 158.34	92	142 300	0.6465214336
		250	3 356 486.258	91	141 400	0.6435643564
		200	3 088 672.854	84	144 000	0.5833333333
		150	3 238 054.972	96	118 600	0.8094435076
		100	7 372 313.07	107	105 100	1.018078021
		50	3 278 109.211	297	150 000	1.98
	600	600	2 960 135.132	52	105 800	0.4914933837
		585	2 960 135.132	51	105 800	0.4820415879
		570	2 960 135.132	52	105 800	0.4914933837
		555	2 960 135.132	54	105 800	0.5103969754
		540	2 960 135.132	52	105 800	0.4914933837
		525	4 325 968.1	14	19 300	0.725388601
		450	2 886 936.092	40	55 800	0.7168458781
		375	2 961 131.54	6	7 500	0.8
		300	2 900 120.687	7	8 900	0.7865168539
		225	3 822 302.971	49	46 000	1.065217391
		150	3 236 510.467	51	45 700	1.115973742
		75	3 036 593.486	113	52 400	2.15648855

Random seed	Datapoints	Batch size	Error	Training time [s]	Epochs	Time per epoch [ms]
2854	200	200	6 562 011.887	7	18 600	0.376344086
		195	6 562 011.887	7	18 600	0.376344086
		190	6 562 011.887	7	18 600	0.376344086
		185	6 562 011.887	7	18 600	0.376344086
		180	6 562 011.887	7	18 600	0.376344086
		175	6 438 322.763	50	85 800	0.5827505828
		150	4 753 525.008	19	32 400	0.5864197531
		125	7 509 660.445	33	55 900	0.5903398927
		100	2 777 192.827	43	71 700	0.59972106
		75	7 334 575.652	22	25 600	0.859375
		50	3 063 919.059	60	56 400	1.063829787
		25	3 063 919.059	61	56 400	1.081560284
	400	400	3 117 948.043	80	183 200	0.4366812227
		390	3 117 948.043	78	183 200	0.4257641921
		380	3 117 948.043	81	183 200	0.442139738
		370	3 117 948.043	82	183 200	0.4475982533
		360	3 117 948.043	83	183 200	0.4530567686
		350	3 452 680.118	89	133 700	0.6656694091
		300	4 768 228.182	86	126 600	0.6793048973
		250	2 778 398.819	84	126 500	0.6640316206
		200	2 843 343.103	59	98 600	0.5983772819
		150	4 266 263.002	178	200 000	0.89
		100	3 428 702.378	64	60 300	1.061359867
		50	3 170 331.213	272	141 800	1.91819464
	600	600	2 897 654.224	50	97 600	0.512295082
		585	2 897 654.224	50	97 600	0.512295082
		570	2 897 654.224	48	97 600	0.4918032787
		555	2 897 654.224	55	97 600	0.5635245902
		540	2 897 654.224	58	97 600	0.5942622951
		525	2 687 787.194	83	99 700	0.8324974925
		450	3 844 736.662	27	32 800	0.8231707317
		375	3 850 194.8	42	51 200	0.8203125
		300	3 556 543.266	25	30 200	0.8278145695
		225	4 118 980.456	55	47 000	1.170212766
		150	3 588 175.024	98	76 100	1.287779238
		75	3 964 719.41	211	99 900	2.112112112

Random seed	Datapoints	Batch size	Error	Training time [s]	Epochs	Time per epoch [ms]
2751	200	200	2 421 136.097	10	27 700	0.3610108303
		195	2 421 136.097	10	27 700	0.3610108303
		190	2 421 136.097	10	27 700	0.3610108303
		185	2 421 136.097	10	27 700	0.3610108303
		180	2 421 136.097	10	27 700	0.3610108303
		175	2 632 204.399	44	73 300	0.6002728513
		150	3 897 435.054	6	10 200	0.5882352941
		125	3 064 082.244	50	85 100	0.5875440658
		100	3 869 625.305	5	8 600	0.5813953488
		75	5 310 461.374	66	80 500	0.8198757764
		50	5 102 646.372	44	41 200	1.067961165
		25	5 014 179.893	176	89 000	1.97752809
	400	400	2 910 790.271	44	95 700	0.4597701149
		390	2 910 790.271	44	95 700	0.4597701149
		380	2 910 790.271	45	95 700	0.4702194357
		370	2 910 790.271	43	95 700	0.4493207941
		360	2 910 790.271	43	95 700	0.4493207941
		350	3 180 094.338	74	109 400	0.676416819
		300	4 354 537.863	131	195 500	0.6700767263
		250	3 744 195.508	93	138 400	0.6719653179
		200	4 718 050.445	80	129 600	0.6172839506
		150	3 417 455.194	101	121 300	0.8326463314
		100	2 758 495.967	205	193 100	1.0616261
		50	3 176 003.778	384	192 300	1.996879875
	600	600	2 949 288.05	10	17 800	0.5617977528
		585	2 949 288.05	10	17 800	0.5617977528
		570	2 949 288.05	11	17 800	0.6179775281
		555	2 949 288.05	10	17 800	0.5617977528
		540	2 949 288.05	10	17 800	0.5617977528
		525	3 020 529.755	69	85 500	0.8070175439
		450	4 577 116.997	59	73 700	0.8005427408
		375	2 816 256.38	50	61 200	0.8169934641
		300	4 406 585.98	9	10 600	0.8490566038
		225	3 497 427.626	9	7 700	1.168831169
		150	3 993 119.082	79	66 200	1.193353474
		75	2 938 503.02	5	2 200	2.272727273

F

Raw data - Activation function

This page is intentionally left blank.

Table F.1: Activation function results for geometry 1 using different random seeds

Random seed	Activation function	Error	Training time [s]	Epochs	Time per epoch [ms]
0	tanh	8 645.172063	55	33 500	1.641791045
	Linear	439 704.4462	< 1	160	0
	ReLU	393 903.5415	< 1	1	0
	ELU	8 620.32482	64	40 000	1.6
	Sigmoid	9 402.659198	173	120 000	1.441666667
2854	tanh	6 666.783332	97	64 000	1.515625
	Linear	440 261.2861	< 1	140	0
	ReLU	393 903.5415	< 1	1	0
	ELU	6 963.906949	63	40 000	1.575
	Sigmoid	9 854.643575	80	50 000	1.6
2751	tanh	7 264.433908	238	128 000	1.859375
	Linear	455 071.423	< 1	130	0
	ReLU	23 788.0222	6	5 548	1.0814708
	ELU	8 488.10933	65	40 000	1.625
	Sigmoid	8 371.439542	80	50 000	1.6

Table F.2: Activation function results for geometry 2 using different random seeds

Random seed	Activation function	Error	Training time [s]	Epochs	Time per epoch [ms]
0	tanh	3 148 189.956	62	34 000	1.823529412
	Linear	28 106 227.07	< 1	160	0
	ReLU	6 816 010.563	< 1	1	0
	ELU	3 055 830.493	37	26 000	1.423076923
	Sigmoid	2 984 459.75	44	33 000	1.333333333
2854	tanh	2 932 422.207	149	98 000	1.520408163
	Linear	29 154 930.97	< 1	110	0
	ReLU	6 816 010.563	< 1	1	0
	ELU	3 002 598.874	98	58 000	1.689655172
	Sigmoid	2 982 465.423	118	80 000	1.475
2751	tanh	2 906 217.473	137	80 000	1.7125
	Linear	28 389 786.9	< 1	120	0
	ReLU	6 816 010.563	1	1 300	0.7692307692
	ELU	3 074 377.986	52	30 000	1.733333333
	Sigmoid	2 986 419.261	103	70 000	1.471428571

G

Raw data - Dataset size study

Table G.1: Dataset size study result for geometry 1

Dataset size	Random seed	Error	Training time [s]	Epochs	Time per epoch [ms]
10	0	537 354.6315	< 1	8	0
	2854	476 214.2375	482	498 069	0.9677374018
	2751	364 307.5624	333	343 625	0.9690796653
50	0	159 445.9156	18	17 251	1.043417773
	2854	35 571.05565	62	57 826	1.072182063
	2751	87 653.29429	566	438 940	1.289470087
100	0	49 739.32046	271	178 508	1.518139243
	2854	54 183.94596	132	101 786	1.296838465
	2751	79 053.14668	63	53 046	1.187648456
200	0	15 526.75111	323	198 307	1.628787688
	2854	5 742.560405	62	49 763	1.245905593
	2751	7 728.001035	58	43 252	1.340978452
400	0	6 404.451577	63	41 694	1.511008778
	2854	8 075.270893	550	331 216	1.6605478
	2751	9 596.092955	716	419 687	1.706033306
600	0	5 544.005772	61	35 000	1.742857143
	2854	5 709.465824	813	434 351	1.871758094
	2751	7 236.297396	606	345 433	1.75431994
800	0	5 523.857348	134	66 027	2.029472791
	2854	5 408.604256	58	30 819	1.881955936
	2751	4 803.228081	676	336 759	2.007370256
1000	0	4 995.397882	1 039	498 128	2.08580927
	2854	5 336.197148	1 027	499 915	2.054349239
	2751	4 877.909388	198	87 171	2.271397598

Table G.2: Dataset size study result for geometry 2

Dataset size	Random seed	Error	Training time [s]	Epochs	Time per epoch [ms]
10	0	31 197 916.6	< 1	1	0
	2854	12 697 243.8	197	243 622	0.8086297625
	2751	20 781 963.33	8	10 751	0.7441168263
50	0	9 713 032.482	587	433 170	1.355126163
	2854	5 051 435.44	157	112 880	1.390857548
	2751	8 189 139.149	8	5 128	1.560062402
100	0	3 037 383.743	547	437 402	1.250565841
	2854	3 917 652.358	6	6 111	0.9818360334
	2751	2 846 945.732	280	202 571	1.382231415
200	0	6 775 691.847	33	27 620	1.194786387
	2854	5 936 328.832	38	27 839	1.364991559
	2751	2 361 651.964	42	34 462	1.218733678
400	0	3 876 854.395	471	295 857	1.591985317
	2854	5 231 156.66	277	176 833	1.566449701
	2751	4 733 603.279	637	394 973	1.612768468
600	0	2 918 646.947	220	115 745	1.900730053
	2854	2 783 037.908	194	125 512	1.54566894
	2751	3 023 574.044	376	202 246	1.859122059
800	0	2 831 695.605	284	141 909	2.001282512
	2854	5 592 047.925	930	498 962	1.863869393
	2751	4 302 673.587	714	381 927	1.869467202
1000	0	2 978 909.196	230	118 374	1.942994239
	2854	2 974 726.878	669	342 393	1.953895085
	2751	3 243 609.101	1 049	489 844	2.14149811
1200	0	2 985 481.536	312	149 864	2.081887578
	2854	2 961 114.655	839	329 004	2.550120971
	2751	3 138 502.828	258	124 955	2.064743308
1400	0	3 194 996.088	1 294	498 989	2.593243538
	2854	3 027 837.512	99	45 207	2.189926339
	2751	4 572 636.001	319	14 6286	2.180659803

H

Raw data - Gaussian Process

This page is intentionally left blank.

Table H.1: GPM results for geometry 1 using different random seeds

Random seed	Error	Training time [s]	Covariance	Mean	Optim. method	Restart points	Max no. of SM eval.	Max optim. iter.	Max mat. size
1014	6 931.233656	94	Matérn 3/2	Constant	Direct	Auto (8)	10 000	500	2 000
	6 925.604667	106	Matérn 5/2	Constant	Direct	Auto (8)	10 000	500	2 000
	7 192.218017	152	Squared exp.	Constant	Direct	Auto (8)	10 000	500	2 000
	7 699.935065	149	Single lay. NN	Constant	Direct	Auto (8)	10 000	500	2 000
	6 912.380198	69	Matérn 3/2	Linear	Direct	Auto (8)	10 000	500	2 000
	6 922.499549	81	Matérn 3/2	Quadratic	Direct	Auto (8)	10 000	500	2 000
	6 931.233656	109	Matérn 3/2	Constant	Monte Carlo	Auto (8)	10 000 (no max)	-	2 000
	6 928.852719	122	Matérn 3/2	Constant	Direct	10	10 000	500	2 000
	6 928.708393	197	Matérn 3/2	Constant	Direct	20	10 000	500	2 000
	6 931.233656	103	Matérn 3/2	Constant	Direct	Auto (8)	20 000	500	2 000
	6 931.233656	109	Matérn 3/2	Constant	Direct	Auto (8)	30 000	500	2 000
	6 931.233656	108	Matérn 3/2	Constant	Direct	Auto (8)	10 000	1 000	2 000
	6 931.233656	110	Matérn 3/2	Constant	Direct	Auto (8)	10 000	2 000	2 000
	6 931.233656	97	Matérn 3/2	Constant	Direct	Auto (8)	10 000	500	4 000
	6 931.233656	98	Matérn 3/2	Constant	Direct	Auto (8)	10 000	500	8 000
	6 905.649861	86	Matérn 5/2	Linear	Direct	10	10 000	500	2 000
9750	6 930.728677	76	Matérn 3/2	Constant	Direct	Auto (8)	10 000	500	2 000
	6 921.199318	114	Matérn 5/2	Constant	Direct	Auto (8)	10 000	500	2 000
	7 189.506242	128	Squared exp.	Constant	Direct	Auto (8)	10 000	500	2 000
	7 699.935065	152	Single lay. NN	Constant	Direct	Auto (8)	10 000	500	2 000
	6 912.741858	75	Matérn 3/2	Linear	Direct	Auto (8)	10 000	500	2 000
	6 924.521644	85	Matérn 3/2	Quadratic	Direct	Auto (8)	10 000	500	2 000
	6 930.872961	93	Matérn 3/2	Constant	Direct	10	10 000	500	2 000
	6 931.161519	220	Matérn 3/2	Constant	Direct	20	10 000	500	2 000
7545	6 929.862914	79	Matérn 3/2	Constant	Direct	Auto (8)	10 000	500	2 000
	6 924.810467	100	Matérn 5/2	Constant	Direct	Auto (8)	10 000	500	2 000
	7 191.592313	148	Squared exp.	Constant	Direct	Auto (8)	10 000	500	2 000
	7 699.935065	132	Single lay. NN	Constant	Direct	Auto (8)	10 000	500	2 000
	6 912.669528	59	Matérn 3/2	Linear	Direct	Auto (8)	10 000	500	2 000
	6 921.849464	63	Matérn 3/2	Quadratic	Direct	Auto (8)	10 000	500	2 000
	6 931.233656	119	Matérn 3/2	Constant	Direct	10	10 000	500	2 000
	6 930.945101	185	Matérn 3/2	Constant	Direct	20	10 000	500	2 000

Table H.2: GPM results for geometry 2 using different random seeds

Random seed	Error	Training time [s]	Covariance	Mean	Optim. method	Restart points	Max no. of SM eval.	Max optim. iter.	Max mat. size
1014	2 961 536.763	77	Matérn 3/2	Constant	Direct	Auto (8)	10 000	500	2 000
	2 963 595.789	76	Matérn 5/2	Constant	Direct	Auto (8)	10 000	500	2 000
	3 033 315.018	143	Squared exp.	Constant	Direct	Auto (8)	10 000	500	2 000
	2 989 799.324	138	Single lay. NN	Constant	Direct	Auto (8)	10 000	500	2 000
	2 960 641.822	84	Matérn 3/2	Linear	Direct	Auto (8)	10 000	500	2 000
	2 970 000	68	Matérn 3/2	Quadratic	Direct	Auto (8)	10 000	500	2 000
	2 961 536.763	70	Matérn 3/2	Constant	Monte Carlo	Auto (8)	10 000 (no max)	-	2 000
	2 961 519.88	117	Matérn 3/2	Constant	Direct	10	10 000	500	2 000
	2 961 519.88	226	Matérn 3/2	Constant	Direct	20	10 000	500	2 000
	2 961 536.763	67	Matérn 3/2	Constant	Direct	Auto (8)	20 000	500	2 000
	2 961 536.763	72	Matérn 3/2	Constant	Direct	Auto (8)	30 000	500	2 000
	2 961 536.763	74	Matérn 3/2	Constant	Direct	Auto (8)	10 000	1 000	2 000
	2 961 536.763	70	Matérn 3/2	Constant	Direct	Auto (8)	10 000	2 000	2 000
	2 961 536.763	68	Matérn 3/2	Constant	Direct	Auto (8)	10 000	500	4 000
	2 961 536.763	72	Matérn 3/2	Constant	Direct	Auto (8)	10 000	500	8 000
	2 960 624.934	87	Matérn 3/2	Linear	Direct	10	10 000	500	2 000
9750	2 961 519.88	78	Matérn 3/2	Constant	Direct	Auto (8)	10 000	500	2 000
	2 963 612.66	70	Matérn 5/2	Constant	Direct	Auto (8)	10 000	500	2 000
	3 033 315.018	82	Squared exp.	Constant	Direct	Auto (8)	10 000	500	2 000
	2 989 799.324	146	Single lay. NN	Constant	Direct	Auto (8)	10 000	500	2 000
	2 960 641.822	62	Matérn 3/2	Linear	Direct	Auto (8)	10 000	500	2 000
	2 970 000	78	Matérn 3/2	Quadratic	Direct	Auto (8)	10 000	500	2 000
	2 961 536.763	92	Matérn 3/2	Constant	Direct	10	10 000	500	2 000
	2 961 536.763	214	Matérn 3/2	Constant	Direct	20	10 000	500	2 000
7545	2 961 553.646	72	Matérn 3/2	Constant	Direct	Auto (8)	10 000	500	2 000
	2 963 595.789	72	Matérn 5/2	Constant	Direct	Auto (8)	10 000	500	2 000
	3 033 315.018	105	Squared exp.	Constant	Direct	Auto (8)	10 000	500	2 000
	2 989 782.601	133	Single lay. NN	Constant	Direct	Auto (8)	10 000	500	2 000
	2 960 658.71	72	Matérn 3/2	Linear	Direct	Auto (8)	10 000	500	2 000
	2 970 000	71	Matérn 3/2	Quadratic	Direct	Auto (8)	10 000	500	2 000
	2 961 519.88	100	Matérn 3/2	Constant	Direct	10	10 000	500	2 000
	2 961 536.763	200	Matérn 3/2	Constant	Direct	20	10 000	500	2 000

I

Raw data - Polynomial Chaos Expansion

Table I.1: PCE results for geometry 1

Error	Training time [s]	Max pol. degree	Q norm	Rel. tolerance	Max mat. size
20 284.23033	< 1	Auto	Auto	10^{-3}	2 000
7 832.368735	< 1	Auto	Auto	10^{-4}	2 000
6 839.517527	5	Auto	Auto	10^{-5}	2 000
20 284.23033	< 1	Auto	Auto	10^{-3}	4 000
20 284.23033	< 1	Auto	Auto	10^{-3}	8 000
20 298.02946	< 1	30	0.5	10^{-3}	2 000
20 298.02946	< 1	50	0.5	10^{-3}	2 000
20 298.02946	< 1	70	0.5	10^{-3}	2 000
21 278.15781	< 1	30	0.3	10^{-3}	2 000
20 339.37069	< 1	30	0.7	10^{-3}	2 000

Table I.2: PCE results for geometry 2

Error	Training time [s]	Max pol. degree	Q norm	Rel. tolerance	Max mat. size
2 633 059.057	1	Auto	Auto	10^{-3}	2 000
2 434 563.616	5	Auto	Auto	10^{-4}	2 000
2 434 563.616	5	Auto	Auto	10^{-5}	2 000
2 633 059.057	1	Auto	Auto	10^{-3}	4 000
2 633 059.057	1	Auto	Auto	10^{-3}	8 000
2 656 219.117	1	30	0.5	10^{-3}	2 000
2 656 219.117	1	50	0.5	10^{-3}	2 000
2 656 219.117	1	70	0.5	10^{-3}	2 000
2 633 590.705	< 1	30	0.3	10^{-3}	2 000
2 658 119.636	2	30	0.7	10^{-3}	2 000

J

Raw data - Uniformity study

Table J.1: Uniformity study result

Radius	Random seed	Error	Training time [s]	Epochs	Time per epoch [ms]
0.01	0	5.02533581	43	33 608	1.279457272
	2854	4.720487263	10	8 318	1.0221207
	2751	5.491538946	9	6 017	1.495762008
0.12	0	9.130115005	113	84 413	1.338656368
	2854	6.700447746	291	147 799	1.968890182
	2751	5.66762737	689	425 769	1.618248393
0.25	0	9.905604474	42	26 131	1.607286365
	2854	9.227350649	40	26 931	1.48527719
	2751	8.347275004	31	18 874	1.642471124

K

Raw data - Streamer Dataset size study

Table K.1: Dataset size study result

Dataset size	Random seed	Layers	Neurons	Error	Training time [s]	Epochs	Time per epoch [ms]
10	0	4	4	44.69563737	2	6 100	0.3278688525
			6	107.3359213	< 1	80	0
			8	146.6151425	27	99 400	0.2716297787
		5	4	36.9945942	4	11 800	0.3389830508
			6	60.73631533	5	17 600	0.2840909091
			8	181.256724	< 1	42	0
	2854	4	4	36.18563251	< 1	950	0
			6	51.04311903	< 1	843	0
			8	119.9749974	< 1	397	0
		5	4	29.29573348	1	1 900	0.5263157895
			6	40.79583312	< 1	319	0
			8	188.4887265	< 1	989	0
	2751	4	4	75.40092838	140	500 000	0.28
			6	79.82606091	143	500 000	0.286
			8	104.2928569	137	500 000	0.274
		5	4	32.06711711	24	74 100	0.3238866397
			6	234.2071732	167	500 000	0.334
			8	83.6247571	171	500 000	0.342

Dataset size	Random seed	Layers	Neurons	Error	Training time [s]	Epochs	Time per epoch [ms]
50	0	4	4	26.43369062	103	366 400	0.2811135371
			6	14.7424557	3	11 400	0.2631578947
			8	13.28570661	1	4 300	0.2325581395
		5	4	11.47039668	63	157 600	0.3997461929
			6	42.91037171	5	14 300	0.3496503497
			8	11.1350797	2	6 200	0.3225806452
	2854	4	4	10.79212676	1	4 400	0.2272727273
			6	31.86063402	6	18 600	0.3225806452
			8	87.13208364	162	500 000	0.324
		5	4	11.90462095	24	70 600	0.3399433428
			6	18.31065264	20	56 900	0.3514938489
			8	80.51024779	74	219 700	0.3368229404
	2751	4	4	12.10082642	1	4 400	0.2272727273
			6	17.72032731	1	4 500	0.2222222222
			8	11.02950588	2	4 700	0.4255319149
		5	4	16.75977327	15	45 800	0.327510917
			6	14.86909547	2	5 900	0.3389830508
			8	15.06651917	2	6 500	0.3076923077
200	0	4	4	6.89492567	47	150 600	0.3120849934
			6	8.877612292	146	434 400	0.3360957643
			8	10.60754448	4	12 500	0.32
		5	4	9.115042512	31	88 700	0.3494926719
			6	10.94760248	4	11 200	0.3571428571
			8	9.982835269	19	52 900	0.359168242
	2854	4	4	8.59982558	164	500 000	0.328
			6	10.15086203	8	24 800	0.3225806452
			8	9.111695781	18	53 100	0.3389830508
		5	4	9.463297523	149	399 900	0.3725931483
			6	20.60121356	59	155 000	0.3806451613
			8	11.24410957	70	185 800	0.3767491927
	2751	4	4	11.52085066	79	214 900	0.3676128432
			6	8.468529979	69	207 700	0.3322099182
			8	8.429175523	97	272 100	0.3564865858
		5	4	10.18871925	70	193 700	0.3613835829
			6	12.53475169	26	67 800	0.383480826
			8	11.27785441	39	105 100	0.3710751665

Dataset size	Random seed	Layers	Neurons	Error	Training time [s]	Epochs	Time per epoch [ms]
400	0	4	4	6.599090846	164	473 000	0.3467230444
			6	17.95912025	214	499 900	0.4280856171
			8	15.90503065	149	347 100	0.4292711034
		5	4	5.38729988	173	443 900	0.3897274161
			6	10.98954048	135	275 400	0.4901960784
			8	10.86185988	7	13 600	0.5147058824
	2854	4	4	5.4886246	83	236 700	0.3506548373
			6	7.735761113	7	16 200	0.4320987654
			8	8.616960021	33	77 300	0.4269081501
		5	4	6.352479831	37	95 000	0.3894736842
			6	9.999649994	5	9 600	0.5208333333
			8	8.299337323	11	19 200	0.5729166667
	2751	4	4	8.452810184	34	109 400	0.310786106
			6	9.876183473	120	303 300	0.625
			8	8.253241787	83	198 700	0.4960727573
		5	4	5.13234839	169	455 500	0.3710208562
			6	9.876183473	87	139 200	0.625
			8	8.253241787	120	241 900	0.4960727573

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY