ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

PAUL SCHERRER INSTITUT
PSI

# Accelerating accelerators: Fast surrogate models for beam prediction

## Master Thesis

in Computational Science and Engineering

Department of Mathematics

ETH Zurich

written by

Renato Bellotti

supervised by

Dr. A. Adelmann (ETH)

Scientific Collaborators

Dr. Alexander Zholents
Dr. John Power
Dr. Gwanghui Ha

June 9, 2020

**Abstract**

Particle accelerators are important tools for various fields. They employ thousands of control knobs and take enormous advantage from using computational models to find the best setup for a given operating point. The conventional models are very time consuming to evaluate and not usable in a real-time setting. We develop two kinds of data-driven surrogate models to mitigate this problem.
First, a neural network is trained to predict beam characteristics (beam size, emittances, energy spread and energy) at various positions along the machine. At the locations of YAG screens, the model has a prediction error of approximately 20% at 90% confidence. The model enables optimisation in under 5 min on 12 CPU cores that would take days on a high performance cluster.
The second model, the invertible model, allows to sample machine settings in order to achieve user-imposed beam characteristics. The model is able to sample configurations according to the empirical distribution obtained with a computational model. The sampling error (desired beam characteristics vs. sampled characteristics) lies around $20 - 30\%$ at 75% confidence. A possible application could be initialisation of an optimisation to start closer to an optimal configuration.
Both models are made accessible to users in the form of interactive web interfaces.

# Contents

# Chapter 1

# Introduction

## 1.1 Goals

Particle accelerators have been an important tool for decades, not only for basic research in high energy physics, but also as an auxiliary instrument for other disciplines such as biology, medicine, chemistry and materials science [1].

Such big machines feature many parameters and often operate in regimes when the response to a parameter change is highly non-linear. For this reason, design and operation rely heavily on computer models implemented in frameworks like OPAL [2]. Since these codes solve the N-body problem for a large number of particles ($> 10^5$ is needed for accuracy reasons), they are computationally expensive and therefore unsuitable for guiding real-time operation.

This thesis aims to make a contribution towards a solution to the problem above: A huge speedup can be achieved by training a surrogate model, a fast-to-evaluate approximation to the computer model. Edelen et al. [3] reported a speedup of up to a factor of $10^6$. A more complicated particle accelerator with more degrees of freedom is modelled in our work. Models of two different kinds are developed and examined:

The forward model will be used as a drop-in replacement for OPAL. It allows to quickly evaluate the effect of a machine setting on the beam characteristics. The invertible model will try to solve the inverse problem: Given a desired set of beam characteristics, the inverse model will suggest machine settings to achieve this beam.

Both model types can be trained to approximate the dynamics of an arbitrary linear accelerator. We choose the Argonne Wakefield Accelerator (AWA) because there is a planned experiment. The goal of the experiment is to create an initial beam intensity modulation at the beginning of the accelerator and preserve it to the end, where a short wiggler magnet is used to transform it to an energy modulation in the beam. Modelling of physics processes guiding this transformation in the wiggler is not part of this work, but can be found in the master's thesis by Arnau Albà [4]. Finding a good operating point for the experiment needs multi-objective optimisations with many constraints. The surrogate models are used to solve the optimisation problem.

The AWA will be presented in the next section, along with a more mathematical problem formulation. The theoretical foundations of artificial neural networks are discussed afterwards.

## 1.2 Problem Description

### 1.2.1 The Argonne Wakefield Accelerator

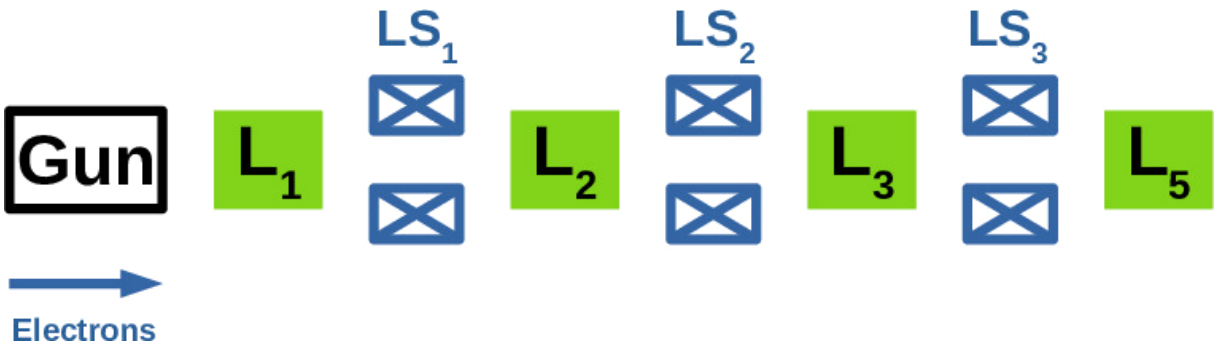



Figure 1.1: Schematic of the Argonne Wakefield Accelerator as modelled in this work. The electrons leave the gun and travel along the beamline towards the right. They receive an energy gain in the four cavities $L_1, ..., L_5$ (no $L_4$ because there is not enough energy in the machine) and are focussed by three solenoids (Linac Solenoids) $LS_1, LS_2, LS_3$. Downstream after $L_5$, there is a drift (not shown).

Both the forward and the inverse model will be approximations to the dynamics of the Argonne Wakefield Accelerator (AWA). The schematic can be found in Fig. 1.1. The gun contains a bucking, a focusing and a matching solenoid whose strengths can be controlled by changing the current flowing through them. The bucking and focusing solenoids are scaled together. Further, the bunch charge and the laser spot size can be changed, as well as the gun phase. The current through the linac solenoids can be tuned as well. The laser follows a uniform distribution in the transversal directions. In the longitudinal direction, it follows a Gaussian mixture of four Gaussian peaks separated by the variable peak-to-peak distance $\lambda$.

In total, there are 9 tunable knobs. They form the space of design variables (DVAR). Each setup of the accelerator is uniquely described by a vector $\mathbf{x} \in \mathbb{R}^9$. The ranges of the design variables are determined by the machine parts available at the AWA facility.

The electron beam is described by its transversal beam sizes $\sigma_x$ and $\sigma_y$, the normalised emittances $\epsilon_x$ and $\epsilon_y$, the mean bunch energy $E$, the energy spread of the beam $\Delta E$ and the correlations between transversal position and the corresponding momentum $corr\,(x, p_x)$, $corr\,(y, p_y)$. These are the quantities of interest (QOI). They are represented by vectors $\mathbf{y} \in \mathbb{R}^8$.

The positions of beam elements can be found in Appendix A, a summary of all design variables and quantities of interest in Appendix B.

### 1.2.2 OPAL: A reliable model of particle accelerators

It is not possible to measure the quantities of interest at every position along the beam line. However, a computer model allows to solve the relativistic equations of motion using first principles and numerical algorithms. This provides an approximation to the physical world. The big advantage is that it is possible to evaluate the quantities of interest at any longitudinal position.

We use the Object Oriented Parallel Accelerator Library OPAL [2] as ground truth. It is well tested (see [5] for an example) and can therefore be considered an accurate model for particle accelerators. On an abstract level, a computer model is a mapping $\mathbf{f}$ between the space of design variables and the space of the quantities of interest. Given a setting for the accelerator $\mathbf{x}$, the program calculates the beam characteristics at any longitudinal position $s \in \mathbb{R}$.

$$\begin{aligned} f &: \mathbb{R}^9 \times \mathbb{R} \to \mathbb{R}^8 \\ \mathbf{f}&\begin{pmatrix} \mathbf{x} \\ s \end{pmatrix} = \mathbf{y}(s), \end{aligned} \tag{1.1}$$

where $s$ is the longitudinal position along the beamline, $\times$ denotes the Cartesian product, and $\mathbb{R}$ is the set of real numbers. Not all configuration vectors $\mathbf{x}$ are feasible due to technical limitations of the AWA itself. More details about the input and output spaces are given in Sec. 2.2.

### 1.2.3 Surrogate Models

Simulation codes come with numerical guarantees regarding precision, but they are computationally expensive. The higher the precision of the calculations, the bigger are the corresponding costs. In contrary, data-driven surrogate models do not provide any precision guarantees, but are fast to evaluate.

The hunger for resources of classical (i. e. non-data-driven) computational models becomes prohibitive when many model evaluations are necessary. A typical task for particle accelerators is optimisation: Many different machine configurations are tried out in order to find one that meets some constraints and is optimal with respect to objectives.

Surrogate models are a valuable tool to speed such calculations up. They provide a fast-to-evaluate approximation to the computational model. In the case of optimisation, they can be used for exploration steps. Once an optimal region is found, the found configurations can be evaluated with the classical model. Even if it turns out they are not optimal yet, only a few more iterations with the simulation code have to be performed. In this way, a speedup can be obtained without sacrificing precision.

We examined many different model families, such as random forests, linear models, nearest neighbour and boosted trees. They did not perform well due to the highly non-linear dynamics of the beam. For this reason, only artificial neural networks are considered in the remainder of this thesis.

## 1.3 Forward Model: Feedforward Neural Networks

A forward surrogate model is an approximation to OPAL,

$$\tilde{\mathbf{f}}\left(\mathbf{x}, s\right) \approx \mathbf{f}\left(\mathbf{x}, s\right). \tag{1.2}$$

We use the short hand notation

$$\mathbf{y}_{\text{true}} := \mathbf{f}\left(\mathbf{x}, s\right),$$
$$\tilde{\mathbf{y}}_{\text{pred}} := \tilde{\mathbf{f}}\left(\mathbf{x}, s\right).$$

Our approximations are implemented as (artificial) neural networks. A neural network is a collection of neurons or nodes (see Fig. 1.2). The neurons are arranged in $L$ layers. In each layer $l \in \{1, ..., L\}$, there are $N_l$ neurons. The $i$-th neuron in layer $l$ is a function

$$f_{l,i}(\mathbf{x}) = \sigma\left((\omega_b)_{l,i} \cdot b_{l,i} + \sum_{j=1}^{N_l} (\omega_{l,i})_j \cdot x_j\right),$$

where $\mathbf{x}$ is the input vector, $\omega_{l,i}$ entries of the weight matrix, $b_{l,j}$ bias terms and $\sigma$ is a non-linear "activation function".

Multiple neurons form a layer $\mathbf{f}_l(\mathbf{x})$. Multiple layers can be stacked by using the output of one layer as the input of the other one. If all neurons from the previous layer are connected to all neurons of the subsequent layer, the network is called fully connected. For example, the i-th output of a fully connected network of depth 3 is $f_{3,i}\left(\mathbf{f}_2\left(\mathbf{f}_1\left(\mathbf{x}\right)\right)\right)$. The analogue case with 5 layers is depicted in Fig. 1.3, where each neuron in the input layer is an entry of the input vector $\mathbf{x}$.

The weights of a neural network are free ("trainable") parameters that are fitted against the data. They are chosen so that they minimise a loss function $\mathcal{L}$. In this thesis, the mean absolute error (MAE) is used:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \left\| \tilde{\mathbf{f}}(\mathbf{x}_i, s_i; \theta) - \mathbf{y}_i(s_i) \right\|. \tag{1.3}$$

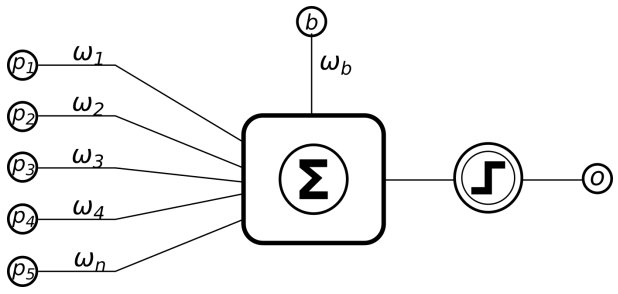



Figure 1.2: A neuron of a neural network [6]. Multiple input values are multiplied by weights and summed up. After that, an activation function is applied. This defines the output of the neuron.

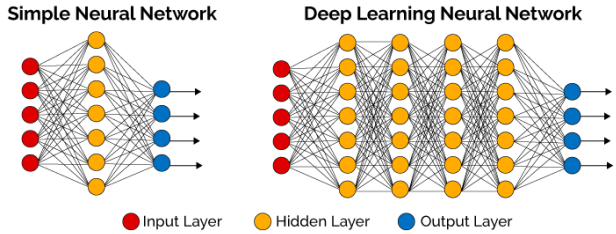



Figure 1.3: Single- and multilayer networks [7].

The notation $\tilde{\mathbf{f}}(\mathbf{x}_i, s_i; \theta)$ implies that the output of the neural network depends on the trainable parameters, indicated by $\theta$. The integer $i \in \{1, ..., N\}$ is the index of the $i$-th entry in the dataset and $N$ is the number of samples.

The optimal parameters are found by minimising the loss function with an iterative optimisation algorithm such as the Adam algorithm, a modified version of gradient descent [8]. An optimal weight vector $\theta^*$ is defined as:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(\theta).$$

Only fully connected deep neural networks are examined as a forward model.

## 1.4 Invertible Model: Invertible Neural Networks

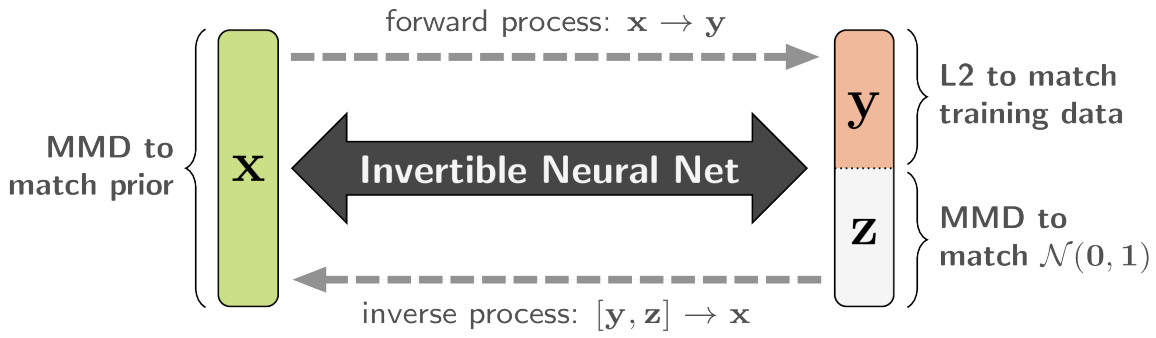



Figure 1.4: Scheme of an Invertible Neural Network as described by Ardizzone et al. [9].

Solving the inverse problem is more complicated than solving the forward problem because the solution is not unique. There are possibly several DVAR configurations that realise the same or a very similar beam. An invertible model needs to be able to randomly select one of them. This is why the inverse prediction is also called sampling. One selection mechanism is presented by Ardizzone et al. [10]. Since we rely on their work, the reader is encouraged to read the original paper for additional details.

The selection mechanism (see Fig. 1.4) is realised by mapping the DVAR space $\mathbb{R}^9$ not only to the space of quantities of interest $\mathbb{R}^8$, but also to a latent space $\mathcal{Z}$ of dimension $d_z$. The latter consists of vectors $\mathbf{z}$ sampled from a Gaussian or uniform distribution. Other than the DVAR and QOI configurations, they are not part of the dataset. The dimension $d_z$ is a hyperparameter. Selecting a random point in the $\mathcal{Z}$ space allows to select one design variable configuration.

### 1.4.1 Description

An invertible model provides two predictions. The forward prediction implements an approximation to OPAL, see Equ. 1.1:

$$\hat{f} : \mathbb{R}^9 \times \mathbb{R} \rightarrow \mathbb{R}^8 \times \mathbb{R} \times \mathcal{Z}$$

$$\hat{\mathbf{f}} \begin{pmatrix} \mathbf{x} \\ s \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{y}}_{\text{pred}} \\ s \\ \mathbf{z} \end{pmatrix},$$

$$\hat{\mathbf{y}}_{\text{pred}} \approx \mathbf{y}_{\text{true}} = \mathbf{f}(\mathbf{x}, s).$$

The hat symbol is used to distinguish between the forward prediction of an invertible model and the one of a forward model. The inverse prediction is given by:

$$g : \mathbb{R}^8 \times \mathbb{R} \times \mathcal{Z} \rightarrow \mathbb{R}^9 \times \mathbb{R}$$

$$\mathbf{g} \begin{pmatrix} \mathbf{y} \\ s \\ \mathbf{z} \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ s \end{pmatrix}.$$

For both of the equations above, the vectors $\mathbf{z}$ are randomly sampled from a Gaussian or uniform distribution. Other than the vectors $\mathbf{x}$, $\mathbf{y}$ and the positions $s$, they are not part of the dataset.

The model presented by Ardizzone et al. is exactly invertible. This means the following equations hold:

$$\hat{\mathbf{f}} \left( \mathbf{g} \begin{pmatrix} \mathbf{y} \\ s \\ \mathbf{z} \end{pmatrix} \right) = \begin{pmatrix} \mathbf{y} \\ s \\ \mathbf{z} \end{pmatrix} \text{ and } \mathbf{g} \left( \hat{\mathbf{f}} \begin{pmatrix} \mathbf{x} \\ s \end{pmatrix} \right) = \begin{pmatrix} \mathbf{x} \\ s \end{pmatrix}. \tag{1.4}$$

The longitudinal position is needed as a parameter for both passes in order to fulfil the invertibility conditions Equ. 1.4.

### 1.4.2 Invertible Architecture

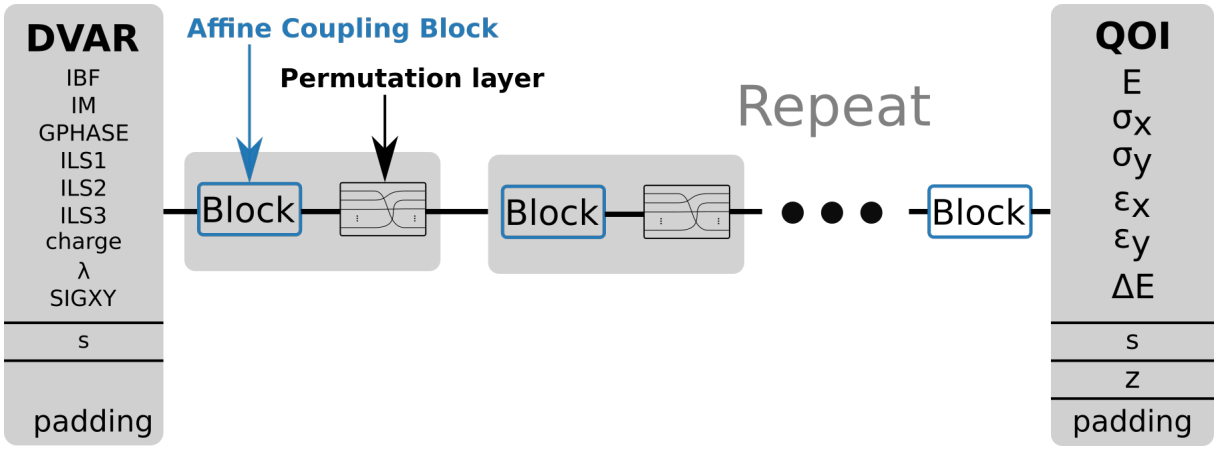



Figure 1.5: Schematic of the invertible architecture. It consists of a series of affine coupling blocks, separated by permutation layers.

Both the forward process and the inverse process are performed by the same network with the same weights. It is not possible to use fully connected feed forward networks because they are not invertible. Instead, Ardizzone et al. suggest to use Affine Coupling Blocks (ACB) (see Fig. 1.5) with permutation layers in between. Both layers are exactly invertible. Therefore, the entire architecture is. The forward process of an ACB can be calculated by separating the first and second half $\mathbf{u}_1, \mathbf{u}_2$ of the input vector $\mathbf{u}$ (analogue for the corresponding output vector $\mathbf{v}$) and performing the following calculations, where $\odot$ denotes elementwise multiplication [10]:

$$\begin{aligned}\mathbf{v}_1 &= \mathbf{u}_1 \odot \exp\left(s_2(\mathbf{u}_2)\right) + t_2(\mathbf{u}_2) \text{ and} \\ \mathbf{v}_2 &= \mathbf{u}_2 \odot \exp\left(s_1(\mathbf{v}_1)\right) + t_1(\mathbf{v}_1).\end{aligned}$$

The calculations for the inverse process are:

$$\begin{aligned}\mathbf{u}_2 &= \left(\mathbf{v}_2 - t_1(\mathbf{v}_1)\right) \odot \exp\left(-s_1(\mathbf{v}_1)\right) \text{ and} \\ \mathbf{u}_1 &= \left(\mathbf{v}_1 - t_2(\mathbf{u}_2)\right) \odot \exp\left(-s_2(\mathbf{u}_2)\right).\end{aligned}$$

The coefficient functions $s_i, t_i$ do not have to be invertible; they can be any function. In order to be able to adapt to the training data, each of these functions is a deep fully connected network itself. To reduce the number of parameters, $s_1, t_1$ share parameters, and so do $s_2, t_2$. The calculations of the forward pass for this special case of parameter sharing are depicted in Fig. 1.6. Because of the parameter sharing, the network $s_1$ is the same as $t_1$, and $s_2$ is the same network as $t_2$.

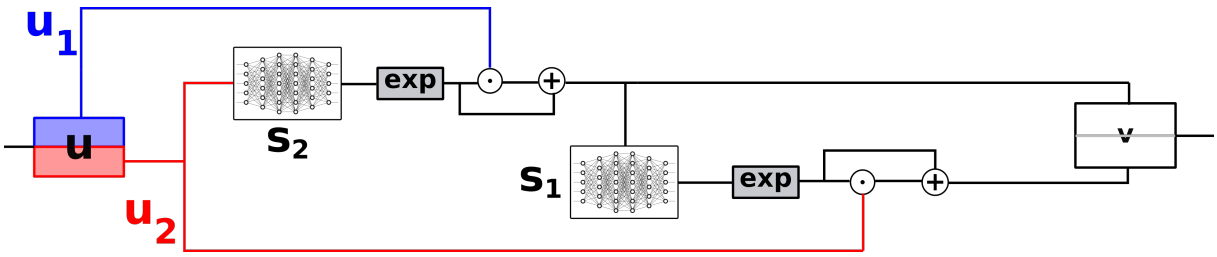



Figure 1.6: Schematic of the forward pass of an affine coupling block. The symbol $\mathbf{u}$ denotes the input vector of the layer, $\mathbf{v}$ is the layer output, $\odot$ denotes componentwise multiplication, $\oplus$ componentwise addition, and exp is the componentwise exponential function. The picture of the neural network is taken from [11].

### 1.4.3 Losses

The architecture discussed in the previous section is exactly invertible. The next step is to find a good set of model parameters to represent the data well. To define what "good" means, a loss function is needed. Ardizzone et al. [10] choose to use a linear combination of 5 loss functions to reward good parameter configurations:

1. $\mathcal{L}_{\mathbf{x}}$**:** This loss ensures that the sampled design variable distribution $\hat{p}_x(\mathbf{x})$ matches the one in the dataset $p_x(\mathbf{x})$:

$$\mathcal{L}_x \text{ is minimal} \Leftrightarrow \hat{p}_x(\mathbf{x}) \stackrel{!}{=} p_x(\mathbf{x})$$

   In many real-world applications, the distributions are not known analytically. For this reason, they can only be compared by comparing samples taken from them. A loss function that fulfils this requirement is the Maximum Mean Discrepancy (MMD), which was first developed by Gretton et al. [12]. A description is given at the end of this section.

2. $\mathcal{L}_{\mathbf{y}}$**:** This loss makes sure that the relation between the design variables and the quantities of interest is captured by the model. It can be any supervised loss function. Ardizzone et al. [10] suggest to use the mean squared error. They also base their theoretical guarantees on this specific loss function. For this reason, it is also used here:

$$\mathcal{L}_y(\theta) = \sum_{i=1}^{N} \left\| \hat{\mathbf{f}}(\mathbf{x}_i, s_i; \theta) - \mathbf{y}_i \right\|^2,$$

   where $\{(\mathbf{x}_i, \mathbf{y}_i, s_i)\}_{i=1}^N$ are tuples in the training set and $N$ is the number of training samples.

3. $\mathcal{L}_{\mathbf{z}}$**:** The model needs to know from which distribution it should sample latent vectors $\mathbf{z}$. This guidance is provided by the loss function $\mathcal{L}_z$. Additionally, the function guarantees that the latent space is sampled from independently of the desired quantities of interest $\mathbf{y}$:

$$\mathcal{L}_z \text{is minimal} \Leftrightarrow \hat{p}_z(\mathbf{z}|\mathbf{y}) \stackrel{!}{=} p_z(\mathbf{z})$$

   The quantity $\hat{p}_z$ denotes the distribution from which the invertible model samples. The quantity $p_z$ denotes the distribution the latent space shall follow. It can be any distribution, as long as it is possible to sample from it. Ardizzone et al. suggest to use $\mathcal{N}(0, 1)$ for each component of $\mathbf{z}$. The design variables in our datasets follow a uniform distribution, so we use Unif$[0, 1]$ instead. Following Ardizzone et al. [10], the MMD loss is chosen for $\mathcal{L}_z$.

4. $\mathcal{L}_{\mathbf{r}}$**:** The invertible model should be robust w. r. t. perturbations:

$$\mathbf{g}\left(\hat{\mathbf{f}}(\mathbf{x}) + \epsilon\right) \approx \mathbf{x},$$

   where $\epsilon$ is Gaussian noise of small magnitude (sampled independently in each component). It is straightforward to assert the condition by using the mean squared error as the reconstruction loss:

$$\mathcal{L}_r = \sum_{i=1}^{N} \left\| \mathbf{g}\left(\hat{\mathbf{f}}(\mathbf{x}_i) + \epsilon\right) - \mathbf{x}_i \right\|^2.$$

5. $\mathcal{L}_{\text{artificial}}$**:** One detail has been omitted in the descriptions so far. In order for the network to be invertible, the dimensions of the network input and output have to match. Ardizzone et al. [10] point out that this is not an issue: The number of independent dimensions of the quantities of interest $\mathbf{y}$ might be smaller than the dimension of the quantities of interest in the datasets. This is the case if the features are not linearly independent. Nevertheless, affine coupling layers expect matching dimensions. A remedy to this technical issue is to pad both the design variables $\mathbf{x}$ and the quantities of interest $\mathbf{y}$ with uniform noise of low amplitude. Notice that the total dimension of input and output vectors must be even due to the splitting within an ACB.

   In our case, we have 10 input and 8 output quantities. Trial and error showed that using vectors of length 12 leads to good results. Therefore, the input vectors are padded by two dimensions and the output vectors by four. The resulting padded vectors are given by:

$$\mathbf{x}_{\text{pad}} = \begin{pmatrix} \mathbf{x} \\ s \\ \mathbf{x}_n \end{pmatrix} \in \mathbb{R}^{12} \text{ and } \mathbf{y}_{\text{pad}} = \begin{pmatrix} \mathbf{y} \\ s \\ \mathbf{z} \\ \mathbf{y}_n \end{pmatrix} \in \mathbb{R}^{12}.$$

The components of the noise vectors $\mathbf{x}_n \in \mathbb{R}^2$ and $\mathbf{y}_n \in \mathbb{R}^4$ are sampled uniformly from the interval $[-0.05, +0.05]$.

The artificial loss $\mathcal{L}_{\text{artificial}}$ encourages the network not to encode any information in the padding dimensions. This can be done by applying the mean squared error to the padding dimensions:

$$\mathcal{L}_{\text{artificial}} = \|\mathbf{x}_n\|^2 + \|\mathbf{y}_n\|^2.$$

Finally, the total loss function is calculated as a weighted sum of all the losses above:

$$\mathcal{L}_{\text{tot}} = w_x\mathcal{L}_x + w_y\mathcal{L}_y + w_z\mathcal{L}_z + w_r\mathcal{L}_r + w_{\text{artificial}}\mathcal{L}_{\text{artificial}} \tag{1.5}$$

This is the function to be minimised during training.

**Mean Field Discrepancy (MMD)**
The discussions above outlined that one needs to be able to compare distributions. The only available information are samples taken from them. Gretton et al. [12] developed MMD to solve this task. Their Lemma 6 gives instructions on how to calculate the squared MMD:

Given $x$ and $x'$ independent random variables with distribution $p$, and $y$ and $y'$ independent random variables with distribution $q$, the squared population MMD is

$$\text{MMD}^2[p, q] = \mathbb{E}_{x,x'}[k(x, x')] - 2\mathbb{E}_{x,y}[k(x, y)] + \mathbb{E}_{y,y'}[k(y, y')],$$

where $x'$ is an independent copy of x with the same distribution, $y'$ is an independent copy of y, and $\mathbb{E}_{x,x'}$ denotes the expectation value over $x$ and $x'$. An unbiased empirical estimate is given by:

$$\text{MMD}^2[X, Y] = \frac{1}{m(m-1)}\sum_{i=1}^{m}\sum_{j\neq i}^{m} k(x_i, x_j) + \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i}^{n} k(y_i, y_j) - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} k(x_i, y_j),$$

where $X \in \mathbb{R}^m, Y \in \mathbb{R}^n$ are vectors whose entries are samples taken from $p$ and $q$, respectively.

The only thing that is missing now is the kernel function $k$. Ardizzone et al. state that the inverse quadratic kernel

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \frac{1}{1 + \|(\boldsymbol{x} - \boldsymbol{x'})/h\|_2^2} \tag{1.6}$$

with $h = 0.2$ yielded the best results [10]. We follow their decision.

### 1.4.4 Summary: Hyperparameters

There are many hyperparameters that need to be tuned for an invertible neural network. To give an overview, they are listed below.

- **Model architecture**
  The architecture is crucial for model capacity, i. e. it determines which functions can be represented by the model. There are several parameters that determine the architecture:
  - Number of affine coupling blocks.
  - Depth and width of the coefficient networks $s_i, t_i$.
  - Activation functions of the neurons in the coefficient networks.
- **Loss weights**
  According to Equ. 1.5, the total loss is a weighted sum of five different loss functions. When optimising the model parameters during training, we are only interested in minimising the total loss. Without loss of generality, the weights of the partial losses are normalised to sum up to one, leaving 4 independently tunable hyperparameters.
- **Parameters related to the optimisation**
  The most prominent examples are the learning rate $\eta$ and the batch size $b$. Additionally, optimisation algorithms such as Adam [8] involve even more parameters, and the choice of the optimisation algorithm itself is another hyperparameter. The number of epochs to train for is one more degree of freedom.
- **Dimension of the latent space and nominal dimension**
  All the hyperparameters discussed so far have counterparts in feedforward neural networks. One peculiarity of invertible neural networks is the presence of a latent space, and the need for padding on both sides of the network. The number of padding and latent dimensions represent one hyperparameter each.

## 1.5 Notation

We use the notation below.

- **Ground truth (OPAL):**

$$\mathbf{y}_{\text{true},i} = \mathbf{f}(\mathbf{x}_i, s_i)$$

denotes the true QOI vector for the $i$-th sample. It is calculated with OPAL.

- **Prediction of a forward model:**
We use the following notation

$$\tilde{\mathbf{y}}_{\text{pred},i} = \tilde{\mathbf{f}}(\mathbf{x}_i, s_i)$$

to denote the model prediction corresponding to the $i$-th sample in the dataset.

- **Forward prediction of an invertible model:**
The analogue notation is used for the forward prediction of an invertible model:

$$\hat{\mathbf{y}}_{\text{pred},i} = \hat{\mathbf{f}}(\mathbf{x}_i, s_i)$$

The hat symbol is used instead of the tilde to distinguish between a forward prediction by a forward model and one by an invertible model. The prediction returns also a vector in the latent space and the position $s_i$. Both are omitted for the sake of readability.

- **Sampled design variables:**
The sampling, or also inverse pass, of an invertible network produces a design variable vector:

$$\mathbf{x}_{\text{sampled},\mathbf{i}} = \mathbf{g}(\mathbf{y}_i, s_i, \mathbf{z}_i),$$

where $\mathbf{z}_i$ is the latent vector corresponding to the $i$-th sample. $\mathbf{z}_i$ is sampled from the distribution of the latent space.

- **Sampled quantities of interest**
A sampled design variable configuration corresponds to a single configuration of quantities of interest. The design variables can serve as input for OPAL:

$$\mathbf{y}_{\text{sampled},i} = \mathbf{f}\left(\mathbf{x}_{\text{sampled},i}, s_i\right)$$

A single OPAL run takes several minutes. The method above is therefore not suitable for real-time user interaction. There are two ways to get an approximation for OPAL.
For one, a trusted forward surrogate model can be used:

$$\tilde{\mathbf{y}}_{\text{sampled},i} = \tilde{\mathbf{f}}\left(\mathbf{x}_{\text{sampled},i}, s_i\right) \tag{1.7}$$

Another possibility is to use the inverse surrogate model itself to perform the forward prediction:

$$\hat{\mathbf{y}}_{\text{sampled},i} = \hat{\mathbf{f}}\left(\mathbf{x}_{\text{sampled},i}, s_i\right)$$

All these quantities are column vectors. For the evaluation, we usually want to calculate predictions for multiple samples. In that case, we can use a data matrix to represent many samples. It is built by stacking many of the vectors discussed above as rows. For example, the data matrix for the ground truth is given by

$$Y_{\text{true}} = \begin{pmatrix} \mathbf{y}_{\text{true},1}^T \\ \vdots \\ \mathbf{y}_{\text{true},N_s}^T \end{pmatrix}$$

$$(Y_{\text{true}})_{i,j} = y_{\text{true},i,j},$$

where the row index $i \in \{1, ..., N_s\}$ denotes the sample and the column index $j \in \{1, ...N_{\text{QOI}}\}$ the quantity of interest. The quantity $N_s$ is the number of samples, and $N_{\text{QOI}}$ is the number of quantities of interest.

# Chapter 2

# Methods

The following steps are needed to train a fast surrogate model. First, a dataset to learn from needs to be built. After that, several models are trained. Their performance is evaluated on a validation set. The results allow to select the best model. As a last step, the final model is evaluated on a test set, and the quality of its predictions are analysed.

Once the model is trained and validated, a user interface is needed. Web-based interactive dashboards are provided as an easy-to-use interface to the surrogate models. They allow people without any prior knowledge in machine learning to utilise a surrogate model and estimate the uncertainty of its predictions.

The following sections describe in detail how the workflow above is implemented.

## 2.1 Hardware

All the work was carried out on the Merlin6 cluster at the Paul Scherrer Institute. It contains compute nodes with 2 Intel® Xeon® Gold 6152 Processors, each providing 22 cores (2 threads per physical core) [13].

It was found that the training time decreases when more cores are used, up to 12 CPU cores, and then increases again for more cores. For this reason, all models were trained on 12 CPU cores.

## 2.2 Dataset

As already discussed in Sec. 1.1, the goal is to build a fast approximation to the computer model implemented in OPAL [2]. To achieve this, one needs to run many simulations with different configurations of design variables (DVARs) and save the corresponding beam characteristics, the quantities of interest (QOIs).
All design variables and quantities of interest, along with a short description, can be found in Appendix B. The design variables ranges are shown in Tab. B.1 and reflect the settings that are possible with the AWA.

### 2.2.1 Definitions

Some definitions are needed to describe the datasets and their structure.

**OPAL Run.** One design variable configuration $\mathbf{x}_\alpha$ is fed to the model. The relativistic equations of motion are solved, and the QOIs $\{\mathbf{y}_{\text{true},\alpha,j} = \mathbf{f}(\mathbf{x}_\alpha, s_{\alpha,j})\}_{j=1,\dots,N_\alpha}$ are logged at multiple longitudinal positions $\{s_{\alpha,j}\}_{j=1,\dots,N_\alpha}$[1]. This is called an OPAL run. We also say that configuration $\mathbf{x}_\alpha$ is evaluated with OPAL.

**OPAL Sampler Run.** OPAL comes with a built-in sampler that allows to sample design variable configurations from the prescribed ranges. We use uniform sampling and the latin hypercube algorithm [14] in order to have better exploration of the design variables space. For each of the sampled DVAR configurations, an OPAL run is performed.

[1] The notation hints that both position and number of the longitudinal evaluation points $N_\alpha$ depend on the chosen resolution and the design variables.

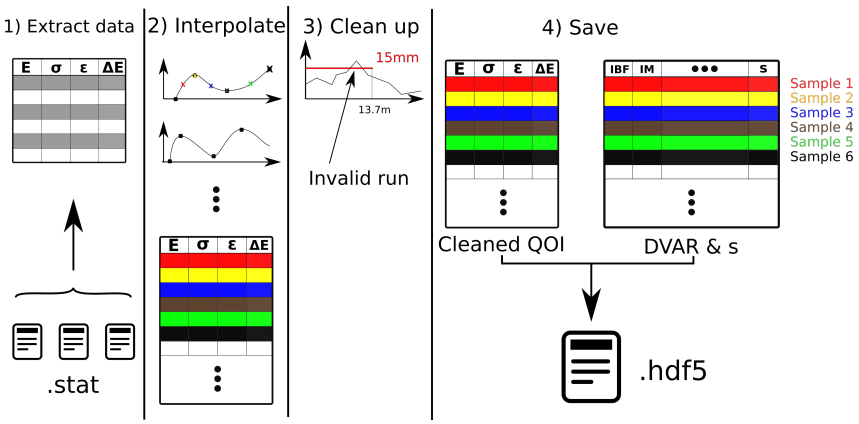



Figure 2.1: Steps needed to build the datasets. First, each run is extracted from the corresponding `.stat` file written by OPAL. Afterwards, the runs are interpolated and evaluated at the same longitudinal positions to build the data matrix. For some runs, the beamsize is bigger than the pipe in the first half of the machine. Those runs are considered invalid and removed before saving the final dataset. Each row in the QOI and DVAR matrices is a sample.

**Sample.** A sample is a tuple

$$(\mathbf{x}_i, \mathbf{y}_{\text{true},i}, s_i)$$
$$\text{s. t. } \mathbf{y}_{\text{true},i} = \mathbf{f}(\mathbf{x}_i, s_i),$$

where $\mathbf{f}(\mathbf{x}, s)$ represents OPAL (see Equ. 1.1). We dropped the $(\alpha, j)$ index and use a single index $i$ to identify samples from now on. Notice that all the samples that originate from the same OPAL run also share the same design variables.

### 2.2.2 Data format and splitting

There are two types of datasets: A training/validation (train/val) set, and a test set. The models are trained on the samples corresponding to the first 70% of OPAL runs in the train/val set. The remaining 30% form the validation set. It is used to choose the best set of hyperparameters. The final evaluation of the best model is performed on the test set. This ensures that neither the trainable nor the non-trainable parameters learn patterns that appear only in the training or validation set.

Both the design variables and the quantities of interest are organised in design matrices, see Sec. 5.1.5 in [15]. Each row contains one sample, and the columns represent the design variables or quantities of interest, respectively. The number of samples in each set can be found in Appendix C, which also contains information about the efficiency of the dataset creation and its computational cost.

### 2.2.3 Differences between the dataset for the forward and the inverse model

Both the forward models and the invertible models are trained on the same design variable configurations. The only differences are the number of positions and the processing steps. The dataset for the forward model contains longitudinal positions $s \in [0, 26]$ m. This is the region depicted in Fig 1.1, plus a drift that is not shown. Each OPAL run contributes 1541 samples. The dataset for the inverse model is slightly different. Here, the drift $[13.7, 26]$ m is not part of the dataset. Additionally, the positions that lie within a cavity or a solenoid are not part of the dataset. Therefore, each OPAL run contributes only 475 samples to the dataset.

Besides containing less positions, the datasets for the invertible model treats the values of the emittances differently: They are clipped to $[0, 200]$ mm mrad. Removing these locations and clipping the emittances makes the quantities of interest as functions of the longitudinal position smoother. It is crucial for achieving good inverse predictions. Without this step, the invertible network was not able to learn how to sample correctly.

### 2.2.4 Creation and Processing

All steps for building a dataset are depicted in Fig 2.1. First, an OPAL sampler run is performed and all the values of the quantities of interest are extracted from the OPAL output files (`.stat` and JSON files). OPAL logs the samples at a high resolution ($\approx 3$ mm), and not all runs contain the same number

of samples. We are not interested in patterns at such a high resolution. To save time during training and simplify the performance evaluation of the surrogate models, the quantities of interest for each run are interpolated. The interpolated functions are all evaluated at the same positions. The resolution between the samples of one run is higher inside the cavities (see Tab. 2.1).

Additionally, the positions of the YAG screens (see Sec. A.3) are added to the evaluation positions. This will allow to compare model predictions to OPAL exactly at the positions where measurement is possible in a lab experiment. More details are discussed in Sec. 2.5.

| Position | $\Delta s$ |
|---|---|
| Inside cavities | 5 mm |
| Outside cavities | 50 mm |

Table 2.1: Longitudinal resolution $\Delta s$ of the samples.

Not all OPAL runs are considered valid. The vacuum pipe in which the electrons travel has a diameter of 15 mm. This means that beams with a bigger transversal size cannot be produced by the AWA. Their behaviour deviates from the behaviour of a machine near the operating point. Therefore, such beams are removed if they become bigger than the pipe within the first 13.7 m. This region is depicted in Fig. 1.1.

Further downstream, there is only a drift, and most beams diverge. This region was not considered during the cleanup step. The reason is that even configurations that diverge in the drift region can still be useful for optimisations. For example, the surrogate model might be able to learn the relationship between the position of a beam waist and the design variables. The earlier the waist, the stronger the beam divergence. An optimisation can then find converging beams based on the learned relationship.

However, if the beam explodes shortly after leaving the gun, the beam cannot be used. The surrogate model is trained to be used for an upcoming experiment at Argonne National Laboratory (ANL). Its goal is to bring an initial energy modulation to the end of the beam. If the beam has a waist so early, the enrgy modulation disappears. Therefore, these configurations are not of interest for the surrogate.

## 2.3 Forward Model

Forward surrogate models are implemented in Python. The code makes use of the MLLIB framework [16]. The author took part in the design and implementation of the framework, together with Mélissa Zacharias and Sichen Li. MLLIB is developed internally at the Paul Scherrer Institute. It is an abstraction layer to provide a unified interface to various machine learning libraries. At the time of writing, there is support for TensorFlow [17] and scikit learn [18].

### 2.3.1 Data Transformation

The ranges of the design variables differ by three orders of magnitude. A model might become more sensitive to changes in the bigger design variables just because their values are bigger. To counteract this, the design variables and the path length are transformed to the range $[-1, 1]$. In principle, neural networks are able to learn this linear transformation themselves, but it is a good idea to make use of prior knowledge.

For the quantities of interest, another transformation is needed. The challenge is that the emittances have outliers that are one to two orders of magnitude bigger than the 99%-percentile, see Fig. 2.2. They are caused by the linac solenoids. The emittance has very narrow peaks at the positions of these beam elements. This renders linear transformations unsuitable. A linear transformation would have the problem that most values would be scaled to be very close together near $-1$ so that the maximum emittance values could be mapped to 1. One could try to choose the transformation so that the 99%-percentile of the emittances (not the maximum) is mapped to 1. But then the peak values are still very far away from the bulk of the values. This means that the network has to learn to predict values of different orders of magnitude.

Therefore, we need a transformation that separates close values better and reduces the difference between the solenoid peaks and the majority of the solenoid values. We decided to use the following transformation:

1. The values are shifted to be positive.
2. The logarithm is taken.
3. The resulting range is scaled to $[-1, 1]$.

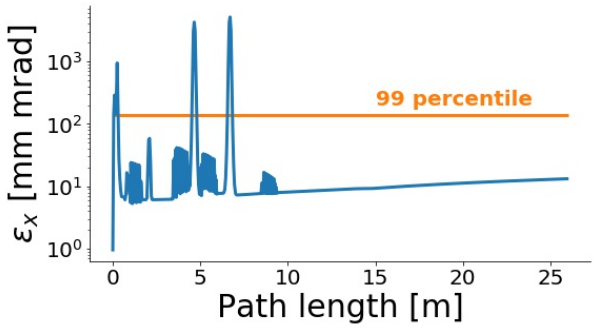



Figure 2.2: Emittance for a single OPAL run. 99 percent of all samples in the test set have have values that lie below the orange curve.

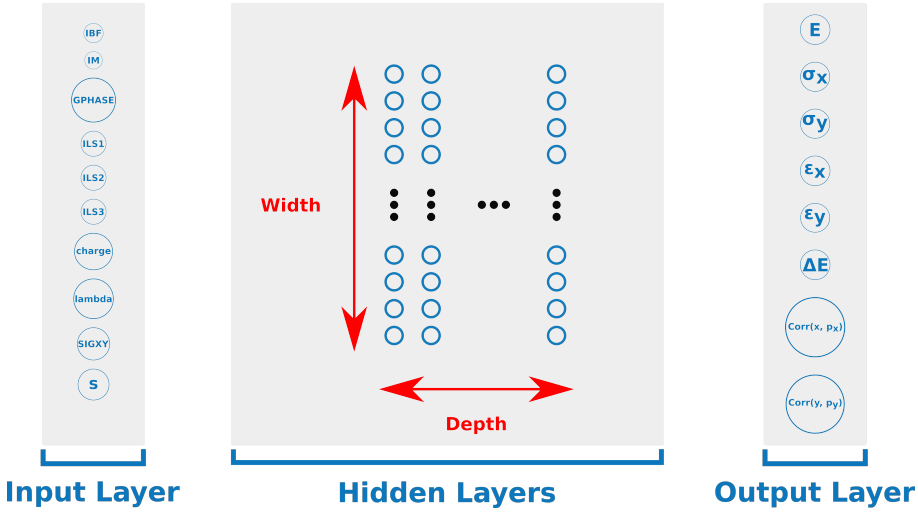



Figure 2.3: Architecture of the forward models. The size of the neurons is only due to the text size and does not represent any information. All layers are fully connected, the connections are not shown to improve readability. The shapes of the hidden layers determine the model capacity.

### 2.3.2 Network Architectures & Parameters

**Architecture**

We examined fully connected feedforward networks, see Fig. 2.3. Such a network has an input layer of size 9, which is the number of design variables plus one for the longitudinal position $s$. The hidden layers are organised in $n_d$ ($d$ for depth) layers of width $n_w$. We will use the shorthand notation $n_d \times n_w$ to refer to such a network. For example, an $8 \times 20$ network has 8 hidden layers of 20 neurons. The output layer has a size of 8 neurons and no activation function. The last transformation is therefore linear.

**Parameters**

The family of fully connected networks comes with many hyperparameters: Each hidden layer needs an activation function ($n_d$ parameters). The number of neurons in each layer has also to be fixed, requiring another $n_d$ parameters. The number of hidden layers is an additional parameter.

The choice of the optimisation algorithm represents one more hyperparameter, and the algorithm itself can possibly be tuned with parameters of its own. Most gradient-based methods require a learning rate $\eta$ and a minibatch size $b$. Finally, one needs to decide how many epochs (iterations over the dataset) one should train for.

In order to restrict the number of hyperparameter configurations to be checked, we decided to use the following fixed set of parameters:

- Each hidden layer has the same number of units.
- Each hidden neuron uses the ReLU activation function.
- The Adam algorithm [8] with default parameters was used for training. Manual trial and error showed that a learning rate of $\eta = 10^{-4}$ leads to good convergence. Higher values of $\eta$ lead to

| Parameter | Options |
|---|---|
| Batch size | 128, 256 |
| Width of the hidden layers | 300, 400, 500, 600, 700 |
| Depth of the hidden layers | 6, 7, 8, 9, 10 |

Table 2.2: Hyperparameters to be evaluated and compared.

constant or almost constant training and validation losses.

- Trial and error showed that the models should be converged after at most 200 epochs. To speed up training and avoid overfitting, early stopping is used: If the validation MAPE does not decrease by at least 1% for 20 epochs, the training is stopped.

The remaining parameters are to be optimised during a hyperparameter scan. They are summarised in Tab. 2.2.

### 2.3.3 Performance Metrics and Model Selection

**Loss**
The loss function describes what the trainable parameters in a neural network are optimised for. As already described in Sec. 1.2, the mean absolute error (MAE) is used (Equ. 1.3) to compare a prediction of the surrogate model $\mathbf{y}_{\text{pred},i}$ to the ground truth calculated by OPAL $\mathbf{y}_{\text{true},i}$. Because this loss is calculated during training, both $\mathbf{y}_{\text{pred},i}$ and $\mathbf{y}_{\text{true},i}$ are assumed to be in the scaled space, i. e. in the range $[-1, 1]$.
**Measure of convergence**
Empirical trial and error showed that the MAE is good for the convergence of the network. However, we need to define a smallest significant improvement of convergence in order to implement early stopping. It is not so easy to define a threshold for the MAE. Therefore it is decided to use the Mean Absolute Percentage Error (MAPE) as an in interpretable measure of convergence. First, we define the percentage error of a forward prediction:

$$\epsilon_{i,j} = \left| \frac{y_{\text{pred},i,j} - y_{\text{true},i,j}}{y_{\text{true},i,j}} \right| \cdot 100\%, \tag{2.1}$$

where $i$ is the sample index and $j$ denotes the quantity of interest. Next, the MAPE is calculated as follows:

$$\text{MAPE} = \frac{1}{N_s \cdot N_{\text{QOI}}} \sum_{i=1}^{N_s} \sum_{j=1}^{N_{\text{QOI}}} \epsilon_{i,j} \tag{2.2}$$

During training, it is calculated after applying the preprocessing, i. e. both $y_{\text{pred}}$ and $y_{\text{true}}$ are in the range $[-1, 1]$.
**Performance metrics**
The final goal is to predict the QOIs at different positions. For practical concerns, it is also important to know the limitations of a surrogate model. Therefore, a measure of prediction error at a specific position is needed.

First, we aggregate the percentage error for each QOI at each position in the test set. Let $I_s := \{i \in \{1, ..., N_s\} | s_i = s\}$ be the ordered set of sample indices at position $s$. Then $\epsilon^s_{j,\alpha}$ is the $\alpha$ percentile of $\epsilon_{i,j}$ over all $i \in I_s$. One can write them as components of a vector and obtain the vector $\epsilon^s_\alpha$.

This quantity allows to define something similar to confidence levels. By calculating the percentiles of $\epsilon$ over the test set, we get an estimate of how good we can expect a model to perform on a previously unseen DVAR configuration at a certain position. The "confidence interval" for a prediction at the confidence level $\alpha$ is then given by

$$\tilde{\mathbf{f}}(\mathbf{x}, s) \left(\mathbf{1} \pm \epsilon^s_\alpha\right), \tag{2.3}$$

where addition/subtraction is executed componentwise and $\mathbf{1}$ is the vector with 1 in every entry.
**Model selection**
For the hyperparameter scan, we need to compare the quality of many models so we can select the best one. Ideally, a single indicator should summarise the quality of the prediction. A straightforward measure of prediction error is the maximum percentage error over all QOIs

$$e_i = \max_j \epsilon_{i,j}, \tag{2.4}$$

where the index $i$ indicates that the error is calculated for the $i$-th sample.

One can calculate the $\alpha$-quantile over all samples, $e_\alpha$. This scalar quantity summarises the prediction error of a forward model. It is calculated in the unscaled space, i. e. in physical units. Therefore, it allows to select the best end-to-end model, taking into account the entire prediction pipeline, including preprocessing.

## 2.4 Invertible Model

The invertible models are implemented in MLLIB as well. The affine coupling block and the permutation layer are not available in TensorFlow, so they are implemented based on PyTorch code from the FrEIA library [19]. During the thesis, it was considered to use the library directly, but documentation was very sparse. At the end of the thesis, the situation has improved very much, and it is recommended to use FrEIA for future work.

### 2.4.1 Data Transformation

| QOI | $a$ | $b$ |
|---|---|---|
| $E$ [MeV] | 0 | 50 |
| $\sigma_x$ [mm] | 0 | 15 |
| $\sigma_y$ [mm] | 0 | 15 |
| $\epsilon_x$ [mm mrad] | 0 | 200 |
| $\epsilon_y$ [mm mrad] | 0 | 200 |
| $\Delta E$ [MeV] | 0 | 1 |

Table 2.3: Values used for the scaling of the quantities of interest (QOIs) for the inverse models. A linear transformation maps the range $[a, b]$ to $[-1, 1]$.

The DVARs are transformed with the `QuantileTransformer` taken from scikit-learn. It maps the quantiles of each design variable to the ones of a normal distribution. Afterwards, a MinMax transformation is applied to make sure all DVARs lie in $[-1, 1]$.

If the `QuantileTransformer` was not used, the invertible model would sample design variables that lie orders of magnitude outside of the ranges in the dataset, as trial and error showed.

The quantities of interest are scaled linearly to the range $[-1, 1]$. The parameters of the transformation are listed in Tab. 2.3.

### 2.4.2 Network Architectures & Parameters

The hyperparameters for an invertible network are discussed in detail in Sec. 1.4.4. It is clear that some parameters need to be fixed to make the hyperparameter scan computationally feasible. Here are their values:

- The activation function for each neuron is the ReLU activation.
- The loss weights are crucial for convergence of the training. If they are not chosen correctly, learning is impossible. Trial and error lead to the following weights:
  $w_x = 400$, $w_y = 400$, $w_z = 400$, $w_{\text{recon}} = 3$, $w_{\text{artificial}} = 1$
- The batch size was chosen to be 256 because it lead to the best convergence and training time (on 12 CPU cores on Merlin6).
- Adam with default parameters [8] is chosen as the optimisation algorithm. The learning rate is $\eta = 10^{-4}$.

The remaining hyperparameters are the dimensions of the latent space and the padding. Further, one needs to set the number of neurons and how to arrange them. There is the number of affine coupling blocks. Each block contains two two coefficient networks. We assume both coefficient networks have the same architecture, and that this architecture is the same in all layers.

The following shorthand notation is used: "$n_b \times n_d \times n_w$" means that there are $n_b$ affine coupling blocks (separated by a permutation block), and each coefficient network contains $n_d$ layers of $n_w$ neurons.

### 2.4.3 Performance Metrics and Model Selection

**Loss**
The loss function is described in detail in Sec. 1.4 and shown in Equ. 1.5.

**Performance metric for the forward prediction**
This is the same task as the forward model solves. Therefore, the same metric, the MAPE, is used (see Equ. 2.2).

**Performance metric for the inverse prediction (sampling)**
For the inverse prediction, we start with a target beam configuration. The goal is to obtain a machine setting (DVAR configuration) such that the corresponding beam is as close as possible to the target. It is therefore natural to assess the prediction quality by calculating the distance between the sampled and the target QOI configurations. Analogue to Equ. 2.1, we can define the percentage error of the sampling. The symbol $\xi$ is used to avoid confusion with the percentage error of the forward prediction:

$$\begin{aligned} \xi_{i,j} &:= \left| \frac{y_{\text{sampled},i,j} - y_{\text{true},i,j}}{y_{\text{true},i,j}} \right| \cdot 100\%, \\ \tilde{\xi}_{i,j} &:= \left| \frac{\tilde{y}_{\text{sampled},i,j} - y_{\text{true},i,j}}{y_{\text{true},i,j}} \right| \cdot 100\%, \end{aligned}$$

where i is the sample index and j denotes the quantity of interest. The tilde indicates that a validation model is used for the evaluation of the sampled design variables (see also Equ. 1.7). Model selection requires a single quantity to compare invertible models. Analogue to Equ. 2.4, we define the maximum percentage error for the inverse prediction:

$$\begin{aligned} \tilde{\chi}_i &:= \max_j \tilde{\xi}_{i,j} \\ \chi_i &:= \max_j \xi_{i,j} \end{aligned} \tag{2.5}$$

We want to validate the invertible model by trying to reproduce all QOI configurations in the test set. There are more than $10^6$ samples. Since OPAL takes about 20 min to evaluate a single DVAR configuration, it is not feasible to evaluate all sampled machine settings with it. Therefore, we need to use a validation surrogate model to estimate the sampling error. We use the following algorithm to calculate the sampling error for the $i$-th sample in the test set:

1. Sample a DVAR config $\mathbf{x}_{\text{sampled},i}$.
2. Predict the corresponding QOI config with a validation model:
   $\tilde{\mathbf{y}}_{\text{sampled},i} = \tilde{\mathbf{f}}(\mathbf{x}_{\text{sampled},i})$
3. Calculate $\tilde{\chi}_i$.

The quality of the inverse prediction can be improved with a simple trick. Instead of simply sampling one configuration, one could sample $n_{\text{tries}}$ DVAR configurations, evaluate them with the forward prediction of the invertible model, and select the DVAR configuration with minimum error. This best-of-n subsampling is already included in step two above. More details about the subsampling trick can be found in Appendix I.

**Model Selection**
The invertible model should be able to predict a DVAR configuration that leads to a beam whose properties are close to some target values. Usually, scientists are interested in beam properties at the very end because that is where the user provided target lies. Therefore, it is our primary goal to make the sampling as accurate as possible at the end of the machine, which is at 13.65 m for the invertible model. The quantiles $\tilde{\chi}_\alpha$ of the maximum percentage error will be used to select the best model from the hyperparameter scan.

## 2.5 Validation

After training the best model, it needs to be validated. First, the prediction error of is evaluated on the test set. The model has not seen this set before, neither during training nor during hyperparamter tuning. This ensures we do not overfit the trainable and untrainable parameters to the training and validation set, respectively. In the end, the predictions of the model are compared to the ones by OPAL.

There is another way to ensure that the surrogate model is a valid replacement for OPAL. A constrained multi-objective optimisation is performed, once using the surrogate model, and once using OPAL. Afterwards, the non-dominated sets are compared to see if both models arrive at similar results.

The optimisation problem was chosen because it emerged from the collaboration with the Argonne National Laboratory. Unfortunately, the built-in optimiser is flawed when too many constraints are imposed. Further, OPAL is too slow to be optimised with an external optimiser. Therefore, the optimisation is only performed with the surrogate. To validate the result, the optimal configurations are evaluated with OPAL. Details are discussed in section 3.5.

Originally it was planned to take lab measurements at the AWA facilities in April 2020. Those would have been compared to the predictions of the surrogate model. However, these plans had to be cancelled due to the outbreak of COVID-19.

## 2.6 Dashboards

A fast surrogate model is most useful when even people without a machine learning background can use it. Further, it is hard to get a feeling for the quality of the predictions, especially as a physicist who is not exposed on a regular basis to the metrics used in machine learning. For these reasons, four browser-based dashboards were developed, using the Dash library [20]. All plots are interactive (selection of axis ranges, zooming, saving of the current view as `PNG` file) thanks to the underlying framework Plotly [21].

The first dashboard provides a GUI for the forward predictions. The second dashboard does the same for the inverse pass of an invertible model. Further, there is a GUI that visually compares the prediction of a forward prediction $\tilde{\mathbf{y}}_{\text{pred},i}$ or $\hat{\mathbf{y}}_{\text{pred},i}$ to the ground truth (OPAL) $\mathbf{y}_{\text{true},i}$. The last dashboard compares the resulting beam of an inverse prediction to a beam in the dataset.

All dashboards can be run locally. They are also uploaded to Heroku, an app hosting platform [22]. In the latter case, one just needs to access a website, no installation is required. This lowers the entry barrier for non-developers additionally and provides further incentive to make use of the model, and Tab. 2.4 for links to the repositories and the Heroku apps.

Please consult Appendix D for details about the dashboards.

| Dashboard | Repo | Heroku URL |
|---|---|---|
| Forward prediction GUI | `https://gitlab.psi.ch/bellotti_r/masters_thesis_forward_surrogate_gui` | `https://awa-wiggler-surrogate.herokuapp.com/` |
| Inverse prediction GUI | `https://gitlab.psi.ch/bellotti_r/masters_thesis_inn_dashboard/tree/master/` | `https://awa-wiggler-inverse.herokuapp.com/` |
| Forward prediction vs. OPAL | `https://gitlab.psi.ch/bellotti_r/masters_thesis_forward_surrogate_vs_truth` | `https://awa-wiggler-surrogate-vs-truth.herokuapp.com/` |
| Inverse prediction vs. OPAL | `https://gitlab.psi.ch/bellotti_r/masters_thesis_inbvertible_surrogate_vs_truth` | `https://awa-wiggler-inn-vs-opal.herokuapp.com/` |

Table 2.4: Links to the repositories the dashboards and to the web apps hosted at Heroku.

# Chapter 3

# Results for the Forward Model

## 3.1 Baseline model

Before the hyperparameter scan is run, manual experimentation is needed to find a good point in hyperparameter space. Additionally, it is not clear how the data should be transformed before feeding it into the model. This manual exploration phase establishes the data preprocessing pipeline and a range of hyperparameters to be scanned. Further, it results in a first surrogate model, the baseline model. Its parameters can be found in Appendix E.

### 3.1.1 Training

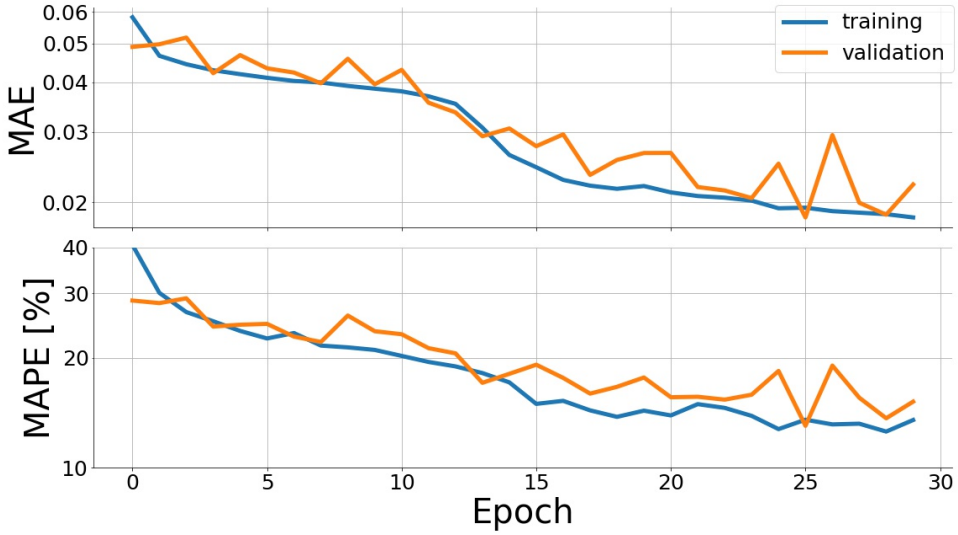



Figure 3.1: Convergence of loss and performance metric during training for the baseline forward surrogate model. The MAE takes the role of the loss function.

The baseline model is trained for 30 epochs. After each epoch, the loss (MAE) and MAPE are calculated over the training and the validation set. The curves are depicted in Fig. 3.1. One can see that the model is not fully converged yet. Training for more epochs would probably decrease the loss function (see also Sec. 3.3.1). This trail is not followed further because the only purpose of the baseline model is to find a good set of hyperparamters. Based on these results, a hyperparameter scan is performed. The latter makes use of early stopping, which relieves us from choosing the number training epochs.

Notice that the loss on the validation set follows the one of the test set. This means that the model does not overfit to the training set yet. The validation curve is much noisier than the training curve. We assume that this comes from the fact that the training set contains more than twice as many samples as the validation set due to our 70:30 split. The bigger variance for the MAE and MAPE on the validation set is then a consequence from the central limit theorem.

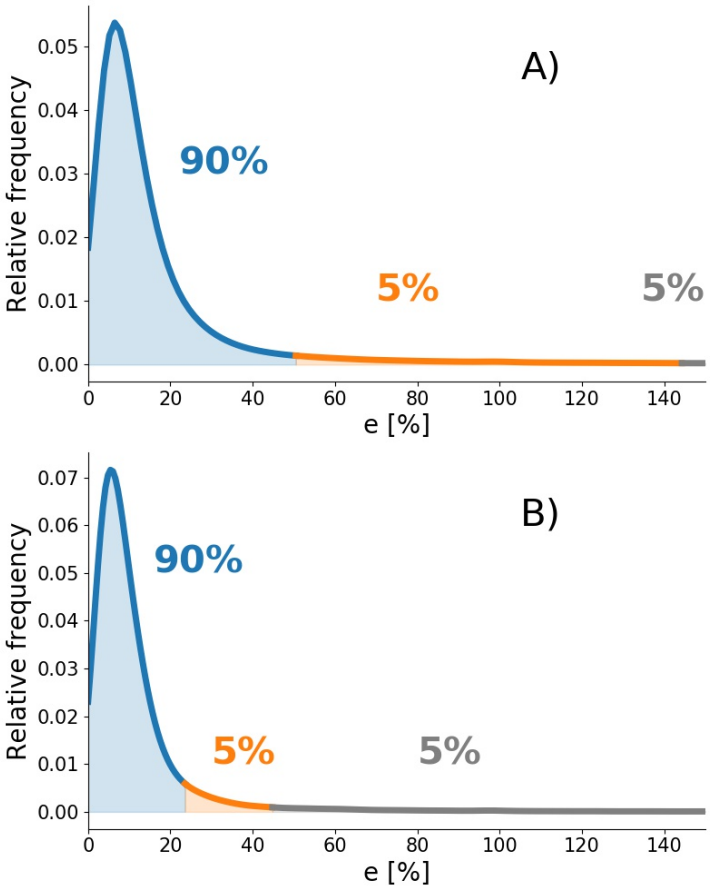



Figure 3.2: Kernel density estimate of the maximum percentage error distribution of the baseline forward surrogate model. The plot shows a kernel density estimation. The tails of the distribution that are bigger than 150% are omitted for better readability. A) The density estimate is calculated taking into account all samples in the test set. B) Samples corresponding to positions within cavities or linac solenoids are excluded from the density estimate.

### 3.1.2 Prediction error

Once the model is trained, we have to validate it. As already discussed in Sec. 2.3.3, the maximum of the absolute percentage error (Equ. 2.4) is used as the measure of model accuracy. Its distribution over the test set is depicted in Fig. 3.2. Both plots depict a kernel density estimate (KDE) of the maximum prediction error. The upper plot takes into account all the samples at all positions, while the lower plot ignores the samples at positions within cavities and linac solenoids. Both plots show that the tail is very long, but does not represent many data points. Therefore, everything bigger than the 95 percentile is considered outliers of the error distribution. A comparison between the subplots shows that many of the outliers correspond to positions inside the cavities or linac solenoids. Within these beam elements, the beam oscillates, which makes prediction difficult. This observation is confirmed in Fig. 3.3. The plot shows several percentiles of the prediction error aggregated by position. For example, one can see that for half of the samples with $s = 26$ m in the test set, the maximum prediction error is smaller than approximately 10%. At the positions of the cavities, the error is between 15 and 20 percent due to the oscillations.

The big prediction errors at positions within cavities and solenoids bear not much relevance. On the one hand, it is impossible to measure the beam within a cavity or solenoid. On the second hand, the experiment for the AWA wanted to optimise the beam at 26m, i. e. at the very end of the region for which the model was trained. Nevertheless, it should be highlighted that even within the cavities, the median error is about $10\% - 20\%$, and the upper quartile of the maximum percentage error is only approximately 30% within the last three cavities.

At the very end, the median error is only 10%. The upper quartile of the error on the test set is about 15%, the 90 percentile lies at 20% error. The 95 percentile lies at 35% error. The maximum prediction error is an upper bound of the prediction error for each individual quantity of interest. As an additional

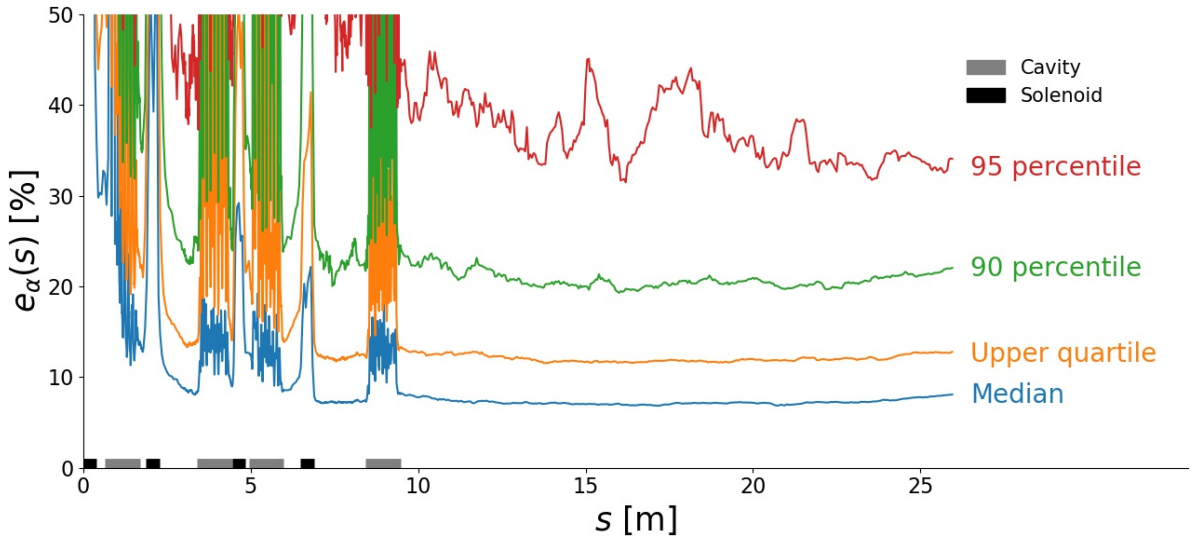



Figure 3.3: Maximum percentage error of the baseline forward surrogate model at different positions. The y-axis was cut off at 50% and the locations of the cavities and linac solenoids are highlighted on the x-axis.

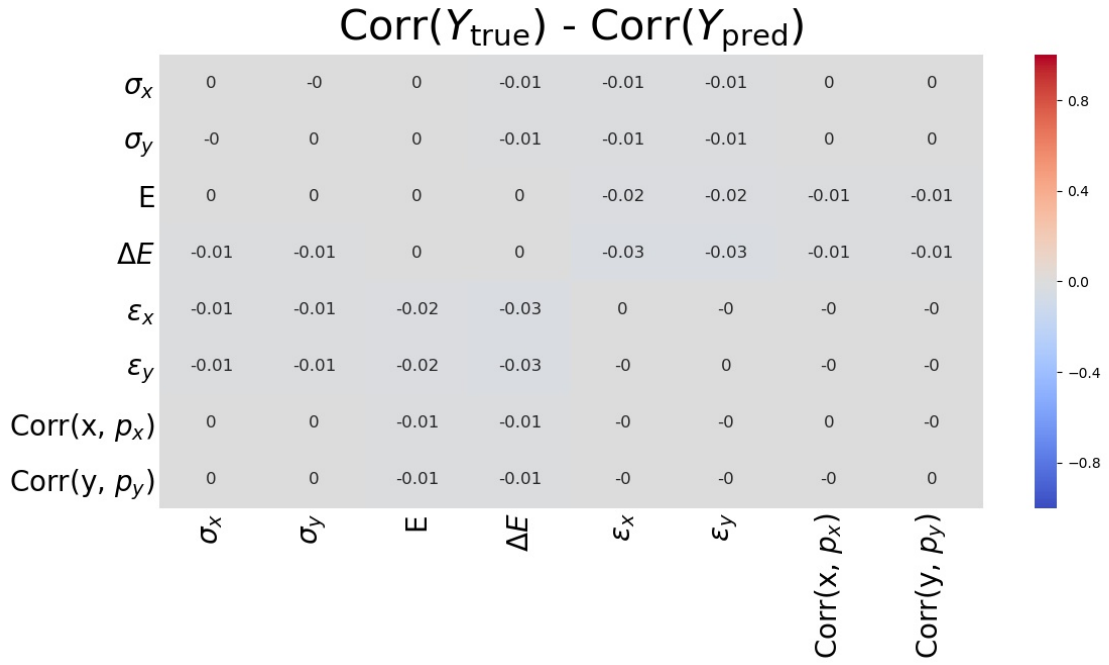



Figure 3.4: Difference between the QOI correlation matrix of the ground truth and the one of the predictions over the test set. The values are rounded to two decimal digits.

validation, the correlation between the quantities of interest can be examined. It is compared to the one calculated over all samples in the test set. The difference of both correlation matrices is shown in Fig. 3.4. It is almost zero. This provides one more hint that the baseline forward model is indeed an accurate surrogate model.

Considering all these arguments, we can conclude that already the baseline forward model is good and useful for applications.

## 3.2 Hyperparameter Scan

### 3.2.1 Overview

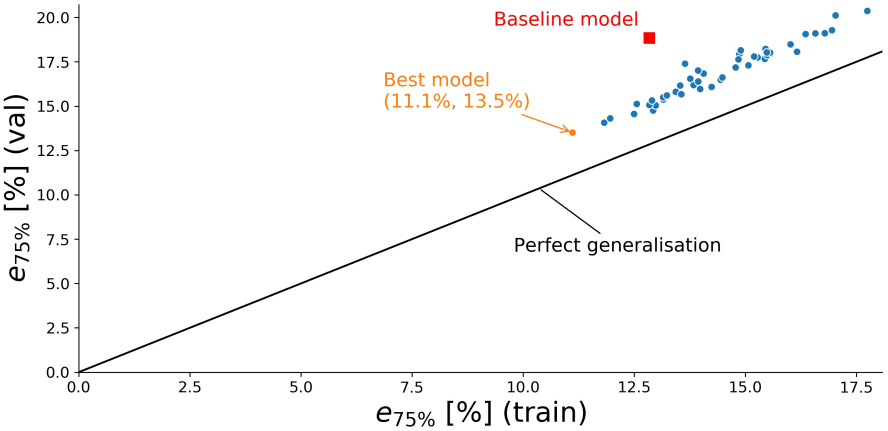



Figure 3.5: Results of the hyperparameter scan.

The results of the hyperparameter scan are given in Fig. 3.5. Each dot represents a single model. The horizontal axis displays the 75 percentile of the maximum percentage error on the training set, the vertical axis the corresponding quantity on the validation set. Perfect generalisation is reached on the diagonal, which is plotted as a reference. The best model is highlighted, and the baseline model is drawn as well.

All the configurations of the hyperparameter scan have much better generalisation than the baseline model. This can be explained by the number of epochs: For the baseline model, a fixed number of 30 epochs is chosen based on a manual tradeoff between convergence and time of training. For the hyperparameter scan, however, early stopping is used to decide how long to train. This strategy seems to be superior: The value of $e_{75\%}$ on the validation set of the best model surpasses the one of the baseline model by almost 5 percentage points.

The choice of the 75 percentile of the maximum percentage error is arbitrary. However, the plots for the 50 and 90 percentile look qualitatively the same, and the same model has the lowest validation error. Only for the 95 and 99 percentiles, another model is slightly better, but we decide that the upper quartile is more important than the tails of the error distribution.

Appendix F contains a table of all hyperparameter configurations, including the hyperparameters for the best forward model. Two configurations have taken more than 4 days for training. Their jobs timed out on the cluster batch system and are ignored.

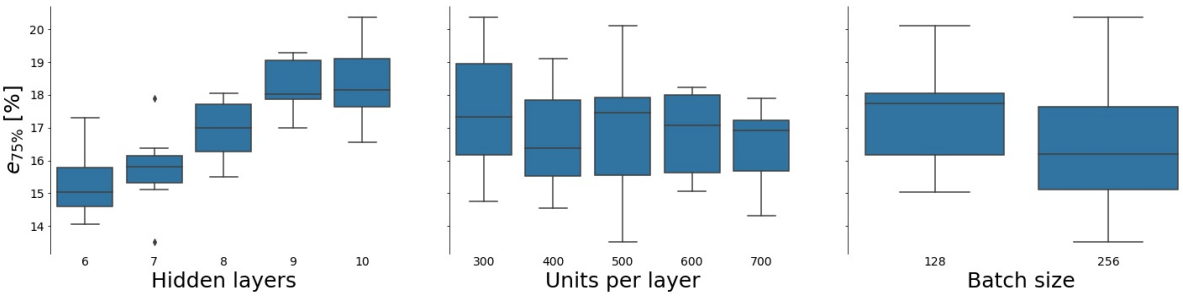



Figure 3.6: Influence of the hyperparameters on model performance. The outliers correspond to points that are further away from the quartiles than 1.5 · Interquartile range.

The influence of different parameters on model performance is shown in Fig. 3.6. Adding more hidden layers increases the prediction error. The optimisation algorithm does probably not find good local optima for these more complicated models. The influence of the number of units per hidden layer is not so clear. There seems to be a small tendency that more units lead to more precise models, but further research is needed. The same is also true for the batch size. One has to mention that training time is significantly shorter for a batch size of 256. Together with the fact that model performance does not depend strongly on the batch size, a value of 256 is recommended.

## 3.3 Best Model

### 3.3.1 Training

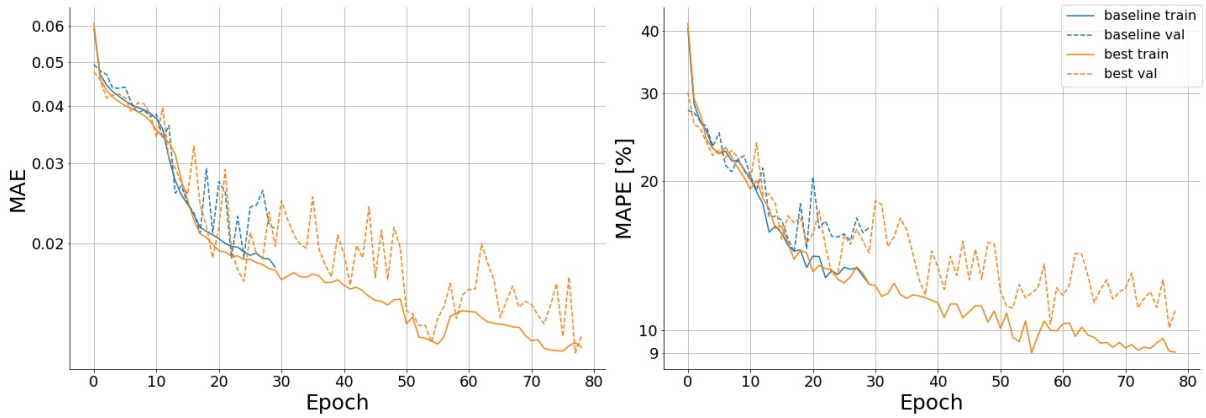



Figure 3.7: Training curves for the best and the baseline forward model.

The training curves depicted in Fig. 3.7 show that the biggest difference between the baseline model and the best model is the number of epochs. One can see that the baseline model is not fully converged yet. It is not clear if convergence would continue like this, or start to flatten out at a higher level than for the best model.

The worse generalisation that is visible in Fig. 3.5 could be explained by oscillations, especially in the validation loss. At the end of training, the validation MAPE has either stopped converging or is in a local maximum due to oscillations. In contrary, the validation MAPE of the best model is in a local minimum of the oscillation at the end of its training, leading to better generalisation.

### 3.3.2 Prediction error

The distribution of the maximum prediction error (see Fig. 3.8 and Equ. 2.4 for the definition) has not improved very much compared to the baseline model. The quantities of interest inside the cavities and linac solenoids are now better captured. This can be seen by comparing the 95 and 99 percentiles of the error to the ones of the baseline model. They are at smaller errors and change less when omitting the cavity and linac solenoid samples.

The reduction in prediction error compared to the baseline model is better visible in Fig. 3.9. The upper quartile is now about 10%, and the median of the error is even smaller than 10%. Both the predictions for positions within the cavities and solenoids are predicted much better than for the baseline model. The upper quantile of the maximum percentage error is around 10% for the cavities and 20% for the solenoids. However, the tail of the error is still very high at these positions. The error of individual quantities of interest is depicted in Fig. 3.10. As expected, all percentiles of the prediction error are smaller than the maximum error (see Equ. 2.4).

A model can only be considered fully validated if its predictions agree with empirical measurement. Unfortunately, lab measurements were not possible due to the outbreak of COVID-19. The 95 percentile for the prediction error at each YAG screen is given in Tab. 3.1. The values can be interpreted as error at 95% confidence.

The energy is predicted almost perfectly, and the energy spread has a very small error (except for the first YAG). The error of the beam sizes is about 13%, the ones of the correlations 15%. The emittances have the biggest error, 23%. It can be concluded that the maximum percentage errors are probably determined by the error on the emittances, at least at 95% confidence.

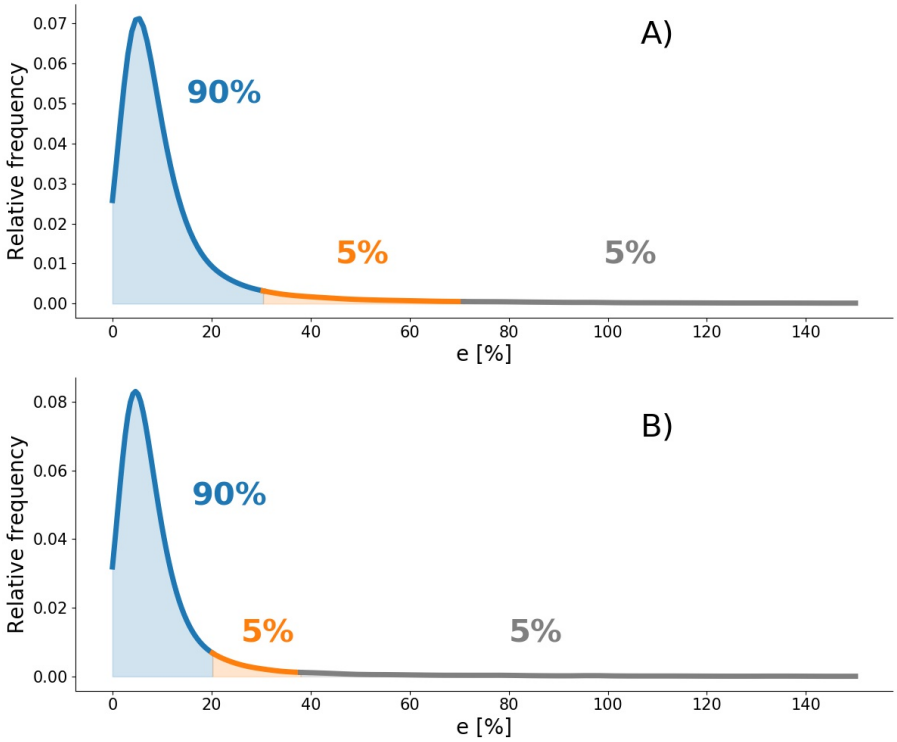



Figure 3.8: Kernel density estimation of the maximum percentage error distribution of the best forward model. Tails of the error that are bigger than 150% are omitted for better readability. A) The density estimate is calculated taking into account all samples in the test set. B) Samples corresponding to positions within cavities or linac solenoids are excluded from the density estimate.

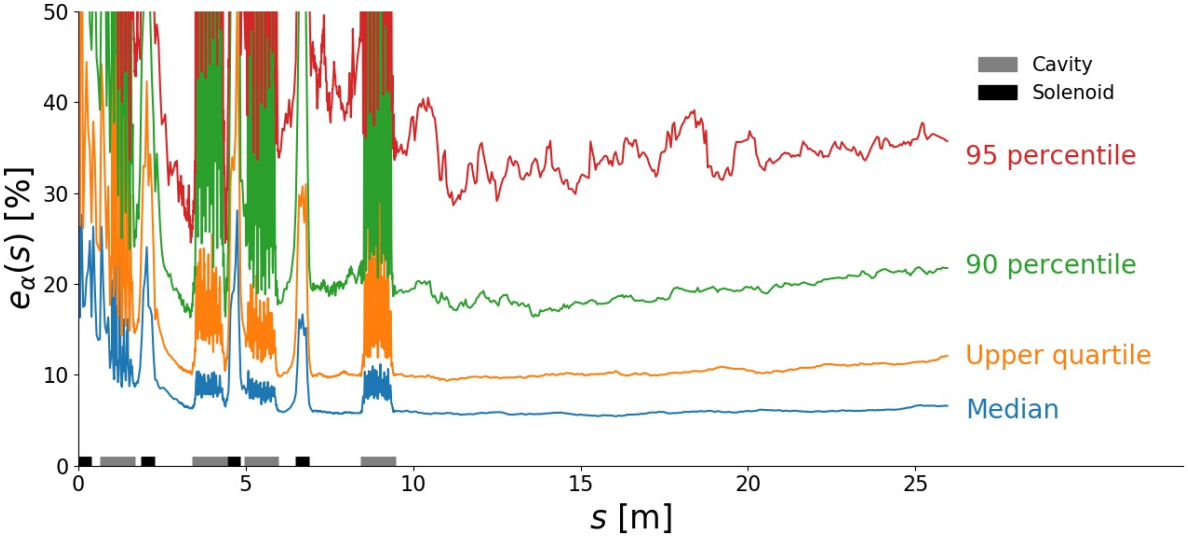



Figure 3.9: Maximum percentage error of the best model depending on the position. The grey and black bars along the x-axis indicate the positions of cavities and linac solenoids, respectively.

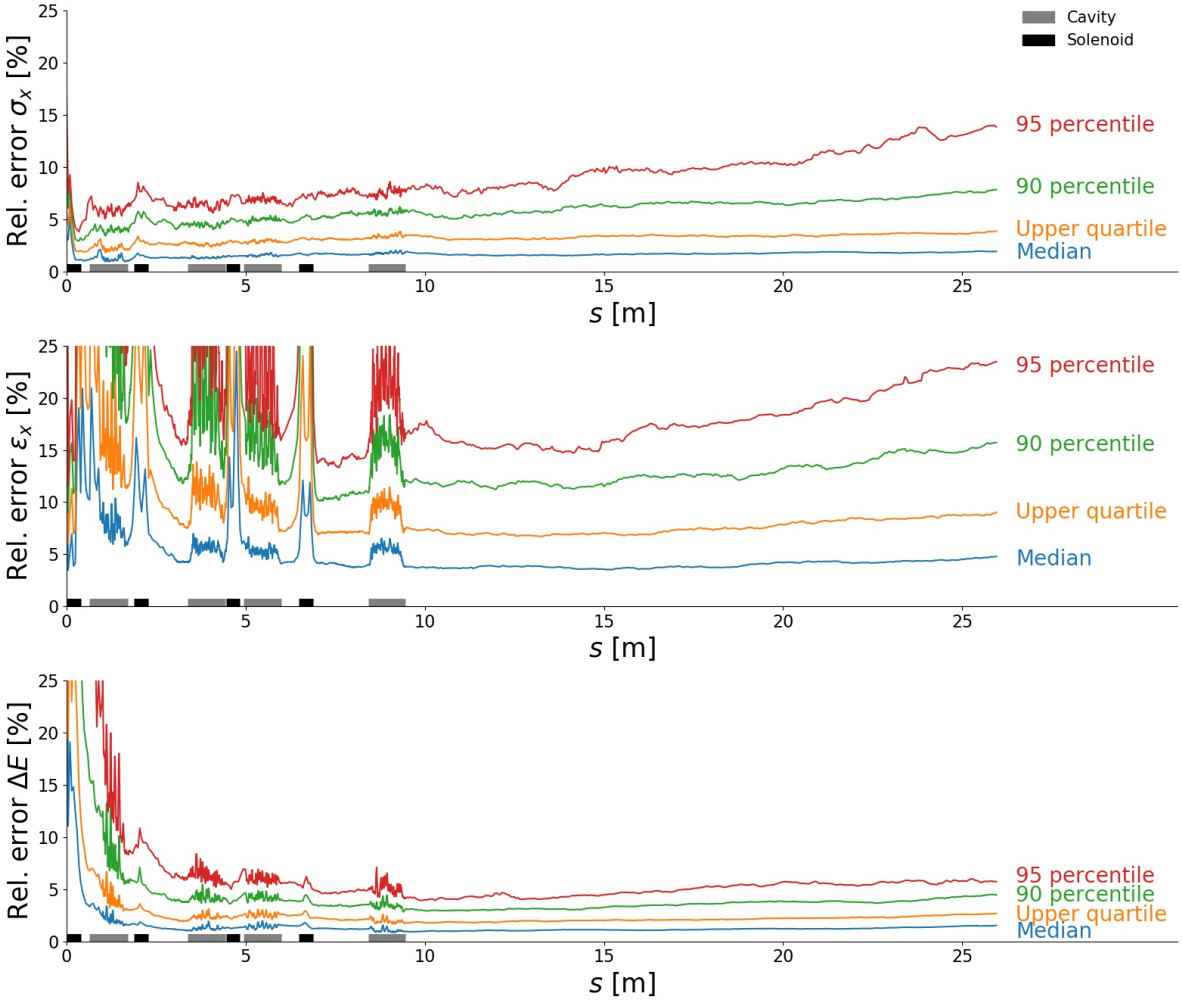



Figure 3.10: Relative prediction error of the best model depending on the position. The grey and black bars along the x-axis indicate the positions of cavities and linac solenoids, respectively.

| | $E$ | $\Delta E$ | $\sigma_x$ | $\sigma_y$ | $\epsilon_x$ | $\epsilon_y$ | Corr$(x, p_x)$ | Corr$(y, p_y)$ |
|---|---|---|---|---|---|---|---|---|
| YAG 0 | 2.3 | 35.1 | 4.9 | 4.8 | 62.2 | 63.6 | 3.5 | 3.5 |
| YAG 1 | 1.0 | 6.3 | 6.2 | 6.0 | 18.1 | 18.3 | 19.0 | 17.4 |
| YAG 2 | 0.2 | 5.7 | 6.5 | 6.5 | 16.4 | 17.9 | 22.3 | 21.2 |
| YAG 3 | 0.1 | 4.2 | 7.6 | 7.5 | 16.3 | 16.6 | 24.1 | 24.0 |
| YAG 4 | 0.1 | 4.3 | 7.7 | 7.0 | 15.2 | 15.5 | 20.3 | 19.2 |
| YAG 5 | 0.1 | 4.2 | 7.5 | 7.3 | 15.2 | 15.1 | 21.6 | 21.1 |
| YAG 6 | 0.1 | 4.5 | 9.8 | 9.4 | 15.6 | 15.8 | 22.9 | 21.1 |
| YAG 7 | 0.1 | 4.8 | 9.5 | 9.5 | 16.7 | 17.0 | 20.7 | 20.3 |
| YAG 8 | 0.1 | 5.4 | 10.3 | 9.7 | 17.4 | 18.2 | 19.1 | 19.0 |
| YAG 9 | 0.1 | 5.5 | 10.7 | 10.0 | 18.7 | 19.0 | 17.4 | 18.0 |
| YAG 10 | 0.1 | 5.7 | 12.7 | 11.9 | 20.8 | 21.0 | 16.8 | 19.1 |
| YAG 11 | 0.1 | 6.0 | 13.4 | 12.4 | 22.9 | 23.3 | 15.1 | 14.8 |

Table 3.1: 95 percentile of the relative error at the positions of the YAG screens. The unit of all values is percentage.

## 3.4 Dependence on training set size

All models are trained on the big train/val dataset containing roughly 20′000 OPAL runs. A priori, it is not clear if the training set is big enough, or if the model would benefit from adding more data. To answer this question, the best model was also trained on the medium train/val dataset (approximately 8′000 OPAL runs) and the small one (4′000 OPAL runs). Details about the datasets can be found in Appendix C.

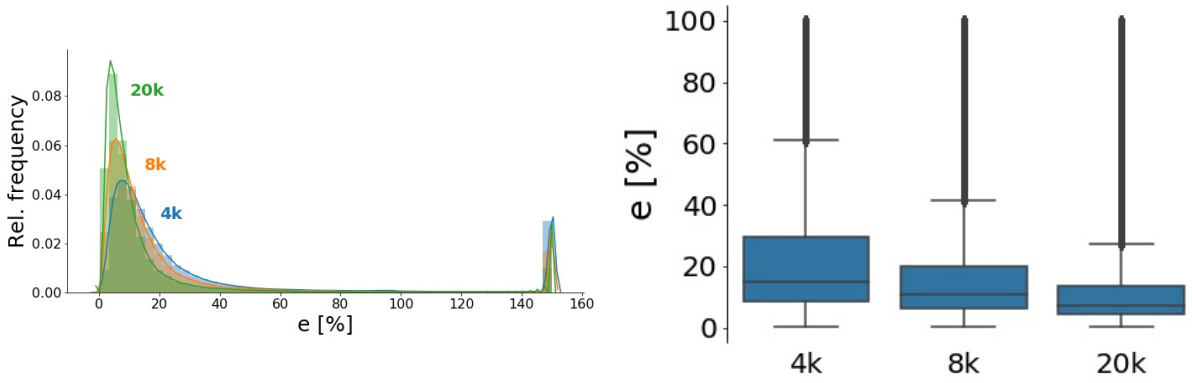



Figure 3.11: Dependence of the maximum percentage error on the number of OPAL runs in the train/val set. The architecture of the best model on 20k samples is used to train all models. In the left plot, the peak at 150% is an artefact from clipping the error to the range [0, 150]%.

All models are evaluated on the test set. The results are depicted in Fig. 3.11. The left plot shows the distribution of the maximum percentage error, the right one the corresponding boxplots. Both plots show that there is a clear decrease in error when more samples are used to train the model.

Several interpretations are possible. It is not clear if the dataset is not fully representative for the underlying model that is implemented in OPAL. Another explanation is that the examined architecture cannot make use of all the information in the training data.

In the plot from the hyperparameter scan, Fig. 3.5, one can see that all the models lie more or less on one line. This means they have a similar generalisation behaviour, which is not perfect yet. The arguments above might also be interpreted as a hint that the examined model families might not be able to fully use the information encoded in the training set.

A table for quantiles of the maximum percentage error for different training set sizes is shown in Tab. 3.2.

| | 0.25 | 0.50 | 0.75 | 0.90 | 0.95 | 0.99 |
|---|---|---|---|---|---|---|
| 4k | 8.5 | 14.8 | 29.5 | 117.2 | 200.2 | 787.6 |
| 8k | 6.1 | 10.8 | 20.2 | 52.5 | 150.4 | 643.9 |
| 20k | 4.2 | 7.3 | 13.4 | 30.4 | 70.8 | 434.7 |

Table 3.2: Dependendence of the maximum percentage errors on the number of OPAL runs in the train/val dataset. The unit of all values is percent. The column headers denote the quantiles of $e$.

## 3.5 Optimisation

We participate in a collaboration with the Argonne National Laboratory (ANL). The goal is to prove that an undulator can be used to convert a density modulation into an energy modulation. The undulator is very sensitive with respect to the characteristics of the entering beam. This imposes many constraints on the beam. Therefore, an optimisation is needed. Since it is a real world application, this provides a good benchmark to confirm that the model can indeed be used as a replacement for OPAL, and to measure the speedup that is gained with the surrogate model.

### 3.5.1 Problem description

The ANL experiment imposes the following requirements. For one, the beam needs to be converging. Mathematically, this means that the correlation between the transversal beam size and the conjugate momentum has to be negative. Further, the beam may not have a sharp waist because that would destroy the initial energy modulation.

Among all design variable configurations that fulfil the constraints above, we are interested in those with minimum beam size, emittance and energy spread right in front of the undulator, i. e. at 26 m.

Taking into account all the considerations above and the fact that only round beams are considered, one arrives at the following constrained multi-objective optimisation problem:

$$
\begin{aligned}
&\min && \sigma_x(26\text{ m}) && \\
&\min && \epsilon_x(26\text{ m}) && \\
&\min && \Delta E(26\text{ m}) && \\
&\text{s. t.} && \sigma_x(s) \geq 2\text{ mm}, && s \in 0, 0.25, 0.5, ..., 10\text{ m} \\
& && \sigma_x(s) \leq 5\text{ mm}, && s \in 0, 0.25, 0.5, ..., 10\text{ m} \\
& && \text{Corr}(x, p_x) \leq 0, && \text{at } 26\text{ m}
\end{aligned}
$$

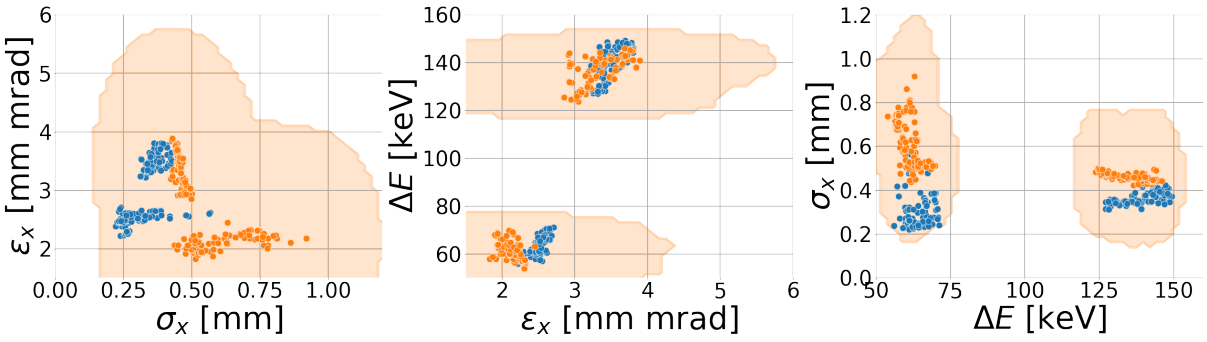
### 3.5.2 Methods and Results

OPAL contains an optimiser that can solve multi-objective optimisation problems with constraints like the one described in the previous section. Unfortunately, it contains bugs in the constraint handling code and cannot be used. Instead, the Python package pymoo [23] is used to solve the optimisation problem. It can use both the surrogate models and OPAL to evaluate individuals thanks to the runOPAL library [24]. This ensures that the model is the only thing that influences the optimisation result.

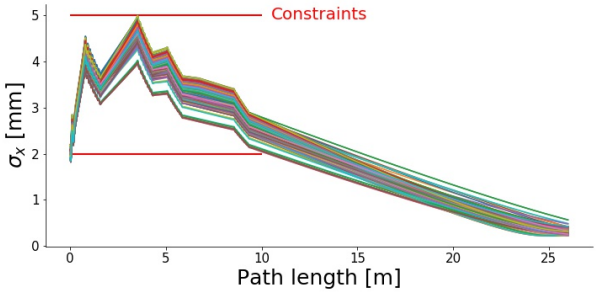
We use the NSGA-II algorithm [25] to solve the optimisation problem. We run it for 100 generations, each containing 200 individuals. According to optimisations with the surrogate model, the constraints are only fulfilled after generation 39 (7800 individual evaluations). Optimisations with fewer individuals do not find feasible solutions at all. Therefore, the computational effort cannot be reduced by using fewer evaluations.

While the optimisation using the best surrogate model has converged after 261 s, the one using OPAL does not finish at all. The reason is that we had to implement the evaluation of each individual as a single SLURM job on the Merlin6 cluster. That means the algorithm has to wait for the completion of all 200 jobs until it can spawn the jobs for the next generation. Because there are many other jobs on the cluster, the jobs are not started immediately. Further, not all jobs in a generation are executed in parallel because of resource limitations. As a consequence, running 10 generations of the algorithm took 22 h. Assuming that the other generations take the same time, we conclude that an optimisation using OPAL would take more than nine days. This is prohibitively long because the maximum job duration on Merlin6 is seven days. Therefore, OPAL cannot be used to solve the optimisation problem with the available resources.

The non-dominated set for the optimisation with the surrogate model is shown in Fig. 3.12a. The 200 non-dominated design variable configurations are evaluated with the surrogate model (orange dots) and OPAL (blue dots). The surrogate predictions and the values calculated by OPAL agree very. The main pattern seems to be captured: The non-dominated set forms two disconnected clusters in objective space, which is captured for the emittances and the energy spread even when the prediction error at 70% confidence is taken into account. The beam size $\sigma_x$ is overestimated by the surrogate, while the energy spread $\Delta E$ prediction shows almost no deviation from the true values. This can be seen from the plot to the right. The plot to the left shows that the emittances are slightly underestimated by the surrogate.




(a) Non-dominated set obtained with the surrogate model. The plots show the pairwise relations between the objectives. The orange dots represent surrogate model predictions, the blue ones OPAL evaluations of the non-dominated design variable configurations. The orange shaded region indicates the prediction uncertainty at 70% confidence as defined by Equ. 2.3.




(b) Beamsizes of the non-dominated beams. The beams are evaluated with OPAL.

Figure 3.12: Results of the optimisation with the surrogate model.

Compared to the values of the quantities of interest (see Tab. 2.3), the deviations are negligible. Even though it is not possible to get a non-dominated set by using only OPAL, it is clear that the configurations cannot be improved very much: The beam size is very close to zero, and so are the emittances. The values of the energy spread are less close to zero, but the prediction error at 70% confidence is close to the predicted values.

As can be seen in Fig. 3.12b, the constraints are mostly fulfilled. Only at the very beginning of the accelerator, there is a minor violation of the minimum beam size constraint. The magnitude of the violation is so small that it bears no physical consequences, and does not worsen the result of the optimisation at all.

### 3.5.3 Speedup

There are two quantities that compare the speed of the optimisation with OPAL to the one with the surrogate model. The first one is the computational cost, and the second one the time to solution. The latter is calculated under the assumption that infinite computational resources are available.

The computational cost for OPAL is calculated easily. Each simulation takes an average time of approximately 20 min to be calculated. We need 100 generations and 200 individuals in each, resulting in 20′000 runs. Each run utilises 4 CPU cores. In total, the computational cost is:

$$c_{\text{OPAL}} = 20000 \cdot 4 = 80'000 \text{ CPU h.}$$

The surrogate model is trained on 20′000 and evaluated on 1′000 OPAL runs. Further, the model training takes 51 h on 12 CPU cores. The manual exploration phase that resulted in the baseline model took about as many tries as the hyperparameter scan, resulting in a total of 100 trainings. Not all of them trained for the same number of epochs or on the same dataset. For the sake of simplicity, we assign the same training time to all models. The total computational cost for the development of the surrogate is given by:

$$c_{\text{surr}} = 20000 \cdot 4 + 100 \cdot 51 \cdot 12 = 141'200 \text{ CPU h.}$$

The optimisation itself takes only 5 min on 12 cores. Because this is negligible, the equation above represents the total computational cost of the optimisation.

| Number of optimisations | 1 | 2 | 3 | Asymtotic $n$ |
|---|---|---|---|---|
| Cost saving $\frac{c_{\text{surr}}}{c_{\text{OPAL}}}$ | 1.765 | 0.9 | 0.59 | $\mathcal{O}(n^{-1})$ |
| Speedup $\frac{t_{\text{surr}}}{t_{\text{OPAL}}}$ | 0.6 | 1.3 | 1.9 | $\mathcal{O}(n)$ |

Table 3.3: Gain in computational cost and time to solution.

The theoretical time to solution for OPAL can be calculated as follows. The assumption of infinite resources allows to evaluate each individual in parallel, and that subsequent generations start running immediately. Further, assume that each individual has the same runtime of 20 min. Then the time to solution for OPAL is given by:

$$t_{\text{OPAL}} = 100 \cdot 20 \text{ min} \approx 33 \text{ h}.$$

For the surrogate model, the creation of the dataset takes only 20 min because the random sample is embarrassingly parallel and according to our assumptions, infinite resources are at our disposal. Further assume that all model trainings can be performed in parallel. The best model is selected at the end. Then the training phase takes 51 h. Compared to this value, the duration for the optimisation itself (261 s) and the creation of the dataset (20 min) are negligible. The time to solution for the surrogate model is therefore:

$$t_{\text{surr}} = 51 \text{ h}.$$

Summarising the results above, one can say that the surrogate model takes more than 50% more computational resources and time to reach the solution of the optimisation. So why does one need a surrogate model?

Remember that the numbers above describe the cost and time to solution for a single optimisation. In applications, however, many optimisation runs need to be performed. For one, the optimisation algorithm needs tuning. A priori, it is not known how many individuals and generations are needed, or which crossover and mutation operators lead to the best results. Perhaps the exact constraints are not known either and have to be found in an iterative way.

All this can be done easily with a surrogate model, but is prohibitively expensive with OPAL. Table 3.3 contains the improvements that can be achieved by using a surrogate model for different numbers of optimisation runs. If more than one optimisation is needed, the surrogate model is both faster and needs less resources. The gain scales linearly with the number of optimisations[1].

It is important to keep in mind that under real-world conditions, the surrogate model is even more beneficial. The random sample can adapt well to the available computational resources, and the time needed for training and optimisation is hardly affected by limited resources. OPAL on the other hand cannot adapt to limited resources because the queuing system delays the execution of each generation up to the point where solving the optimisation problem is not practically feasible any more.

Further, once the surrogate model is trained, it can substitute any valid OPAL run for the accelerator it has been trained on. The only assumption here is that we are exclusively interested in statistical quantities, but not the phase spaces. FInally, manual real-time interaction with the beam is only possible with a surrogate model, but not with OPAL.

[1] As long as the time to run the optimisations with the surrogate model is negligible compared to the 51 h needed for training. Since an optimisation with the surrogate takes less than 5 min, we are usually in the regime where the assumption of negligible optimisation cost holds.

# Chapter 4

# Results for the Invertible Model

## 4.1 Baseline Model

To the best of our knowledge, nobody has tried to train an invertible neural network on a particle accelerator dataset. Therefore extensive manual trial and error was required to find a good configuration of hyperparameters. The resulting model is described in this section. It serves as a baseline model to which the other models from the hyperparameter scan are compared.

### 4.1.1 Training

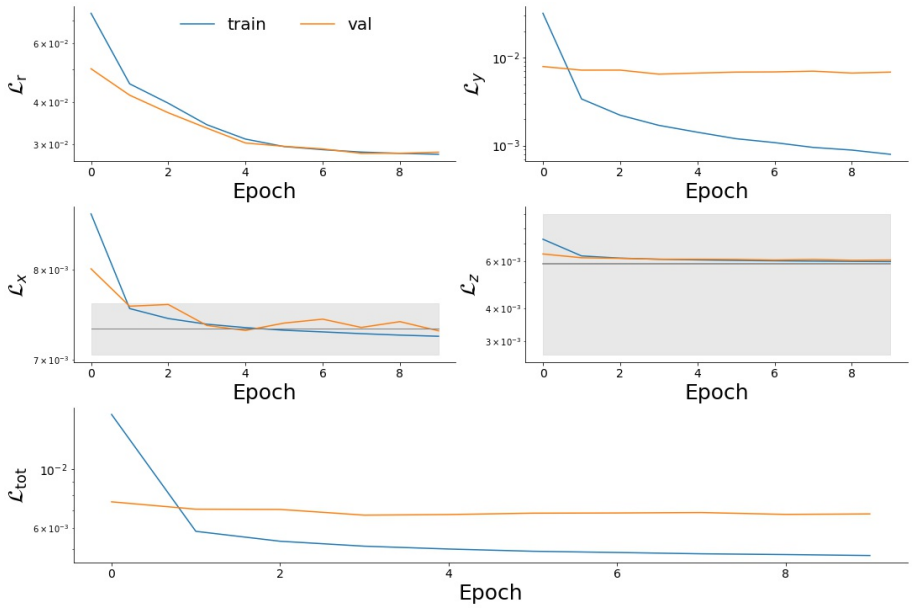



Figure 4.1: Losses of the baseline invertible model during training.

As can be seen in Fig. 4.1, the losses do not need many epochs to converge. There is no significant decrease in the losses after the first three epochs, especially for the validation loss. The grey areas in the $\mathcal{L}_x$ and $\mathcal{L}_y$ plots indicate the minimum MMD. It is calculated by comparing random batches taken from the same distribution (empirical DVAR distribution for $\mathcal{L}_x$, batches sampled from the latent distribution for $\mathcal{L}_z$). The mean value (dark grey line) plus/minus one standard deviation (shaded area) over the batches is plotted.

Notice that the MMD losses have reached their minimum. They cannot be improved further. The artificial loss $\mathcal{L}_{\text{artificial}}$ is not shown because both the training and the validation value are constant at $5 \cdot 10^{-3}$.

### 4.1.2 Inverse Prediction

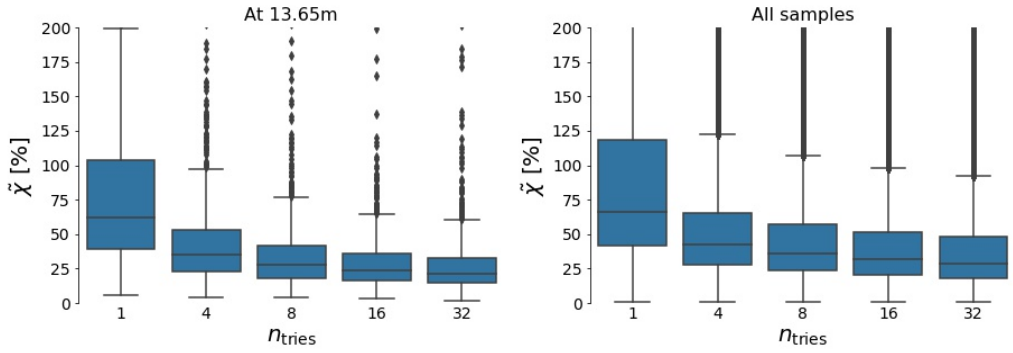



Figure 4.2: Maximum relative error for the inverse prediction.

Since the region just before the drift is of special interest for optimisations, a detailed description of the sampling accuracy is given in Fig. 4.2. Notice that the plot was limited to show the range $[0, 200]$ %. Not all outliers are depicted. The maximum error decreases if more samples are used. The error distribution seems to stabilise at $n_{\text{tries}} = 32$. Further, the median of the overall error is not significantly bigger than the one at the end of the region of interest, but the upper quartile is.

This observation is partially confirmed by Fig. 4.3. The sampling error is biggest in the regions around the solenoids. The percentiles are roughly at the same levels in the regions between the solenoids, but the presence of the solenoids seems to influence regions nearby. This explains why the error decreases along the first cavity, increases along the second one and decreases again in the third one. Notice that the model was not trained on samples inside the solenoids and cavities. As can be seen at the last cavity, this has no major impact on the median of the error.

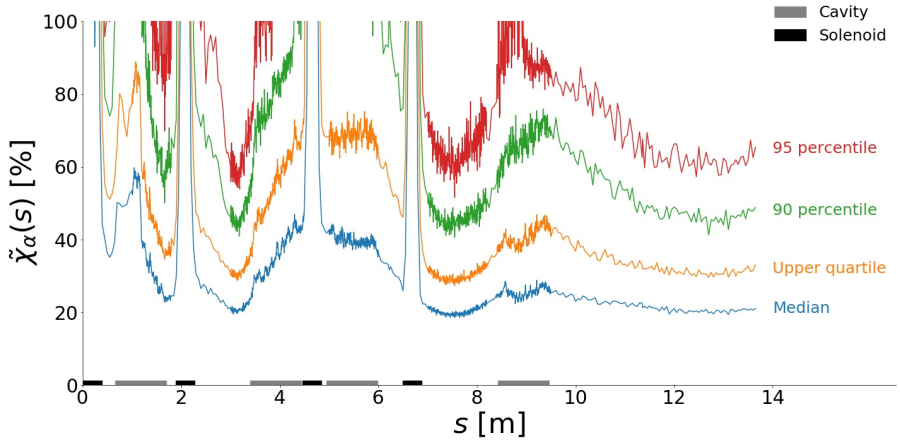



Figure 4.3: Maximum percentage error along the machine for $n_{\text{tries}} = 32$.

The maximum error is an important quantity to give an upper bound of the model performance. However, not all quantities of interest can be obtained to the same accuracy (see Fig. 4.4).

Notice that the percentiles for the maximum error take bigger values than the ones for the separate quantities of interest. This means that the model has problems to predict configurations that match all desired QOI values. However, the distribution of the sampled QOIs matches the one in the test set very well, see Fig. 4.5. The sampled design variables are also uniformly distributed in their space (Fig. 4.6). Deviations from the uniform marginal distributions mainly originate from the best-of-n sampling strategy.

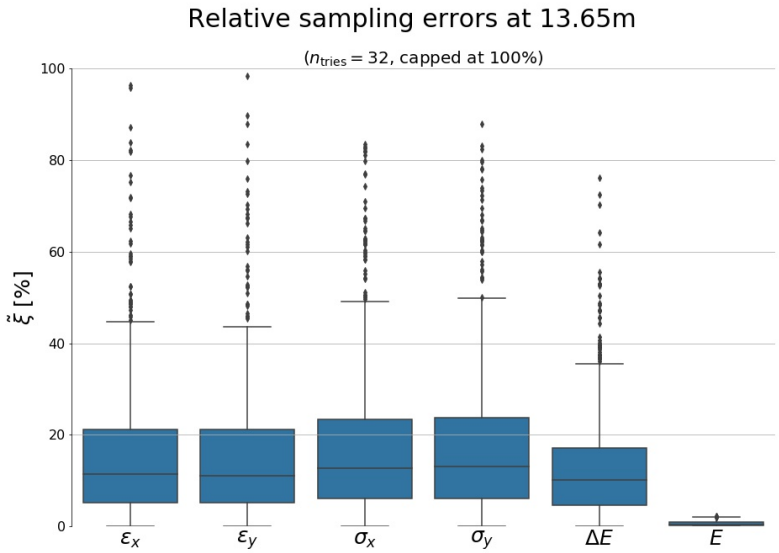



Figure 4.4: Error for the inverse prediction per QOI for $n_{\text{tries}} = 32$.

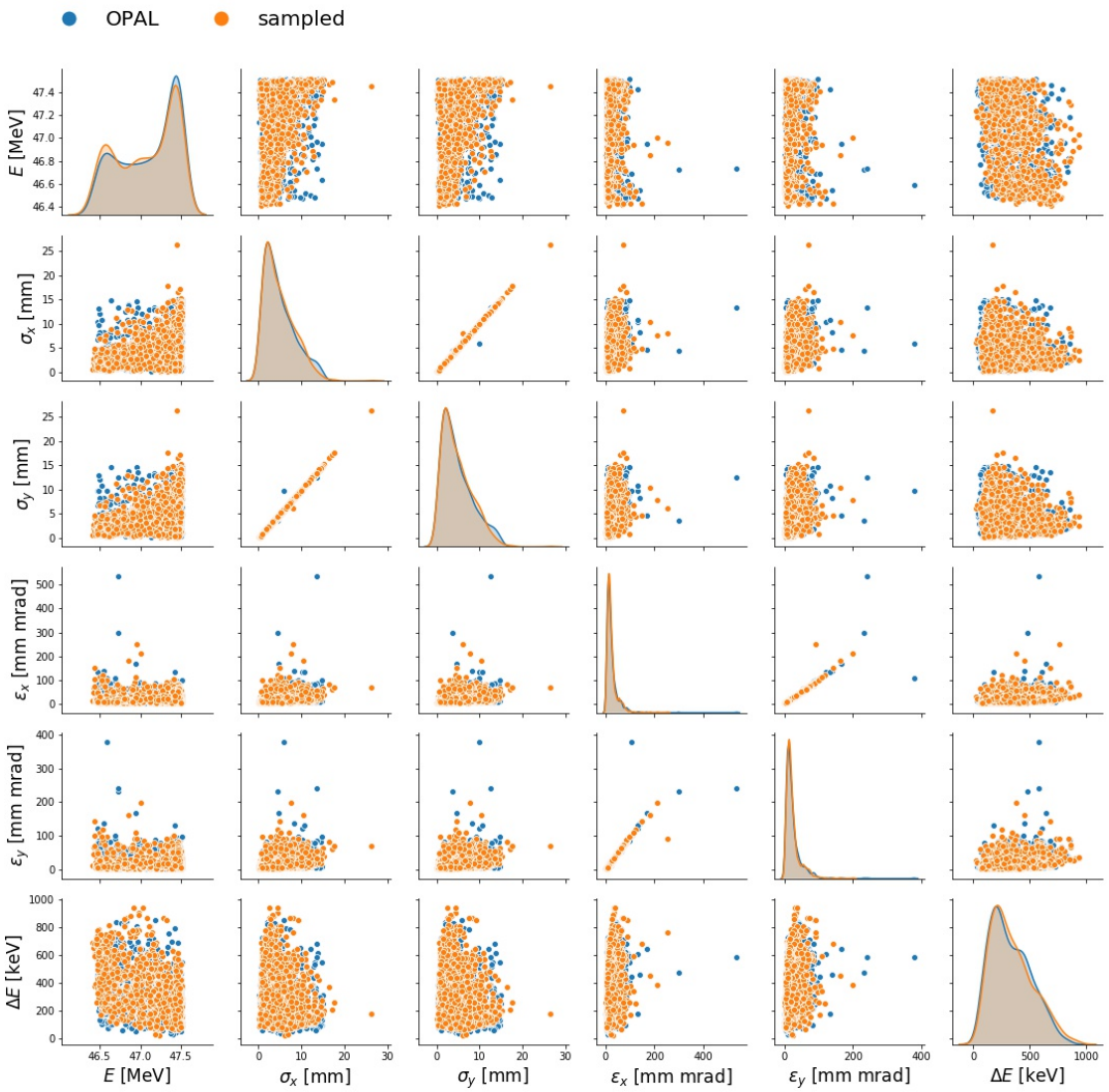



Figure 4.5: The distributions and pairwise relationships between the quantities of interest. Blue dots are samples from the test set, orange dots are sampled with the baseline model and evaluated with the best forward model. Only samples at the end of the cavities (13.65 m) are depicted.

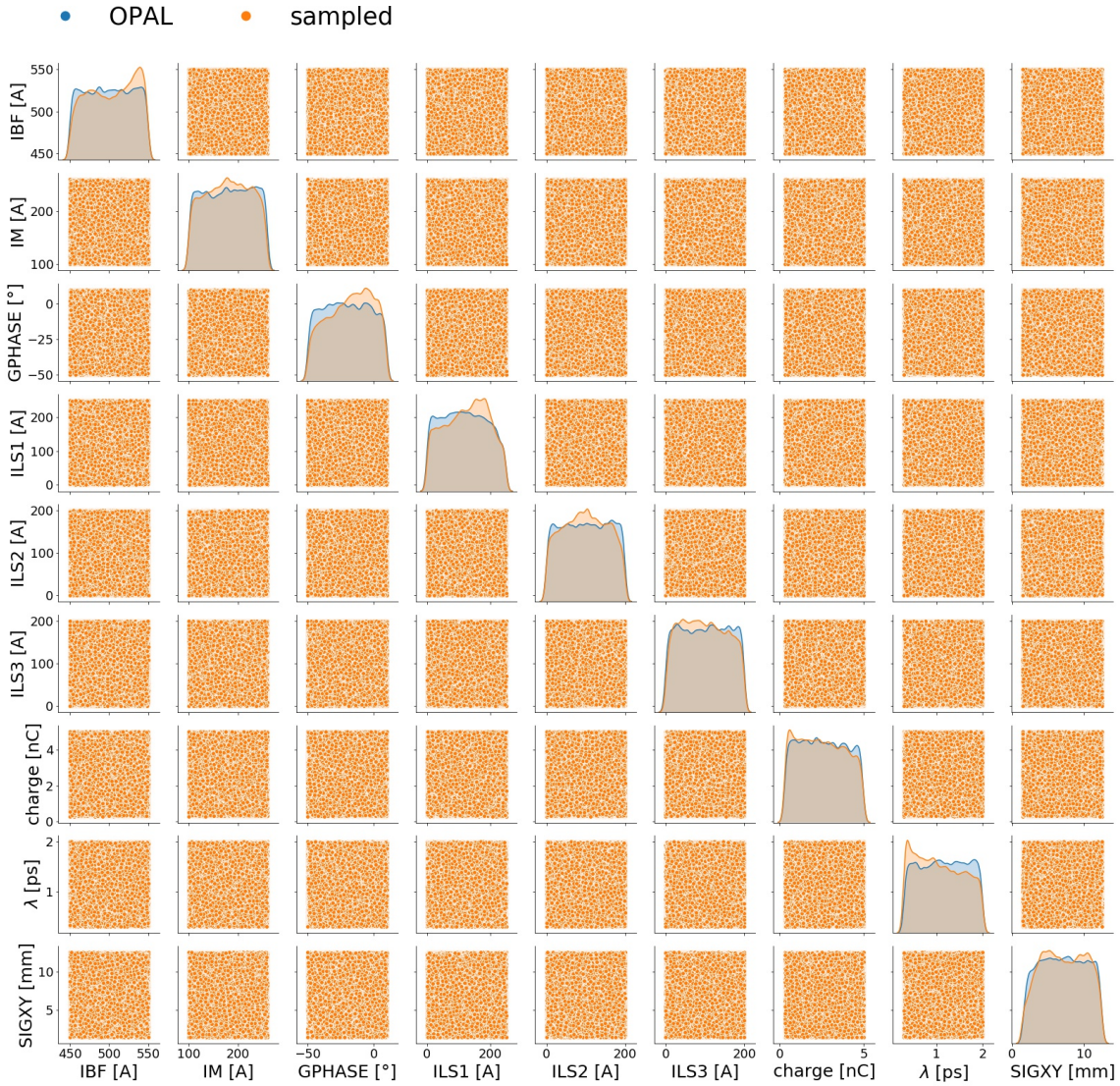



Figure 4.6: The distributions and pairwise relationships between the design variables. Blue dots are samples from the test set, orange dots are sampled with the baseline model. The diagonals depict the marginalised distributions. The plot shows a random selection of 300′000 samples.

## 4.2 Hyperparameter Scan

| Hyperparameter | Values |
|---|---|
| $n_b$ | 4, 6, 8, 10, 12 |
| $n_d$ | 3, 5, 7 |
| $n_w$ | 60, 80, 100, 120, 140 |

Table 4.1: Parameters of the hyperparameter scan for the invertible model.

The results from the baseline model suggest that the dimensions of the latent space and the padding are sufficient to sample from the distribution of the dataset. Therefore, these parameters are not varied during the hyperparameter scan. The full list of parameters that are visited by the grid search algorithm can be found in Tab. 4.1.

The result is depicted in Fig. 4.7. Each model is a dot. The "best model" is the one with the smallest values for $\tilde{\chi}_{90\%}$, $\tilde{\chi}_{95\%}$ and $\tilde{\chi}_{99\%}$. There is a model that has slightly ($< 1\%$) better values for $\tilde{\chi}_{50\%}$ and $\tilde{\chi}_{75\%}$, but much worse higher quantiles. Therefore, it is not considered the best model.

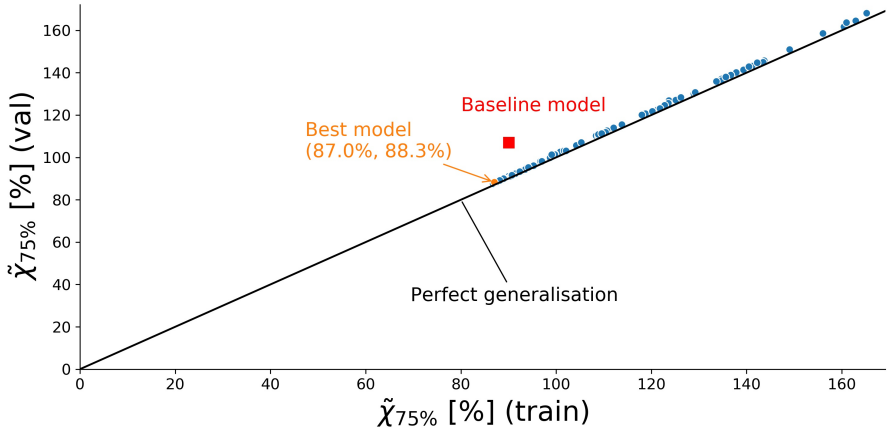



Figure 4.7: Results of the hyperparameter scan for the invertible surrogate.

Notice that all models generalise almost perfectly. The only exception is the baseline model. One can conclude that the quality of the inverse predictions is mainly limited by the capacity of the model, not by the optimisation of the loss function.

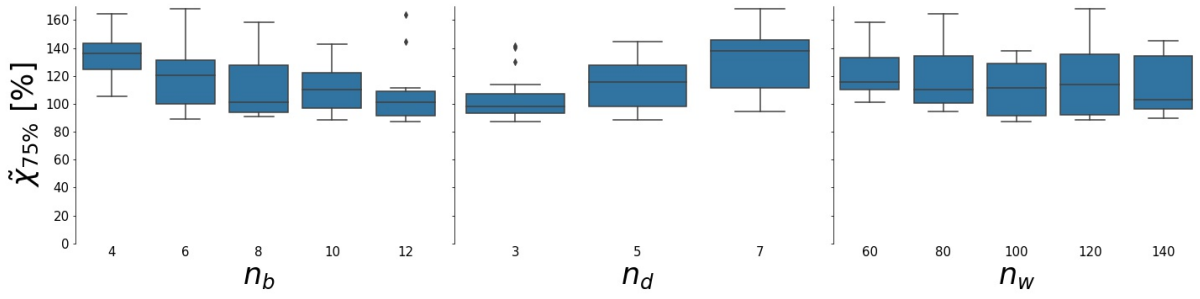



Figure 4.8: Influence of the hyperparameters on sampling performance. Left to right: Number of affine coupling blocks, depth and width of the coefficient networks.

The dependence of the model performance is shown in Fig. 4.8. The boxplots represent the marginalised distributions. One can see that more affine coupling blocks tend to return better models, while adding more layers to the coefficient subnetworks decreases the quality of the sampling. No clear trend is visible for the width of the coefficient networks.

Despite the trend, all values of $n_b, n_d$ and $n_w$ have at least one model with a performance that is comparable to the one of the best model: The lower whiskers are more or less on the same level for all

parameter values. There is almost no improvement for the best model. We assume that the invertible model would benefit from using more training data (see also Sec. 4.4).

## 4.3 Best Model

### 4.3.1 Training

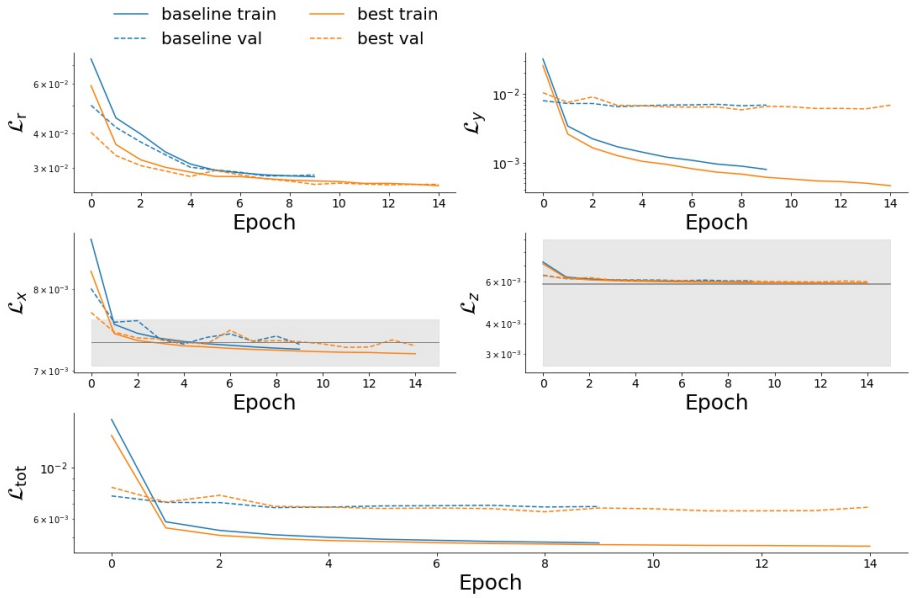



Figure 4.9: Training curves for the best and the baseline invertible model.

The losses of the best model are depicted in Fig. 4.9. The grey line and area denote the minimum MMD. It is calculated by comparing batches taken from the true distribution. The grey line denotes the mean value over the batches, and the shaded are is the mean plus/minus one standard deviation. For the $\mathcal{L}_x$ plot, the true distribution is the one in the training set. For $\mathcal{L}_z$, the batches are sampled from the uniform distribution, i. e. the distribution of the latent space. The number of randomly selected batches is the same as is used for one epoch during training.

The MMD losses of both the baseline model and the best model have fully converged. In other words, the inverse prediction samples from the correct distributions. The limiting factors of the sampling error are therefore the reconstruction loss and $\mathcal{L}_y$.

The curves for the baseline model are added for comparison. It is not clear why the baseline model has a bigger generalisation error than the models from the hyperparameter scan. Most of the losses have the same values up to noise for both models. The only difference is the reconstruction loss. This is a hint that $\mathcal{L}_r$ is the limiting part of the loss function. Another interesting observation is that the validation loss for the forward prediction $\mathcal{L}_y$ is almost constant during training, while the training loss is converging. Nevertheless, the sampling error generalises almost perfectly.

### 4.3.2 Forward Prediction

The distribution of the maximum percentage error is show in Fig. 4.10, and its dependence on the position in Fig. 4.11. The error explodes at the positions of the solenoids. At the cavity locations, the error also increases, even though not as dramatically. Notice that the oscillations within the cavities are visible in the figure.

Even the best invertible model does not reach the performance of the forward surrogate model, especially if the samples corresponding to solenoids and cavities are taken into account. This makes sense because the invertible model was not trained for these regions. In the regions of interest, the 90 percentile of the maximum error lies roughly at 30%.

The errors for the different QOIs are shown in Fig. 4.12. Notice that the errors for the emittances and the beamsizes are comparable, while the ones for the energy and the energy spread are much smaller. One can conclude that the maximum percentage error for the forward prediction is mainly determined by the emittances and the beam sizes.

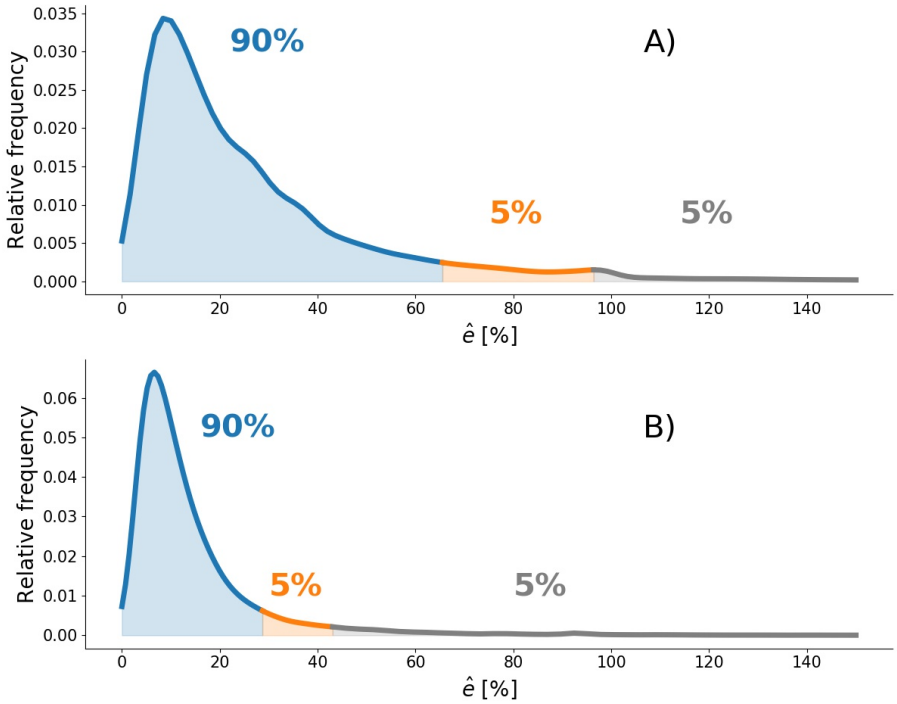



Figure 4.10: Kernel density estimate of the maximum forward prediction error distribution for the best invertible model. Errors greater than 150% are not shown for the sake of readability. A) The density estimate is calculated taking into account all samples in the test set. B) Samples corresponding to positions within cavities or linac solenoids are excluded from the density estimate.

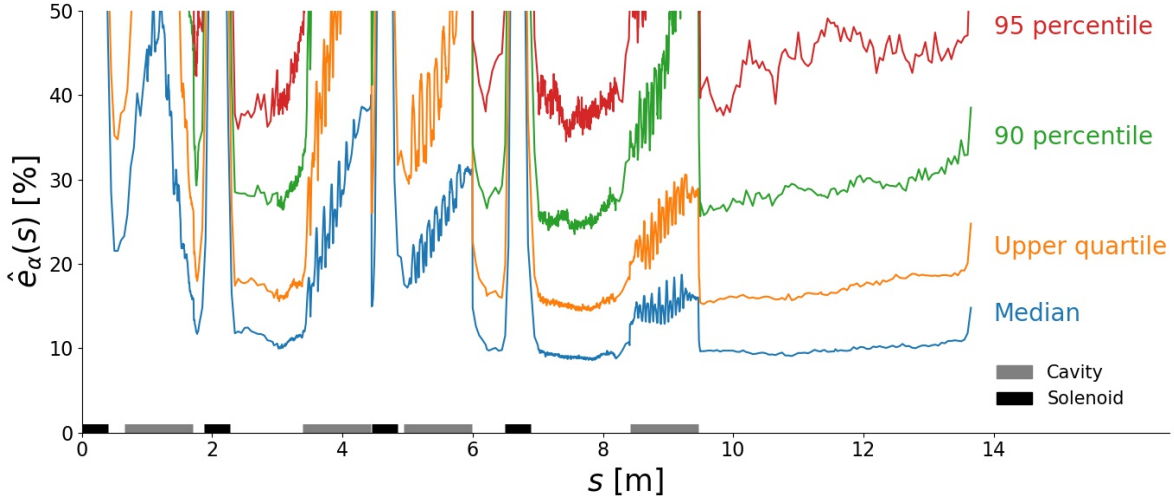



Figure 4.11: Maximum prediction error (see Equ. 2.4) of the forward prediction of the best invertible model.

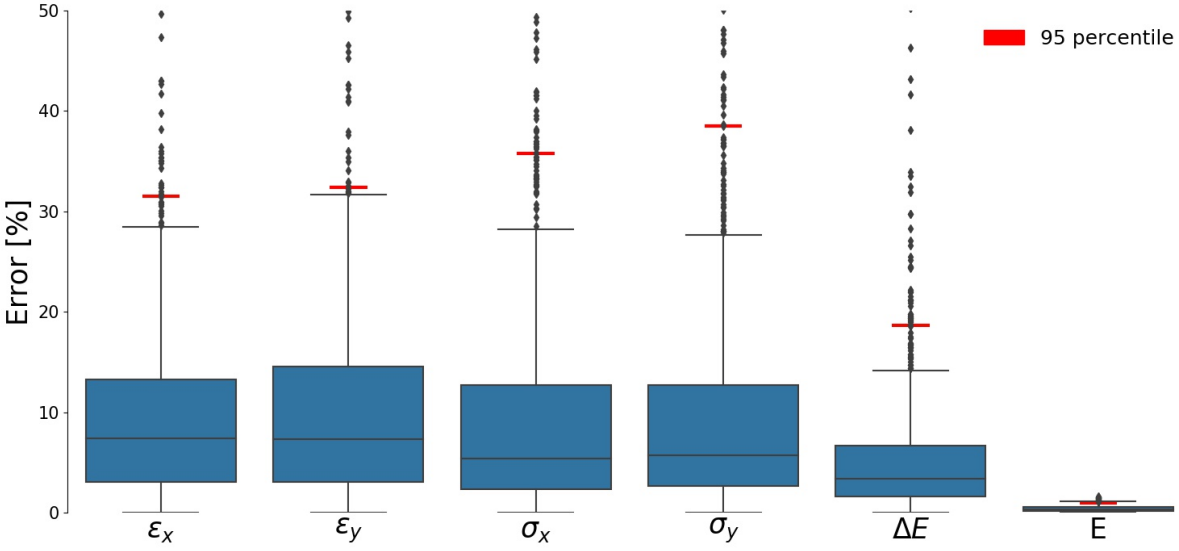


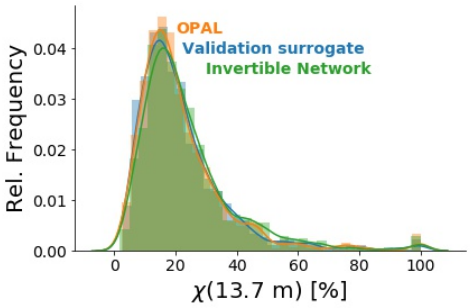

Figure 4.12: Error (see Equ. 2.1) of the forward prediction at 13.65m. Outliers with values bigger than 50% are omitted.

### 4.3.3 Inverse Prediction

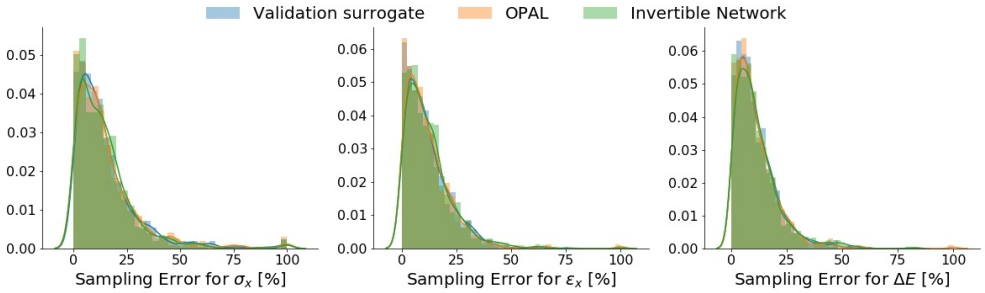
In the entire section, $n_{\text{tries}} = 32$ is assumed unless explicitly stated.

#### Error estimate with validation model vs. OPAL

The performance of an invertible model is assessed by predicting the effect of a sampled DVAR config with the best forward model, and comparing the resulting beam to the desired one at the desired position. First, it has to be shown that the validation model is indeed capable of representing the sampling error accurately. The plots in Fig. 4.13 show the distributions of the maximum percentage error and the percentage error for several quantities of interest separately.




(a) Distribution of the maximum sampling error (see Equ. 2.5).




(b) Sampling error distribution for individual QOIs.

Figure 4.13: Sampling error estimated with different models. The best invertible model is used to sample one configuration for each QOI vector at 13.65 m in the test set. Then the invertible model itself (green), the validation model (blue) and OPAL (orange) evaluate the DVAR predictions. The resulting QOI vector is compared to the target vector to obtain the errors. The plots show the distributions of these errors.

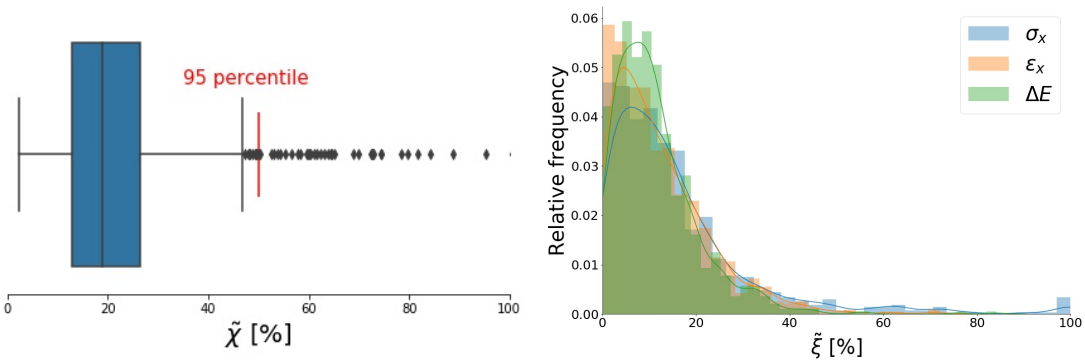
To create the plots, the following steps are taken. First, the best invertible model samples a design variable configuration for each sample at 13.65 m in the test set. Second, OPAL is run to evaluate these quantities of interest and obtain the error. Additionally to running OPAL, the invertible model and the best forward model are used to evaluate the sampled DVAR configs.

In principle, such plots could be made including all samples in the test set instead of just those at 13.65 m. Unfortunately, this is infeasible because OPAL runs are too expensive in terms of CPU hours. The plots above required 913 OPAL runs. They took about 2.75 h on 132 CPU cores on Merlin6. Evaluating the sampling error on the entire test set with more than $10^6$ samples is not feasible.

For these reasons and because we have no other choice, we assume that the validation model can also be used at other locations to estimate the sampling error. Nevertheless, one should keep in mind that this is only an assumption.

**Error estimated with validation model**

1. *Error at 13.65m (end of region of interest)*




(a) Maximum sampling error. Outliers bigger than 100% error are omitted.

(b) Sampling error of the most important QOIs. The values are clipped to $[0, 100]$%.

Figure 4.14: Sampling error at 13.65 m, i. e. after the cavities.

The maximum percentage error of the sampling at the end of the region of interest is shown in Fig. 4.14a. The entire distribution of the error at 13.65 m is depicted next to it in Fig. 4.14b. The errors along the $y$-axis are not shown because the beams are symmetric up to noise, and the bunch energy is more or less constant, which leads to very small errors.

The plots show the error distribution over the DVAR configurations in the test set. In the image to the right, it can be seen that the tails of the errors start roughly at 30%, and that the error for the bulk of the DVAR configurations is located around $10 - 20$%.

The 95 percentile of the error is at 50%, which is a high value. This implies that for 95% of the DVAR configurations in the test set, it is possible to sample beams where the biggest error is smaller than 50%. For single QOIs, the situation is better. In other words, it is possible to sample machines that agree well with the imposed QOIs, but it is not always the case that all QOIs are hit equally well.

The sampled design variable distributions and their pairwise relations are depicted in Fig. 4.15. The distribution of the sampled quantities of interest can be found in Fig. 4.16. It agrees very well with the one obtained with OPAL.

2. *Dependence on position*
   The maximum sampling error depends on the position along the machine. This relationship is depicted in Fig. 4.17. The median and upper quartile of the error are about at the same level in the spaces between the solenoids and cavities, while the 90 and 95 percentiles of the error decay only slowly after the last cavity.

   Altogether, it can be concluded that the error of the sampling is about at the same level along the beam line, as long as one takes care not to sample within a solenoid or cavity.

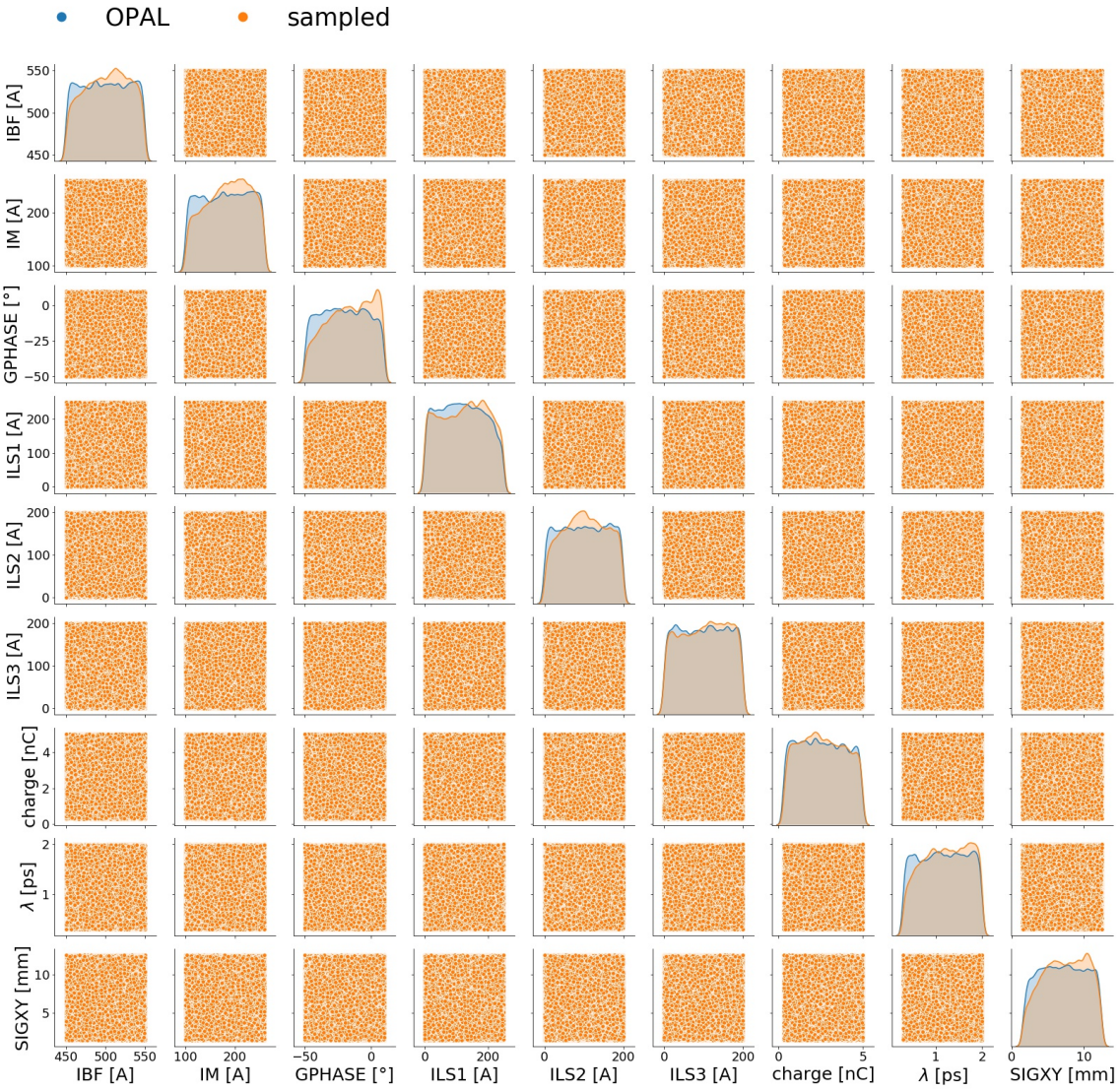



Figure 4.15: Distribution of the sampled design variables at all positions.

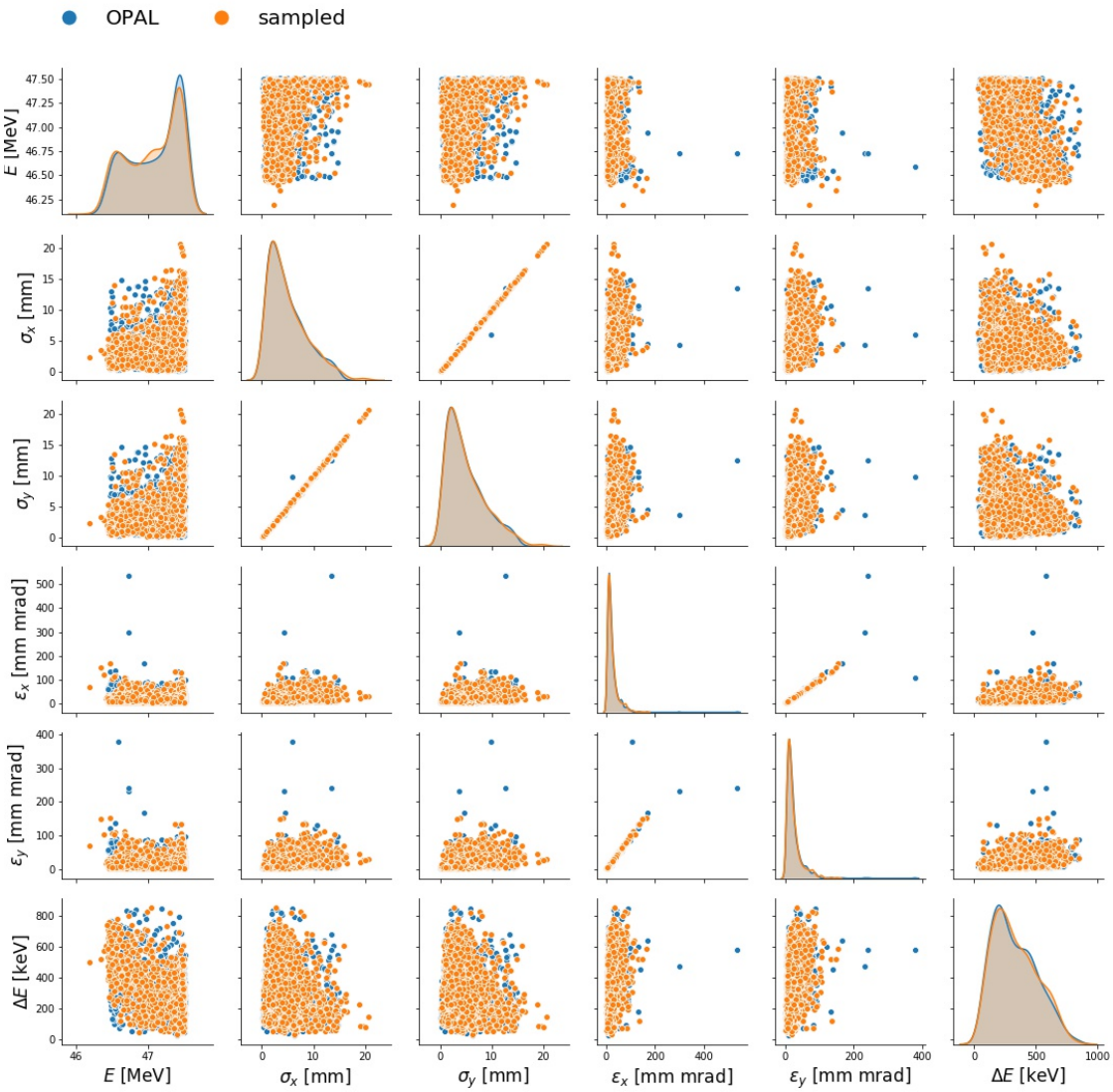



Figure 4.16: Distribution of the sampled quantities of interest after the cavities.

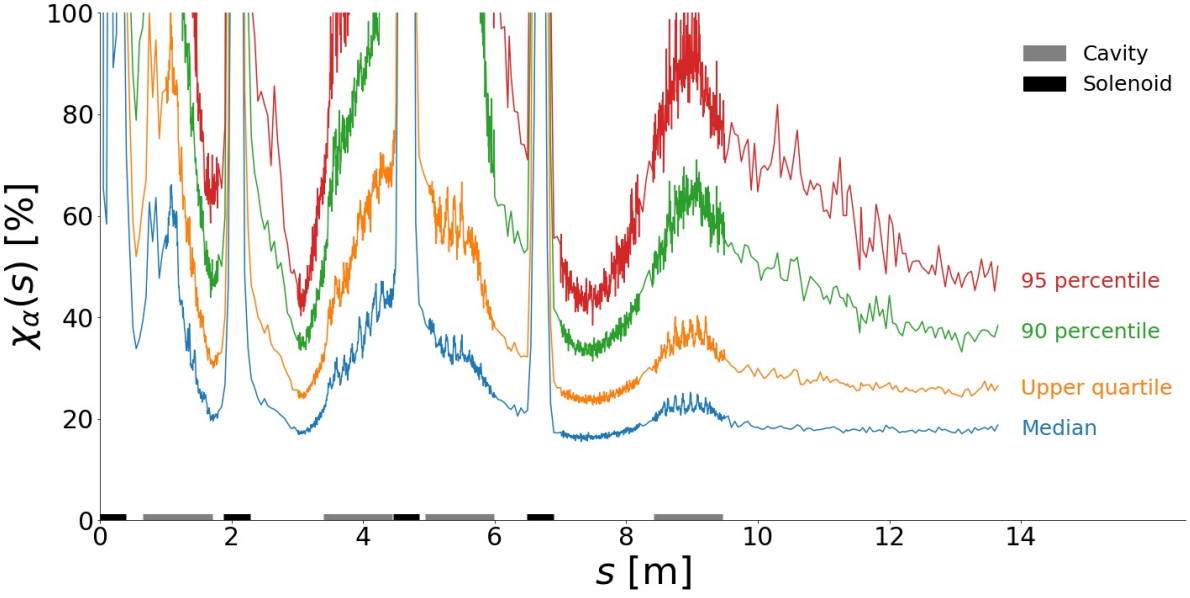
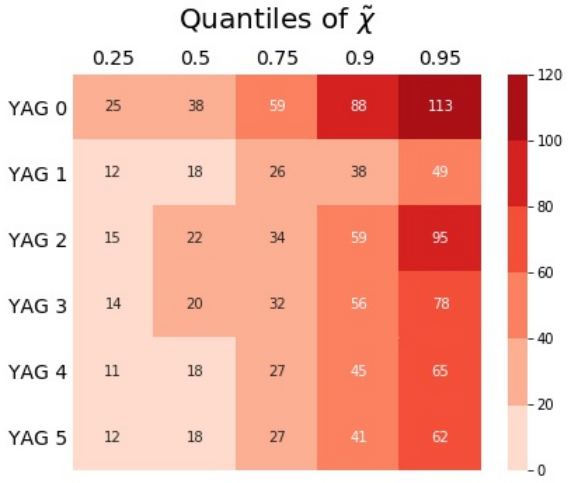
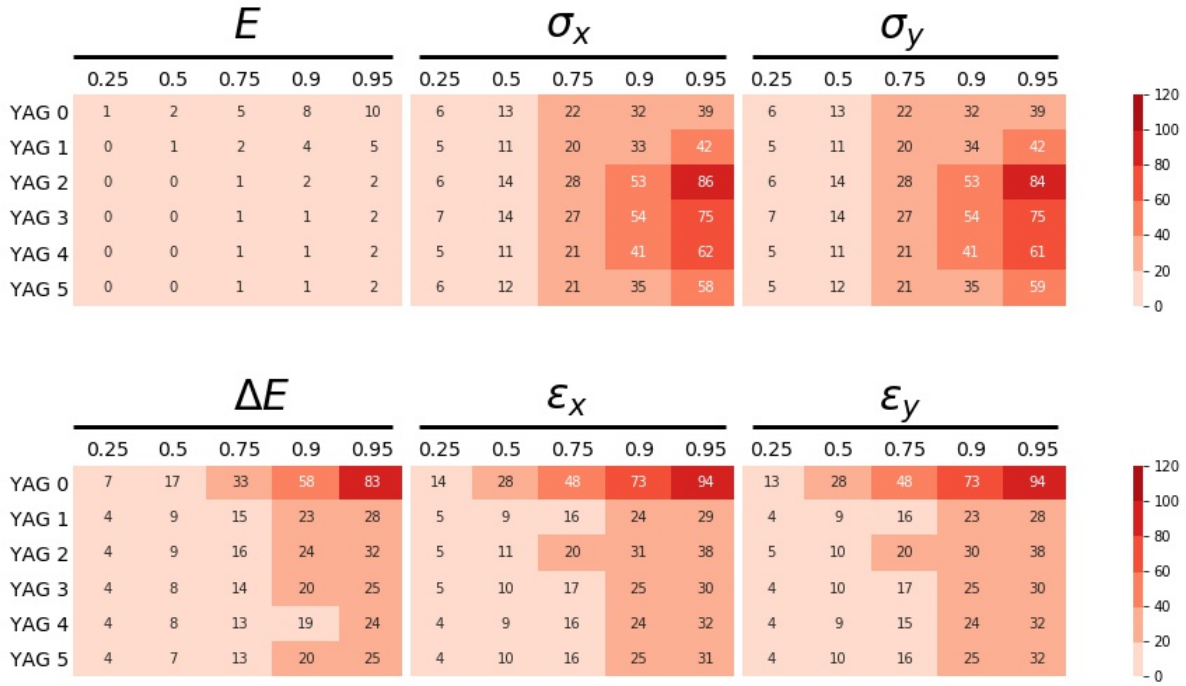



Figure 4.17: Dependence of the sampling error on the position.

3. *Sampling error at YAG screens*
   The sampling error at the positions of the YAG screens are of special interest because lab mea-




(a) Maximum sampling error at the YAG screens.




(b) Sampling error of each QOI at the YAG screens.

Figure 4.18: Sampling error at the YAG screens. The error is estimated using the validation model.

surements are possible only there. A table depicting $\chi_\alpha$ can be found in Fig. 4.18a. It shows the quartiles of the error (horizontal direction) at different YAG screens (vertical direction). The colors visualise the magnitude of the error. Darker means bigger error.

The error at the first screen is about twice as big as the one for the other screens. Apart from that, the performance at the other YAG screens is comparable. At YAG 2 and 3, the error is slightly higher.

The median error is about 20%, and the upper quartile error is around 30%. That means the model fulfils its purpose, though the sampling error could be smaller. At 95% confidence level, the maximum error is too big.

Instead of considering the maximum error of an inverse prediction, one can also examine the error separately for each quantity of interest. This is done in Fig. 4.18b, Fig. 4.18b.

Both quartiles and the median of the individual sampling errors are smaller than the maximum error. The emittances and the energy spread are predicted with less error than the beam sizes, except at YAG 0. This implies that the maximum percentage error is mostly determined by the beam sizes. The 95 percentiles of the $\sigma_x, \sigma_y$ errors are very close to the one of the maximum error. As already said, the only exception is YAG 0, where the emittances are the performance bottleneck.

For the emittances, the 95 percentile of the error is around 30% at the YAG screens. This is a good result.

It should also be noted that the energy has an error of 5% or less for 95 percent of the QOI configurations at screens in the test set. The only exception is the first screen, but also there the error of the energy is negligible.

## 4.4 Influence of Training Set Size

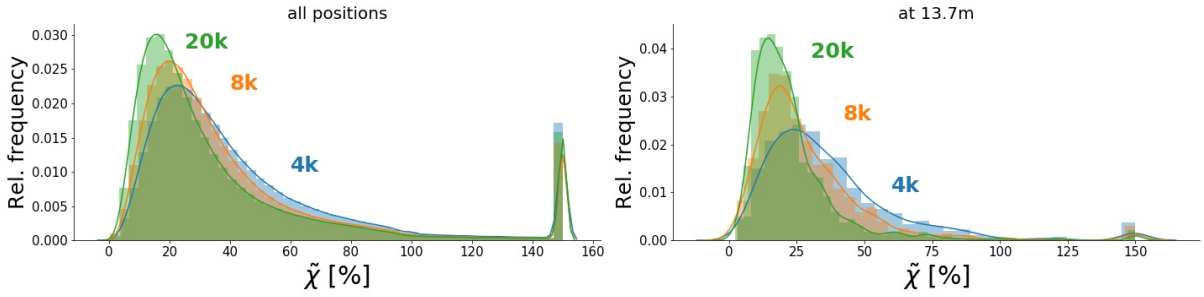



Figure 4.19: Distribution of the maximum sampling error. The plot to the left shows the distribution over all samples in the test set, the right one only at the end of the region of interest. The distribution labels, e. g. "4k", refer to the number of OPAL runs in the training/validation set. The error is clipped to $[0, 150]$%. This is the origin of the peaks at 150%.

The sampling error decreases when more samples are added, as can be seen in Fig. 4.19 and Tab. 4.2. The error quantiles generally decrease by a few percentage points. It is not clear if this trend would continue beyond a training set consisting of more than 20′000 samples. Nevertheless, there is a hint that future work might benefit from using a bigger training set.

The peaks at 150% appear because the errors are clipped, i. e. all values bigger than 150% are considered to be 150%. This means that the tails of the error distribution are extensive if all samples are taken into account. However, they disappear almost entirely if only the samples at 13.65 m are considered. A plausible explanation is that the tails mainly originate from the samples at the solenoid and cavity positions where the model is not trained.

| | 0.25 | 0.50 | 0.75 | 0.90 | 0.95 |
|---|---|---|---|---|---|
| 4k | 22 | 33 | 54 | 88 | 150 |
| 8k | 19 | 29 | 47 | 79 | 127 |
| 20k | 15 | 25 | 42 | 77 | 139 |

Table 4.2: Quantiles of the maximum sampling error (horizontal) over the test set for different training sets (vertical).

# Chapter 5

# Discussion

The results from chapter 4 show that it is possible to use a neural network as a fast and accurate surrogate model and a replacement for OPAL. Based on the empirical distribution on the test set, the model error at 95% confidence is around 30% at all positions of interest. The error of single quantities of interest at the positions of the YAG screens is at 20%. The emittances are shown to have the biggest prediction error, followed by the correlations. The beam sizes are predicted better. Their prediction error is around 10%, i. e. only half as big as the one for the emittances. The energy spread has only a prediction error around 5%, and the energy is predicted almost perfectly with 0.1% error. Even very fine patterns are resolved by the model, e. g. the oscillations of the correlations and the beam sizes inside the cavities, as can be seen in the dashboards (Fig. D.3).

There are hints that the prediction error can still be decreased by a few percentage points by adding more samples to the training set. How much there is to be gained by doing so is an open question.

The forward surrogate model is successfully applied to solve an optimisation problem. It is able to solve the problem in less than five minutes on 12 CPU cores on Merlin6. The same problem cannot be solved using OPAL. This shows that fast machine learning models are enablers for problems that are too expensive to solve with existing software.

A big part of the problem is that OPAL does not provide a working solver for constrained multi-objective optimisation problems. But even if this functionality is added in the future, the surrogate model is more efficient as long as more than a single optimisation is needed. Further, the use of a surrogate model allows to quickly optimise for different objectives and try multiple sets of constraints. Once a good surrogate model is trained and validated, this empirical workflow represents a powerful tool.

Thanks to the web interfaces, an operator can even perform a quick optimisation "by hand". This provides valuable insight and intuition about the machine and how the parameters interact with each other, even for positions in the beam where lab measurement is not possible.

The forward model allows to estimate the error of an invertible model precisely enough that there is almost no difference to the error obtained by running OPAL. It can therefore be used as a validation model, which allows to estimate the error of the inverse prediction and therefore validate the invertible model. The latter is able to sample from the correct distributions, both in the space of design variables and in the space of the quantities of interest. The forward prediction is significantly worse than the one of the forward model, but still acceptable. The error at 90% confidence is around 20%.

The sampling error at the YAG screens is biggest for the beam sizes. For these, the model is only usable at 75% confidence, where it has an error between $20 - 30\%$. The emittances and energy spread have this error at 95% confidence. This hints that it might be possible to build an invertible model that samples the beam sizes better.

We observed that the preprocessing of both the design variables and the quantities of interest has a significant influence on the sampling performance. Therefore, a different kind of preprocessing might improve the sampling error for the beam sizes.

Another improvement might be gained by using more training samples. For one, the hyperparameter scan suggests that the models generalise perfectly. There also seems to be a hard error boundary that no parameter configuration can break. The second hint is that lowering the number of training samples decreases performance. It is plausible that increasing the number will result in better sampling performance.

Finally, there is one observation we cannot explain. Both the forward model and the invertible model have problems predicting emittances at the first YAG screen. Further work is required to find out the reason for this.

# Chapter 6

# Conclusion and Outlook

It was shown that accurate and fast data-driven surrogate models can be trained on particle accelerators. They allow to perform optimisations that are not possible with classical computer models.

To the best of our knowledge, invertible neural networks have never been applied to particle accelerators so far. We have shown that this is possible with acceptable error bars. It is not clear yet what applications such models might have. If the sampling quality increases, they could be used as a tool to obtain beams tailored for specific properties.

Another application might be to use an invertible model to initialise an optimisation algorithm. This might not seem very useful at the first glance. A forward surrogate model is needed to develop an invertible one. Why not just use the forward model for the optimisation? There might be time constraints, e. g. for real-time optimisations. Right now, a simulation with 100 generations and 200 individuals in it still takes $\mathcal{O}(5\text{ min})$, which is too much for real-time applications. However, if the invertible model initialises the algorithm well enough, only a couple of generations might be needed. Assume that the evaluation of a single generation with a forward surrogate model takes 3 s, the invertible model needs 5 s for the initialisation, and that afterwards only 10 generations are needed. Then the entire optimisation takes 35 s, which corresponds to a speedup of approximately 8.5.

Future work is also needed to analyse the influence of the various parameters of an invertible neural network. Our hyperparameter scan shows that more affine coupling blocks with more shallow and narrower coefficient networks might improve the network. It was not possible in this thesis to investigate thoroughly due to time constraints.

Both the forward and the invertible models show good generalisation. The invertible models seem even to generalise almost perfectly. This could be taken as a hint that more samples are needed to improve the models. More work is needed to examine this question in more detail.

On the other hand, it is not clear how low the error should be. At some point, the prediction error will be smaller than the error of the underlying numerical model. Where exactly this happens needs to be shown by future work.

# Bibliography


[1] Helmut Wiedemann. *Particle Accelerator Physics*. Springer, 2015.

[2] Andreas Adelmann et al. "OPAL a Versatile Tool for Charged Particle Accelerator Simulations". In: *arXiv e-prints*, arXiv:1905.06654 (May 2019), arXiv:1905.06654. arXiv: 1905.06654 [physics.acc-ph].

[3] Auralee Edelen et al. "Machine learning for orders of magnitude speedup in multiobjective optimization of particle accelerator systems". In: *Phys. Rev. Accel. Beams* 23 (4 Apr. 2020), p. 044601. DOI: 10.1103/PhysRevAccelBeams.23.044601. URL: https://link.aps.org/doi/10.1103/PhysRevAccelBeams.23.044601.

[4] Arnau Albà. "Start-to-End Modelling of the AWA Microbunched Electron Cooling POP-Experiment". MA thesis. ETH Zürich, Paul Scherrer Institute, 2020.

[5] Nicole Neveu. "Beam Line Design for Fully Staged Two Beam Acceleration at the Argonne Wakefield Accelerator Facility". English. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Zuletzt aktualisiert - 2020-01-27. PhD thesis. 2018, p. 137. ISBN: 978-0-438-90619-8. URL: https://search.proquest.com/docview/2189100970?accountid=28439.

[6] Caio Davi. *Fundamentals of deep learning: why do neurons in perceptron of neural network need bias term?* Online. Retrieved 2020-05-30. May 2020. URL: https://developpaper.com/fundamentals-of-deep-learning-why-do-neurons-in-perceptron-of-neural-network-need-bias-term/.

[7] Favio Vázquez. *Deep Learning made easy with Deep Cognition*. Retrieved 2020-03-11. Dec. 2017.

[8] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (Dec. 22, 2014). arXiv: 1412.6980v9 [cs.LG].

[9] *Analyzing Inverse Problems with Invertible Neural Networks*. Online. Retrieved 2020-03-12. URL: https://hci.iwr.uni-heidelberg.de/vislearn/inverse-problems-invertible-neural-networks/#inverse-problems.

[10] Lynton Ardizzone et al. "Analyzing Inverse Problems with Invertible Neural Networks". In: (Aug. 14, 2018). arXiv: 1808.04730v3 [cs.LG].

[11] Jason Jinhuan. *Training and Implementing AlphaZero to play Hex*. Online. May 2018. URL: https://notes.jasonljin.com/projects/2018/05/20/Training-AlphaZero-To-Play-Hex.html.

[12] Arthur Gretton et al. "A kernel two-sample test". In: *Journal of Machine Learning Research* 13.Mar (2012), pp. 723–773. URL: http://www.jmlr.org/papers/volume13/gretton12a/gretton12a.pdf.

[13] *Intel® Xeon® Gold 6152 Processor*. Tech. rep. URL: https://ark.intel.com/content/www/us/en/ark/products/120491/intel-xeon-gold-6152-processor-30-25m-cache-2-10-ghz.html.

[14] M. D. McKay, R. J. Beckman, and W. J. Conover. "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code". In: *Technometrics* 21.2 (May 1979), p. 239. DOI: 10.2307/1268522.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[16] Renato Bellotti, Mélissa Zacharias, and Sichen Li. *MLLIB*. Online. URL: https://gitlab.psi.ch/adelmann/mllib.

[17] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[18] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[19] *Framework for Easily Invertible Architectures (FrEIA)*. Online. 2020. URL: https://github.com/VLL-HD/FrEIA#documentation.

[20] Plotly. *Introducing Dash*. Online. June 2017. URL: https://medium.com/plotly/introducing-dash-5ecf7191b503.

[21] Plotly Technologies Inc. *Collaborative data science*. 2015. URL: https://plot.ly.

[22] Online. URL: https://www.heroku.com/home.

[23] Julian Blank and Kalyanmoy Deb. *pymoo: Multi-objective Optimization in Python*. 2020. arXiv: 2002.04504 [cs.NE].

[24] Online. URL: https://gitlab.psi.ch/OPAL/runOPAL.

[25] K. Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197.

# Appendix A

# Positions of the beamline elements

## A.1 Cavities

- $[0.65, 1.71]$ m
- $[3.39, 4.44]$ m
- $[4.94, 5.99]$ m
- $[7.08, 8.13]$ m (defect)
- $[8.42, 9.47]$ m
- $[9.83, 10.89]$ m (defect)

Due to a bug, all train and test datasets use the following (wrong) cavity locations. They stem from previous communication with the experimental team at ANL.

- $[1., 1.8]$ m
- $[3., 4.5]$ m
- $[5., 6]$ m
- $[7., 8.2]$ m
- $[8.4, 9.5]$ m

This only matters for the resolution of the interpolation and should have negligible consequences, but should be mentioned for the sake of completeness.

Due to another bug, the samples at the positions of the YAG screens $0, 3$ and $7$ appear twice in the dataset. This concerns both the training/validation and the test set. Since these are only three positions out of over 1000, the effects of this bug are negligible.

## A.2 Solenoids

- $[1.88, 2.28]$ m
- $[4.45, 4.85]$ m
- $[6.49, 6.89]$ m

## A.3 YAG screens

- 0.5 m
- 2.93 m
- 6.22 m
- 9.47 m

- 11.36 m
- 11.57 m
- 15.13 m
- 16.70 m
- 19.26 m
- 20.69 m
- 22.99 m
- 25.27 m

# Appendix B

# Features

**Design Variables (DVARs)**

- **IBF [A]:**
  Current through buck focusing solenoid
- **IM [A]:**
  Current through matching solenoid
- **GPHASE [°]:**
  Gun phase
- **ILS1 [A]:**
  Current through linac solenoid 1
- **ILS2 [A]:**
  Current through linac solenoid 2
- **ILS3 [A]:**
  Current through linac solenoid 3
- **Mean bunch charge [nC]**
- **$\lambda$ [ps]:**
  Peak-to-peak separation between the 4 peaks of the initial bunch distribution
- **SIGXY [mm]:**
  Laser spot size

| Bound | IBF [A] | IM [A] | GPHASE [°] | ILS1 [A] | ILS2 [A] | ILS3 [A] | charge [nC] | $\lambda$ [ps] | SIGXY [mm] |
|---|---|---|---|---|---|---|---|---|---|
| Lower | 450 | 100 | -50 | 0 | 0 | 0 | 0.3 | 0.3 | 1.5 |
| Upper | 550 | 260 | 10 | 250 | 200 | 200 | 5 | 2 | 12.5 |

Table B.1: The design variables and their ranges.

Quantities of Interest (QOIs)

- **Mean bunch energy** $E$
- $\sigma_{\mathbf{x}}, \sigma_{\mathbf{y}}$**:**
  Transversal root mean square beam sizes
- $\epsilon_{\mathbf{x}}, \epsilon_{\mathbf{y}}$**:**
  Transversal normalized emittances
- $\mathbf{\Delta E}$**:**
  RMS energy spread over one bunch
- $\mathrm{Corr}(\mathbf{x}, \mathbf{p_x}), \mathrm{Corr}(\mathbf{y}, \mathbf{p_y})$**:**
  Correlations between transversal position and momentum.
  These were considered only for the forward models.

# Appendix C

# Datasets

The dataset for the invertible model contains the same design variable configuration as the one for the forward model. However, less the positions of the solenoids and cavities are omitted, as well as the drift from $13 - 26$ m. Further, the emittances are clipped to the range $[0, 200]$ mm mrad. The cost of these steps is negligible compared to the one of running OPAL. Therefore, the same computational cost is assumed for generating the dataset for the invertible model as for the forward model.

## C.1 Big train/val dataset

| | **Forward model** | **Invertible model** |
|---|---|---|
| OPAL runs | 20'000 | 20'000 |
| Invalid runs | 1935 (9.675%) | 1935 (9.675%) |
| Valid samples train/val set | 27'838'165 | 8'598'940 |
| CPU h on Merlin6 | 13'992[1] | - |

Table C.1: Information about the datasets.

Names of the files:

`4_peak_distr_lower_charge_gun_and_cavities_20k slinear 2D_filtered.hdf5`

`4_peak_distr_lower_charge_gun_and_cavities_20k slinear 2D_filtered_filtered_epsilon_clipped_at_0.0002_no_solenoids2_no_cavities.hdf5`

## C.2 Medium train/val dataset

| | **Forward model** | **Invertible model** |
|---|---|---|
| OPAL runs | 8'000 | 8000 |
| Invalid runs | 804 (10.05%) | 804 (10.05%) |
| Valid samples train/val set | 11'089'036 | 3'425'296 |
| CPU h on Merlin6 | 5'544 | - |

Table C.2: Information about the datasets.

Names of the files:

`4_peak_distr_lower_charge_and_IM slinear 2D_filtered corrected 8k slinear 2D_filtered.hdf5`

`4_peak_distr_lower_charge_gun_and_cavities_8k slinear 2D_filtered_filtered_epsilon_clipped_at_0.0002_no_solenoids2_no_cavities.hdf5`

## C.3 Small train/val dataset

Names of the files:

`4_peak_distr_lower_charge 4k slinear 2D_filtered.hdf5`

`4_peak_distr_lower_charge_gun_and_cavities_4k slinear 2D_filtered_filtered_epsilon_clipped_at_0.0002_no_solenoids2_no_cavities.hdf5`

| | **Forward model** | **Invertible model** |
|---|---|---|
| OPAL runs | 4'000 | 4'000 |
| Invalid runs | 413(10.325%) | 413(10.325%) |
| Valid samples train/val set | 5'527'567 | 1'707'412 |
| CPU h on Merlin6 | 6'336[1] | - |

Table C.3: Information about the datasets.

## C.4 Test dataset

| | **Forward model** | **Invertible model** |
|---|---|---|
| OPAL runs | 1'000 | 1'000 |
| Invalid runs | 87 (8.7 %) | 87 (8.7 %) |
| Valid samples | 1'406'933 | 1'176'857 |
| CPU h on Merlin6 | 633 | - |

Table C.4: Information about test dataset. The same test set is used for the forward and the invertible model.

The test set is the only dataset for the invertible model that also contains samples within the cavities and solenoids.

Names of the files:
`4_peak_distr_lower_charge_and_IM slinear 2D_filtered corrected 1k slinear 2D_filtered.hdf5`
`4_peak_distr_lower_charge_gun_and_cavities_1k slinear 2D_filtered.hdf5`

[1] There is a bug in OPAL that causes random samples to hang. The jobs continues running, but no output is written. The SLURM job has to be killed, and some OPAL runs have to be restarted manually. This causes the job to take longer than necessary, leading to a higher resource consumption.

# Appendix D

# Dashboards

## D.1 GUI for forward predictions

### D.1.1 Description

The dashboard is designed to be as easy to use as possible (Fig D.1). No prior knowledge about the dataset and the model is required for end users.

The user enters values of the design variables into the text fields in the top right corner. The program then evaluates the underlying surrogate model at many positions. The resulting curves are plotted, together with a measure of prediction error.

### D.1.2 Estimating the accuracy

The measure of accuracy is calculated in advance with the following algorithm:

1. Evaluate all DVAR configurations in the test set with the surrogate model. This gives a matrix $\tilde{Y}$, where each row corresponds to one design variable configuration in the test set, and each column contains the values of one quantity of interest.
2. Calculate the residuals $R = \left| Y_{\text{OPAL}} - \tilde{Y} \right|$. Subtraction and taking the absolute value are applied componentwise.
3. Group the rows of $R$ by longitudinal position $\{R_s\}_{s \in S}$, where $S$ is the set of all longitudinal positions in the test set.
4. Calculate the quantiles (over all DVAR configurations in the test set) for each quantity of interest at each position. $r_{\sigma_x,s,\alpha}$ is the $\alpha$-quantile for the transverse beamsize along $x$ at position $s$. This means that a fraction of $\alpha$ samples at position $s$ have a residual that is smaller than (or equal) $r_{\sigma,s,\alpha}$.

   The analogue quantities are calculated for the other quantities of interest.

The error bars in the plots are calculated with the following equation:

$$y_{\sigma_x,\text{upper/lower}}(s_i) = \tilde{y}_{\sigma_x}(\mathbf{x}, s) \pm r_{\sigma_x,s,\alpha} \tag{D.1}$$

The confidence level $\alpha$ is set by the user.
This formula overestimates the model uncertainty. For the calculation of the residuals, only the magnitudes are considered, but not their sign. As a consequence, the error bars would be smaller if the sign information was taken into account.

### D.1.3 Advantages

The GUI allows scientists to evaluate the effect of the design variables in real time. To the best of our knowledge, there has not been a similar tool for particle accelerators yet. A single evaluation of OPAL takes approximately 20 minutes. This is prohibitive to playful trial and error. Our GUI makes this possible. It is even possible to perform a quick optimisation "by hand". This ability has proven to be a useful tool during the planning phase of the AWA experiment.

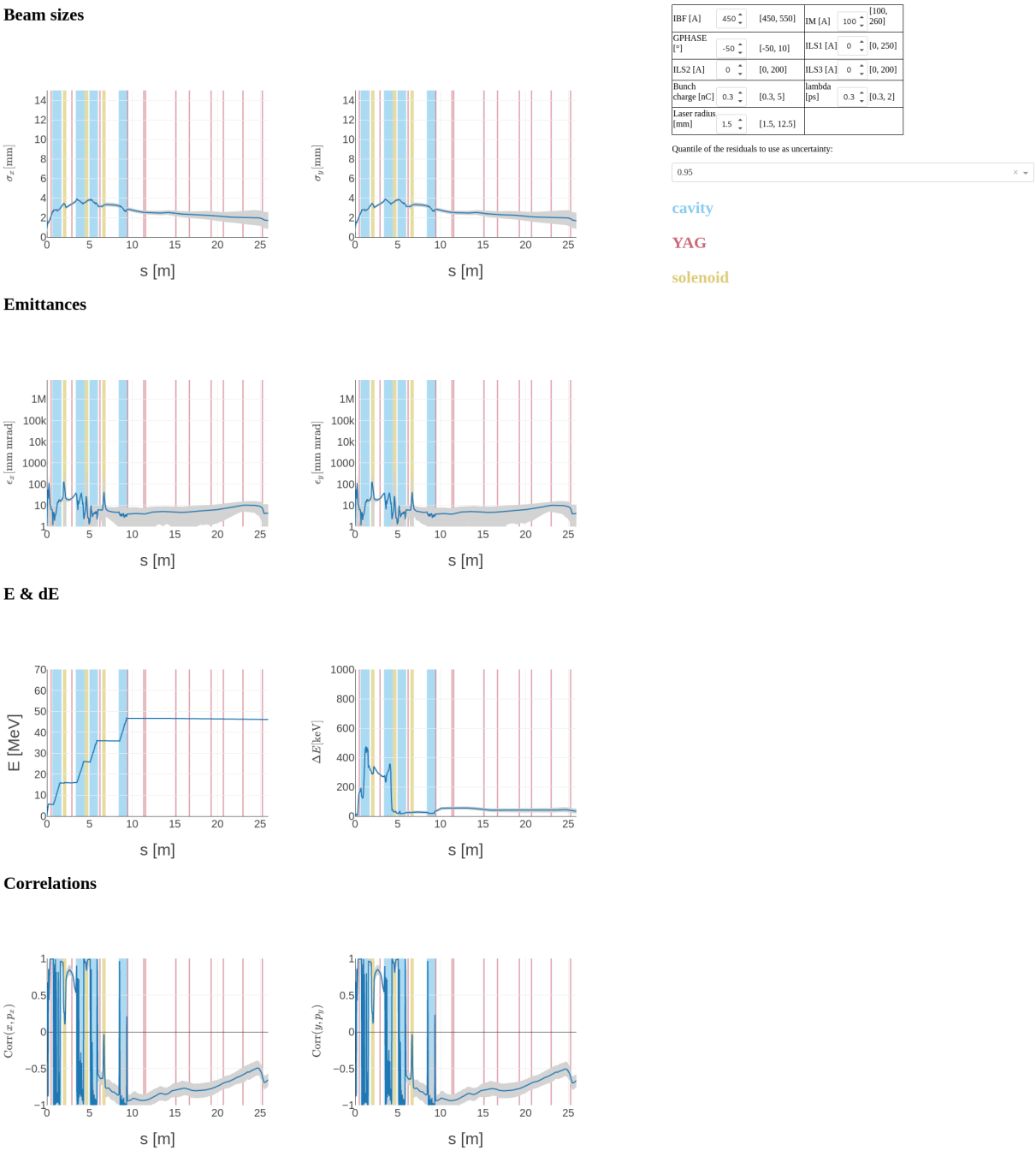



Figure D.1: **Screenshot of the GUI for a forward prediction.** The dashboard can be used both for a forward model and the forward prediction of an invertible model. The user types the design variable configuration into the input fields in the top-right corner. The annotated intervals correspond to the ranges on which the model is trained. Internally, the surrogate model is evaluated for many longitudinal positions. The resulting predictions are plotted. The area shaded in gray is a measure of uncertainty. The confidence level can be set with a drop down menu below the text boxes. The vertical colored stripes indicate the positions of the beam elements. The color legend can be seen to the right.

### D.1.4 Shortcomings

A fast and easy to use GUI has many advantages. Nevertheless, the user has to be cautious. Not all possible design variable configurations are captured equally well. However, the confidence intervals are precomputed. They do not depend on the values the user enters in the GUI. This means that they are rather a global indicator of the prediction accuracy than a confidence interval for an individual prediction.

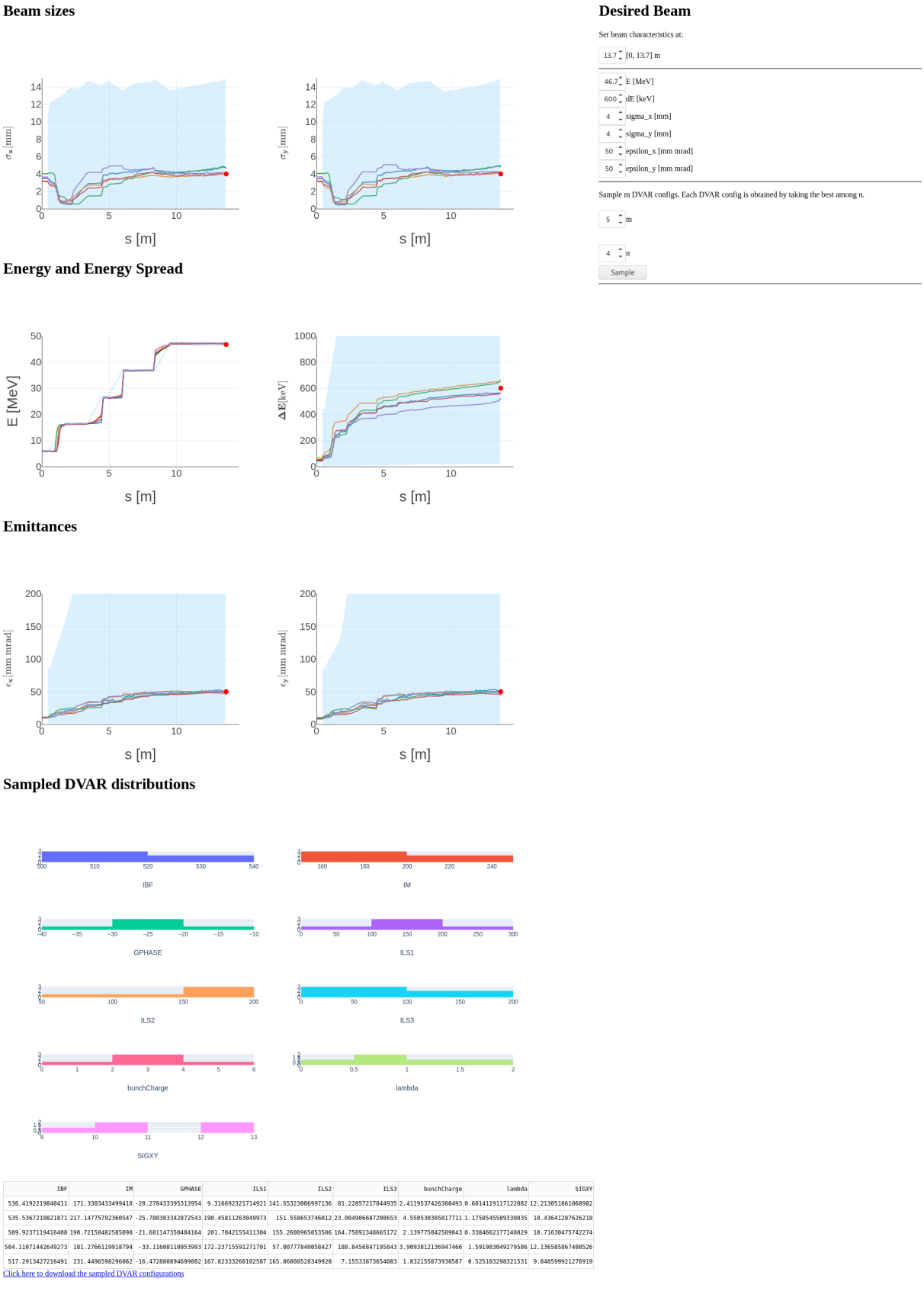
Another issue is related to this. The model is not trained on all possible DVAR configurations (see Sec. 2.2.4). Some configurations are excluded. As a consequence, it cannot predict some types of configurations correctly. The model is not trained on them. Even worse, there is no way to tell if such a configuration has been entered in the text fields. Users must keep this in mind.

## D.2 GUI for inverse prediction

A screenshot of the GUI for the inverse prediction is shown in Fig. D.2. The user can enter the values of the quantities of interest he/she wants to realise at a certain position. A click on the "Sample" button generates $n$ DVAR configurations. The one that comes closest to the desired beam according to the maximum percentage error is selected and plotted. This is repeated $m$ times, leading to $m$ curves for each QOI.

The reader might wonder why $m$ is needed. Of course, $n$ would suffice to select the best beam according to the MAPE. However, displaying multiple beams allows the user to make a trade off. For example, in Fig. D.2, purple curve is good for the beam sizes, but not so good for the emittances. If the user has stricter constraints on the emittances, he/she might prefer the blue curve instead.

It would also be possible to make the tradeoff by leaving away $m$ and just press the "Sample" button multiple times, until a good enough beam is achieved. We chose to add the $m$ textfield because it gives the user the possibility to estimate at one glance how much variance there is in the sampling.




| IBF | IM | GPHASE | ILS1 | ILS2 | ILS3 | bunchCharge | lambda | SIGXY |
|---|---|---|---|---|---|---|---|---|
| 536.4192219848411 | 171.3303433499418 | -20.278433395313954 | 9.316692321714921 | 141.55323006997136 | 81.22857217844935 | 2.4119537426308493 | 0.6014119117122082 | 12.213051861068982 |
| 535.5367210821871 | 217.14775792360547 | -25.780383342872543 | 190.45811263049973 | 151.558653746012 | 23.004906687208653 | 4.550530385017711 | 1.1758545589330835 | 10.436412876262 18 |
| 509.9237119416408 | 190.72158482585098 | -21.681147350404164 | 201.7042155411304 | 155.26009650535 86 | 164.75892348665172 | 2.139775042509843 | 0.3384662177140829 | 10.716304757422 74 |
| 504.1107144264927 3 | 181.2766119918794 | -33.11608110953993 | 172.23715591271701 | 57.00777840058427 | 188.8456847195843 | 3.9093812136947466 | 1.5919830492795 86 | 12.136585867408526 |
| 517.2913427216491 | 231.4496598296062 | -16.472880894699802 | 167.82333260102587 | 165.86808528349928 | 7.15533873654083 | 1.832155873930587 | 0.5251032983215 31 | 9.840599921276919 |

Click here to download the sampled DVAR configurations

Figure D.2: **GUI for the inverse prediction.** A user enters the beam he/she desires into the textfields. They are located in the top right corner. The network will make $m$ suggestions of design variables. For each suggestion, $n$ DVAR configurations are sampled, and the one that gets closest to the desired beam is selected. The distribution of the DVARs in the $m$ suggested configurations is plotted as a histogram below the QOI plots. The values are shown in a table and can be downloaded as a TSV file by clicking on the link at the very bottom. The format is readable by the OPAL sampler.

## D.3 Visualising forward prediction performance

### D.3.1 Description and Advantages

This dashboard allows to visually compare model predictions to the ground truth, i. e. the values calculated by OPAL. Ideally, both curves overlap. This form of validation comes very close to the use case of a forward surrogate model: To be able to replace the computational model. Even without a machine learning background, it is straightforward to see which patterns along the beamline are captured well, just by taking a look at a handful of configurations.

### D.3.2 Shortcomings

Due to storage limitations on the free hosting plan of Heroku, only 200 OPAL runs from the test set can be uploaded. They are selected randomly.

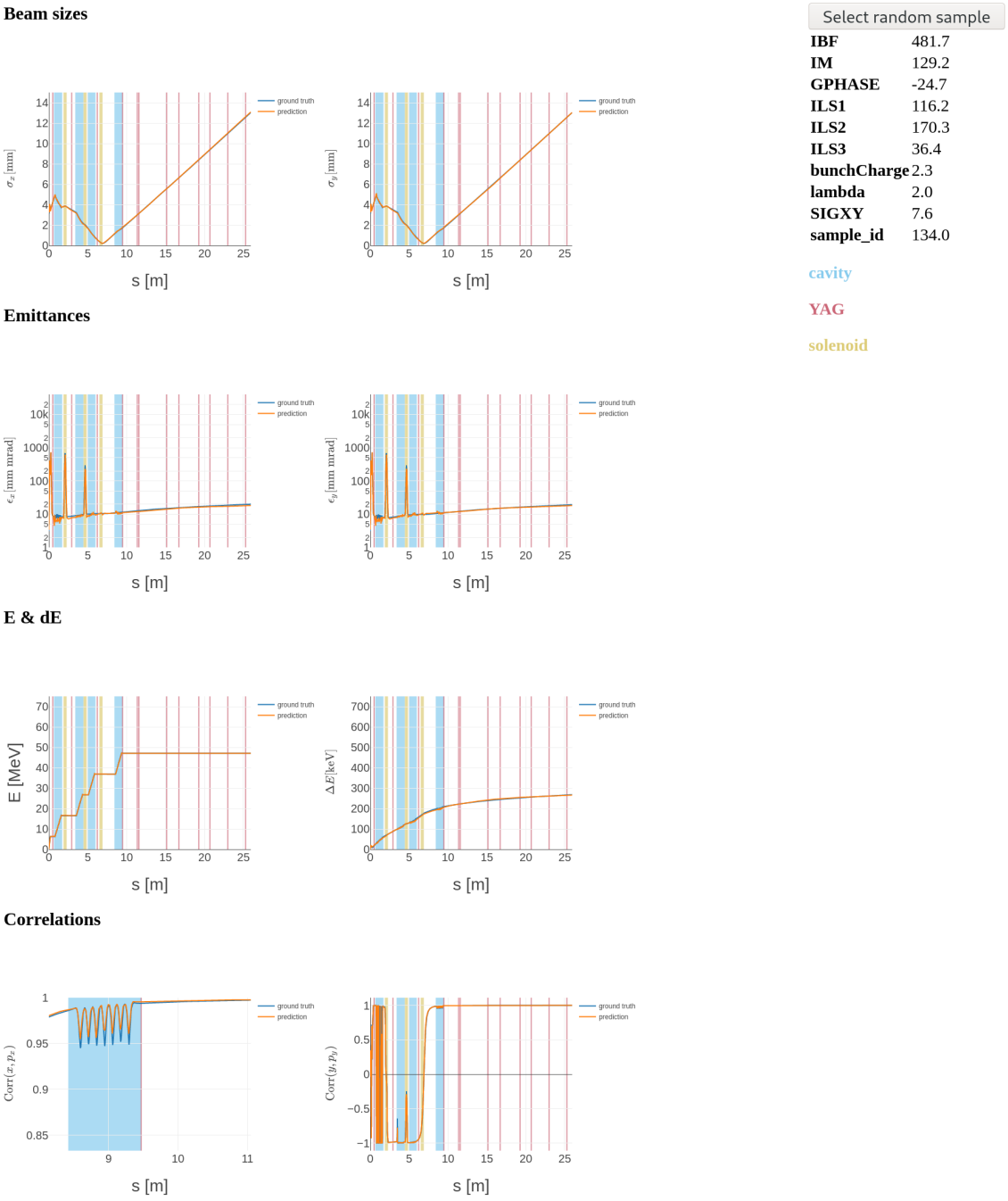



Figure D.3: **Screenshot of the dashboard to compare forward predictions to OPAL.** A random configuration from the test set can be chosen by pushing the button on the top right. The design variable configuration and its ID are shown in the table below the button. The configuration is evaluated at many positions, and the resulting predicted beam is plotted along with the one calculated with OPAL. The colors of the shaded vertical stripes indicate positions of beam elements. A legend for the colors is shown to the right.

## D.4 Visualising inverse prediction performance

### D.4.1 Description

A screenshot of the dashboard can be found in Fig. D.4. The dashboard contains a subset (200 OPAL runs) of the test set. More runs could not be included due to disk space constraints on Heroku. One OPAL run is randomly chosen whenever the user presses the "New configuration" button. That configuration is plotted as the green line. Its last value (i. e. the one at 13.65m) is used as the target $\mathbf{y}_{\text{true}}$ for the inverse sampling. This is shown in the plots as the red dots, and the values are shown in the table in the top right corner.

Then, $n$ DVAR configurations $\mathbf{x}_1, ..., \mathbf{x}_n$ are sampled, where $n$ is set by the user (in the screenshot, $n = 4$). The inverse model runs a forward prediction for each of the $n$ DVAR configurations. The best one $\mathbf{x}_b$ is chosen based on the maximum percentage error $e$ compared to the desired QOI configuration at 13.65m:

$$b = \underset{i}{\operatorname{argmin}}\, e\left(\hat{\mathbf{f}}(\mathbf{x}_i), \mathbf{y}_{\text{true}}\right) \tag{D.2}$$

Afterwards, the both the inverse model and its corresponding validation model are used to predict the QOIs at many positions. This is represented by the blue and the orange line, respectively.

### D.4.2 Advantages and Shortcomings

This dashboard offers much insight into an invertible model.
First, one can see how good the sampled DVAR configurations are according to the model itself. This helps to decide if the sampling is the problem: If the model predicts that the sampled DVARs lead to curves that are far away from the desired values (red dots), then the sampling is not good.

Of course, it can also be that the invertible model has a faulty forward prediction. To see this, a more precise forward model is used for the forward prediction. If the invertible model and the forward model disagree, then the forward prediction of the invertible model is probably not good enough.

The validation model is not without fault. A final validation is only obtained by running OPAL for the sampled DVARs. This is nothing that can be done interactively, and is therefore omitted in the dashboard.

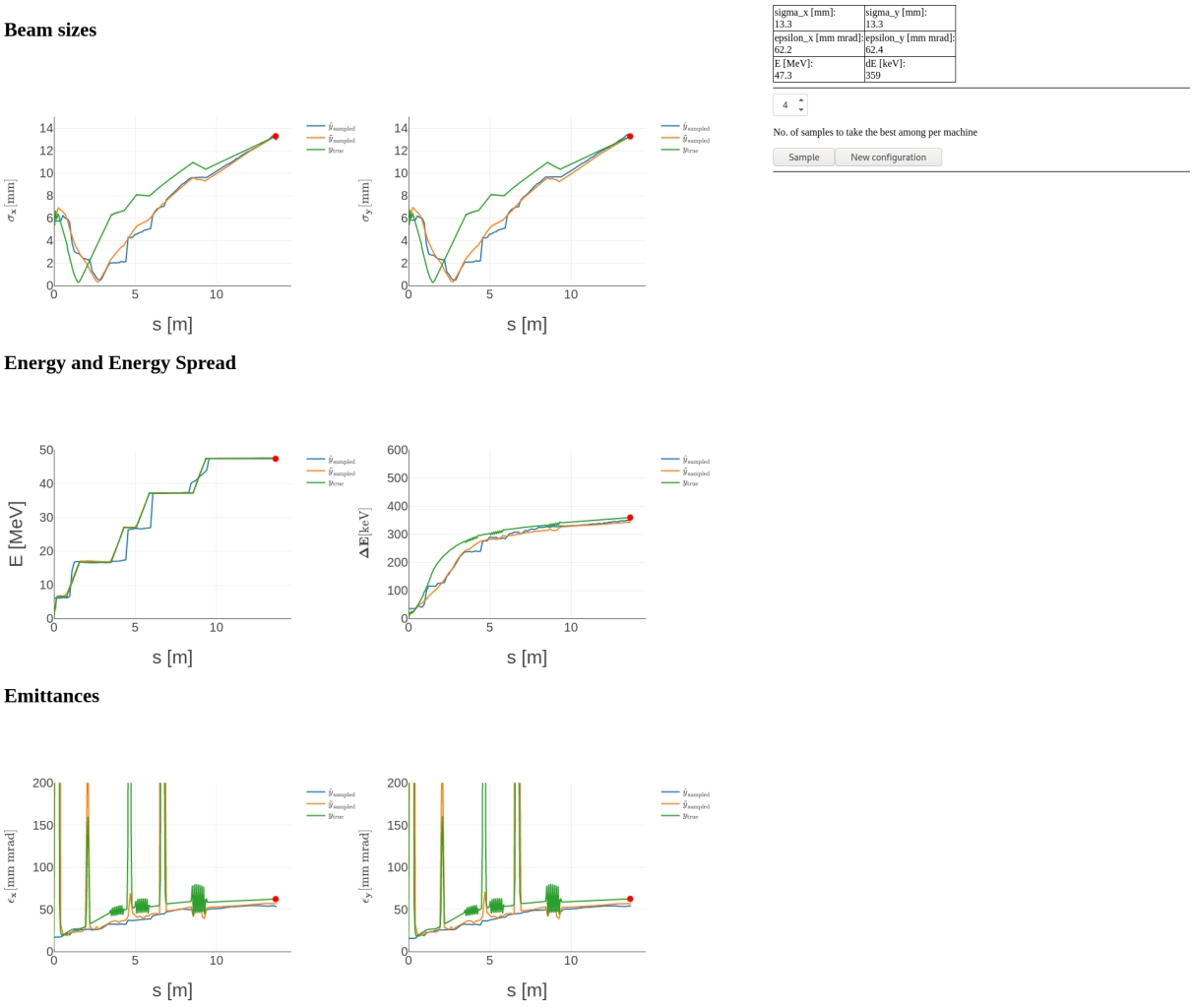



Figure D.4: **Screenshot of the dashboard to compare inverse predictions to OPAL.** The dashboard program contains 200 samples from the test set. One configuration is chosen at random by clicking the "New configuration" button in the top right. The corresponding quantities of interest at 13.65m are shown in the table above the button, and marked in the plots as red dots. The green curve represents the OPAL run corresponding to this configuration. A click on the "Sample" button triggers the sampling of some (here: 4) design variable configurations, among which the best according to the maximum percentage error is chosen. The forward predictions according to the invertible and the validation model of these configurations are also plotted (blue and orange curves).

# Appendix E

# Parameters of the baseline models

## E.1 Forward Model

- Trained on the big train/val set
- Number of hidden layers: 8
- Number of neurons per hidden layer: 500
- Activation function for each hidden neuron: ReLU
- Optimiser: Adam
- Learning rate: $10^{-4}$
- Batch size: 128
- Epochs: 30

## E.2 Invertible Model

- Trained on the big train/val set
- Dimension of latent space: 1
- Distribution of latent space: uniform
- Nominal dimension: 12
- Architecture: $8 \times 5 \times 100$
- Activation function for each hidden neuron: ReLU
- Loss weights: $w_x = w_y = w_z = 400, w_r = 3, w_{\text{artificial}} = 1$
- Optimiser: Adam
- Learning rate: $10^{-4}$
- Batch size: 256
- Epochs: 10

# Appendix F

# Hyperparameter Scan for the Forward Model

| | hiddenLayers | unitsPerLayer | batch_size | 0.25 | 0.5 | 0.75 | 0.9 | 0.95 | 0.99 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 6 | 300 | 128 | 5.0 | 8.9 | 16.5 | 36.2 | 79.6 | 411.1 |
| 1 | 6 | 300 | 256 | 4.5 | 8.0 | 14.8 | 34.8 | 83.6 | 419.1 |
| 2 | 6 | 400 | 128 | 4.6 | 8.0 | 15.0 | 33.9 | 74.5 | 379.0 |
| 3 | 6 | 400 | 256 | 4.5 | 8.0 | 14.6 | 34.5 | 93.9 | 414.4 |
| 4 | 6 | 500 | 128 | 5.2 | 9.3 | 17.3 | 40.0 | 97.0 | 458.7 |
| 5 | 6 | 500 | 256 | 4.4 | 7.7 | 14.1 | 30.0 | 67.6 | 381.8 |
| 6 | 6 | 600 | 128 | 4.8 | 8.4 | 15.7 | 34.0 | 74.3 | 388.1 |
| 7 | 6 | 600 | 256 | 4.5 | 7.9 | 15.1 | 40.7 | 114.7 | 476.3 |
| 8 | 6 | 700 | 128 | 4.7 | 8.4 | 15.8 | 38.9 | 104.5 | 466.7 |
| 9 | 6 | 700 | 256 | 4.4 | 7.7 | 14.3 | 34.3 | 87.8 | 437.0 |
| 10 | 7 | 300 | 128 | 5.2 | 9.0 | 16.0 | 36.1 | 88.3 | 442.3 |
| 11 | 7 | 300 | 256 | 4.9 | 8.7 | 16.1 | 37.9 | 85.5 | 455.0 |
| 12 | 7 | 400 | 128 | 4.7 | 8.3 | 15.7 | 39.1 | 101.3 | 440.3 |
| 13 | 7 | 400 | 256 | 4.9 | 8.7 | 16.2 | 35.7 | 80.5 | 416.6 |
| 14 | 7 | 500 | 128 | 4.7 | 8.3 | 15.4 | 34.6 | 83.7 | 430.5 |
| 15 | 7 | 500 | 256 | 4.2 | 7.4 | 13.5 | 30.7 | 73.2 | 393.1 |
| 16 | 7 | 600 | 128 | 4.9 | 8.8 | 16.4 | 37.0 | 89.9 | 484.0 |
| 17 | 7 | 600 | 256 | 4.6 | 8.0 | 15.1 | 39.6 | 105.0 | 488.3 |
| 18 | 7 | 700 | 128 | 5.2 | 9.2 | 17.9 | 51.1 | 161.4 | 588.2 |
| 19 | 7 | 700 | 256 | 4.7 | 8.3 | 15.3 | 35.7 | 91.4 | 469.0 |
| 20 | 8 | 300 | 128 | 5.3 | 9.6 | 18.1 | 40.5 | 99.8 | 493.1 |
| 21 | 8 | 300 | 256 | 5.0 | 8.9 | 16.6 | 37.9 | 92.5 | 439.9 |
| 22 | 8 | 400 | 128 | 5.5 | 9.8 | 18.0 | 41.4 | 99.5 | 485.5 |
| 23 | 8 | 400 | 256 | 4.9 | 8.5 | 15.5 | 32.7 | 72.5 | 410.3 |
| 24 | 8 | 500 | 128 | 5.2 | 9.4 | 17.7 | 40.4 | 97.4 | 468.8 |
| 25 | 8 | 500 | 256 | 5.0 | 8.7 | 16.2 | 40.0 | 111.9 | 493.5 |
| 26 | 8 | 600 | 128 | 5.4 | 9.5 | 17.7 | 45.6 | 124.1 | 524.6 |
| 27 | 8 | 600 | 256 | 4.8 | 8.5 | 15.6 | 35.0 | 87.2 | 415.9 |
| 28 | 8 | 700 | 128 | 5.2 | 9.2 | 17.2 | 38.7 | 97.2 | 479.5 |
| 29 | 8 | 700 | 256 | 5.1 | 9.0 | 16.8 | 39.4 | 100.9 | 489.7 |
| 30 | 9 | 300 | 128 | 6.4 | 10.9 | 19.3 | 44.1 | 103.7 | 486.4 |
| 31 | 9 | 300 | 256 | 5.5 | 9.9 | 19.1 | 49.5 | 139.9 | 556.6 |
| 32 | 9 | 400 | 128 | 5.4 | 9.7 | 17.9 | 41.6 | 109.9 | 535.0 |
| 33 | 9 | 400 | 256 | 5.4 | 9.6 | 17.8 | 42.3 | 110.6 | 519.7 |
| 34 | 9 | 500 | 128 | 5.5 | 9.8 | 18.0 | 40.6 | 103.4 | 498.5 |
| 35 | 9 | 500 | 256 | 5.6 | 10.0 | 19.1 | 52.9 | 155.7 | 560.6 |
| 36 | 9 | 600 | 128 | 5.6 | 10.0 | 18.2 | 41.9 | 102.3 | 483.9 |
| 37 | 9 | 600 | 256 | 5.5 | 9.6 | 17.9 | 41.3 | 97.4 | 536.3 |
| 38 | 9 | 700 | 128 | NaN | NaN | NaN | NaN | NaN | NaN |
| 39 | 9 | 700 | 256 | 5.2 | 9.1 | 17.0 | 42.7 | 121.0 | 474.0 |
| 40 | 10 | 300 | 128 | 5.6 | 10.0 | 18.5 | 40.5 | 87.6 | 463.5 |
| 41 | 10 | 300 | 256 | 6.1 | 10.8 | 20.4 | 52.8 | 154.6 | 600.2 |
| 42 | 10 | 400 | 128 | 5.7 | 10.2 | 19.1 | 42.7 | 103.6 | 474.8 |
| 43 | 10 | 400 | 256 | 5.1 | 9.0 | 16.5 | 34.6 | 77.5 | 455.7 |
| 44 | 10 | 500 | 128 | 6.1 | 10.8 | 20.1 | 48.7 | 140.2 | 566.6 |
| 45 | 10 | 500 | 256 | 5.6 | 9.8 | 17.6 | 36.7 | 83.6 | 457.2 |
| 46 | 10 | 600 | 128 | 5.3 | 9.6 | 18.0 | 40.6 | 95.9 | 519.4 |
| 47 | 10 | 600 | 256 | 5.4 | 9.6 | 18.1 | 41.4 | 100.8 | 500.1 |
| 48 | 10 | 700 | 128 | NaN | NaN | NaN | NaN | NaN | NaN |
| 49 | 10 | 700 | 256 | 5.3 | 9.4 | 17.4 | 37.4 | 89.9 | 473.6 |

Table F.1: Parameters of the hyperparameter scan for the forward model. The numbers $0.25, 0.5, 0.75, 0.9, 0.95$ and $0.99$ denote quantiles of the maximum percentage error. The two rows highlighted in red mark parameter configurations that have not finished training after 4 days. These configurations were ignored. Row number 15 (highlighted in yellow) is the configuration of the best forward model.

# Appendix G

# Hyperparameter Scan for the Invertible Model

| | number_of_blocks | depth | width | 0.25 | 0.5 | 0.75 | 0.9 | 0.95 | 0.99 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 3 | 60 | 48.9 | 74.1 | 140.1 | 328.5 | 573.8 | 1675.5 |
| 1 | 4 | 3 | 80 | 48.1 | 73.4 | 141.3 | 326.0 | 584.4 | 1973.7 |
| 2 | 4 | 3 | 100 | 44.4 | 68.8 | 129.9 | 331.9 | 623.5 | 2011.1 |
| 3 | 4 | 3 | 120 | 42.5 | 66.6 | 113.9 | 254.0 | 431.4 | 1319.5 |
| 4 | 4 | 3 | 140 | 41.6 | 64.1 | 105.6 | 246.1 | 430.2 | 1320.1 |
| 5 | 4 | 5 | 60 | 43.4 | 67.1 | 115.5 | 271.0 | 485.8 | 1741.7 |
| 6 | 4 | 5 | 80 | 47.1 | 71.4 | 130.1 | 322.4 | 557.8 | 1529.9 |
| 7 | 4 | 5 | 100 | 45.2 | 70.6 | 137.0 | 338.2 | 592.6 | 1703.8 |
| 8 | 4 | 5 | 120 | 44.1 | 69.2 | 127.4 | 301.4 | 521.2 | 1491.0 |
| 9 | 4 | 5 | 140 | 46.6 | 70.1 | 122.5 | 283.9 | 495.1 | 1627.6 |
| 10 | 4 | 7 | 60 | 50.9 | 76.8 | 145.6 | 320.3 | 533.5 | 1508.2 |
| 11 | 4 | 7 | 80 | 53.6 | 80.0 | 164.5 | 379.1 | 662.1 | 2024.4 |
| 12 | 4 | 7 | 100 | 48.0 | 73.9 | 136.3 | 298.7 | 508.9 | 1639.1 |
| 13 | 4 | 7 | 120 | 54.9 | 80.5 | 161.7 | 367.3 | 615.4 | 1695.9 |
| 14 | 4 | 7 | 140 | 47.8 | 74.4 | 144.9 | 351.1 | 613.9 | 1741.3 |
| 15 | 6 | 3 | 60 | 40.9 | 64.7 | 110.5 | 243.4 | 412.2 | 1200.4 |
| 16 | 6 | 3 | 80 | 39.7 | 62.0 | 102.6 | 250.1 | 455.7 | 1539.3 |
| 17 | 6 | 3 | 100 | 38.0 | 59.3 | 93.6 | 210.7 | 367.8 | 1099.5 |
| 18 | 6 | 3 | 120 | 36.2 | 56.3 | 89.3 | 207.0 | 390.4 | 1249.5 |
| 19 | 6 | 3 | 140 | 39.8 | 61.4 | 98.0 | 243.7 | 447.3 | 1478.3 |
| 20 | 6 | 5 | 60 | 46.0 | 71.6 | 135.9 | 314.7 | 535.9 | 1594.5 |
| 21 | 6 | 5 | 80 | 45.1 | 70.2 | 126.8 | 294.8 | 512.9 | 1645.7 |
| 22 | 6 | 5 | 100 | 37.1 | 59.0 | 92.3 | 210.9 | 375.3 | 1285.5 |
| 23 | 6 | 5 | 120 | 40.5 | 63.5 | 101.5 | 217.0 | 364.8 | 1094.0 |
| 24 | 6 | 5 | 140 | 45.6 | 69.9 | 125.5 | 298.4 | 525.2 | 1645.2 |
| 25 | 6 | 7 | 60 | 45.4 | 69.5 | 127.0 | 315.0 | 573.8 | 1878.0 |
| 26 | 6 | 7 | 80 | 50.1 | 74.1 | 138.9 | 341.0 | 592.1 | 1666.1 |
| 27 | 6 | 7 | 100 | 44.5 | 68.6 | 120.6 | 278.0 | 471.1 | 1403.9 |
| 28 | 6 | 7 | 120 | 57.1 | 82.6 | 168.1 | 381.1 | 627.1 | 1635.5 |
| 29 | 6 | 7 | 140 | 47.4 | 74.1 | 143.2 | 329.2 | 556.0 | 1621.4 |
| 30 | 8 | 3 | 60 | 40.2 | 62.6 | 100.8 | 232.5 | 402.3 | 1190.6 |
| 31 | 8 | 3 | 80 | 38.1 | 60.3 | 98.1 | 232.6 | 432.7 | 1700.5 |
| 32 | 8 | 3 | 100 | 36.5 | 58.2 | 93.3 | 218.3 | 406.8 | 1407.4 |
| 33 | 8 | 3 | 120 | 34.6 | 55.5 | 91.1 | 212.3 | 387.0 | 1328.5 |
| 34 | 8 | 3 | 140 | 37.3 | 59.9 | 95.0 | 221.7 | 394.5 | 1288.4 |
| 35 | 8 | 5 | 60 | 48.1 | 72.6 | 130.7 | 299.1 | 512.1 | 1507.1 |
| 36 | 8 | 5 | 80 | 41.1 | 64.6 | 112.6 | 267.7 | 462.1 | 1354.4 |
| 37 | 8 | 5 | 100 | 35.6 | 56.7 | 91.0 | 202.1 | 348.1 | 1101.9 |
| 38 | 8 | 5 | 120 | 43.5 | 68.3 | 120.0 | 280.4 | 501.4 | 1887.9 |
| 39 | 8 | 5 | 140 | 35.6 | 57.0 | 91.6 | 224.3 | 414.6 | 1354.8 |
| 40 | 8 | 7 | 60 | 53.4 | 79.9 | 158.5 | 364.1 | 619.5 | 1755.1 |
| 41 | 8 | 7 | 80 | 52.0 | 77.7 | 150.9 | 357.6 | 611.8 | 1660.7 |
| 42 | 8 | 7 | 100 | 44.9 | 70.0 | 124.6 | 267.8 | 442.0 | 1263.9 |
| 43 | 8 | 7 | 120 | 48.1 | 73.9 | 143.1 | 344.8 | 593.8 | 1736.3 |
| 44 | 8 | 7 | 140 | 36.8 | 58.0 | 94.2 | 232.5 | 427.6 | 1504.2 |
| 45 | 10 | 3 | 60 | 41.2 | 65.1 | 111.8 | 249.7 | 415.6 | 1230.3 |
| 46 | 10 | 3 | 80 | 36.3 | 58.2 | 94.3 | 225.8 | 404.6 | 1297.0 |
| 47 | 10 | 3 | 100 | 36.0 | 56.5 | 90.0 | 208.6 | 383.9 | 1281.6 |
| 48 | 10 | 3 | 120 | 34.7 | 55.3 | 88.4 | 194.4 | 352.1 | 1208.2 |
| 49 | 10 | 3 | 140 | 40.2 | 62.8 | 103.1 | 241.9 | 424.6 | 1360.9 |
| 50 | 10 | 5 | 60 | 44.8 | 68.5 | 121.7 | 287.5 | 493.6 | 1464.8 |
| 51 | 10 | 5 | 80 | 39.5 | 62.0 | 101.3 | 234.2 | 407.4 | 1235.5 |
| 52 | 10 | 5 | 100 | 43.9 | 69.4 | 128.3 | 305.5 | 532.5 | 1520.1 |
| 53 | 10 | 5 | 120 | 36.6 | 59.2 | 96.1 | 229.9 | 414.4 | 1354.3 |
| 54 | 10 | 5 | 140 | 38.3 | 60.6 | 98.1 | 225.2 | 394.2 | 1266.0 |
| 55 | 10 | 7 | 60 | 42.4 | 66.0 | 110.2 | 268.1 | 492.4 | 1494.6 |
| 56 | 10 | 7 | 80 | 43.4 | 65.8 | 110.1 | 251.2 | 435.3 | 1410.1 |
| 57 | 10 | 7 | 100 | 46.9 | 73.4 | 138.0 | 313.2 | 529.0 | 1550.6 |
| 58 | 10 | 7 | 120 | 45.7 | 70.2 | 123.1 | 291.0 | 504.3 | 1389.6 |
| 59 | 10 | 7 | 140 | 48.7 | 73.4 | 142.8 | 366.3 | 654.6 | 1959.9 |
| 60 | 12 | 3 | 60 | 39.9 | 63.0 | 107.0 | 259.8 | 452.6 | 1286.1 |
| 61 | 12 | 3 | 80 | 38.1 | 59.9 | 95.2 | 214.8 | 373.4 | 1102.0 |
| 62 | 12 | 3 | 100 | 34.4 | 54.8 | 87.5 | 194.8 | 356.9 | 1207.4 |
| 63 | 12 | 3 | 120 | 36.3 | 58.0 | 93.3 | 217.9 | 393.6 | 1248.1 |
| 64 | 12 | 3 | 140 | 34.8 | 56.1 | 90.0 | 202.0 | 359.4 | 1099.5 |
| 65 | 12 | 5 | 60 | 41.2 | 64.5 | 110.8 | 259.1 | 452.4 | 1355.9 |
| 66 | 12 | 5 | 80 | 39.0 | 60.9 | 99.8 | 230.2 | 398.5 | 1247.5 |
| 67 | 12 | 5 | 100 | 34.7 | 55.5 | 89.1 | 195.6 | 338.3 | 1002.3 |
| 68 | 12 | 5 | 120 | 34.4 | 55.0 | 88.3 | 184.2 | 310.4 | 906.6 |
| 69 | 12 | 5 | 140 | 49.4 | 75.5 | 144.7 | 318.4 | 533.4 | 1531.7 |
| 70 | 12 | 7 | 60 | 41.6 | 64.5 | 103.4 | 245.4 | 429.9 | 1258.7 |
| 71 | 12 | 7 | 80 | 40.7 | 62.8 | 101.3 | 238.0 | 423.8 | 1279.4 |
| 72 | 12 | 7 | 100 | 42.4 | 66.4 | 111.2 | 256.9 | 431.4 | 1218.2 |
| 73 | 12 | 7 | 120 | 52.3 | 79.4 | 163.6 | 387.1 | 658.4 | 1983.4 |
| 74 | 12 | 7 | 140 | 41.4 | 63.9 | 103.1 | 286.2 | 545.9 | 1733.2 |

Table G.1: Parameters of the hyperparameter scan for the forward model. The numbers $0.25, 0.5, 0.75, 0.9, 0.95$ and $0.99$ denote quantiles of the maximum percentage error. The two rows highlighted in red mark parameter configurations that have not finished training after 4 days. These configurations were ignored. Row number 15 (highlighted in yellow) is the configuration of the best forward model.

# Appendix H

# Best invertible model

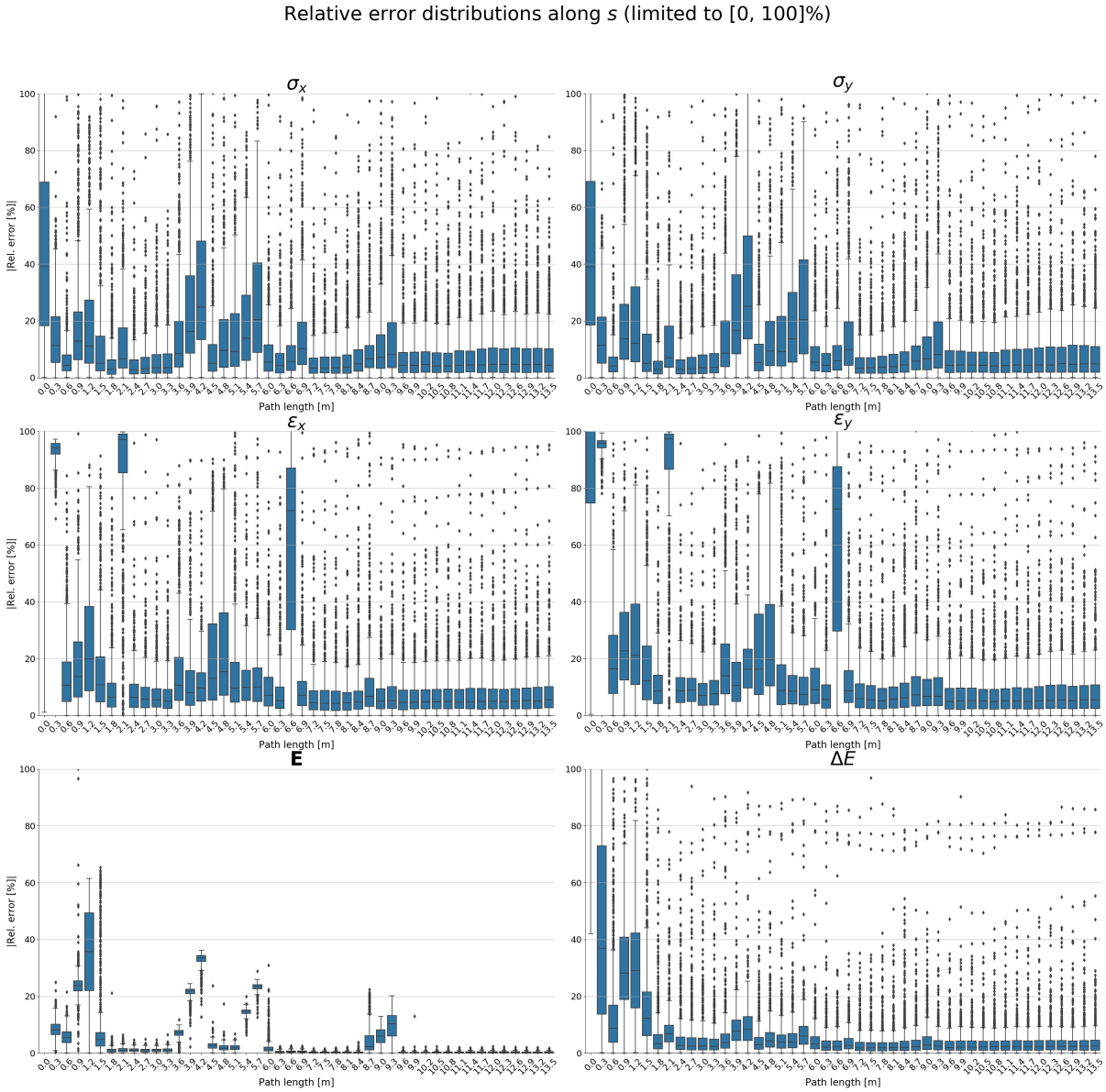



Figure H.1: Forward prediction error of the best invertible model.

# Appendix I

# Subsampling trick

The invertible surrogate model is built because of its ability to sample design variable configurations for a given target vector of quantities of interest, and a position. The goal is to achieve DVAR configurations such that all corresponding quantities of interest are close to their target values. This is measured by the maximum prediction error, see Equ. 2.5. There is a simple trick to reduce this error: Instead of sampling a single DVAR configuration, one can sample $n_t$ of them and take the one with the smallest error. The best DVAR configuration is returned by the sampling function of the model.

Assume that we want to have design variables for $n_s$ target beams. For each of them, $n_{\text{tries}}$ design variable configurations are sampled, among which the best one is selected. The goal of the algorithm below is to formulate the best-of-n trick in terms of matrix operations in order to allow efficient implementations.

$n_s$: number of target samples (beams)
$n_t$: Number of DVAR configs to try
$n_D$: Number of design variables
$n_Q$: Number of quantities of interest
$n_{\text{tries}}$: Number of design variable configurations to assess for each target beam.
$Y_{t,1}, ..., Y_{t,n_s} \in \mathbb{R}^{n_Q}$: Target QOI vectors.

1. Sample $n_{\text{t}}$ design variable configurations:

   $$X_{s,1}, ..., X_{s,n_{\text{tries}}} \in \mathbb{R}^{n_s \times n_D}.$$

   The index $s$ indicates that the matrices are sampled. Each of these matrices $X_{s,k}$ is a data matrix: The $i$-th row contains the DVAR configuration for the $i$-th target vector, and the $j$-th row contains the values of the $j$-th design variable.

   The goal of the subsequent steps is to select the best row from each of these matrices.

2. Evaluate the sampled DVAR configs by forward predicting with the INN:
   (Remove path length column as well!)

   $$Y_{s,1}, ..., Y_{s,n_t} \in \mathbb{R}^{n_s \times n_Q},$$
   $$Y_{s,k} = \hat{\mathbf{f}}^{-1}(\mathbf{x}).$$

3. Calculate the (estimated) relative errors corresponding to each of the tries:

   $$r_{s,k} := \left| \frac{Y_{t,k} - Y_{s,k}}{Y_{t,k}} \right|$$
   $$r_{s,1}, ..., r_{s,n_t} \in \mathbb{R}^{n_s \times n_Q}.$$

   All operations are performed componentwise.

4. Take the maximum (indicated by bars) over all features of the relative errors:

   $$(\bar{\mathbf{r}}_{s,k})_i := \min_j (r_{s,k})_{i,j}$$
   $$\bar{\mathbf{r}}_{s,1}, ..., \bar{\mathbf{r}}_{s,n_t} \in \mathbb{R}^{n_s}.$$

   These vectors of maximum sampling errors can be summarised as a matrix:

   $$R_{ij} := (\bar{\mathbf{r}}_{s,j})_i,$$
   $$R \in \mathbb{R}^{n_s \times n_t}$$

5. For each target QOI vector (i. e. row), find out which of the residuals is the minimum.

   First we define the mask that tells us if row $i$ contains the best DVAR configuration to realise the target beam with index $j$:

   $$m_{ij} := \begin{cases} 1 & \text{if DVAR config } j \text{ has smallest error for the } i\text{-th sample} \\ 0 & \text{else.} \end{cases}$$

   Minimum for the $i$-th row:

   $$(m_r)_i := \min_j R_{ij}.$$

   Replicate this by stacking copies horizontally in order to obtain a 2D matrix:

   $$(M_r)_{ij} := (m_r)_i.$$

   This can be used to obtain the mask matrix $m$ as the result of a componentwise equality test:

   $$\begin{aligned} m_{ij} &= (R_{ij} == (M_r)_{ij}), \\ m &= (R == M_r)_{\text{componentwise}} \in \mathbb{R}^{n_s \times n_t}. \end{aligned}$$

6. As the last step, we can calculate the final matrix of sampled design variables $X_f$. Each row contains the best DVAR configuration out of $n_{\text{tries}}$.

   $$\begin{aligned} (X_f)_{ij} &= \sum_k m_{ik} \cdot (X_{s,k})_{ij} \\ &= \sum_k X_{s,ijk} \cdot (m^T)_{k,i}. \end{aligned}$$

   This expression can be efficiently evaluated with `numpy.einsum()`.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

## Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

---

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

**Title of work** (in block letters):

Accelerating Accelerators:
Fast Surrogate Models for Beam Prediction

**Authored by** (in block letters):
*For papers written by groups the names of all authors are required.*

| **Name(s):** | **First name(s):** |
|---|---|
| Bellotti | Renato |
| | |
| | |
| | |
| | |

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

| **Place, date** | **Signature(s)** |
|---|---|
| Urdorf, 01.06.2020 | Renato Bellotti |
| | |
| | |
| | |
| | |

*For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.*