Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Using Machine Learning Techniques to Identify Magnetic Field Errors in SLS

Master Project

Department of Physics

ETH Zurich

written by

BSc ETH Phys
Güney Erin Tekin

supervised by

Dr. A. Adelmann (ETH)

scientific adviser

Dr. S. Bettoni

July 5, 2025

Abstract

We explored the use of transformer-based neural networks to predict quadrupole field errors within the Swiss Light Source (SLS). The trained networks exhibited remarkable accuracy in simulations, enabling a drastic reduction in β-beat after only one iteration. While these results are promising, the transition from the simulated environment to the real world may be a limiting factor for current networks.

Contents

1 Introduction **1**

2 Background **2**
2.1 LOCO 2
2.2 Machine Learning Techniques 3
2.2.1 Transformers & Attention 4

3 Results **6**
3.1 Results for Quadrupole Errors 6

4 Conclusion & Discussion **9**
4.1 Applicability of the Current Network to Real World 9
4.2 Future Outlook 10

A Appendix **13**
A.1 Additional Figures 13
A.2 Generative AI Usage 14
A.3 Special Thanks 14

Chapter 1

Introduction

The Swiss Light Source (SLS) is a synchrotron light source used to provide high-energy x-rays for chemistry, biology, and physics experiments. In 2024, it was upgraded to improve its operating parameters and reduce its energy use. During this upgrade, the lattice design was changed to have additional higher-order magnets to correct for higher-order optical effects [1]. Inspired by the changes in the lattice, we decided to explore machine learning (ML) methods for various control and identification tasks in SLS. In our previous work, we tested using ML techniques to steer the electron orbit. The AI methods were better at controlling the orbit, requiring fewer iterations, compared to the standard response matrix method [2].

Inspired by the previous success, we decided to research AI-based methods for detecting optical errors. The widely used method for error detection in accelerators is Linear Optics from Closed Orbits (LOCO) [3]. This method is based on the assumption that the response matrix (RM), which represent the orbit response to changes in corrector strengths, responds linearly to changes in certain machine parameters. Thus, making error detection possible by measuring the deviation of the real-world RM from the design RM. In this report, we will first introduce LOCO and ML. Afterwards, we will present our results for using ML to detect errors. Finally, we will discuss our results and the future of ML in accelerator sciences.

Chapter 2

Background

2.1 LOCO

In SLS, the LOCO method is the main method [3] used for optical error detection. LOCO is a common method used for optical error detection in accelerators. It can correct for various errors, such as magnetic field misalignments, magnet rolls, or BPM gain errors. In LOCO, a linear response model $M$is calculated such that:

$$\Delta RM = M\vec{\theta} \tag{2.1}$$

where ΔRM is the change in the response matrix caused by the parameters $\vec{\theta}$. M is calculated in simulation and inverted using the SVD decomposition [3]. Then ΔRM, the deviation of the response matrix from design parameters, is measured and $\vec{\theta}$ is calculated. By applying $-\vec{\theta}$ to the machine, we can bring the machine closer to the design parameters. Or conversely, $\vec{\theta}$ can be applied to the simulation to align the simulation with the real world. Regardless of the direction, this measurement and parameter change process is iterated until convergence.

Another critical part of LOCO is the assumption of linearity between the parameters and changes in the ΔRM. This linearity is questionable in the presence of non-linear effects and higher-order magnets. SLS has, in addition to the 112 quadrupole magnets, 288 sextupole and 264 octupole magnets [4]. The presence of non-linearities may also have an adverse effect on the convergence speed of LOCO.

2.2 Machine Learning Techniques

Machine Learning (ML) is a rapidly accelerating field, but at its core, the main task of any machine learning algorithm is to map a certain set of input vectors to output vectors (cf. [5, 2, 6] to for further reading). In reality, machine learning has been applied to many different fields, from classifying images of bees to creating realistic agents that can handle complex tasks. In this project, we focused on training networks that can predict magnetic field errors in SLS. This task can be approached from many different angles:

- Using conventional measurements and training a network to process the results (i.e., a network to replace LOCO).
- Training agents that can intelligently choose which measurements to run (as an example: choosing kicker(s) for response measurements) and that can predict the magnetic field errors from these measurements.
- Training potentially invertible surrogate models that can emulate the accelerator and using these to indirectly predict the magnetic field errors.

Due to the limited time and scope of this project, we limited ourselves to the first option. The main challenge of the first option is the dimensionality of the inputs and outputs. The natural choice for input is the RM. The SLS employs, excluding spares, 115 correctors and 115 BPMs for both horizontal and vertical axes. Disregarding inter-axis coupling, this yields an RM with dimensions of $115 \times 115 \times 2$. If the RM were to be used as the input vector of a fully-connected neural network, this would result in an input dimension of 26450, causing the first layer in the network to have around 700 million optimization parameters. This requires the use of different architectures to limit the number of weights.

The common techniques for such cases are Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Transformers. CNNs are commonly used for image processing applications and are based on learnable convolution kernels for information extraction. Due to the nature of the convolution process, the positional information of the data is not fully conserved. This is advantageous for image processing tasks, as the location of the object is often irrelevant. In our case, this is a big limitation. The location of the BPM's in relation to the magnets is critical for magnetic field error detection.

On the other hand, RNNs and Transformers allow us to consider a series of vectors as the input. The RM is suited to such interpretations as its columns can be read as a series of kicks and their corresponding responses. RNNs achieve this by processing the vectors in a serial fashion and using the previous outputs of the neurons to enhance the current input. This allows for information transfer between each vector in the series. However, this process introduces significant challenges for gradient computation, primarily due to vanishing or exploding gradients, which severely impede the effective training of RNNs.

Transformers achieve the same purpose with a completely different mechanism: Self-Attention. All of the vectors are input at the same time, and the Attention mechanism allows for the transfer of information between vectors, allowing us to gain insight from the information contained in the whole matrix.

2.2.1 Transformers & Attention

Transformers were first popularized by the famous paper "Attention is all you need" [7] by Google. The full Transformer architecture contains an Encoder and a Decoder, both consisting of Multi-Head Attention blocks. In the natural language processing context, the Encoder is used to analyze the input sentence, while the Decoder is used to predict the next word. Since our application is only a predictive task, as opposed to a generative one, we only used the Encoder part of the architecture.

Before the Encoder, we use a learnable linear transformation to increase the dimension of the input vector. The Encoder itself is made out of a Multi-head Attention block connected to a feed-forward network for further processing. The output vectors of the Encoder were averaged into a single vector through Adaptive Average Pooling, a method where the pooling parameters (such as stride and kernel size) are learned during training. Additional neural layers with Batch Normalization and Dropout were utilized to predict Quadrupole errors from the resulting vector. We found that this architecture achieves the ideal balance between the number of parameters and performance.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.2}$$

Equation 2.2 shows the simplified equation for computing attention. The Q, K, V matrices consist of the input vectors packed together into a matrix such that QK^T is the dot-product of the vectors with each other. When the attention is done using the multi-head method, the matrices are first projected to lower-dimensional vector spaces using learnable transformations. Then, Attention is calculated on each vector space and afterward joined together:

$$MultiHead(Q, K, V) = Concatenate(head_1, ..., head_n)W^O \tag{2.3}$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.4}$$

where W^O is a learnable transformation and $(W_i^j)_{i\in\{1,...,n\}}$ $\forall j \in \{Q, K, V\}$ are also sets of learnable transformations. Multi-Head Attention offers performance benefits over the single-head version [7].

In our network, an initial linear transformation increased the input dimension from 115 to 230. We then employed four sequential transformers, each with two heads, for the results presented in this paper, although models with three transformers demonstrated identical performance. We also used a linear transformation before the transformer to increase the input dimension from 115 to 230. Figure 2.1 shows an overview of the network architecture.

The Transformer blocks had in total ca. 11 million parameters. The 3 MLP layers for the quadrupole error prediction had 65432 parameters. To optimize these parameters, we used the AdamW optimizer, a derivative of the ADAM optimizer, for the training. AdamW provides inherent weight regularization capabilities and is more resilient to changes in the training parameters [8].

During training, we also experimented with various training schedules to improve the training outcome & speed. We saw some success using a WarmUp schedule [9], but training with low learning rates was the better method. We trained our networks on the Merlin/Gwendolen

cluster using an A100 GPU and the PyTorch framework [10]. The training of the transformer networks took 2-3 hours. The prediction speed of the classical PyTorch implementation is acceptable for our use case, but further performance could be gained by using the FlashAttention implementation [11].

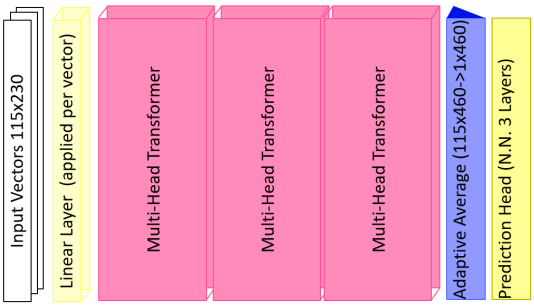


Figure 2.1: Schematic Overview of the Network Architecture. Each Multi-Head Transformer consists of two Transformers running in parallel, using a different projection of the input vector.

Chapter 3

Results

3.1 Results for Quadrupole Errors

For the quadrupole case, we considered convolutional networks before deciding on the Transformer architecture. We considered two convolutional architectures, one where each column of the Response Matrix was considered a channel, and one where there was only a single channel. The single-channel version had serious performance problems. This was most probably a limitation caused by the translation invariance of the convolution operation. These kinds of networks were unable to capture enough information from the response matrix.

The other convolutional architecture was more promising, but the high number of channels caused the network to be too large, causing problems during training. We considered convolution layers, where the number of channels increased between each layer, and since the number of weights per convolutional layer is $C_{in} \times H \times W \times C_{out}$, the number of weights grew too large. Due to this, we decided to move away from the convolutional architecture, although there were some successful projects using CNNs to detect errors [12, 13].

When we moved to the transformer architecture, the performance improved, and we trained two networks with different error levels. We found that a single network gained no additional benefit from iterative application, unlike LOCO. Therefore, we trained two networks that can be applied sequentially to the machine to improve the performance, with comparable effort to LOCO. Figure 3.2 shows the performance of both networks.

Both networks were trained on uniform distributions with ranges $[-5 \times 10^{-3}, 5 \times 10^{-3}]$ and $[-5 \times 10^{-2}, 5 \times 10^{-2}]$,; and the mean absolute errors (MAE) on the validation datasets are 3.95×10^{-4} and 5.35×10^{-3} respectively. The MAE increases slightly more than the order of magnitude increase in the uniform distribution range. This is caused by the network struggling to learn the larger distribution. Since the increase is minimal, we did not take steps to improve the performance.

We also measured the β-beat achieved by the application of both networks. β-beat is a measure of the deviation of machine β-parameters from the design β-parameters. It is defined as $(\beta_{design} - \beta_{machine})/\beta_{design}$ along the length of the machine. Figures 3.1 and A.1 show the decrease in β-beat by the networks. The coarse network is responsible for an order-of-magnitude decrease in β-beat in both axes in a single iteration. In contrast, LOCO is estimated to require

3-4 iterations to reach a similar level of β-beat.

In Figure 3.1, the fine network demonstrates a disproportionate effect on the y-axis, a random phenomenon dependent on the specific random seed. Figure A.1 then illustrates the network performance when additional errors are introduced in the sextupoles and quadrupoles. Here, we observe that the coarse network remains largely unaffected by these additional errors, while the fine network struggles considerably.

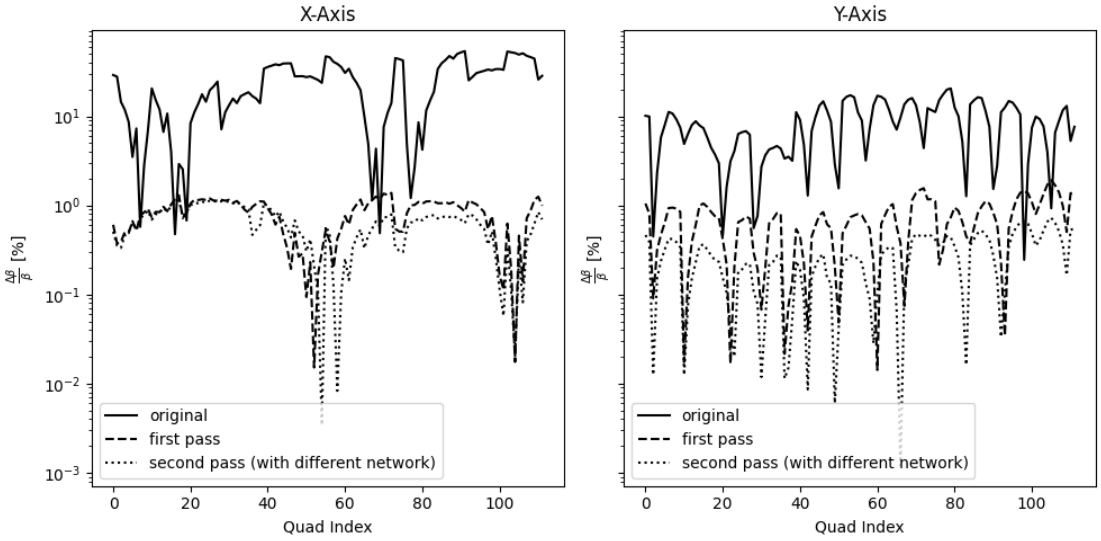
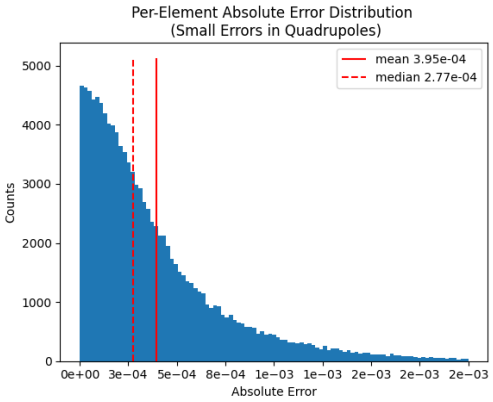
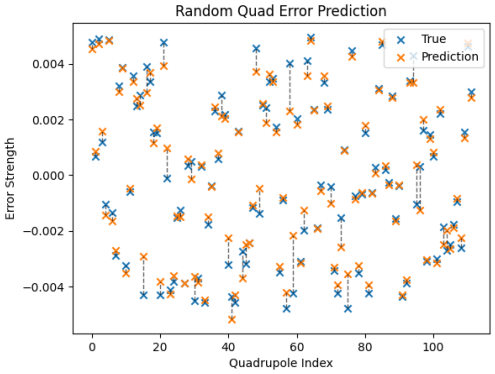


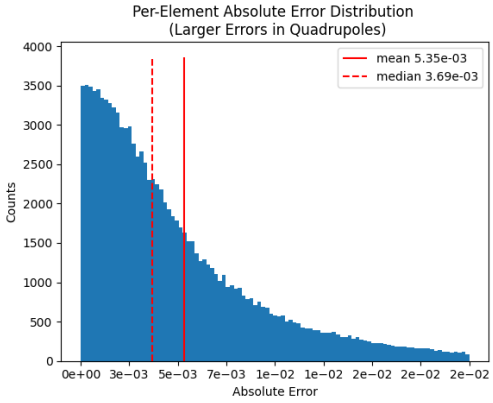
Figure 3.1: Reduction in β-beat at the quadrupole entrances. Calculated in simulation with only quadrupole errors present. The two networks are applied sequentially in the simulation.



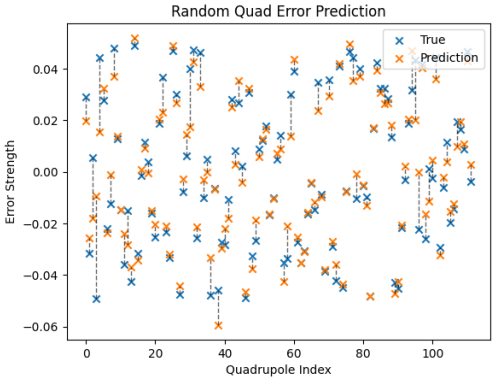
(a) AE Distribution for Fine-Network



(b) Single Prediction for Fine-Network



(c) AE Distribution for Coarse-Network



(d) Single Prediction for Coarse-Network

Figure 3.2: Performance Characteristics of Both Networks. Figures 3.2(b) and 3.2(d) show the network results for a single element in the validation dataset. In contrast, Figures 3.2(a) and 3.2(c) display the overall distribution of per-element absolute errors across the validation dataset.

Chapter 4

Conclusion & Discussion

We have demonstrated that transformer-based models constitute a viable approach for detecting optics errors in SLS. The coarse-network alone can reduce the β-beta to approximately 1% in a single iteration. This represents a substantial improvement over classical LOCO, which is estimated to require 4-5 iterations to achieve a similar performance level.

The ML approach benefits significantly from the presence of higher-order magnets in SLS. Unlike the purely linear LOCO method, ML-based approaches can learn and adapt to these system nonlinearities, which are crucial for performance in such complex systems. Furthermore, the use of transformers represents an advancement over other ML architectures. For instance, a similar study [13] found that convolutional architectures are limited, requiring multiple iterations to achieve comparable performance.

However, ML-based approaches also present additional challenges when applied in real-world scenarios. Specifically, network performance is highly dependent on a close match between the simulation and the physical machine; any deviations can lead to unexpected network outputs.

4.1 Applicability of the Current Network to Real World

During the writing of this report, we found a mistake in the function we used to generate the response matrix. This function was used extensively and consistently across scripts to train and test the network. The main mistake was that vertical kicks were being applied using the horizontal kicker. Therefore, the kick location was 45 millimeters off from the correct location. Since there are no other elements between the horizontal and vertical kickers, the mistake should not affect the viability of the method, but the current networks suffer heavily when tested with the correct kick location (cf. Figure A.2).

Such mistakes are typical when simulation-trained ML networks are applied in the real world. The application of such networks in SLS might be especially challenging, since the time allocated for development on SLS is limited. This is especially hard for direct ML methods that use real-world measurements to predict machine parameters, as any deviation from the simulation might fall outside of the network's capabilities. To preempt this, the networks would need to be trained with a more robust dataset, containing data points from a wider distribution of machine states.

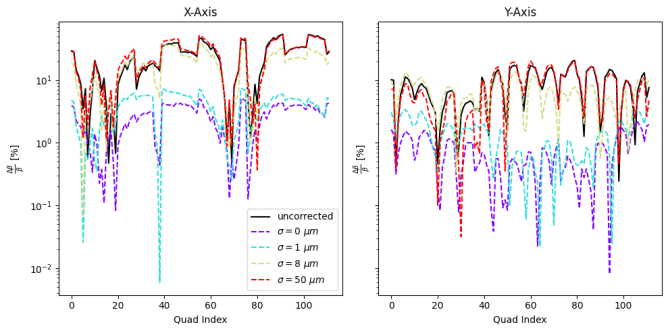


Figure 4.1: Noise sensitivity of the coarse network.

Another limitation of the current architecture is its susceptibility to noise. We tested the beta-beat performance with additional Gaussian noise on the RM measurement. The network performance decreased when noise with $\sigma = 1\mu m$ was added, with the performance deteriorating further with more noise up to ca. $8\mu m$. For further noise levels, the network increases the beta-beat (cf. Figure 4.1). The resilience to noise can be increased by adding additional MLP layers or a denoising autoencoder before the transformer. Currently, the RM is directly fed to the Transformer after a linear transformation.

4.2 Future Outlook

With further research and refinement, such models could be applied for error detection in SLS, but other concepts mentioned in Section 2.2 might provide better alternatives for such tasks. Especially, agent-like models could provide interesting alternatives. Such a model would be trained on BPM measurements as inputs and kickers as control outputs. The network could then discover novel methods to detect errors. Such a network was trained for quantum error correction, and it found novel measurement strategies to extend the lifetime of the qubit state [14]. For accelerators, the main challenges for such agent-like approaches are slow simulation speeds and the high number of inputs and outputs.

Another possibility is to forgo the error detection and train agents that drive the SLS to a desired condition. As an example, such an agent could use continuous BPM measurements to ensure that SLS stays close to desired operating parameters. The agent would always be online and would adjust the magnet strengths as necessary. Potentially, such agents could improve SLS performance in adverse or unexpected conditions, as they could adjust the machine parameters in real-time to optimize for desired properties.

Bibliography

[1] A. Streun, "Sls 2.0, the upgrade of the swiss light source," in *13th Int. Particle Acc. Conf.*, JACoW Publishing, 2022.

[2] S. Bettoni, J. Kallestrup, G. E. Tekin, M. Böge, and R. Boiger, "Machine learning for orbit steering in the presence of nonlinearities," *Journal of Synchrotron Radiation*, vol. 32, pp. 609–621, 5 2025.

[3] J. Safranek, "Experimental determination of storage ring optics using orbit response measurements," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 388, pp. 27–36, 3 1997.

[4] H. Braun, T. Garvey, M. Jörg, A. Ashton, P. Willmott, R. Kobler, H. Braun, T. Garvey, D. Reinhard, A. S. S. Ring, M. Aiba, S. Bettoni, M. Böge, P. Braschoss, J. Buchmann, C. Calzolaio, A. Citterio, M. Dehler, S. Dordevic, R. Fortunati, A. Gabard, N. Gaiffi, R. Ganter, M. Gaspar, C. S. G. Hellmann, R. Ischebeck, H. Jöhri, R. Kalt, B. Keil, P. Lerch, F. Marcellini, G. Montenero, M. Negrazus, C. Ozkan-Loch, M. Paraliev, M. Pedrozzi, B. Riemann, M. Roggli, B. Ronner, C. Rosenberg, N. Samadi, S. Sanfilippo, V. Schlott, T. Schmidt, L. Schulz, S. Sidorov, H. Siebold, D. Stephan, L. Stingelin, A. Streun, W. Tron, J. V. Comamala, V. Vrankovic, M. Wurm, A. Zandonella, R. Zennaro, E. Z. P. science, M. Brügger, M. Calvi, S. Danner, U. Flechsig, W. Glettig, L. Huber, D. Just, R. Kinjo, X. Liang, C. Pradervand, B. Röster, M. Schmidt, P. Willmott, K. Zhang, L. Pedrazzi, W. Rendler, C. Schmid, R. Sieber, A. Weber, J. Wickström, and E. Zehnder, "Sls 2.0 storage ring technical design report," tech. rep., PSI, 11 2021.

[5] S. Russell and P. Norvig, *Artificial Intelligence, Global Edition A Modern Approach*. Pearson Deutschland, 2021.

[6] G. E. Tekin, "Deblurring of images with the richardson-lucy-network," 6 2024.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[8] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 11 2017.

[9] A. Kosson, B. Messmer, and M. Jaggi, "Analyzing & reducing the need for learning rate warmup in gpt training," 10 2024.

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[11] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," 5 2022.

[12] E. Fol, J. M. C. de Portugal, G. Franchetti, and R. Tomás, "Optics corrections using machine learning in the lhc," in *Proc. 10th International Particle Accelerator Conference (IPAC'19), Melbourne, Australia, 19-24 May 2019*, pp. 3990–3993, JACoW Publishing, 2019.

[13] R. Li, B. Jiang, Q. Zhang, Z. Zhao, C. Li, and K. Wang, "Linear optics calibration in a storage ring based on machine learning," *Applied Sciences*, vol. 13, p. 8034, 2023.

[14] T. Fösel, P. Tighineanu, T. Weiss, and F. Marquardt, "Reinforcement learning with neural networks for quantum feedback," *Phys. Rev. X*, vol. 8, p. 031084, Sep 2018.

Appendix A

Appendix

A.1 Additional Figures

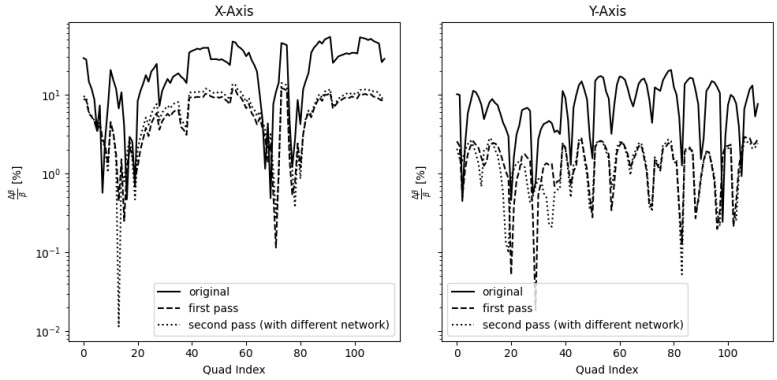


Figure A.1: β-beat performance of the networks. This is a realistic scenario, with errors in sextupole and quadrupole magnets also present.

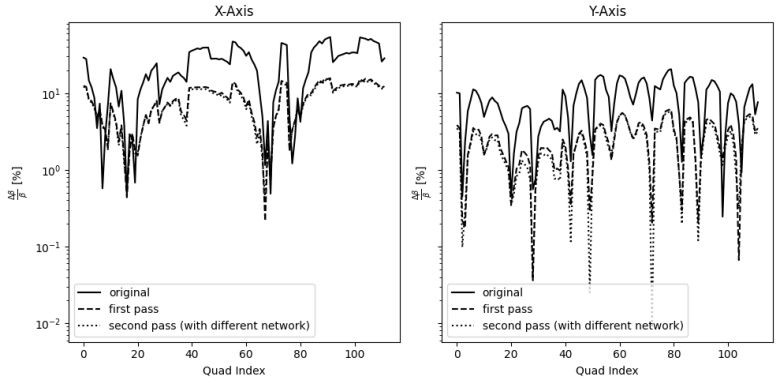


Figure A.2: Same as Figure 3.1, but the correct function for the RM was used during testing.

A.2 Generative AI Usage

Generative LLMs especially (Google Gemini 2.0 Flash and Gemini 2.5 Flash Experimental) were used to improve the grammar, writing, style, and LaTeX-typing of this report. They were not used to generate full sentences or factual information. Furthermore, various LLMs (OpenAI 4o & 4o-mini; Google Gemini 1.5 Flash Gemini 2.0 Pro; Anthropic Claude Sonnet 3.5) were used to generate parts of the code. The LLMs were used to write helper functions for the ML training. These include: functions for saving and loading models, example implementations of AI architectures, data management (loading, moving to gpu etc.), logging functions, and the rough training loop.

A.3 Special Thanks

We thank the following persons for their help with the project:

- Jonas Kalestrup, for providing the SLS ".mat" file for PyAT.
- Simona Bettoni, for her help with the project.
- Efe Ongan, for his discussions regarding different ML architectures.