A Machine Learning Approach for Particle Accelerator Errant Beam Prediction using Spatial Phase Deviation

Yigit A. Yucesan[a,*], Willem Blokland[a], Pradeep Ramuhalli[a], Alexander Zhukov[a], Charles Peters[a], David Brown[a] and Cary Long[a]

[a]*Oak Ridge National Laboratory, Oak Ridge, Tennessee 37830, USA*

ARTICLE INFO

Keywords:
Particle accelerators
Data-driven fault detection
Neural architecture search
Sustainable machine learning framework

ABSTRACT

Particle accelerators are extremely complex systems that are expected to operate on high availability. Predicting impending failures only by utilizing data collected from diagnostic equipment already on board can help operators to avoid installing expensive sensors, unscheduled downtime and associated costs. For this purpose we explore the predictive power of Machine Learning algorithms to detect faulty beams prior to the failure. In this study, we propose a Machine Learning approach to model mapping from a pair of sensors located across the accelerator. While the model is trained to represent normal operation, we evaluate the predictive performance on known faulty beam pulses. We also investigate the model performance on unseen data through k-fold cross-validation. Then we recap the analysis with a neural architecture search and hyperparameter optimization study to fine tune our initial model. This paper will also introduce a sustainable framework that can standardize Machine Learning workflow applied to particle accelerators.

1. Introduction

The Spallation Neutron Source (SNS) of the Oak Ridge National Laboratory (ORNL) is the world's most powerful pulsed proton accelerator to date. The accelerator provides high-intensity proton beam to a liquid mercury target for neutron production to be used for various experiments in research fields such as drugs [1], batteries [2], and engines [3]. In the design phase of the accelerator the availability requirement was set to be at least 90%, with the accelerator ultimately achieving 95% during operation [4].

Several factors contribute to the requirement for continuous operation. The main drawback of an unexpected beam failure in any type of system is the disrupted planned schedule for operation. Unscheduled maintenance ultimately leads to loss of production and postponing the experiments that take months and in some cases years to prepare for. One particular unexpected beam failure at the SNS has been labeled "errant beam". Errant beam is a beam pulse that does not transport properly through the SNS linac. This errant beam pulse has the potential to degrade the expensive equipment located across the linear accelerator, which can further increase the downtime, reduce accelerator capability, and lead to additional maintenance costs. Considering all these negative impacts of an errant beam, it becomes a necessity to develop a method to detect the incoming anomaly with far enough lead time to allow the existing machine protection system or the operators to take action.

Even though state-of-the-art condition monitoring systems can help with detecting anomalies for some industrial systems and subsystems by collecting and processing various types of diagnostic data (i.e. vibration [5] and temperature [6]), these sophisticated systems tend to be very expensive to install. Especially for cutting edge systems as complex as SNS, it is near impossible to track the condition of each component that ensure reliable operation, which are in the order of thousands. Therefore, utilizing data obtained from existing instrumentation to build diagnostic models becomes a more attractive option for the operators. Another exclusive challenge of developing anomaly detection models for a linear accelerator comes from the fact that the system is unique; in other words, there are no fleet of similar particle accelerators operating simultaneously from which failure data can be collected. Therefore, one can only rely on the historical data obtained from the same individual system to build a diagnostic model.

Machine Learning (ML) holds significant promise for enhancing the operations and availability of particle accelerators [7–9]. With the increasing complexity and demands of modern accelerator facilities, ML techniques offer

*Corresponding author
yucesanya@ornl.gov (Y.A. Yucesan); blokland@ornl.gov (W. Blokland); ramuhallip@ornl.gov (P. Ramuhalli); zhukovap@ornl.gov (A. Zhukov); peterscc@ornl.gov (C. Peters); browndl1@ornl.gov (D. Brown); longcd@ornl.gov (C. Long)
ORCID(s): 0000-0002-2627-0821 (Y.A. Yucesan); 0000-0002-2385-6894 (W. Blokland)

valuable tools for optimizing and automating accelerator operations, thereby improving efficiency and availability. By analyzing large volumes of operational data, ML algorithms can identify patterns, correlations, and anomalies that may not be readily apparent to human operators. This enables the development of predictive models that can forecast potential failures. Furthermore, ML-based anomaly detection can proactively identify and address performance issues, minimizing downtime and maximizing accelerator availability. Overall, the application of ML in particle accelerator operations has the potential to enhance reliability, efficiency, and performance, ultimately contributing to advancements in accelerator science and enabling new frontiers in high-energy physics research.

In this study, we investigate the use of historical diagnostic beam data combined with a Neural Network to predict anomalous beam behavior in the SNS Linac. We focus on two specific Beam Position Monitors (BPMs) located at different points along the tunnel and develop a mapping model based on normal beam phase data obtained during regular operations. Subsequently, we utilize this model to infer the behavior of pulses leading to an errant beam, comparing the downstream signal construction and performing binary classification to identify upcoming faulty beam pulses. To ensure the robustness of our approach, we conduct cross-validation analysis, examining the impact of data shuffling, and employ neural architecture search and hyperparameter optimization techniques to fine-tune the model. Moreover, we propose a scalable and sustainable ML framework that includes a user-friendly graphical interface for initiating model training, a high-performance computer capable of parallel GPU training, and a browser-based experiment tracking server to enhance and standardize ML workflow.

The outline of this paper is designed as follows. In Section 2 we scan the literature on errant beam detection methods for particle accelerators and other ML applications in this domain. In Section 3 we describe the data source, acquisition process, and pre-processing applied. Section 4 provides the ML architecture of our choice for this use case and classification setup. In Section 5 we introduce the ML framework for streamlined implementation of aforementioned methods. Finally Section 6 presents the performance result of our approach and Section 7 concludes this article with final remarks and future directions. There are two appendices at the end of this paper, where Appendix A lays out the cross-validation technique used in the paper, and Appendix B provides details on model fine tuning approach.

2. Literature Review

Errant beam prediction is a critical task in particle accelerators to anticipate and mitigate potential hazards caused by beam misbehavior. Several studies have been conducted to develop accurate and efficient prediction models for errant beams in these complex systems. Blokland et al. [7] propose a Siamese Neural Network (SNN) model that compares normal and anomalous beam pulses to detect anomalies and provide uncertainty estimates, thereby improving the prediction of equipment failures without the need for additional monitoring devices. By utilizing existing diagnostic data from a beam monitoring device of the SNS, the proposed Siamese Neural Network model proves to identify anomalies and provide uncertainty estimates. This approach has the potential to improve accelerator availability and minimize downtimes caused by equipment failures. Motivated by the need for improved anomaly prediction in particle accelerators, Li et al. [8] explore various forecasting methods for short-term prediction. These methods, including machine learning models like Random Forest, Lasso, CNNs, and Autoencoders, have shown promising results in accurately predicting anomalies related to beam loss trips, interlocks, power supply faults, and cavity faults. The key takeaway is that while these models have performed well on historical data and offer the potential for real-time implementation, challenges remain in addressing false positives, handling longer-term forecasts, and enhancing interpretability, requiring further research in the field. Rescic et al. [9] focus on the utilization of popular machine learning algorithms such as k-Nearest Neighbors (kNN) and Decision Trees (DT) for tasks such as pulse classification and machine trip prediction. The authors present a comprehensive evaluation of these algorithms. The findings demonstrate the potential of machine learning in enhancing the reliability of particle accelerators and provide valuable insights into the performance and effectiveness of these algorithms in this domain.

Machine learning has emerged as a promising tool not only for anomaly prediction but also for various applications in particle accelerators, offering opportunities for improved performance and operational efficiency. Edelen et al. [10] present a comprehensive review of these opportunities, highlighting the potential of machine learning techniques in accelerator physics and control systems. Mishra et al. [11] focus on uncertainty quantification in deep learning models for particle accelerator applications, addressing the challenge of assessing and mitigating uncertainties associated with predictive models. Scheinker et al. [12] propose an adaptive approach to machine learning for compact particle accelerators, enabling real-time adaptation and optimization of accelerator settings based on machine learning feedback. Furthermore, Emma et al. [13] introduce a machine learning-based longitudinal phase space prediction method for

particle accelerators, demonstrating the ability to accurately forecast accelerator behavior and optimize beam parameters. These studies collectively emphasize the potential of machine learning in improving various aspects of particle accelerators, including control, optimization, uncertainty quantification, and predictive modeling.

There are also other efforts at SNS to apply Machine Learning to improve system availability and design, including time series anomaly detection in power electronics signals [14], uncertainty quantification of the physical model parameters for the liquid mercury target [15], and cryogenic moderator system parameter optimization [16].

3. Data and System Description

The SNS consists of three main parts, a linear accelerator to generate the H^- beam at the proper energy, an accumulation ring that stacks the output of the accelerator to allow all the energy of a pulse to be delivered to the target in less than a 1 μs, and a liquid mercury target where neutrons are produced. The H^- beam at SNS has several layers of structure to it. At the highest level the accelerator operates at 60 Hz to deliver beam to the target. One such 60 Hz event is deemed a macro-pulse, and is nominally 1 ms in duration. Each macro-pulse consists of approximately 1000 mini-pulses, each nominally 1 μs, at approximately a 70% duty cycle. The time for a mini-pulse to make one revolution around the ring is also nominally 1 μs, i.e. one turn around the ring, because of this mini-pulses are often referred to as *turns*. A result of this timing is each subsequent mini-pulse stacks in the ring exactly in time with the previous mini-pulses, resulting at the end of a macro-pulse with a single pulse stored in the ring that is nominally 1000 times stronger than a single mini-pulse, which is then sent to the target. Each mini-pulse is further composed of micro-pulses derived from the 402.5 MHz RF used by the Radio Frequency Quadrupole (RFQ) at the beginning of the linear accelerator.

This section will cover the data sources utilized in this work, applied pre-processing, and metrics for success.

3.1. Data Source

Beam Position Monitors (BPM) are installed along the accelerator from start to end. The SNS BPMs measure horizontal and vertical position, beam current and beam phase. It is understood that most errant beam pulses are caused by malfunctions in the accelerating structures at the beginning of the linac. Due to this the phase data are especially interesting as it directly reflects the performance of the accelerating cavities and is expected to have early indications of a failing acceleration process. The BPM systems provide a waveform of the macro-pulse with each sample representing the position, intensity, and phase of a mini-pulse. When a beam is aborted, up to 300 macro-pulses are saved. In addition, macro-pulses are saved on a regular basis to also acquire normal pulses.

For the analysis carried out in this study, we utilized a dataset consisting pulses collected from March 2021 to February 2023, covering approximately two years of operation. In this dataset, there are 6500 normal pulses and 1150 'faulty' pulses. Note that we define a *faulty* pulse as the pulse just before the errant beam pulse. The chosen *upstream* and *downstream* BPMs are located at the beginning and ending of High Energy Beam Transport (HEBT) section of the linac. Fig. 1 illustrates the raw phase data for upstream, downstream, and the difference for both normal and faulty beam instances. This illustration makes it clear that it is almost impossible for humans to make a distinction and draw conclusions from raw data, hence the need for a data-driven approach.

3.2. Data Pre-processing

Beam phase is subjected to modification during the operations of the accelerator. As the accelerating cavities are tuned, the phases measured by the BPMs change. This discrepancy can cause difficulties for training of data-driven models (i.e. prioritizing large values to minimize loss function). There are several techniques to normalize data to prevent this. In this study, we apply local standardization of the phase data per each pulse:

$$\phi_{\mathbf{i}}^{\mathbf{S}} = \frac{\phi_{\mathbf{i}} - \bar{\phi}_{\mathbf{i}}}{\sigma_{\mathbf{i}}} \tag{1}$$

where $\phi_{\mathbf{i}}$ is the phase signal, $\bar{\phi}_{\mathbf{i}}$ is the mean of the phase, $\sigma_{\mathbf{i}}$ is the standard deviation of the phase, and $\phi_{\mathbf{i}}^{\mathbf{S}}$ is the standardized phase for signal i. Figure 2 illustrates data before and after standardization. It can be clearly observed from the figure that before pre-processing, phase signals obtained from same BPM sensor can vary drastically, however standardization allow us to bring the pulses from different time stamps into same scale.

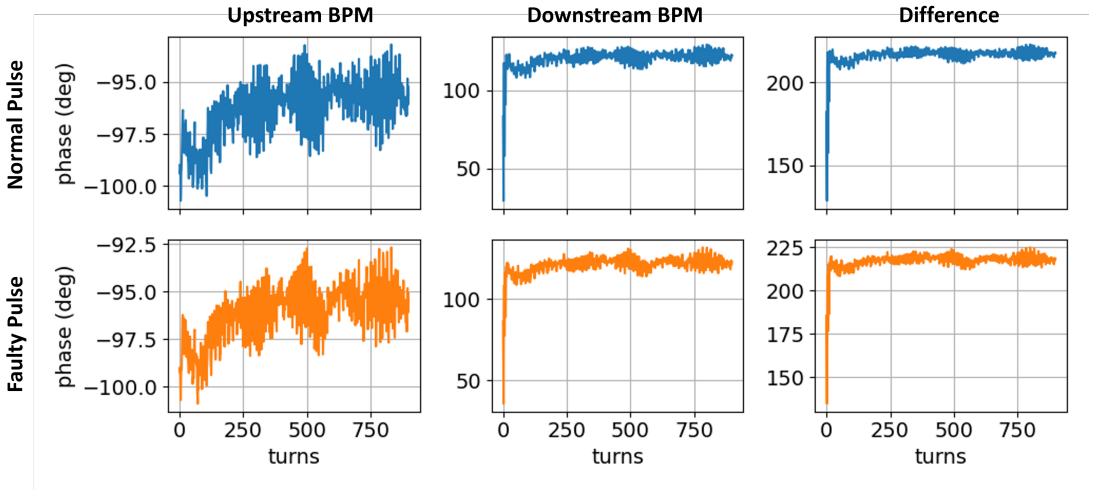
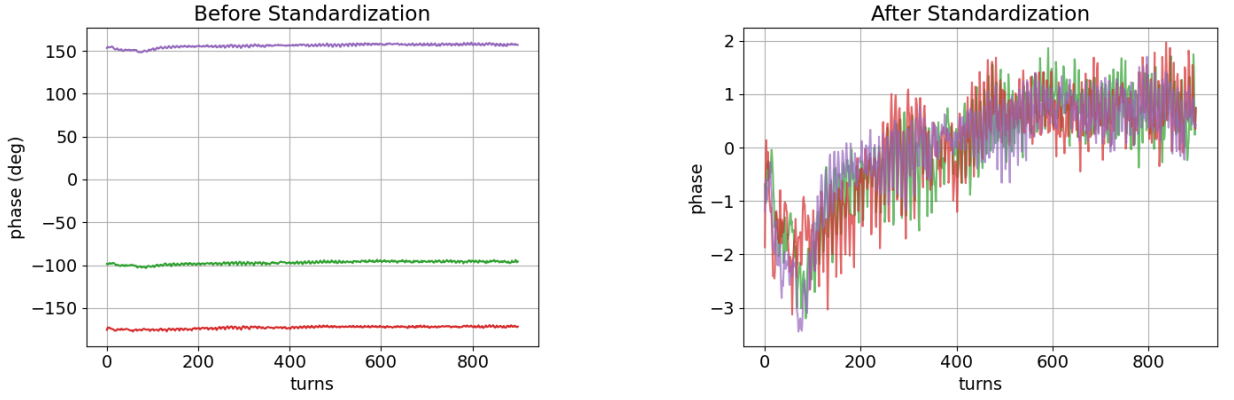


Figure 1: Illustration of phase data for upstream and downstream BPMs and difference, for normal and faulty beams.



(a) Raw BPM phase signals.

(b) Standardized BPM phase signals.

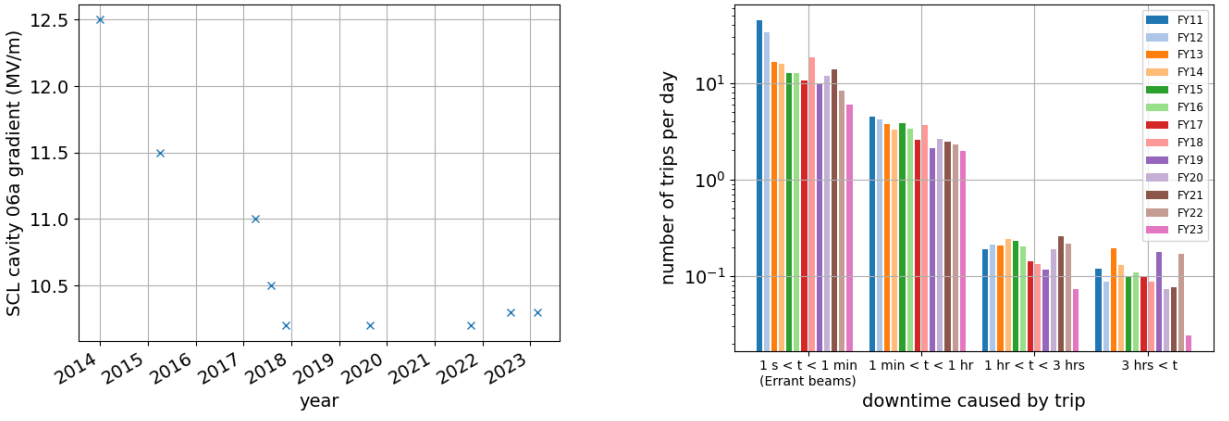
Figure 2: Historical phase data collected from the upstream BPM. Three random timestamps are sampled from the dataset to illustrate the variation of the phase for a single BPM, and we can locally standardize signals to scale. Note that signals from the same timestamp are color coded in both subfigures.

3.3. Errant Beam Induced Downtime

Most errant beam pulses result in total beam loss within the Superconducting Cavity Linac (SCL). While most errant beam induced trips can be recovered to full beam power within a minute this beam loss can induce longer downtimes. In some cases the beam loss trips SCL RF cavities, which may require the reduction of RF cavity gradient in order to recover the cavity to high availability. Figure 3a shows approximately 25% gradient reduction for a single SCL RF cavity over the years as a preventive action against errant beams. Even though the gradient reduction helps with increasing the availability, further reduction could result in lower linac beam energy output which would translate to reduced beam power on target. On the other hand, if the frequency or amount of errant beam could be reduced it may be possible to increase SCL RF gradients for the cavities most impacted by errant beam loss, which would increase the output energy of the linac or provide additional energy margin.

Figure 3b lays out the system downtime statistics over different fiscal years (FY), binned by the length of the trip. We categorize trips that cause downtimes more than a second and less than a minute as errant beams. Errant beams are observed significantly more than other trips causing longer downtimes (i.e. more than a minute and less than an hour, more than an hour and less than three hours, and more than three hours). Currently the frequency of errant beam pulses

is approximately 11 per day, and most errant beam pulses result in very short downtimes on the order of 40 seconds to recover to full power beam operations (see first bin in Figure 3b). For 60 Hz beam pulse operation (5,184,000 possible daily pulses) approximately 0.5% are lost due to errant beam pulses.



(a) SCL cavity gradient over years.

(b) SNS yearly downtime statistics per trip severity.

Figure 3: Data and operational downtime statistics showing the effect of errant beam over system downtime.

Similar to an errant beam pulse wrongly aborting a good pulse, a false positive, has as consequence a reduction in the integrated beam power on target and thus less neutrons are sent to the experimenters. As integrated beam power is a tracked performance criteria, it is important that false positives do not in a significant way affect the integrated beam power. Therefore, our goal in agreement with Operations is to keep false positive rate below 0.5%, to match the current daily loss of integrated beam power from errant beam. Ultimately, the success for the Machine Learning model is defined as maximizing the true positive rate (i.e. percentage of errant beams correctly predicted) while keeping the false positive rate below 0.5% (i.e. percentage of normal pulses falsely flagged as errant beam by the model).

4. Neural Network Design and Training

Our goal is to build a mapping model that will take phase signal from the upstream BPM and accurately predict the downstream BPM phase signal. While we expect a predictable mapping from the upstream BPM to downstream BPM during normal operation, the model should fail to reconstruct the downstream signal in the presence of imminent failure. Note that, in this study, *faulty* signals are essentially those pulses occurring just before (one pulse before) an errant beam pulse, and presumably contain a signature of impending failure.

For the purpose of predicting the downstream BPM signal from the upstream BPM signal, we propose to use a multi-layered perceptron (MLP) neural network, with multiple hidden layers and dropouts in between to prevent overfitting. An MLP is composed of multiple fully connected dense layers between the input and output layers, and are useful architectures for both classification and regression problems, including the regression mapping problem in the present case. One can also simply attempt to use an MLP for direct classification over a single BPM data. However, we intuitively assume signature of a fault would have an adverse affect on the acceleration process. Therefore, we hypothesize that utilizing multiple BPM sensors located across the Linac would provide more information rather than single source of data. Table 1 summarizes the architecture of MLP model where *relu* is the Rectified Linear Unit activation.

After training the MLP with a dataset consisting of signals from normal operation, we use the trained model to predict labeled faulty (signal leading to errant beam) and normal (not included in training dataset) signals. Then we compare the prediction with the actual downstream signal using the Root Mean Squared Error (RMSE) metric:

$$\mathrm{RMSE}_i = \sqrt{\frac{\sum_{j=0}^{L}(\phi_{i,j}^{S} - \hat{\phi}_{i,j}^{S})^2}{L}} \tag{2}$$

Table 1
Architecture of MLP. Total number of trainable parameters: 630,900

Layer	Dense 1	Dropout 1	Dense 2	Dropout 2	Output Dense
Nodes/Rate	300	0.095	300	0.095	900
Activation	relu	-	relu	-	linear
# parameters	270,300	0	90,300	0	270,300

where $\phi^S_{i,j}$ is the actual standardized phase signal, $\hat{\phi}^S_{i,j}$ is the predicted standardized phase signal, L is the length of the signal for the signal i. Subsequent to evaluating the RMSE of both faulty and normal signals, we can apply a threshold value to calculate the True Positive Rate (TPR) (percentage of faulty signals classified as faulty) and the False Positive Rate (FPR) (percentage of normal signals classified as faulty). By changing the threshold, we build a Receiver Operating Characteristic (ROC) curve, which is an essential performance indicator for binary classification. Figure 4 depicts the designed training, inference, and post-processing scheme as a flowchart.

We train the MLP with 500 epochs, with an initial learning rate of 0.0001, along with an adaptive learning rate callback that factors down the learning rate if validation loss is not improved for certain amount of epochs. The reason for the large number of epochs used is to ensure the optimizer can converge to global optima. We also use the popular RMSprop[1] optimizer that proves decent convergence across different networks.

5. Machine Learning Framework

Any type of Machine Learning application often require exhausting number of experiments, trained and tested on various types and vast amounts of data, optimizing models with large number of parameters. It is very crucial to build a sustainable framework for training, testing, and optimizing ML models using reliable pipelines and standardized processes. For this purpose we designed and established the workflow depicted in Figure 5.

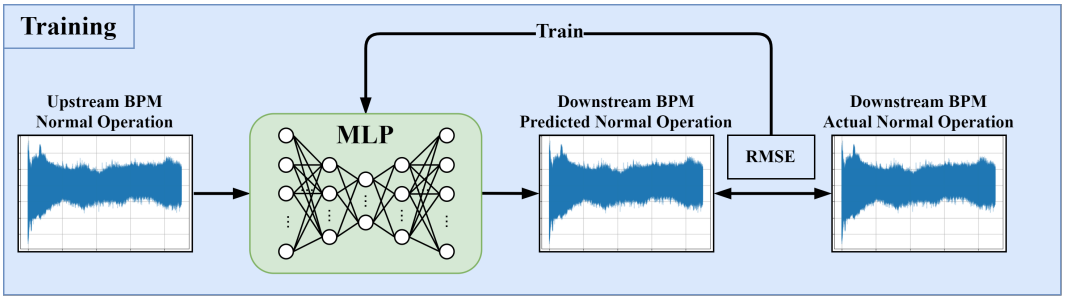
First, we build a user-friendly graphical interface for data selection, model selection, and training or hyperparameter optimization settings. When the user initiates model training, modified source codes are deployed to high-performance GPU for efficient training, while the cluster extracts the requested data from file server. While the user can track the progress from the interface, relevant model parameters and metrics are simultaneously logged to the cloud server where the MLflow [17] tracking server resides. MLflow is a Python package that enables logging and tracking ML experiments and runs via browser based interface, providing consolidated ML ecosystem with customizable artifacts. After a run is completed, users can evaluate model performance, compare different models, and visualize an overview of an HPO/NAS experiment.

6. Results

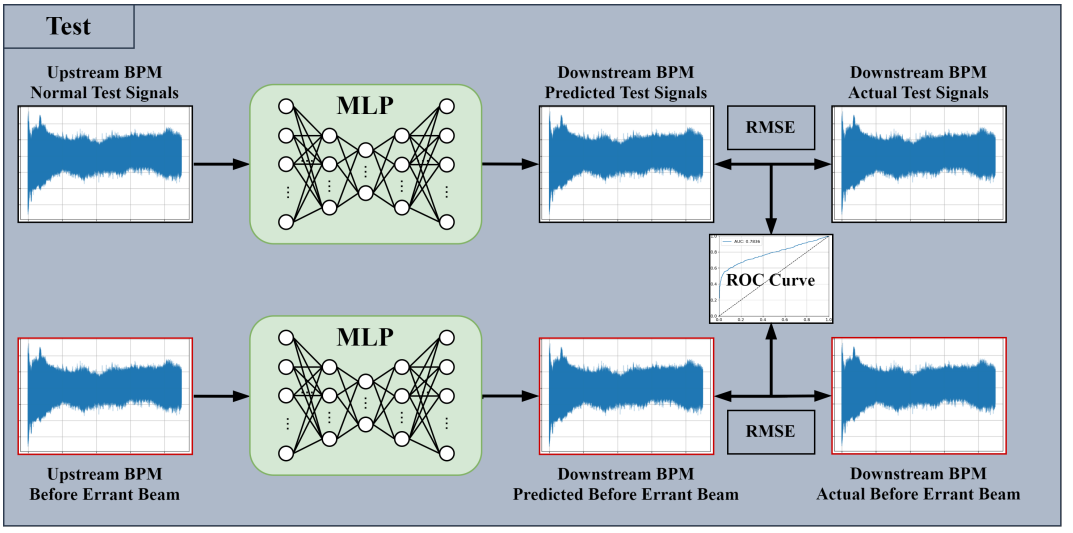
Figure 6 illustrates the learning curve as an indicator of training performance and the ROC curve to evaluate prediction performance of trained model on unseen normal and faulty signals. In Figure 6a, we observe a smooth training trend through epochs. As the model converges, lower loss evaluated on validation set throughout the training indicates decent generalization from the training set, as opposed to an overfitting issue. Figure 6b is the ROC curve constructed using trained model. Even though the overall classification performance implies good distinction between faulty and normal signals with a steep rise of the curve, we need to focus on very low FPR region as justified in Section 3.3. The zoomed in plot shows that model has TPR $\widetilde{=}$ 0.2568 for FPR = 0.005, which means by preserving 99.5% of our normal signals, model can identify more than a quarter of errant beams one pulse prior to the errant beam occurring.

In an ML training, one should always ensure the data is not biased as we split into training and validation. We will use k-fold cross-validation technique (explained in Appendix A) to shuffle the data and systematically switch the validation set and train a model for each fold. With Figure 7 we investigate the cross-validation results. Figure 7a shows the ROC curves constructed for each fold of the cross-validation analysis. The zoomed in version of these curves suggests minor deviation on TPR for each fold, for our target FPR of 0.005. In Figure 7b, we illustrate this distribution as a function of the target FPR. As expected, as we increase the number of misclassified normal signals (increased

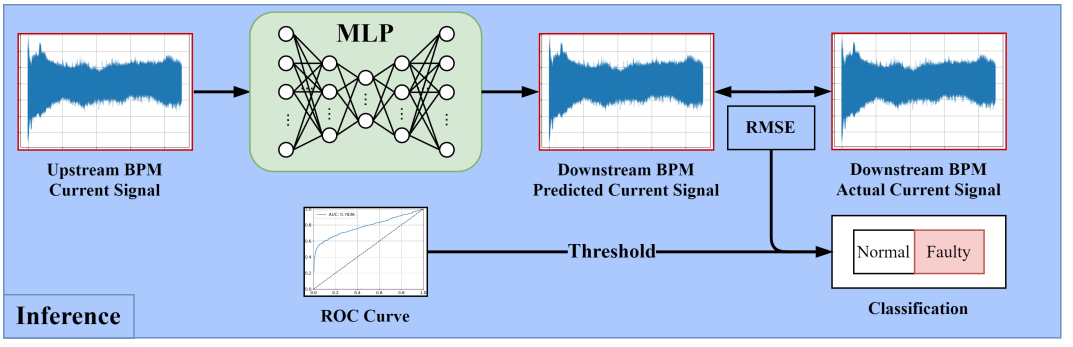
[1] https://keras.io/api/optimizers/rmsprop/



(a) Training.



(b) Testing and post-processing.



(c) Inference.

Figure 4: Flowchart for training, testing, post-processing, and inference.

FPR), we can increase the number of errant beams correctly identified and is the essential operational trade-off for classification problems.

Finally, we conduct a hyperparameter / neural architecture optimization with the method and settings prescribed in Section B. We take the training and validation datasets used in the initial training and use the obtained TPR = 0.2568 metric as a baseline for improvement. We set 100 iterations limit for the optimization. Figure 8 provide illustrations that summarize the optimization and insights gained. In Figure 8a, the effect of the network depth on performance is presented. We can clearly conclude here is that single hidden layer networks tend to overperform more deep networks, even though the difference is marginal. Figure 8b depicts how the activation function used in hidden

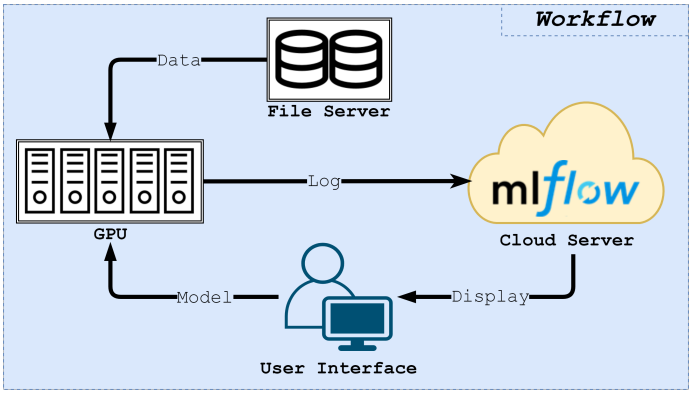
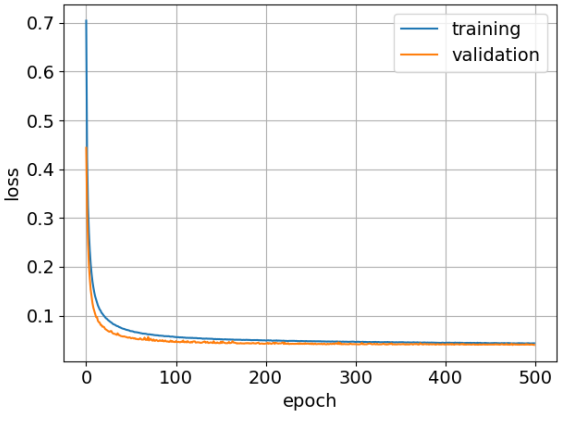
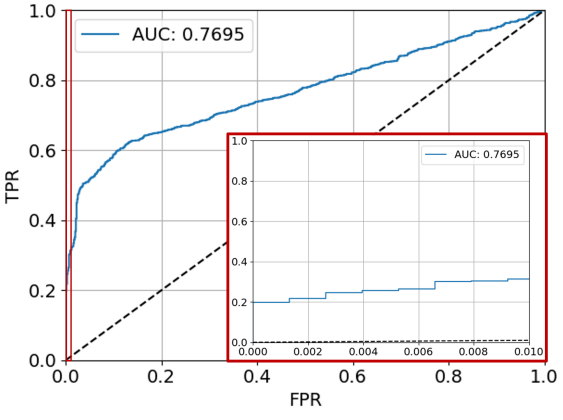


Figure 5: Flowchart of designed ML workflow.

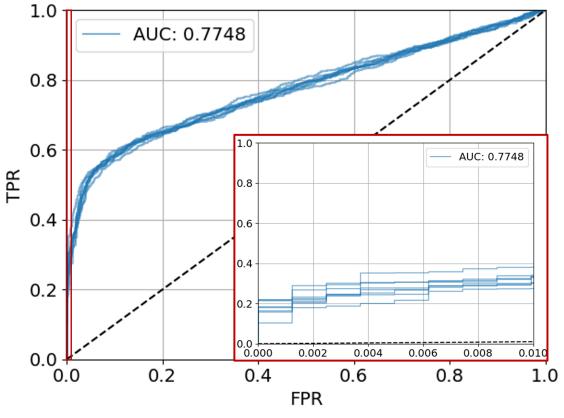


(a) Learning curve.

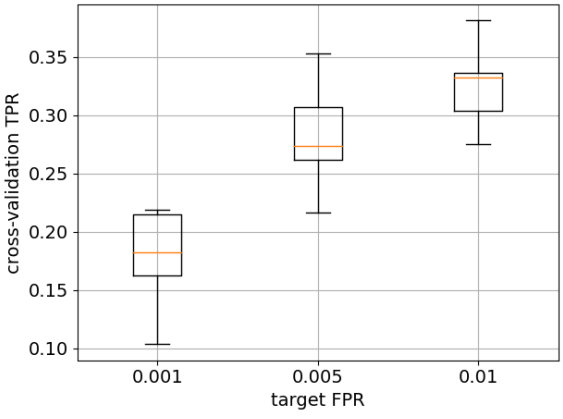


(b) ROC curve, zoomed into low FPR region.

Figure 6: Training and prediction performance.



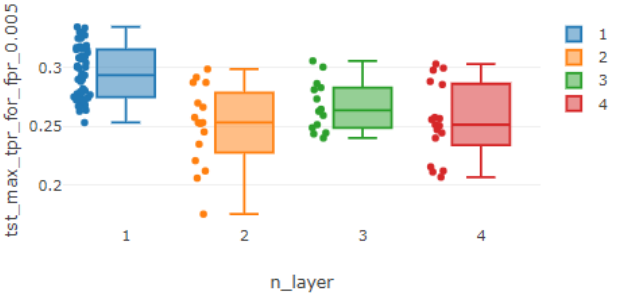
(a) ROC curves from all cross-validation folds.



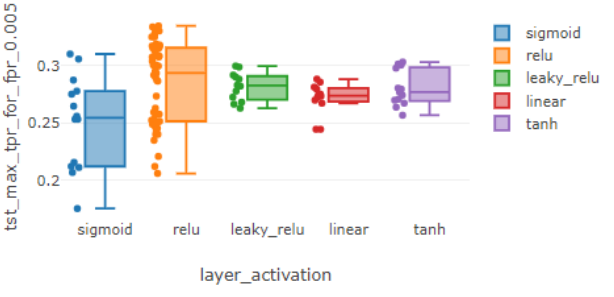
(b) Fold-wise TPR distribution for different FPR targets.

Figure 7: Training and prediction performance.

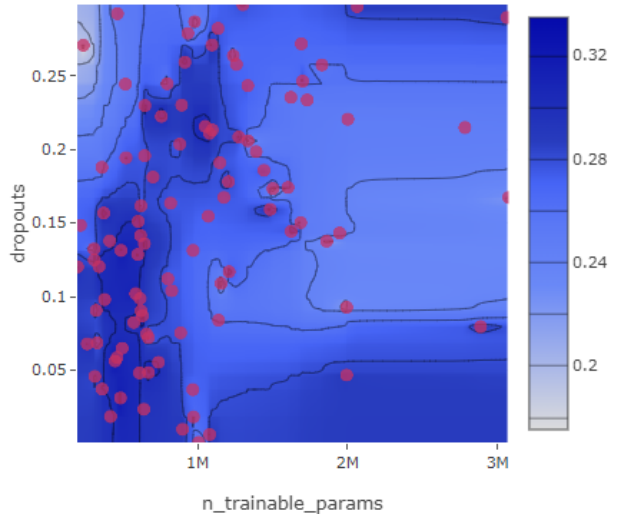
layers can affect the performance. Despite the large interval between the upper and lower quantiles, networks that utilize relu activation yields to the best performance obtained compared to the other options. This is not a surprising outcome, since relu activation provides more reliable convergence in network training and widely used in all type of deep learning models [18]. Figure 8c shows the contour plot constructed with number of trainable parameters on the X-axis and layer dropout percentage on the Y-axis, and the target metric TPR as the Z-axis. This landscape can be interpreted as how the optimization algorithm focuses on certain regions of the search space. As can be deduced from the results, a less complex architecture (less number of trainable parameters), results in better performance, with lower dropout rates tending to contribute to this performance also. This result also aligns with Figure 8a, in the claim that "simple is better". However, as observed from Figure 8d, the width of the network does not have a simple monotonic effect on the performance. This may be caused by the limits we defined for the number of nodes as the search space was defined.



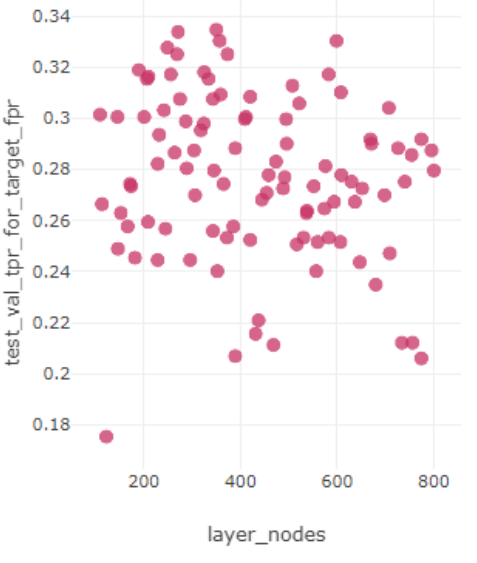
(a) Effect of number of hidden layers (i.e. depth of the network).



(b) Effect of the activation function used in hidden layers.



(c) Effect of model complexity (number of model parameters) and dropout percentage on the performance. Note that color scaled Z-axis shows model TPR at FPR = 0.005.



(d) Effect of number of hidden layer nodes (model width) on model performance.

Figure 8: HPO/NAS insights. Note that every single dot corresponds to a model architecture / set of hyperparameters trained during optimization.

Finally, Figure 9 presents the ROC curve of the best performing model obtained from HPO/NAS study. This result indicates that the best-performing model has a TPR $\eqsim 0.3348$ for FPR $= 0.005$, which is an increase of 8% over the initial model and enables identification of more than a third of errant beams.

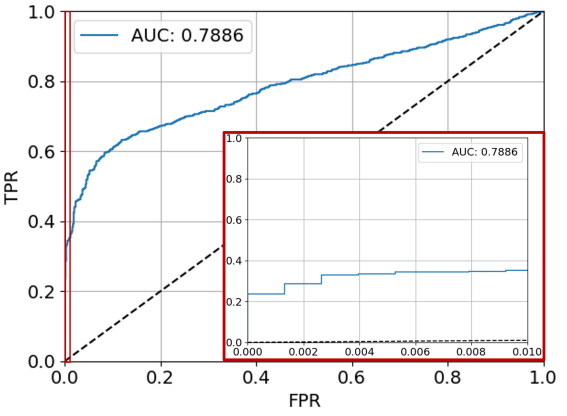


Figure 9: ROC curve of the best performing model after optimization.

7. Conclusion

In this contribution we propose a Machine Learning method to utilize two diagnostic sensors located across the linear accelerator tunnel of SNS, and identify errant beams by mapping the phase signals from the upstream sensor to downstream sensor. We construct an initial Neural Network model and establish a baseline performance. Then we apply cross-validation, in order to reveal the sensitivity to data shuffling. Finally, we conduct a hyperparameter / neural architecture search to fine tune our initial model and gain insights on different aspects of proposed model on performance. We also build a sustainable framework for conducted ML experiments for accelerated training and optimization.

Our findings prove that with the proposed approach, more than a quarter of errant beams can be predicted with a simple initial model. Furthermore, this model can be optimized to increase the performance to over 33% correct classification of errant beams by only misidentifying 0.5% of the normal operation beams. The optimization study also revealed that:

- Single hidden layer architecture outperforms deeper designs.
- Despite large spread, best results obtained with relu activation layer.
- Simpler models tend to perform better.
- Model width does not have an obvious effect on the performance.

Based on these results, as future work, we would like to explore following avenues to improve errant beam prediction performance:

- More data: one can easily notice the step-wise behaviour in our ROC curves, which is essentially the result of limited number of signals. We aim to increase the frequency of data acquisition from BPM sensors and enhance our dataset.
- Higher quality data: improve data acquisition and train the model with higher resolution phase signals.
- Data source optimization: test different pairs of upstream and downstream sensors, or even optimize which pair gives the best result.
- Data fusion: incorporate other variables (beam current, beam position) that are suspected to include failure signature before errant beam occurs.
- Exploring time horizon: investigating the pulses two or more before an errant beam to potentially expand the time horizon for errant beam detection.

As this contribution proved that we can indeed predict significant amount of errant beams prior to failure, we look forward to take another step towards deploying our model to the field for operational benefit. The next step towards operationalization is to expose our model that is trained with historical data to real-time streaming data, and verify online predictive capability. Once we establish the performance of the model is sustainable, then we can grant the model the permission to abort beams, and collect statistics on errant beams to validate there is in fact considerable decrease in downtime caused by these faults. There is however significant need for hardware installation inside the LINAC to enable timely processing of data from two different BPM chassis to conduct inference. We envision the aforementioned system to be an edge computer that is tailored for fast inference and data processing.

Finally, we acknowledge the challenges of translating ML models trained and evaluated offline into operational environment under real-time data. One specific challenge is the necessity for consistent performance with temporal extrapolation. Models trained on historical data naturally infer on future; hence, expected to be robust to any changes in the system or data to a certain extent. For this purpose, we are aiming to continuously test the models trained in this work on the data collected from continued operations. Another potential challenge on deploying proposed model for online inference is the data streaming issues. Since our model utilizes data from two separate sensors located spatially further apart in the tunnel, we need to ensure data streaming from these sensors to an edge computer that conducts model prediction is timely, in addition the inference speed is sufficient to take any action using model output.

Acknowledgments

The authors acknowledge the help of the collaborators from Thomas Jefferson National Accelerator Facility, Malachi Schram and Kishansingh Rajput, and Sarah Cousineau at ORNL, for making this grant work possible.

This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). This research used resources at the Spallation Neutron Source, a DOE Office of Science User Facility operated by the Oak Ridge National Laboratory. The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

References

[1] P. S. Langan, V. G. Vandavasi, C. J. Cooper, K. L. Weiss, S. L. Ginell, J. M. Parks, L. Coates, Substrate binding induces conformational changes in a class a β-lactamase that prime it for catalysis, ACS Catalysis 8 (2018) 2428–2437. doi:10.1021/acscatal.7b04114.

[2] H. Zhou, K. An, S. Allu, S. Pannala, J. Li, H. Z. Bilheux, S. K. Martha, J. Nanda, Probing multiscale transport and inhomogeneity in a lithium-ion pouch cell using in situ neutron methods, ACS Energy Letters 1 (2016) 981–986. doi:10.1021/acsenergylett.6b00353.

[3] M. L. Wissink, Y. Chen, M. J. Frost, S. J. Curran, O. Rios, Z. C. Sims, D. Weiss, E. T. Stromme, K. An, Operando measurement of lattice strain in internal combustion engine components by neutron diffraction, Proceedings of the National Academy of Sciences 117 (2020) 33061–33071. doi:10.1073/pnas.2012960117.

[4] S. Henderson *et al.*, The spallation neutron source accelerator system design, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 763 (2014) 610–673. doi:https://doi.org/10.1016/j.nima.2014.03.067.

[5] R. B. Randall, Vibration-based condition monitoring: industrial, automotive and aerospace applications, John Wiley & Sons, 2021.

[6] Y. Zhu, Y.-Q. Ni, H. Jin, D. Inaudi, I. Laory, A temperature-driven MPCA method for structural anomaly detection, Engineering Structures 190 (2019) 447–458. doi:10.1016/j.engstruct.2019.04.004.

[7] W. Blokland, K. Rajput, M. Schram, T. Jeske, P. Ramuhalli, C. Peters, Y. Yucesan, A. Zhukov, Uncertainty aware anomaly detection to predict errant beam pulses in the oak ridge spallation neutron source accelerator, Physical Review Accelerators and Beams 25 (2022). doi:10.1103/physrevaccelbeams.25.122802.

[8] S. Li, A. Adelmann, Time series forecasting methods and their applications to particle accelerators, Physical Review Accelerators and Beams 26 (2023). doi:10.1103/physrevaccelbeams.26.024801.

[9] M. Rescic, R. Seviour, W. Blokland, Predicting particle accelerator failures using binary classifiers, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 955 (2020) 163240. doi:10.1016/j.nima.2019.163240.

[10] A. Edelen, C. Mayes, D. Bowring, D. Ratner, A. Adelmann, R. Ischebeck, J. Snuverink, I. Agapov, R. Kammering, J. Edelen, I. Bazarov, G. Valentino, J. Wenninger, Opportunities in machine learning for particle accelerators, 2018. URL: https://arxiv.org/abs/1811.03172. doi:10.48550/ARXIV.1811.03172.

[11] A. A. Mishra, A. Edelen, A. Hanuka, C. Mayes, Uncertainty quantification for deep learning in particle accelerator applications, Physical Review Accelerators and Beams 24 (2021). doi:10.1103/physrevaccelbeams.24.114601.

[12] A. Scheinker, F. Cropp, S. Paiagua, D. Filippetto, An adaptive approach to machine learning for compact particle accelerators, Scientific Reports 11 (2021). doi:10.1038/s41598-021-98785-0.
[13] C. Emma, A. Edelen, M. Hogan, B. O'Shea, G. White, V. Yakimenko, Machine learning-based longitudinal phase space prediction of particle accelerators, Physical Review Accelerators and Beams 21 (2018). doi:10.1103/physrevaccelbeams.21.112802.
[14] M. I. Radaideh, C. Pappas, J. Walden, D. Lu, L. Vidyaratne, T. Britton, K. Rajput, M. Schram, S. Cousineau, Time series anomaly detection in power electronics signals with recurrent and ConvLSTM autoencoders, Digital Signal Processing 130 (2022) 103704. doi:10.1016/j.dsp.2022.103704.
[15] M. I. Radaideh, L. Lin, H. Jiang, S. Cousineau, Bayesian inverse uncertainty quantification of the physical model parameters for the spallation neutron source first target station, Results in Physics 36 (2022) 105414. doi:10.1016/j.rinp.2022.105414.
[16] B. Maldonado, F. Liu, N. Goth, P. Ramuhalli, M. Howell, R. Maekawa, S. Cousineau, Data-driven modeling of a high capacity cryogenic system for control optimization, in: Proceedings of the 22nd World Congress of the International Federation of Automatic Control, 2023. Accepted.
[17] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe, et al., Accelerating the machine learning lifecycle with mlflow., IEEE Data Eng. Bull. 41 (2018) 39–45.
[18] P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, 2017. URL: https://arxiv.org/abs/1710.05941. doi:10.48550/ARXIV.1710.05941.
[19] T. Elsken, J. H. Metzen, F. Hutter, Neural architecture search: A survey, Journal of Machine Learning Research 20 (2019) 1–21.
[20] L. Yang, A. Shami, On hyperparameter optimization of machine learning algorithms: Theory and practice, Neurocomputing 415 (2020) 295–316. doi:10.1016/j.neucom.2020.07.061.
[21] Y. Ozaki, Y. Tanigaki, S. Watanabe, M. Onishi, Multiobjective tree-structured parzen estimator for computationally expensive optimization problems, in: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, ACM, 2020. URL: https://doi.org/10.1145/3377930.3389817. doi:10.1145/3377930.3389817.

Appendices

A. K-fold Cross-validation

While training a data-driven model with limited data, one needs to ensure robustness of the model to unseen data through cross-validation. K-fold cross-validation is a prominent technique for this purpose and is carried out by following these steps:

1. Dataset is shuffled.
2. Dataset is split into k subsets
3. Looping over each subset:
   - 3.1 Selected subset is left out of training data.
   - 3.2 Remaining data is assigned as training set.
   - 3.3 Model is trained on training set and evaluated on the left out subset.
4. Evaluation metrics from each fold are combined to provide an overall measure of sensitivity.

In this paper we will apply 8-fold cross-validation.

B. Neural Architecture Search / Hyperparameter Optimization

We discussed the MLP architecture and training hyperparameters determined for this study in Section 4. However, these values are initially selected arbitrarily. In most ML applications, these values are optimized to provide best possible architecture design and set of hyperparameters, with the optimization performed using Neural Architecture Search (NAS) [19] and Hyperparameter Optimization (HPO) [20] approaches.

In this work, we employ Tree-structured Parzen Estimators (TPE) as our hyperparameter optimization algorithm [21]. The TPE is a powerful technique commonly employed for both hyperparameter optimization and neural architecture search. Fundamentally based on Bayesian Optimization theory, TPE operates by modeling the relationship between hyperparameters and the objective function through two separate probability distributions: a prior and a conditional distribution. The prior distribution represents the prior knowledge about the hyperparameters, while the conditional distribution captures the relationship between hyperparameters and the objective function based on historical evaluations. By iteratively updating these distributions, TPE focuses its search on promising regions of the hyperparameter space, effectively balancing exploration and exploitation. This algorithm employs a tree-based search strategy, where promising hyperparameter configurations are selected for evaluation based on the expectation of improvement in the objective function. Its flexibility, scalability, and ability to handle both continuous and categorical hyperparameters

Algorithm 1 Tree-structured Parzen Estimator (TPE)

Require:
N: number of iterations
n: number of samples per iteration
$p(x|y)$: density model for good configurations x given their performance y
$p(y)$: density model for the performance y

Ensure:
Best configuration found: x^*

$D \leftarrow \emptyset$ ▷ Set of observed configurations and their performances
for $i = 1$ to N **do** ▷ Perform N iterations
  Sample n configurations $\{x_1, x_2, \ldots, x_n\}$ from $p(x|y)$ ▷ Generate candidate configurations
  for $j = 1$ to n **do** ▷ Evaluate each candidate configuration
    Evaluate configuration x_j and obtain performance y_j
    Add (x_j, y_j) to D ▷ Store the configuration and its performance
  end for
  Split D into D_{good} and D_{bad} based on a threshold ▷ Partition configurations based on performance
  Update $p(x|y)$ using D_{good} ▷ Update the density model for good configurations
  Update $p(y)$ using D_{bad} ▷ Update the density model for performance
end for
$x^* \leftarrow \arg\max_{x \in D} p(x|y)$ ▷ Find the configuration with highest probability
return x^* ▷ Return the best configuration

Table 2
HPO/NAS search space.

Network/training parameter	Lower limit	Upper limit	Variable type
Number of hidden layers	1	4	Discrete
Number of nodes per layers	100	800	Discrete
Activation of hidden layers	[linear, tanh, relu, leaky relu, sigmoid]		Discrete
Dropout percentage	0.0	0.3	Continuous
Initial learning rate	1e-6	1e-3	Continuous
Reduce learning rate on plateau - factor	0.5	0.99	Continuous
Reduce learning rate on plateau - patience	5	20	Discrete
Early stopping - patience	20	80	Discrete

make TPE a valuable tool for researchers and practitioners seeking optimal hyperparameter settings and neural architecture designs. The pseudo-code of the TPE algorithm is outlined in Algorithm 1.

In this work, we define architectural and hyperparameter space as in Table 2. For the performance metric to be maximized by the employed optimization process, we choose TPR at FPR = 0.005 corresponding to maximizing the number of errant beams correctly identified with only 0.5% of the normal beams misclassified, which also aligns with the system availability requirement defined in Section 3.3.