UNIVERSITY OF
LIVERPOOL

Using Neural Networks to Predict Pixel Thresholds in the LHCb

Byron Vernon
20161512

A Project Report Submitted in partial fulfilment of the requirements for the degree of Bachelor of Science

Under the supervision of David Hutchcroft
At the Department of Physics

Contents

1 Declaration

I hereby declare that this report is my own work and effort and that it has not been submitted anywhere for any award. Where other sources of information have been used, they have been acknowledged.

Byron Vernon November 26, 2025

2 Abstract

The upgraded Vertex Locator (VELO) in the LHCb contains 40 million pixels, the threshold ('trim') value of every pixel must be calibrated at regular intervals. The current process, a full 16-step noise scan, measures each pixel at every possible trim setting. While accurate, the scan keeps the sensors powered for longer, increasing self-heating and therefore radiation damage. This study investigates a data-driven alternative that uses a neural network to predict the optimal trim value from just the two extreme scan points (0x0 and 0xF).

Construction phase noise data from 120 VELO datasets were cleaned (faulty pixels and NaNs removed) and the four input features (noise mean and width at both extremes) were manually normalised with sensors specific offsets. A network built in Keras/TensorFlow reached an average accuracy of 98.6% within ± 1 trim unit on 60 independent construction 256x256 arrays, training in under 20 seconds and predicting a single 256x256 dataset in less than 2 seconds. For a representative dataset, $> 99\%$ of predictions lay within two trim units of the true value.

When the same model was applied to a 2025 operational module, accuracy dropped to 83%, showing condition shifts caused by the different configuration and/or irradiation effects.

These results demonstrate that neural network calibration can potentially replace the full scan with a two point measurement, shortening calibration time and reducing thermal load while retaining high precision. Periodic retraining on updated full scans would be required to keep up with changing sensor conditions and sustain performance over the VELO's lifetime.

3 Introduction

3.1 Problem Overview

At CERN, many detectors are used to study the particles produced in high-energy collisions. One of which, the LHCb [1], studies particles containing beauty quarks to further the understanding of matter, anti-matter and the imbalance between them. Used within this experiment is a VELO (VErtex LOcator), which tracks particles' trajectories and identifies decay and collision points. In 2022, this detector was upgraded so that its sensor arrays now contain a total of 40 million pixels, improving precision. Each of these 40-million pixels must be calibrated to both detect real signals and filter out noise. This calibration is the main focus of this project.

Currently, calibration is achieved using a full noise scan. This process tests every pixel under all 16 possible threshold settings, known as trim values (ranging from 0x0 to 0xF

in hexadecimal). A threshold, in this context, is the minimal signal level a pixel must receive to be recorded. During the scan, the noise response at each setting is measured, allowing for identification of the threshold value that best distinguishes real signals from background noise. Although this accurately finds the optimal trim value for each pixel, it is very time consuming as the system must be tested at all 16 values and then each pixels data evaluated. Also, due to sensor behaviour potentially changing over time, this scan has to be repeated regularly to maintain reliable detector performance.

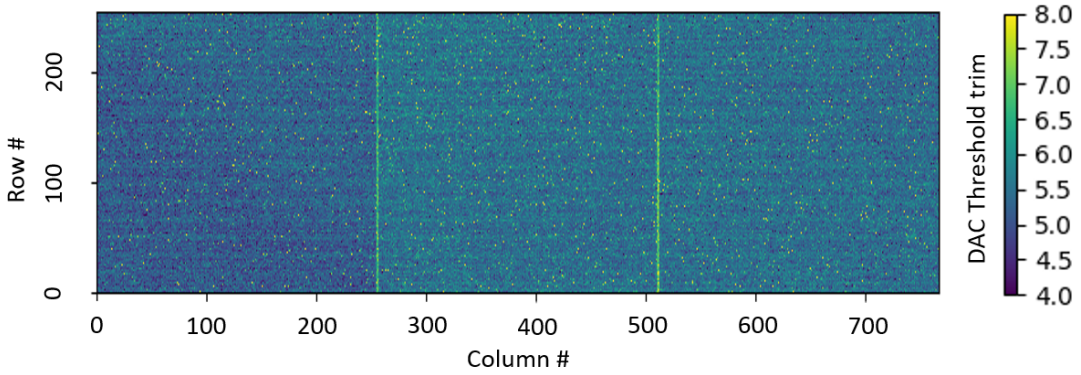


Figure 1: Map of pixel trim values for a single tile. Obtained from Manchester Article on construction of the VELO [2]

Figure 1 shows a map of the final trim values across a "tile," a single piece of silicon which contains three 256×256 pixel sensors within the VELO module. Although the hardware trim can span 0x0–0xF, the colour scale is cut to 4-8 in this figure because the majority of values are within this range. As a result, differences in trim values are easier to visualise. Viewing this map it can be seen that pixels have a natural variation in trim values, coming from small differences in construction and conditions. This variation along with the fact there are no clear observable patterns, highlights the complexity in calibration of threshold values.

As shown in Figure 2, each ASIC (one (256×256) dataset) has a global DAC (digital to analogue converter) threshold that spans a wide, precise range across the 256×256 pixel array. In addition, every pixel has a smaller 4-bit trim (0–F) to correct for pixel-to-pixel variations, this is the value predicted in this project. Although the per-pixel adjustment is relatively small, it is large enough to fine-tune individual pixels and achieve uniform calibration without relying solely on the global threshold.

3.2 Motivation

Due to the time-consuming nature of the calibration process; the goal of this project is to use neural networks to predict the trim values using only a fraction of the data, therefore reducing calibration time significantly.

Fast calibration of the trim values is important as the full noise scan requires the sensors to remain powered, which continues heat generation. The sensors are continuously cooled, however, reducing calibration time allows the detectors to be powered off sooner. This, minimises the period where heat builds up, enabling the system to restore the optimal low operating temperature (around -30°) faster. Maintaining these low temperatures are necessary for mitigation of radiation damage as explained in the article on radiation damage effects [3]. Faster calibration is therefore important in minimising thermal stress

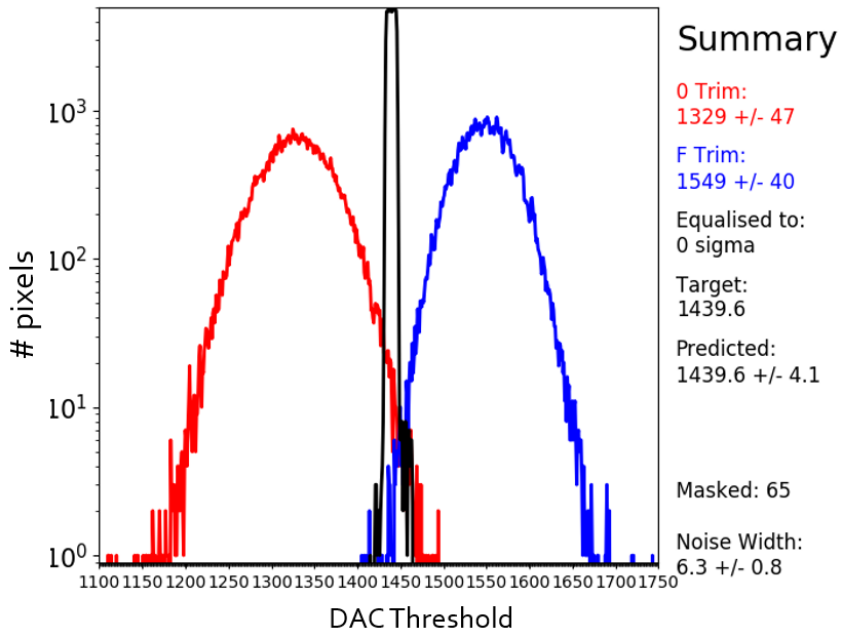


Figure 2: Histogram of Equalization Results for a Single ASIC (Application-specific integrated circuit)

and preservation of the sensors, allowing longer sustained performance over experiment runs.

3.3 Introduction to neural networks

Neural networks are computer systems inspired by the brain, designed to recognise patterns and make predictions based on data patterns. They consist of interconnected layers of nodes (or neurons) that process inputs and improve outputs through learning, making them effective at predicting values. In this project, the goal is to predict pixel trim values, using the construction and performance data available from LHCb through the Supervisor's access to CERN.

A neural network learns complex patterns in data by passing it through its layers. Each neuron in the layers receives the inputs, calculates a weighted sum, then uses an activation function, which adds non-linearity by determining how significant the result of each node is. The final layer of the network produces an output which is either probabilities (classification) or a continuous output (regression).

To improve the predictions over time, initially the data is passed forwards through the network from the input layer, through the hidden layers into the output layer, finally the network's output is compared with the true value and the difference is quantified using a loss function. It then uses a process called back-propagation which is where the network learns from its mistakes by adjusting the weights so future predictions are closer to the answer. The network calculates how each nodes weight contributes to the loss and then adjusts weight in the direction that reduces loss. Repeating these steps across multiple epochs (iterations) gradually improves the network.

Since threshold calibration (full noise scanning) is time-consuming, a trained neural network can potentially predict the optimal trim values for each pixel using just the first and last step of the noise scan, saving significant time. Model performance is evaluated

using both accuracy (within 1 integer of true trim) and cross-entropy loss, which is what the network uses as feedback to improve predictions and quantifies how wrong a guess is, guiding the network towards the correct values. Cross entropy loss is defined as

$$L_{CE}(p, q) = -\sum_{i=1}^{C} p_i \log q_i \tag{1}$$

Where p_i represents the true probability distribution (usually a vector where the correct class is denoted as a 1 and all others a 0) and q_i is the predicted probability for class i, with C being the total number of classes.

As shown in equation 1, the loss is a negative logarithmic function. This means that the further the probability deviates from one (the more wrong the prediction), the loss increases dramatically. This steep penalty involved in the logarithmic loss, provides larger gradients for updates during back-propagation, which speeds up learning, pushing the network towards correct values quicker.

Initially sparse categorical accuracy was used, which only rewarded predictions that exactly matched the true value. However, due to reasons explained later in this report, the accuracy function was changed to reward guesses within 1. This accuracy function is defined as:

$$A_{\text{adj}} = \frac{1}{N} \sum_{j=1}^{N} \mathbb{1}\{|y_j - \hat{y}_j| \leq 1\},$$

where y_j denotes the true label for the jth sample, $\hat{y}_j$ is the predicted label (determined as the index of the highest output probability), N is the total number of samples, and $\mathbb{1}\{\cdot\}$ is the indicator function that returns 1 if the absolute difference between y_j and $\hat{y}_j$ is at most 1 and 0 otherwise. Since the true probability vector p, included in the loss, is one-hot encoded (i.e. the correct class is indicated by a 1 and all others by 0), the true label y_j is simply the index i for which $p_i^{(j)} = 1$. Similarly, the predicted label $\hat{y}_j$ is determined by taking the index of the maximum value in q. This direct mapping from the probability distributions in Equation 1 to discrete class labels allows to evaluate accuracy using the metric defined above.

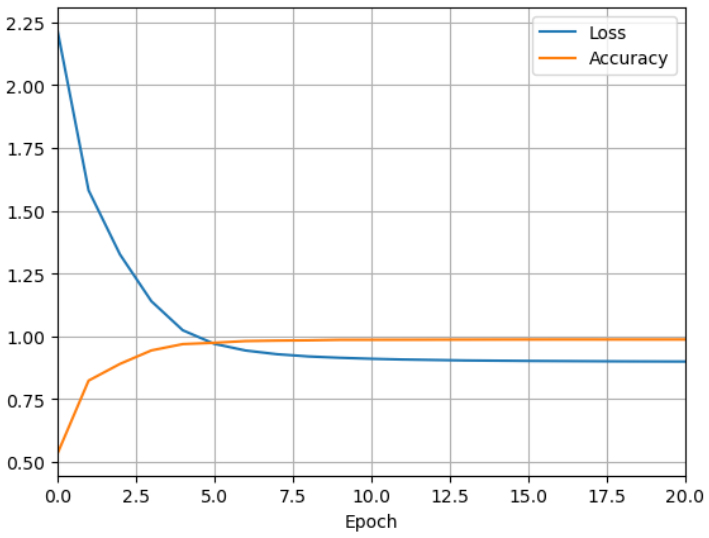


Figure 3: Accuracy and loss over first 20 epochs of training

Figure 3 shows the cross entropy loss, quickly converging to a value below 1 (≈ 0.9) while accuracy rises to around 98%. The increase of accuracy to a high level as loss is quickly optimised demonstrates the effectiveness of cross entropy loss as the networks training feedback.

4 Data

All data used in this project was obtained through the supervisor's access to CERN, specifically from the VELO construction and operational datasets in the LHCb experiment. Each dataset is organised by detector module, providing five key input parameters alongside a final threshold value for a (256×256) pixel matrix.

Each data folder is 1 'module' of the VELO upgrade, the full upgrade is made up of 52. Within a modules data folder, there are 12 different (256×256) datasets (4 tiles, labelled VP0-0 to VP3-2) Each of these datasets is broken up further into separate (256×256) array CSVs with one file for each parameter (mean noise at 0, noise width at F, final trim etc). When a dataset is loaded, these CSVs are read and reshaped from (256×256) grids into 65,536 element arrays, so every pixel becomes one row of data. The parameter arrays are merged to form a unified table with columns for each parameter with each row being a pixel, this is the form that is passed to the network for training and predictions. Further details on the rationale behind the design of individual VELO modules can be found in Ref. [4]

The five input parameters are:

- Mean noise at the lowest threshold (0),
- Mean noise at the highest threshold (F),
- Noise width at the lowest threshold (0),
- Noise width at the highest threshold (F),
- Mask value (0–4) indicating pixel faults (0 = fully functional, 4 = most severe).

During data pre-processing, any pixels containing NaN values were removed to prevent errors during neural network training. Faulty pixels (mask > 0) were also excluded because they consistently produce the same threshold, offering no meaningful information to the network. As a result, the mask parameter was omitted from the input features. The remaining data was then normalised so that all features were on a consistent scale, preserving the relative relationships between the lowest and highest thresholds without distorting critical correlations.

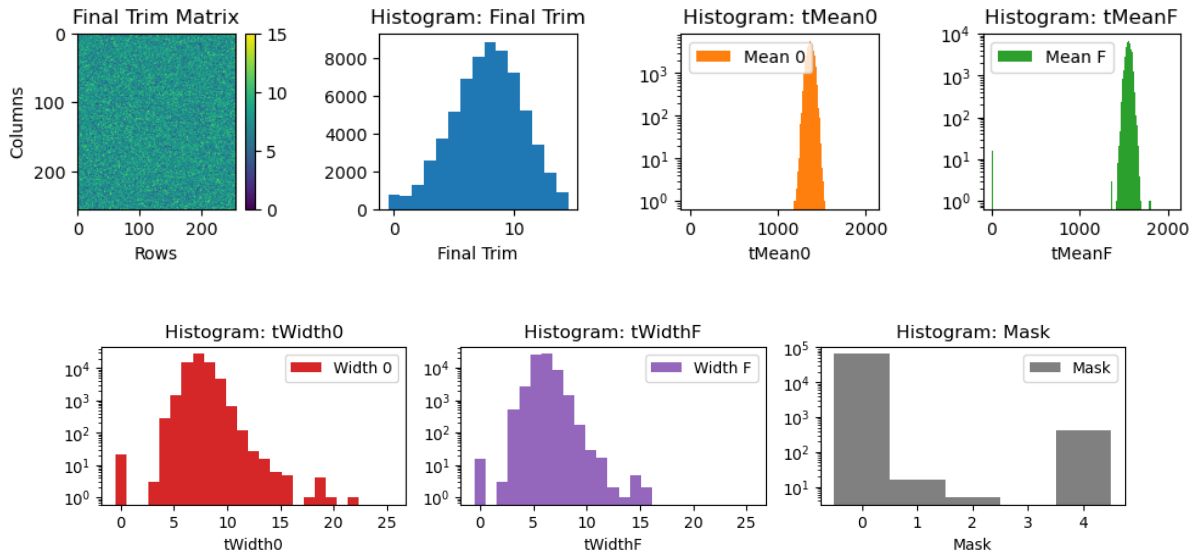


Figure 4: Calibration data for a single construction dataset. The top left plot shows the distribution final trim values across a 256×256 pixel array with a histogram of that distribution next to it, illustrating the optimal threshold settings for individual pixels. The other panels present histograms for each input feature—namely, the mean noise at the lowest threshold, noise width at both trim extremes, and the mask values.

Figure 4 displays one dataset (a single (256×256) array) within a modules data. The top left sub-plot shows the (256×256) matrix of final trim values, which the network and this project is attempting to predict. The remaining plots are histograms, first of those same trim values and then all the different input features follow it. These histograms show the physical distribution of each parameter, offering insight into how the noise and width values change at the 2 different threshold levels.

5 Methodology

5.1 Manual Observations

In addition to training the network with raw data, each data feature was also plotted against the final trim value to gain physical insights into the relationships, as shown in Figure 5 below.

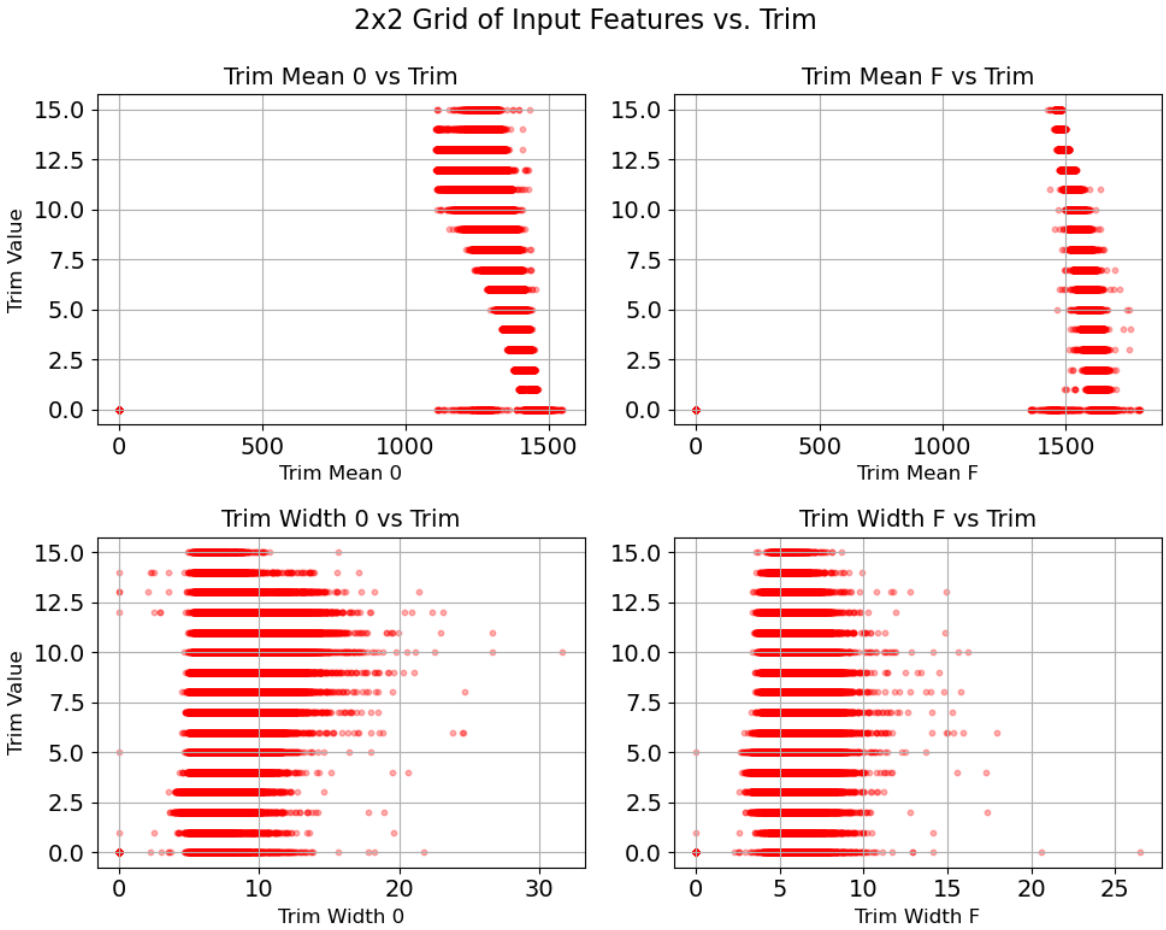


Figure 5: Width and mean features vs. Final Trim for 4 combined datasets.

Observing Figure 5 it can be seen that there is a complex correlation between the mean/width at the different thresholds and the trim values across four combined datasets, with the mean displaying a stronger correlation than the width. To explore the relationship of the final data feature Figure 6 was plotted and examined.

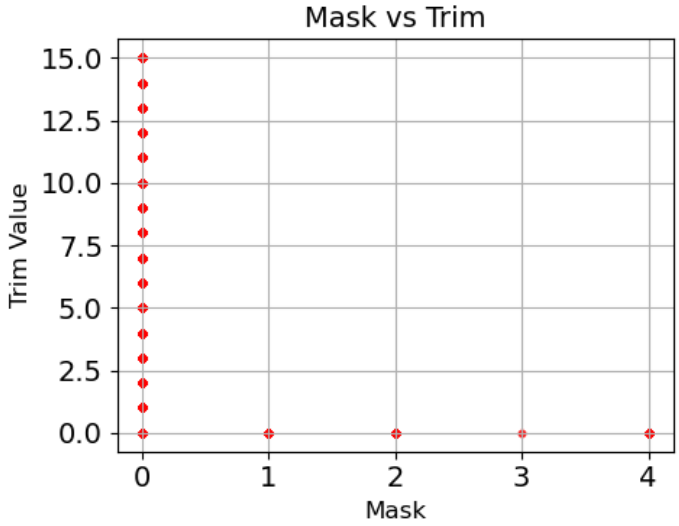


Figure 6: Mask vs. Trim for 4 combined datasets

Figure 6 demonstrates that only pixels with a mask value of 0, those considered "perfect" based on construction data, receive a tuned threshold, while pixels with a non-zero mask value (indicating they are known to be faulty) are always assigned a trim value of 0. From this consistent relationship across datasets it was determined that the trim of 'faulty' pixels (mask > 0) could be predicted accurately and therefore these pixels were filtered out before the data was passed into the network and mask was also removed as an input feature. These pixels were added back into the final prediction map later to get a full dataset prediction.

Alongside plotting the input features against trim values, statistical Pearson tests were performed on input features to test the strength of correlations further, making sure remaining input features offered meaningful input to the network.

Mean 0	Mean F	Width 0	Width F
-0.72	-0.78	0.11	-0.18

Table 1: Average Pearson correlation coefficients between input features and the final trim value. These correlations are illustrated in Figure 5

Table 1 displays the average Pearson correlation coefficients over all datasets, allowing the strength of inputs correlation with the trim to be seen. The results show mean values (at both 0 and F) offer the most information to the network as they have the strongest correlations with the trim values. On the other hand width features display a weaker correlation. Whilst they show a significantly weaker correlation, they still add value to the network. Along with testing these inputs, the correlation of the x and y coordinates were tested due to some observations of trim patterns showing visible patterns however this resulted in coefficients ≈ 0. To look into these patterns further, it was noted that each dataset is constructed of 4x2 superpixels and so the correlation of these super pixel coordinates was also tested.

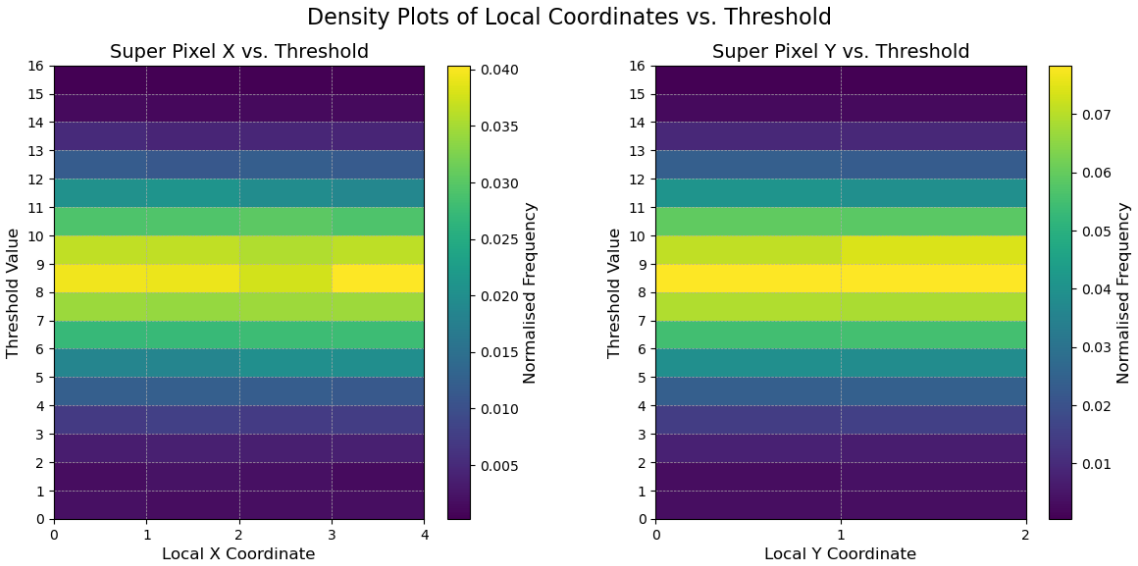


Figure 7: Local Super Pixel Coordinate vs Trim for 4 merged datasets

Figure 7 shows the distribution of trim values for each superpixel coordinate, showing no major difference between distributions across coordinates. This observation supports the conclusion that pixel position (both global and superpixel) does not provide meaningful information to improve trim prediction.

5.2 Framework and Initial Model

In this project, as discussed in the introduction, a neural network was used to predict pixel thresholds in the LHCb experiment, implemented using Keras [5] and TensorFlow [6]. Keras offers a simplified API (Application Programming Interface) making TensorFlow significantly easier to use/write code with. Reducing chances of implementation errors. Although it can limit advanced customisation options, this was not a problem for this project. TensorFlow is one of the most widely adopted deep learning frameworks, providing robust tools and extensive community support. Keras was selected due to the supervisor's familiarity with its libraries along with the availability of relevant documentation online. All code was written in Python.

An initial model provided by the supervisor gave a starting point to hit the ground running, a basic network set up to use one dataset. The dataset was split equally for training and evaluation and this data was passed into the input layer of the network with no preprocessing. This initial model also contained a batch normalisation layer, normalisation of the input data is important so the network processes all data on a similar scale. However when using a batch normalisation layer, it runs the risk of destroying the correlation between the values at 0 and F if they are normalised without the difference kept in mind.

The model architecture contained 1 hidden layer, made up of a number of nodes equal to the number of inputs (currently five) and an output layer made up of 16 nodes corresponding to the 16 discrete trim values (0-F in hexadecimal). The output layer, has consistently used a SoftMax activation function to provide probability distributions over the 16 classes, has remained mostly unchanged across experiments. This output configuration was found to be the most successful across different approaches, as discussed in Section 5.4. The hidden layer utilised the ReLU activation function, one of the most

widely used activation functions for its efficiency and simplicity. Figure 8 below shows three commonly used activation functions including ReLU.

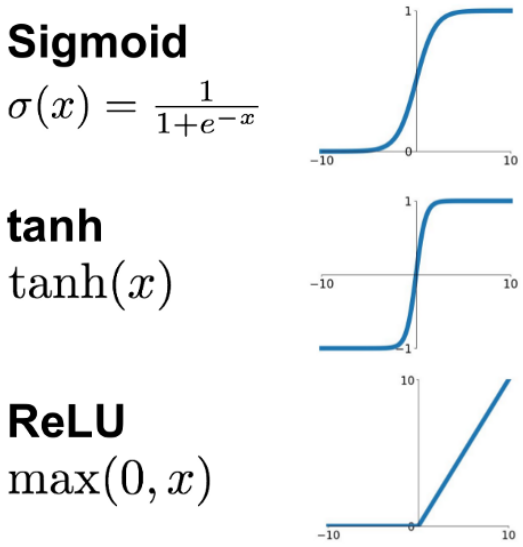


Figure 8: Three widely-used activation functions along with their graphical representations. Source: [7].

Figure 8 shows 3 widely used activation functions, ReLU, sigmoid and tanh. In this project, all 3 of these were tested along with leaky ReLU which addresses the issue within ReLU where negative input neurons are inactive, as shown in the figure, and instead applies a small non zero gradient to negative values, allowing all nodes to continue learning. The figure displays how Sigmoid and tanh provide smoother curves within different ranges. By comparing these functions, it can be investigated how each activation function affects learning and model performance in predicting the trim values. Testing these 4 different functions extensively allowed for an educated decision on the appropriate use for the final model.

5.3 Model Development

5.3.1 Evaluation Metrics

To evaluate each model iteration, results were automatically exported to a CSV file every time a model was evaluated, recording the final accuracy, cross entropy loss and training time. While the loss is what the network uses to improve, the metric served only as a secondary gauge of success, helping distinguish between models that achieved similar accuracies and times. Due to the inherent randomness of neural-network training (due to varying initial guesses and optimisation paths), each model was run 50 runs times to calculate the standard deviation of each metric. This statistical measurement allowed for comparisons to determine whether an improvement or regression in model performance was significant, rather than being the result of random variation.

Table 2 shows the standard deviations of performance metrics; accuracy, loss and training time, over 50 runs of the same model. These values display the natural variability due to randomness in the training process. For example, 68% (1σ) result an accuracy within $\pm 2.23\%$ of the mean, while 95% (2σ) are within $\pm 4.46\%$ and 99% (3σ) within

	Δ Accuracy (%)	Δ Loss	Δ Training Time (s)	Percentage of sample
1σ	2.23	0.111	38.6	68%
2σ	4.46	0.221	77.2	95%
3σ	6.68	0.332	115.8	99%

Table 2: Standard deviation metrics for model evaluation over 50 runs, rounded to three significant figures.

$\pm 6.68\%$. By comparing differences between models against these thresholds it can be determined whether a model is a significant improvement from another to the 3 different significance levels. It was determined that if a model has an observed improvement or regression in these metrics that exceeds the 2σ threshold (a 95% significance level) it was likely a genuine improvement and not due to random fluctuations.

5.3.2 Key Model Improvements

The first significant improvement in model performance was achieved by altering the evaluation metric. Initially the accuracy function rewarded only predictions that matched the final trim values exactly. However given there is 16 discrete trim values, along with that a true optimal threshold is likely to include a fraction rather than being a strict exact integer, a prediction that is off by just one unit (for instance, predicting 8 when the true value is 7) should not be considered completely incorrect. To address this the accuracy metric was updated to reward guesses 1 integer either side of the real final trim value. This adjustment better captures incremental nature of the trim values and acknowledges the uncertainty in determining an exact optimal trim value. This update to the accuracy metric resulted in a jump in performance from around 58% to 98% for the initial single split dataset, indicating the network's guesses were much better than the original metric suggested.

As described in subsection 5.1 the next significant leap in model performance came from addressing 'faulty' pixels. These faulty pixels, pixels containing a mask value greater than 0, consistently produced a trim value of 0. Due to this observation, all masked pixels were removed from the dataset before training and the mask feature was removed as an input. During the final prediction phase, the locations of the removed pixels were recorded and all their predicted trim values were set to 0, adding them to the networks outputted predictions and reshaping the resulting array back into a (256×256) array to create a complete prediction map.

Alongside the masked pixels, any pixels containing NaN values were removed to prevent training errors. Across all 115 evaluated datasets 265 total NaN pixels were observed, an average of 2.3 per set. 56.5% of tiles were free of any NaN data, while the rest contained 1 or more, with the worst case containing 70 NaNs. Even that extreme of 70 is just 0.11% of the 65,536 pixels. Overall, NaN data was a minor issue ($\approx 0.01\%$ of pixels) but still had to be dealt with to prevent errors. The NaN data arose from occasional errors during construction scans. Since the most common (mode) NaN trim values were 0 and they could not be predicted by the network, they were treated like masked pixels, removed from training and reinserted (with a predicted value of 0) at their original locations in the (256×256) prediction map.

As briefly described in the previous subsection, the initial model contained a batch normalisation layer. This layer calculates the mean and variance for each batch (a subset

of the data, with a configurable batch size) and normalises the inputs to ensure they are on a similar scale. While this can improve training, this method of doing so can obscure the differences between values at trim 0 and F, by treating batches independently. Therefore, the batch normalisation layer was removed and manual normalisation was added as part of data processing, ensuring relationships were preserved. In order to do this a normalisation function was created, this function standardised the data using a calculated sensor specific offset to enable comparisons between different sensors. The function first calculates an offset using the mean of input feature Mean 0, then normalises that feature by subtracting the offset and scaling it by a consistent factor. The factors and offsets were obtained by looking at the data and projecting it between -2 and 2, this wider span preserves a higher resolution in differences in values, helping the network determine differences. A similar normalisation is applied to the corresponding Mean F feature, but with an additional offset to account for systematic difference. Likewise, the width features are normalised by subtracting a baseline value and scaling by another constant factor. This approach normalises data whilst preserving relationships and eliminating sensor specific biases for consistent scaling.

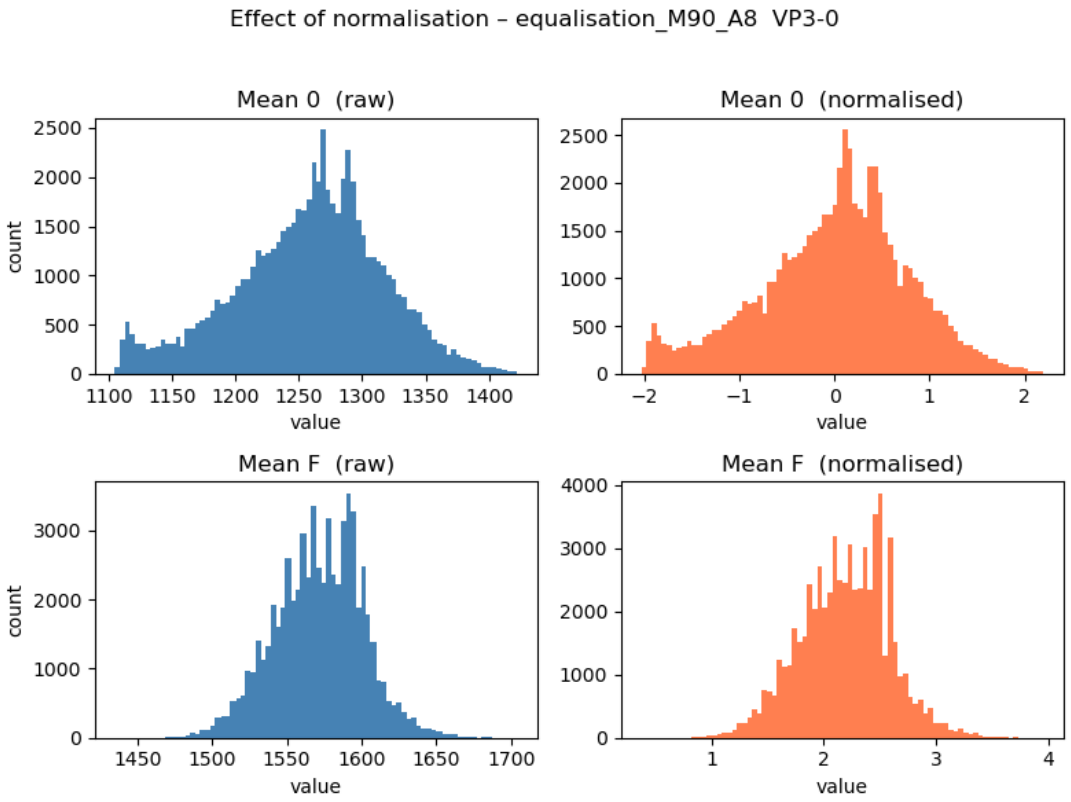


Figure 9: Effect of the `normSet` *routine on one datasets mean features. Left column (blue): raw distributions of the Mean 0 and Mean F features. Right column (orange): the same features after manual normalisation. The procedure recentres Mean 0 at 0 and maps both features into a comparable numerical range while preserving their shapes and the shift between the two.*

Figure 9 confirms that the `normSet` function re-centres feature Mean 0 at zero and rescales it within -2 and 2 while keeping the shape of the distributions intact. It then normalises Mean F with a clear change from that distribution (between around 1 and 3 in this case) to capture the shift between the values at the two trim levels. Because

the two trim levels now occupy distinct, but similarly scaled, regions of the input space, the network can understand the shift without the correlation being destroyed by batch normalisation.

The final major performance upgrade, aside from hyper-parameter tuning, came from addressing the different distributions of datasets. When trained on one dataset and evaluated across many others the average accuracy was around 86%. However, some datasets achieved very high accuracies (97-100%) while other fell as low as 50-60%, indicating poor generalisation (the network was only performing well on datasets similar to its training set). To improve this, initially an additional integer offset application step was added to the final prediction, iterating through offsets from -2 to 2 and returning the best value along with the resulting accuracy from this applied offset. However, in order to predict these offset values the network would need to be trained with varying distributions, therefore the network was retrained on 5 datasets, each representing a different best offset value. This exposure to multiple distributions, improved generalisation, resulting in an average accuracy of 98% and removing the need for offset adjustment post-prediction.

5.3.3 Hyper-Parameter Tuning

The network contains various parameters that can be tweaked to alter how the network learns, these are known as hyper-parameters. Hyper-parameter are both architectural and training based, architectural parameters define the structure of the network (number of layers, number of nodes per layer and activation functions), while training parameters control the learning process (learning rate and batch size). After incorporating all major model improvements discussed in the previous section, hyper-parameters were iterated over to evaluate their effects on performance and get to the optimal network setup. The parameter tuning process followed a logical sequence: first optimising the architectural parameters then fine tuning the training parameters. For every hyper-parameter, a range of values was defined with multiple steps and the corresponding accuracy and training time was recorded across 30 module evaluations to obtain reliable average metrics.

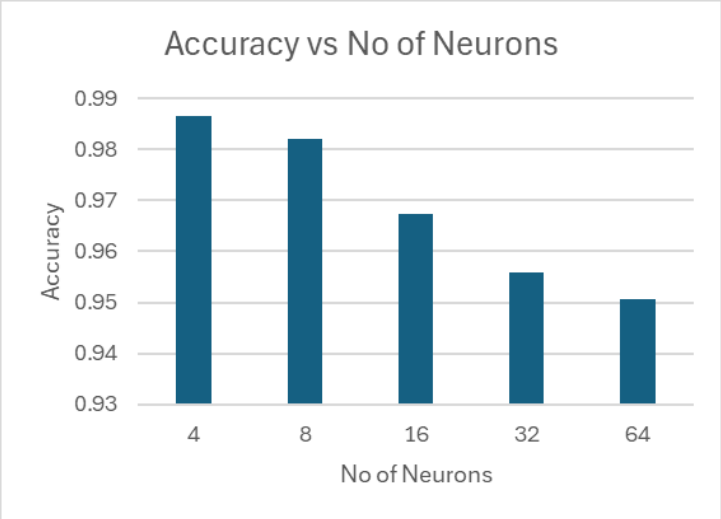


Figure 10: Test of number of neurons parameter, showing average accuracy across 5 different ascending values, each double the last.

Figure 10 displays an example of the parameter tuning process for the number of

neurons. The plot shows a clear downwards trend in average accuracy as number of neurons increases, with the initial value of 4 (lowest multiple of the number of input features) showing the best performance within the range. A similar approach was followed to tune all other parameters, which allowed for an optimal configuration for the final model.

5.4 Other Solutions

5.4.1 Regression based Neural Network

Another version of the model assessed was a regression model. Making the output layer one node and using the linear activation function instead. In this setup, the network produces continuous output values rather than discrete classes, in theory allowing for more precise error feedback and the possibility of decimal predictions that can then be rounded. However, this added extra complexity, as the continuous outputs had to be rescaled and rounded to match the trim values. Alongside this, initial testing of this model indicated poorer prediction accuracy compared with the classification model.

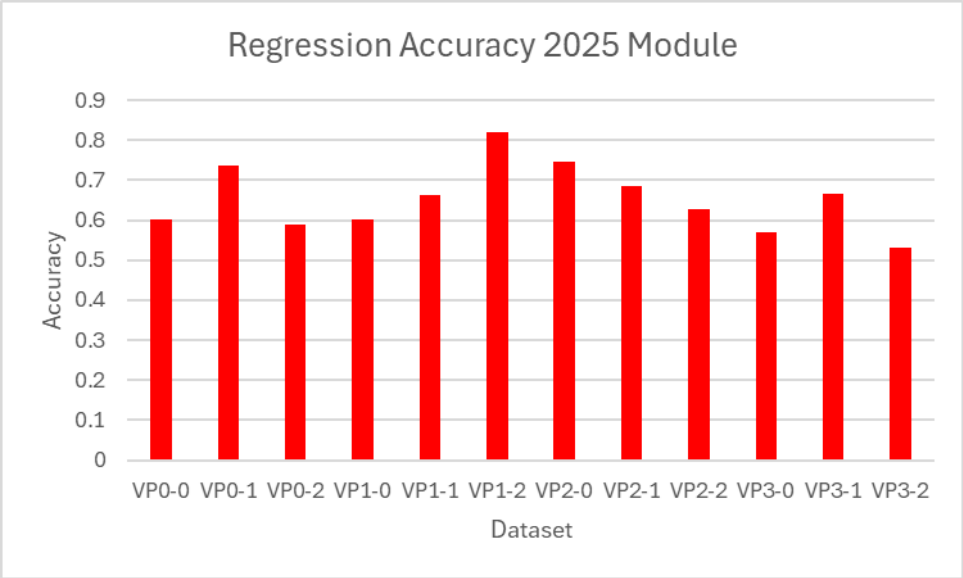


Figure 11: Results from regression model from 2025 module, same module used to later evaluate classification model. Average accuracy = 65.4%

Figure 11 shows the results from the tuned regression model over the 2025 operational module. Displaying a moderately worse accuracy in making predictions than the classification model, results shown in figure 17. This alongside the additional steps to convert outputs promoted the use of the classification model for this application. Hyperparameter tuning was also applied to this model to ensure a fair comparison, with the classification approach emerging as the more effective and simple solution.

5.4.2 Boosted-Decision-Tree (BDTs)

An alternative to neural networks was also considered and evaluated, XGBoost (Extreme Gradient Boosting) [8]. Unlike neural networks, which rely on interconnected nodes in layers, which "learn" through back-propagation, XGBoost builds decision trees in

stages which correct errors from previous trees. This method usually works well on tabular ('spreadsheet style') data and typically requires less tuning than neural networks. However, boosted decision-tree models can struggle to capture the subtler, highly non-linear correlations that a neural network learns from the full 16-class trim-prediction problem.

The same five construction tiles used to train the neural network were used for BDT training, along with the same 60 evaluation datasets, to ensure fair comparison.

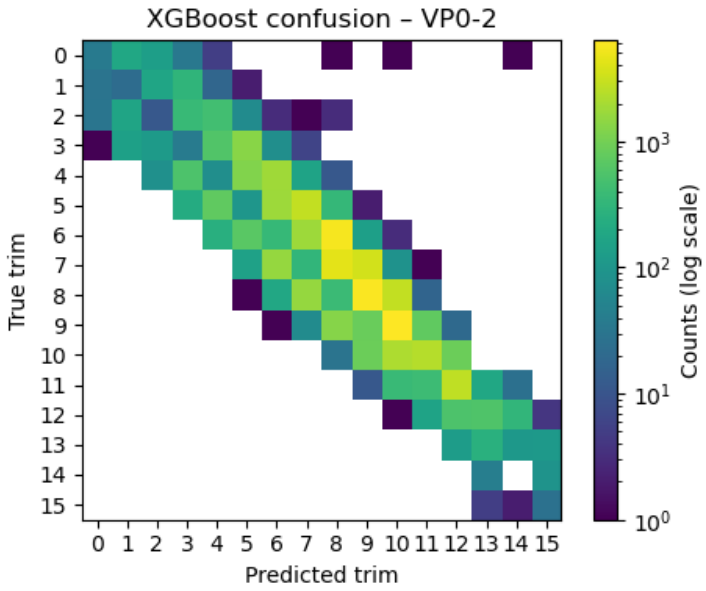


Figure 12: Log-scaled confusion matrix for XGBoost predictions on a single dataset.

Figure 12 shows the log-scaled confusion matrix for one (256×256) dataset. Across all 60 evaluation tiles the BDT achieved an average accuracy of 95.% within ± 1 trim value; well above the 55% obtained in an early, untuned test and only 3% below the best neural-network result (98.6%), a surprisingly impressive result. Training and evaluation time was on a similar scale to that of the final neural network. The confusion pattern shows a clear broader spread along the diagonal than the neural-network equivalent (Figure 16), indicating that trees struggle to capture the most subtle correlations present in the noise–trim relationship. For operational (2025) data the BDT delivered 81.4% (± 1), slightly lower than the network's 83.14%, confirming that both methods are similarly sensitive to shifting detector conditions.

5.5 Final Model

The final model is built up of the architectural and processing elements discussed in the previous sections and is the result of months of testing.

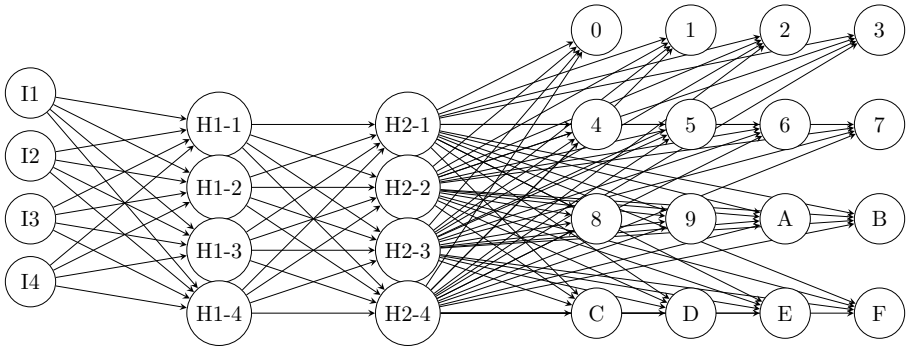


Figure 13: Final network topology diagram. The architecture features an input layer along with 2 hidden layers all made of 4 nodes, and an output layer with 16 nodes (0–F), reflecting the 16 different trim values.

Figure 13 displays the final design of the network, which features an input layer built of 4 nodes (one for each input feature) that receives the processed pixel data. This data then passes through two hidden layers that capture complex patterns in the data using ReLU activation functions. Finally there is an output layer consisting of 16 nodes corresponding to the different trim values (0–F in hexadecimal). A SoftMax activation function in the output layer generates probabilities for all these nodes, with the highest probability indicating the predicted trim value.

This model design is a result of important decisions, removing the batch normalisation layer in favour of manual normalisation, filtering out faulty pixels and tuning the network's parameters. Each part of the network and its preprocessing helps it to predict pixel thresholds to the best of its ability.

This final model is used for all the results presented in the following section, where its performance is thoroughly evaluated.

6 Results and Analysis

This section presents the results recorded from the final network model and evaluates it's performance on both construction and operational datasets. The model is evaluated using accuracy and training time.

6.1 Results

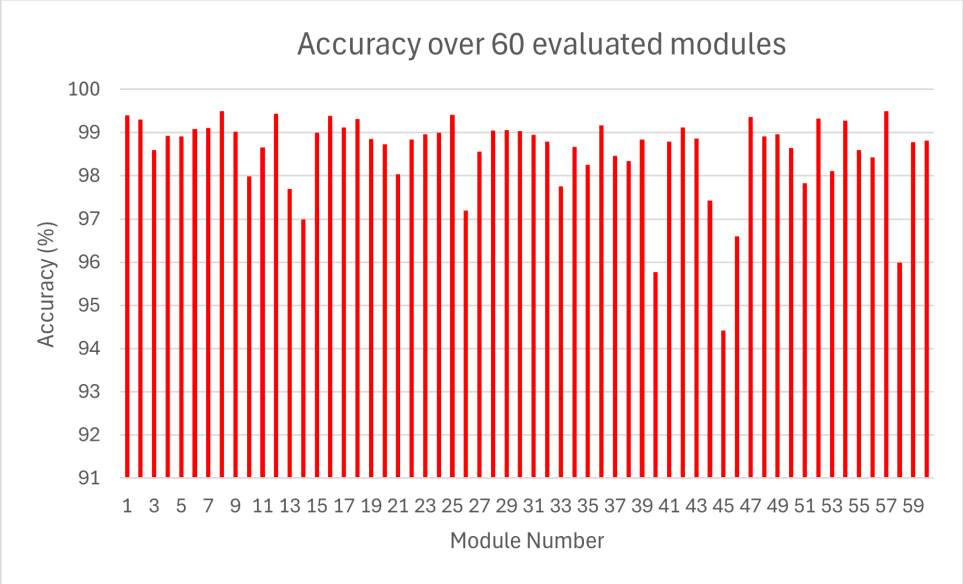


Figure 14: The evaluation results (accuracy) of 60 datasets across 10 different modules construction data. Average = 98.55%, Variance = 0.924 and model took 19.57s to train.

Figure 14 shows the evaluation results from 60 different datasets, with an average prediction accuracy of 98.55%. This high average accuracy shows how effective the network can be when trained on data similar to the evaluation data. The 60 evaluated datasets were randomly selected from a total of 120 available datasets, obtained from 10 of the 56 VELO modules (each module contains 12 datasets). For training, five of the datasets providing different distributions were used to train the network and removed from evaluation. With a training time of just 19.57 seconds on an 11th Gen Intel i7-11370H (3.30GHz), it can be seen that the network very quickly learned patterns in the data involved in predicting the trim values. Additionally, the variance of 0.924 also shows the network is very consistent, with most results being within 1% of the mean accuracy.

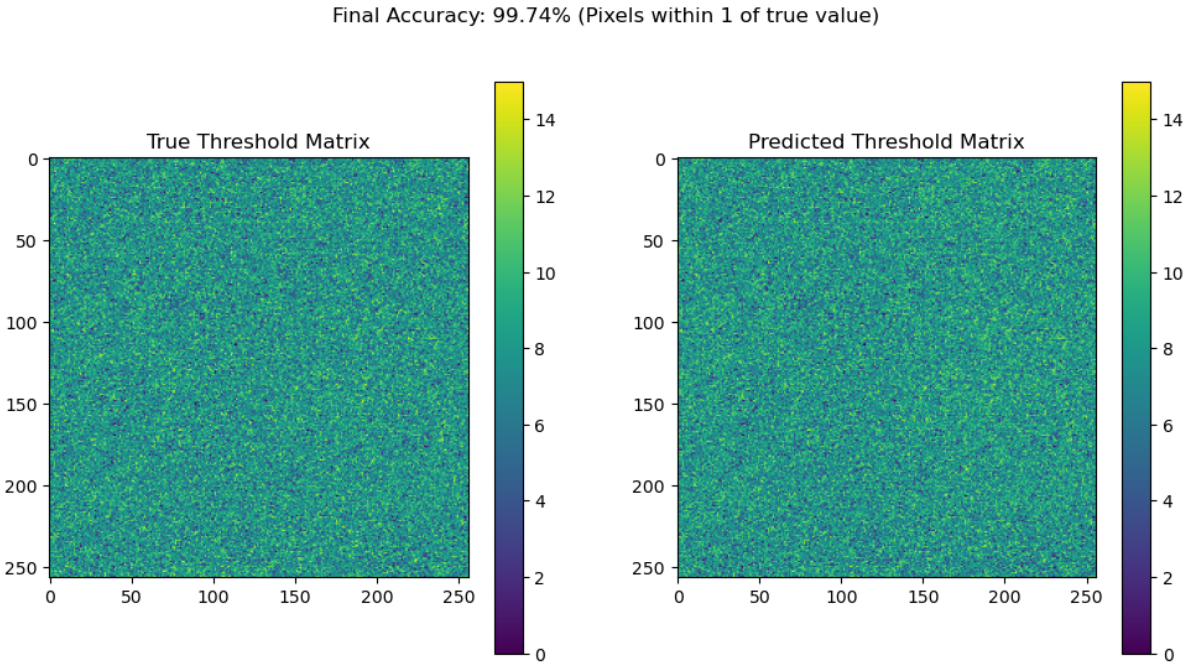


Figure 15: The true trim matrix side by side with a complete prediction of a (256×256) *matrix. Faulty pixels added back in with the predicted values of 0 and the total accuracy recorded in the heading.*

Figure 15 displays a side by side comparison of the true trim matrix and the network's complete prediction for a single (256×256) array. Any pixels filtered out during training (faulty pixels or containing invalid data) are added back with predicted values of 0, resulting in a full final prediction map. This full array is the most successful prediction from the results with a final accuracy of 99.74% of pixels being within 1 of the true trim value, highlighting the peak performance of this network model.

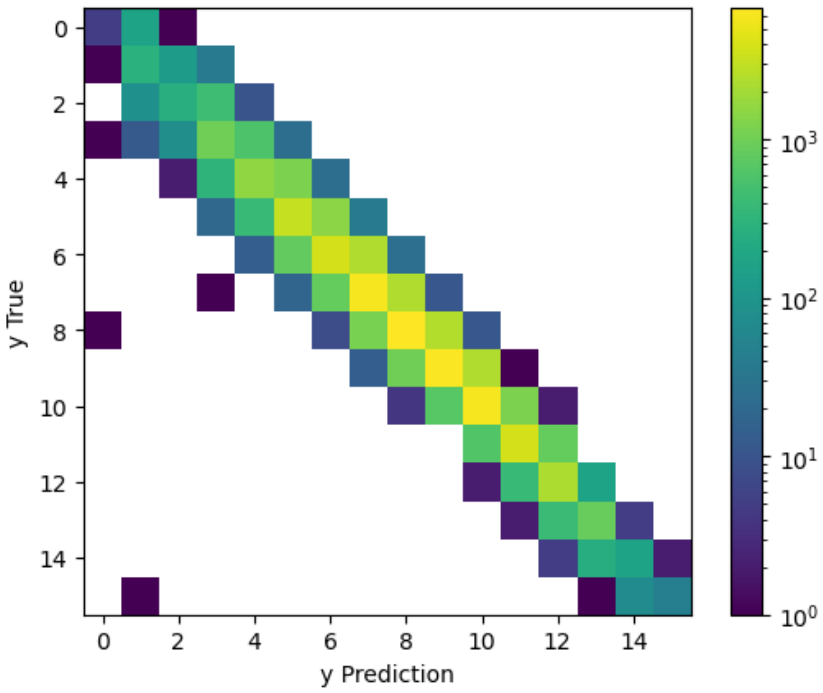


Figure 16: The confusion matrix showing true values in the y axis against the predicted values in the x. Uses a log frequency scale and represents the same dataset as in figure 15.

Figure 16 presents the confusion matrix of the networks predictions, with true trim values on the y and predicted values on the x-axis, using a log frequency scale. It shows that almost all predictions fall within a small margin of the true trim values (typically within 2 integers), indicating that even the guesses not being rewarded by the accuracy function are still very close. Only 3 points on the plot can be seen outside of a reasonable range, each corresponding to about one prediction. Although this shows that the network can sometimes be very far off, the incredibly low frequency of this for an entire (256×256) array demonstrates the consistency and accuracy of the network.

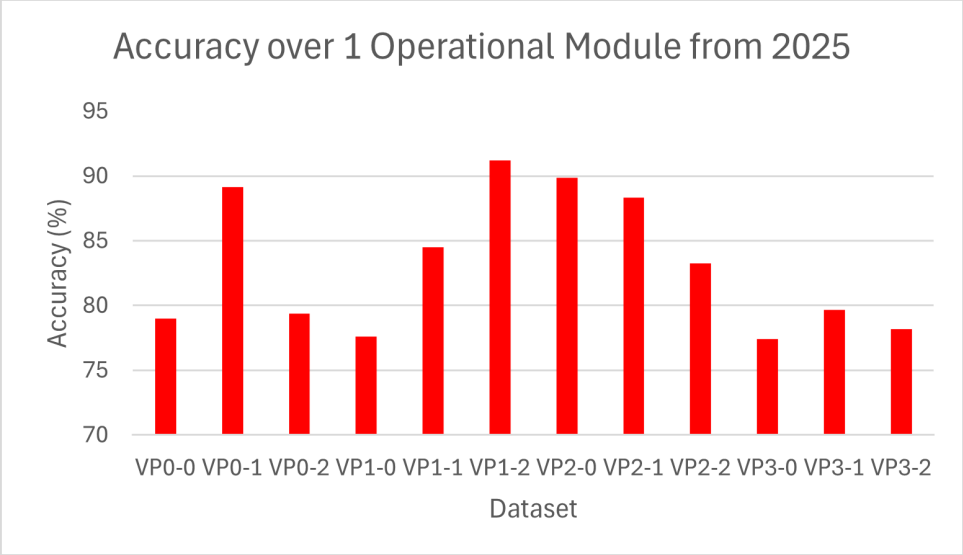


Figure 17: The evaluation results (accuracy) of 1 module from 2025 operational data. Average = 83.14% and uses same model as figure 14.

Finally, Figure 17 illustrated the model's performance on a single module from the 2025 operational data, achieving an average accuracy of just 83.14%. This notable drop in accuracy from the construction data suggests that the model, which was trained on construction data, struggles under operational conditions. This performance drop-off may be attributed to both: differences in operational and construction conditions and the gradual effects of irradiation over time [3]. Possible solutions to this drop in performance will be discussed in the following sections.

6.2 Analysis

As shown in the results the network displays exceptional accuracy and consistency on construction data, which it was trained on, achieving almost perfect predictions at times. However as shown from the networks performance on the construction data as conditions differ the network's performance drops off significantly. Because conditions inevitably change over time, due to multiple factors such as irradiation leading to higher currents being needed [3]. The network's performance will drop off unless retrained on updated full noise scan data periodically to capture shifts in conditions and therefore data.

7 Discussion

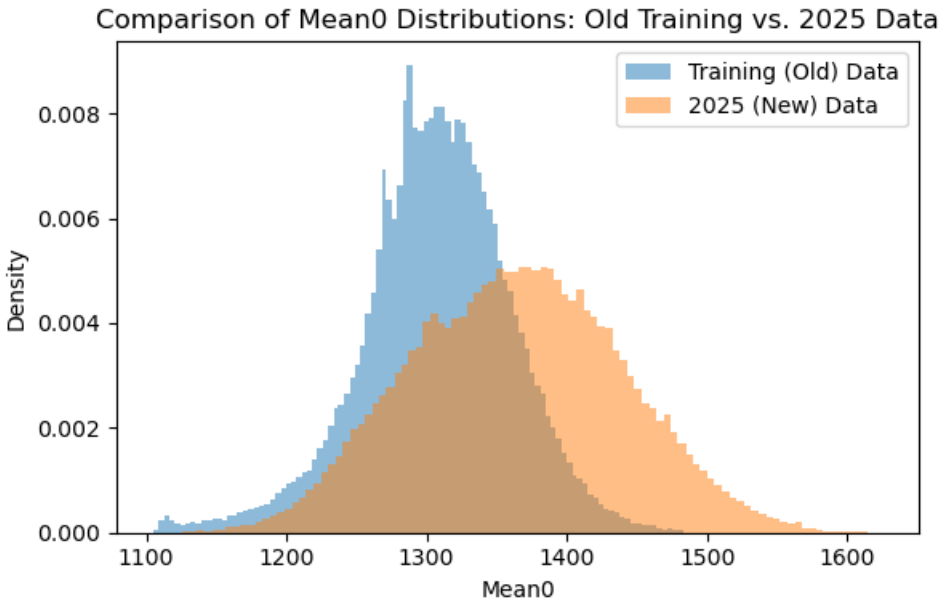


Figure 18: The mean at 0 distributions of the merged training dataset and the 2025 operational data, presented on the same graph

Figure 18 compares the distributions of the mean 0 input feature between all the training data combined and the entire 2025 operational module data on a single graph. The noticeable change between these 2 distributions highlights the changes in sensor behaviour under operational conditions after 3 years of use, compared with the construction environment used for training. It can be seen that in the 2025 data there is a clear shift to a higher mean along with a much less defined peak, suggesting sensors have become more varied in operational conditions. This suggests that changes, possibly factors such as irradiation and sensor ageing, are affecting the values.

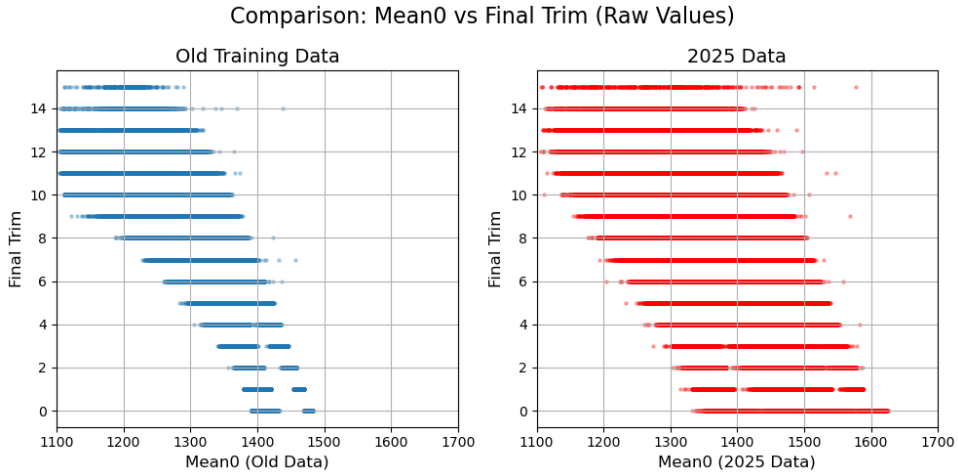


Figure 19: The mean0 vs final trim of all the training data and the 2025 operational data, presented on side by side plots.

Figure 19 reinforces this by presenting side by side plots of the relationship between

the mean 0 input feature and the final trim, for both datasets. The clear difference in these plots show how the correlation has changed over time and under operational conditions. Together, these two figures display that as conditions change, so do the relationships between inputs and the final trim values. This highlights the need for periodic retraining of the network using full noise scans to ensure the network maintains high prediction accuracy as conditions change.

Due to these observations, further research is required to monitor how sensor conditions evolve over time. By recording shifts in thresholds, noise and the correlation between input features and final trim values, it may be possible to identify an optimal interval for retraining the network. This would allow for the development of a retraining schedule that maintains high prediction accuracy even as gradual changes, due to irradiation for example, occur. Ultimately refining the calibration process this way would reinforce and expand the application of machine learning methods in detector calibration.

Research indicates that machine learning methods have been applied in various aspects of the LHCb experiment, for instance in particle identification [9]. However, no studies have been found showing a similar application to this project in calibrating the detector. While this prevents a direct comparison of results, the findings of this project establish a very promising foundation for applying machine learning to enhance detector calibration times.

7.1 Future Work

Several further studies would strengthen the model and move it towards being a fully operational calibration process and were not able to be explored due to time constraints.

Currently the operational test used just one 2025 module. Adding more operational data would confirm whether the accuracy loss is systematic or just module specific.

Re-plotting the Mean-Trim correlations for the same modules at regular intervals (from 2022-2025) would quantify how quickly sensor conditions change. The resulting trend could be used to set an evidence-based re-training period.

Pursuing these directions would convert the baseline method presented here into a dynamic calibration system capable of supporting the VELO throughout its operational lifetime.

8 Conclusion

The upgraded VELO's 40 million pixels, currently require a full 16 step noise scan to determine their optimal trim (threshold) values. This procedure, although accurate, is time consuming and forces sensors to remain powered, generating additional heat. Prolonged operation at higher temperatures increases radiation damage and shortens detector lifetime [3]. Developing a fast calibration method that can infer the trims from just a fraction of the scan would cut calibration time and thermal load, prolonging detector life.

Instead of using a full 16 step noise scan, the calibration task was recreated as a 16 class classification problem. A neural network is given the noise width and mean at the two extreme thresholds (0x0 and 0xF) and predicts the optimal trim value. After data cleaning and normalisation steps explained within this report, the final architecture was able to reach a 98.6% average accuracy (± 1 trim level) on construction data while training in under 20s on an Intel i7-11370H CPU.

On the 60 construction datasets saved for evaluation, the network achieved a mean accuracy of 98.6% within one trim unit. The confusion matrix (Figure 16) shows that ¿ 99% deviate at most by two trim level, and only 3 individual pixels within that single (256×256) dataset fell outside of that range. After adding masked and NaN pixels back (with a trim=0), the reconstructed maps reproduce the trim patterns with minimal difference (Figure 15). The model displays effective performance on the construction data while having under 20s training time and < 2s of prediction time per dataset on CPU hardware [10].

On the other hand, when applied to a 2025 operational module, average accuracy dropped to 83.1%. Histogram and scatter comparisons (Figures 18 and 19) show a clear shift in the noise mean and a broadening of its distribution, changing the correlation learned during original training. The shift is consistent with differences in operational conditions along with threshold drift after 3 years of operation and therefore irradiation. These results emphasise that a static model trained on construction data only stays reliable whilst conditions remain similar, suggesting a need for dynamic retraining.

The presented neural network approach reduces the calibration from a 16-step scan to just the two extreme settings whilst preserving $> 95\%$ accuracy on construction data. Although accuracy degrades under newer conditions, the method shows a clear success to this application, with potential to be retrained on updated scans. These results show that neural network calibration can substantially shorten VELO calibration time and mitigate thermal stress, providing the groundwork for an adaptive trim calibration system that involves with the detector through its lifetime.

8.1 Personal Reflection

Working on this project provided opportunities to develop new skills and experience, as detailed below:

Computational skills: Majority of project time was spent getting experience with Keras/TensorFlow for building the network and XGBoost for a boosted-tree comparison. Along the way the code had to cope with practical issues such as unpacking large .tgz archives, reshaping (256×256) tables, writing a normalisation function to keep the 0 to F correlations intact, alongside the challenges of array shape mismatches, NaNs, and the effect of batch-normalisation. All are examples of problems being isolated and resolved, reinforcing a systematic approach to troubleshooting.

Research approach: The work followed a research based iterative approach: hypothesise, build, test, fix, repeat. Keeping a lab-book of methods, figures and thoughts ensured results could be reproduced and method could be tracked. Technical jargon such as; trim DACs, noise widths and learning schedules, were rewritten in simple language for this report and figures were produced to publication standard. Communicating the methodologies clearly was just as important as the method itself.

Lessons for the future:

- Looking at operational data earlier would have shown threshold drift sooner and potentially changed design choices.
- Speeding up first semester development would have created room later for extra steps such as scheduled retraining and more operational dataset tests.
- Trying a boosted-decision tree baseline proved how a neural network was advantageous and how it picked up on complex patterns more significantly.

9 Bibliography

References

[1] Lhcb – the large hadron collider beauty experiment. `https://home.cern/science/experiments/lhcb`.

[2] Peter Svihra. The design and construction of lhcb velo upgrade modules. In *2019 IEEE Nuclear Science Symposium and Medical Imaging Conference, NSS/MIC 2019*, 2019 IEEE Nuclear Science Symposium and Medical Imaging Conference, NSS/MIC 2019, United States, April 2020. IEEE. 2019 IEEE Nuclear Science Symposium and Medical Imaging Conference 2019, NSS/MIC ; Conference date: 26-10-2019 Through 02-11-2019.

[3] K. Akiba et al. Radiation damage effects and operation of the lhcb vertex locator. *IEEE Transactions on Nuclear Science*, 65(5):1127–1132, 2018.

[4] K. Akiba et al. The lhcb velo upgrade module construction. *Journal of Instrumentation*, 19(06):P06023, jun 2024.

[5] Keras. `https://keras.io/`.

[6] Tensorflow. `https://www.tensorflow.org/`.

[7] Shruti Jadon. Introduction to different activation functions for deep learning. `https://shorturl.at/SF7Wj`, 2018. Accessed: March 14, 2025.

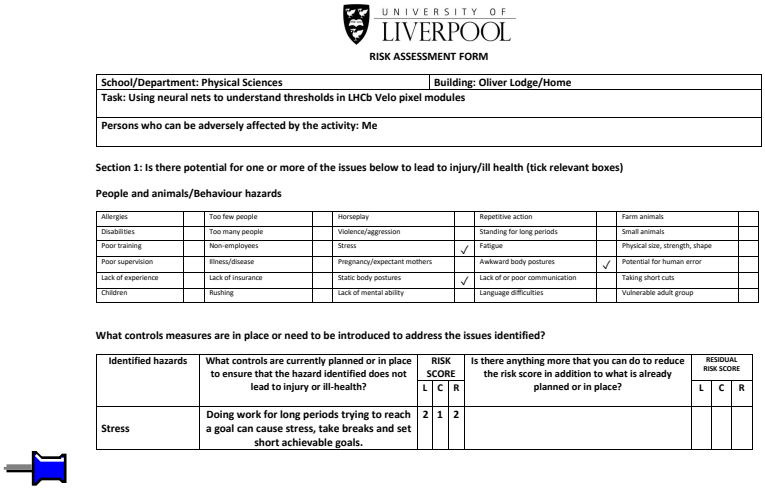
[8] XGBoost Developers. *XGBoost Documentation (release 3.0.0)*, 2024. Accessed: 17 Apr 2025.

[9] Nikita Kazeev et al. Machine learning for particle identification in the lhcb detector. 2020.

[10] ASUS. ASUS TUF Dash F15 FX516PM Technical Specifications, 2021. System used in this work; Intel Core i7-11370H CPU (no GPU usage).

10 Appendix

10.1 Risk Assessment



UNIVERSITY OF LIVERPOOL

RISK ASSESSMENT FORM

School/Department: Physical Sciences	**Building: Oliver Lodge/Home**
Task: Using neural nets to understand thresholds in LHCb Velo pixel modules	
Persons who can be adversely affected by the activity: Me	

Section 1: Is there potential for one or more of the issues below to lead to injury/ill health (tick relevant boxes)

People and animals/Behaviour hazards

Allergies		Too few people		Horseplay		Repetitive action		Farm animals	
Disabilities		Too many people		Violence/aggression		Standing for long periods		Small animals	
Poor training		Non-employees		Stress	✓	Fatigue		Physical size, strength, shape	
Poor supervision		Illness/disease		Pregnancy/expectant mothers		Awkward body postures	✓	Potential for human error	
Lack of experience		Lack of insurance		Static body postures	✓	Lack of or poor communication		Taking short cuts	
Children		Rushing		Lack of mental ability		Language difficulties		Vulnerable adult group	

What controls measures are in place or need to be introduced to address the issues identified?

Identified hazards	What controls are currently planned or in place to ensure that the hazard identified does not lead to injury or ill-health?	RISK SCORE			Is there anything more that you can do to reduce the risk score in addition to what is already planned or in place?	RESIDUAL RISK SCORE		
		L	C	R		L	C	R
Stress	**Doing work for long periods trying to reach a goal can cause stress, take breaks and set short achievable goals.**	**2**	**1**	**2**				

Filename: `RisksProject.pdf`

10.2 Report Plan

Project Report Planning:

Aim to take the reader on journey through the project in logical steps:

Use figures and references where appropriate to make report more interesting than a page of my thoughts

Introduction:

Problem Introduction: Introduce the problem, explain LHCb and VELO detector and then talk briefly about time consuming nature of calibration and how it can be improvedmoving into next section

Motivation: Explain in depth why this is an issue and is calling for possible solutions, (faster calibration times between fills allows for power to be switched off sooner, allowing for cooling, mitigating radiation damage)

Reference: https://ieeexplore.ieee.org/document/8334647

Rewrite in own words and use: Rapid calibration of the sensor thresholds offers significant operational benefits beyond mere time savings. Although the full scan itself is shorter than the LHC recycle time, it requires powering the sensors—which leads to self-heating. By reducing the calibration duration, the detectors can be powered off more quickly, allowing the cooling system to restore the optimal low temperature (from approximately –15°C to –30°C) sooner. Maintaining these lower temperatures is crucial for mitigating radiation damage, as confirmed by studies on the VELO sensors. Faster calibration, therefore, minimizes thermal stress and helps preserve the sensors' radiation tolerance, ensuring sustained performance over the experiment's lifetime.

Solution Introduction: introduce theory and background information of neural networks and why they offer a potential for an improved solution

Methodology:

Data Analysis pre network: explain physical findings from analysing the data ourselves before throwing it into network (mask > 0 resulting in same trim values, statistical correlations between different data, coordinates not showing any correlation, mean being strongest, width being weaker etc)

Add in coordinate and statistical correlation tests

Initial model: explain initial starting model allowing us to hit the ground running and explain problems with it such as batch normalisation layer taking away relationships between threshold levels, mask values being included, nan pixels not being filtered etc

Iterations: Explain any large model developments/changes in chronological order explaining the though process behind decisions and why one model is better than previous version/s

Final model: Explain refinements to get to the model used for the results and why this is the most successful model yet, any final thought processes gone into obtaining this model

Other possible solutions: Xtreme gradient boosting and regression model, show results from both under tests and speak on what neural networks do that XGB doesn't and why this is better

Filename: `report_plan.pdf`

10.3 VELO Data Modules

All datasets were obtained via the supervisor's access to the CERN EOS/LHCb storage.

Construction-phase modules (10)

- equalisation_M98_A1
- equalisation_M94_A2
- equalisation_M96_A3
- equalisation_M59_A4
- equalisation_N22_A5

- equalisation_M116_A6
- equalisation_N030_A7
- equalisation_M90_A8
- equalisation_N029_A9
- equalisation_N013_A10

Sets used for network training (5 of the above)

Module	Dataset (VP)
equalisation_N030_A7	VP0-1
equalisation_M59_A4	VP0-0
equalisation_M98_A1	VP0-1
equalisation_N030_A7	VP3-2
equalisation_M98_A1	VP2-0

2025 operational dataset

- Module25_20230621 (12 sets: VP0-0 ... VP3-2)