

Smart pixel sensors: towards on-sensor filtering of pixel clusters with deep learning

Jieun Yoo¹, Jennet Dickinson², Morris Swartz³, Giuseppe Di Guglielmo^{2,4}, Alice Bean⁵, Douglas Berry², Manuel Blanco Valentin⁴, Karri DiPetrillo⁶, Farah Fahim^{2,4}, Lindsey Gray², James Hirschauer², Shruti R. Kulkarni⁷, Ron Lipton², Petar Maksimovic³, Corrinne Mills¹, Mark S. Neubauer⁸, Benjamin Parpillon^{2,1}, Gauri Pradhan², Chinar Syal², Nhan Tran^{2,4}, Dahai Wen³, Aaron Young⁷

¹ University of Illinois Chicago, Chicago, IL, 60607, USA

² Fermi National Accelerator Laboratory, Batavia, IL 60510, USA

³ Johns Hopkins University, Baltimore, MD 21218, USA

⁴ Northwestern University, Evanston, IL 60208, USA

⁵ University of Kansas, Lawrence, KS 66045, USA

⁶ The University of Chicago, Chicago, IL 60637, USA

⁷ Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

⁸ University of Illinois Urbana-Champaign, Champaign, IL 61801, USA

Abstract. Highly granular pixel detectors allow for increasingly precise measurements of charged particle tracks. Next-generation detectors require that pixel sizes will be further reduced, leading to unprecedented data rates exceeding those foreseen at the High Luminosity Large Hadron Collider. Signal processing that handles data incoming at a rate of $\mathcal{O}(40\text{MHz})$ and intelligently reduces the data within the pixelated region of the detector *at rate* will enhance physics performance at high luminosity and enable physics analyses that are not currently possible. Using the shape of charge clusters deposited in an array of small pixels, the physical properties of the traversing particle can be extracted with locally customized neural networks. In this first demonstration, we present a neural network that can be embedded into the on-sensor readout and filter out hits from low momentum tracks, reducing the detector's data volume by 54.4-75.4%. The network is designed and simulated as a custom readout integrated circuit with 28nm CMOS technology and is expected to operate at less than $300\ \mu\text{W}$ with an area of less than $0.2\ \text{mm}^2$. The temporal development of charge clusters is investigated to demonstrate possible future performance gains, and there is also a discussion of future algorithmic and technological improvements that could enhance efficiency, data reduction, and power per area.

1. Introduction

High granularity silicon pixel detectors are crucial for disentangling the tremendous numbers of particles produced at high energy colliders. With billions of readout channels and event rates as high as 40 MHz, these pixel detectors generate petabytes of data per second. To quickly extract the pixel information necessary for high priority physics, we propose to develop intelligent on-chip data reduction. This paper specifically investigates a neural network that can selectively read out pixel clusters based on the incident particle's momentum.

As the subsystems closest to the interaction point, pixel detectors provide precision spatial measurements that play an essential role in pattern recognition, vertexing, and particle momentum measurements. A traversing charged particle creates electrical signals in a cluster of pixels that, in combination with the pixel sensor's location, provide precise 3D measurements to seed pattern recognition. The pixel size and proximity to the interaction point determine the track impact parameter and momentum resolution. Both the desired spatial precision and the expected density of charged particles are used to decide the pixel dimensions. The current generation ATLAS [1] and CMS [2] experiments at the Large Hadron Collider (LHC) contain pixel detectors with pitches of $50 \times 250 - 400 \mu\text{m}^2$ and $100 \times 150 \mu\text{m}^2$, respectively, and a thickness of $\mathcal{O}(300 \mu\text{m})$. At the High Luminosity LHC (HL-LHC), the pixels will be reduced to roughly $50 \times 50 \mu\text{m}^2$ in size and $\mathcal{O}(100 \mu\text{m})$ thick [3, 4].

The particle properties extracted from pixel detector data play a crucial role in physics measurements. In high-luminosity environments, vertex information is essential to distinguish the primary interaction from additional proton-proton interactions occurring in the same bunch crossing (pileup). Impact parameter measurements from the pixel detector also provide key information for reconstructing particles with relatively long lifetimes that decay a measurable distance from the beam axis, including the tau lepton, charm quark, and bottom quark. Correct identification of these particles is necessary for several high-priority searches and measurements, such as characterizing the Higgs boson's couplings to second and third generation fermions. In addition, many Beyond the Standard Model scenarios predict new particles that would lead to displaced vertex signatures in the detector. Many hidden valley or dark sector [5] models contain signatures with displaced leptons or "dark QCD" hadronic showers [6]. Hypothesized new particles decaying to soft, unclustered energy patterns (SUEPs) [7] are another distinctive signature, which require pixel information to distinguish it from pileup.

Despite its importance to collider physics programs, pixel detector information is difficult to read out. The large data volume calls for multiple methods of data reduction. Zero suppression is employed to read out only active pixels, and the accumulated charge per pixel is digitized and represented with only a few bits. Even with these techniques, data rates at the current CMS and ATLAS experiments exceed bandwidth constraints for read out at the collision frequency of 40 MHz. Collisions of interest are selected using a hardware-based

trigger system, which uses information from the other (non-pixel) detector subsystems to select events at a rate of < 1 MHz. Events that contain new physics only in the pixel data are not selected by the low-level trigger and are lost forever. Furthermore, the more granular pixel detectors built for the HL-LHC and beyond will lead to even higher data rates, which further increases the need for data reduction.

In this paper, we seek to overcome the limitations of pixel readout with local data reduction in dedicated circuits before transmitting information off of the detector. While pixel readout has traditionally been treated as a *lossless compression* task, we explore the paradigm of lossy compression within a given event to enable lossless readout over all possible collisions. To enable this strategy with optimal performance and yet keep the algorithms reconfigurable, we explore the use of neural networks with reprogrammable weights. A machine learning approach is required due to the complicated pulse shapes, a combination of drift and induced currents, generated by the pixel detector. The contribution from those two components is dependent on the sensor geometry where the charged particle impacts the detector and the particle's trajectory. As a first benchmark task, we develop, design, and study a neural network that will filter out pixel clusters originating from low-momentum charged particle tracks. We then optimize, design, and simulate that neural network in an integrated circuit and study its performance, power, area, and latency.

The layout of this paper is as follows. The pixel geometry we assume for this study and the simulated pixel sensor data is detailed in Section 2. Then in Section 3, we introduce the benchmark pixel cluster filtering task, the algorithm designed and optimized for the task, and its performance. We then detail a preliminary implementation of the algorithm for deployment on-sensor in Section 4. Finally in Section 5, we summarize the results from this study and discuss the additional benchmark tasks and their benefits, as well as ways to further improve the performance and efficiency of the algorithm.

Related Work

Data compression of silicon tracking information, in both strip and pixel detectors, is explored in Ref. [8–10]. These works contain detailed studies of both lossless and lossy compression of the digital information content of the pixel and strip tracker detectors. These references provide a good baseline for understanding what data compression factor is possible for current tracking systems. Our work expands on these previous studies by: looking at analog and timing information; considering future pixel dimensions, studying the potential for cluster filtering by the track momentum; and designing a first implementation of on-chip detector algorithms.

Ref. [11] explores the use of pixel cluster shapes offline (off-detector) to extract direction information and reduce tracking combinatorics and complexity. The technique described in this study can be used in the future to extract directional information from charge clusters in a single pixel layer, providing similar benefits in terms of reduced algorithm complexity for tracking downstream, which is particularly important for online data processing systems.

Finally, our study relies on previous work for translating neural network algorithms into circuits using the `hls4ml` [12, 13] workflow. In particular, the first implementation using `hls4ml` to build a reconfigurable ASIC [14] for calorimeter on-detector data compression provides a basis for much of the technology developed in this paper.

2. Sensor geometry and dataset

2.1. Simulated data

The studies in this paper are based on a simulated dataset of silicon pixel clusters produced by charged particles (pions) [15]. The kinematic properties of the incident particles are taken from fitted tracks in CMS 13 TeV collision data. Figure 1 shows the p_T distribution of these particles in blue. Because particles with very low transverse momentum (p_T) are not reconstructed as tracks in CMS, the distribution turns on above 100 MeV. The corresponding p_T distribution corrected for losses due to inefficiency of the CMS tracker is shown in orange.

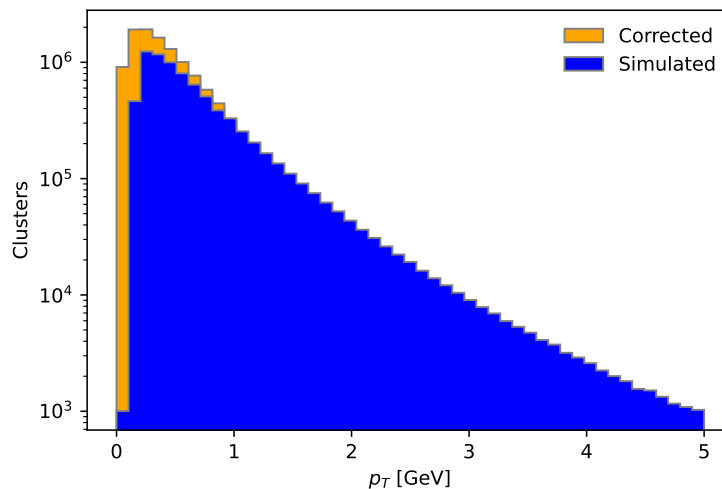


Figure 1: Transverse momentum (p_T) of simulated particles (blue) and the p_T distribution corrected for tracking inefficiency (orange).

To study a concrete sensor configuration, we make the following assumptions about our future pixel sensors:

- The sensor plane is described by coordinates x and y , while the z direction is normal to the sensor. The pixel pitch is taken to be $50 \mu m \times 12.5 \mu m$ in $x \times y$.
- The overall pixel sensor area is $16 \times 16 \text{ mm}^2$ and its thickness is $100 \mu m$.
- The sensor is situated on a cylinder of radius 30 mm, with the particle's origin at its center. Particle interactions are simulated at varying positions of the sensor along the cylinder's axis.
- A bias voltage of -100V is applied.

- The detector is immersed in a 3.8 T magnetic field parallel to the x coordinate.

The detector response is simulated using a time-sliced version of PixelAV [16], which provides: an accurate model of charge deposition by primary hadronic tracks (in particular to model delta rays), a realistic electric field map resulting from the simultaneous solution of Poisson's Equation, carrier continuity equations, and various charge transport models, an established model of charge drift physics including mobilities, Hall Effect, and 3-D diffusion, a simulation of charge trapping and the signal induced from trapped charge, and a simulation of electronic noise, response, and threshold effects. A particularly valuable aspect of PixelAV used in this study is time evolution of the drift and induced currents in the pixel sensor.

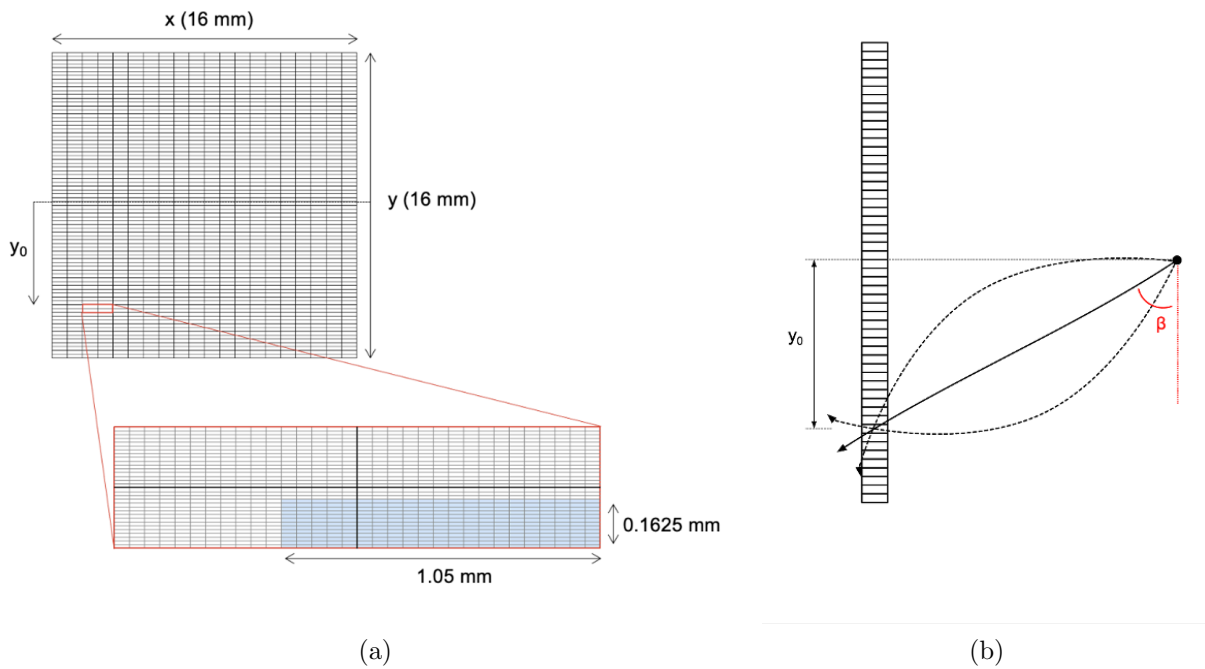


Figure 2: (a): A schematic of the pixel sensor area and the specific region of interest (blue) of 21×13 pixels for a given cluster. The magnetic field is parallel to the sensor x coordinate. (b): A diagram of three charged particles traversing our simulated silicon sensor at the same y_0 position. The sensor is viewed in the bending plane of the magnetic field. The solid track corresponds to a charged particle with high p_T , while the two dashed tracks correspond to low p_T particles with opposite charge.

Figure 2 sketches out key features of the pixel sensor and corresponding strategies employed by this paper. Within the pixel sensor area, we define a *cluster region of interest*, shown in blue, which corresponds to 21×13 pixels in x and y , respectively. This region is large enough to fully encompass a charge cluster and serves as input to the ML algorithm used to extract cluster features. The position (x, y) where the charged particle traverses the sensor mid-plane is uniformly distributed across the central 3×3 pixel array. The shape

of the charge deposited in the pixel array is sensitive to this position and to the particle's angle of incidence. The incident angle in the $x - z$ plane is denoted by α , and by β in the $y - z$ plane. Due to the bending of charged particle tracks in the magnetic field, the shape of the charge cluster also depends on the particle's p_T , which is highly correlated with β . The shape of the cluster also depends strongly on its azimuthal position with respect to the center of the sensor, which is denoted by the coordinate y_0 .

For a given cluster, the sum over pixel columns projects the cluster shape onto the x -axis: this distribution is referred to as the x -profile. The sum over pixel rows, y -profile, which projects the cluster shape onto the y -axis, is sensitive to incident angle β and therefore to the particle's p_T . Two example clusters are shown in Figure 3 with the corresponding x - and y -profile projections.

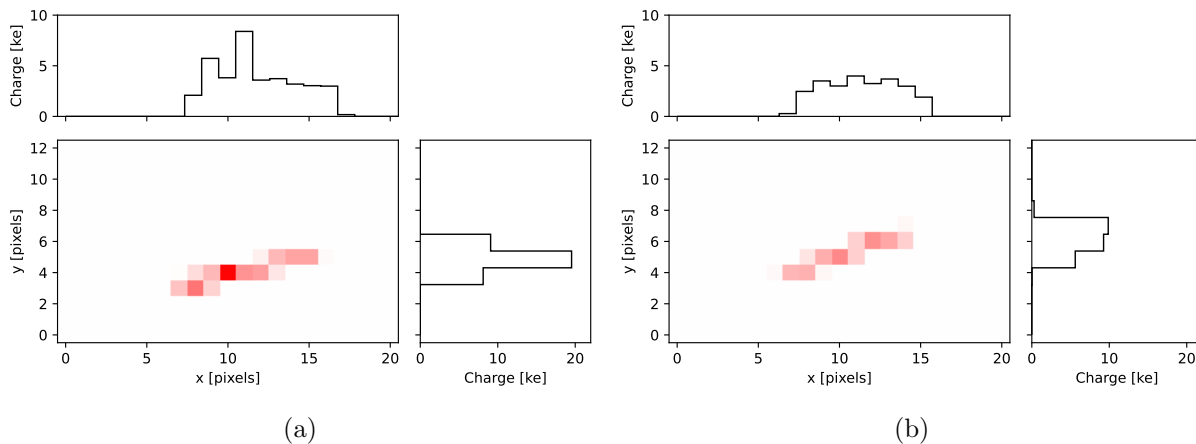


Figure 3: Two example charge clusters and the corresponding x - and y -profile projections. The color scale (common between the panels) represents the collected charge. Both clusters have $y_0 = 2.3$ mm, but different p_T : (a) $p_T = 1.9$ GeV, (b) $p_T = 135$ MeV.

The mean y -profile cluster charge distributions for particles impinging near the center of a sensor ($-1 < y_0 < 1$ mm) are shown in Figure 4a for three populations of clusters. Clusters created by high p_T particles ($p_T > 2$ GeV) are represented by the black distribution, while clusters created by low p_T particles ($p_T < 200$ MeV) are represented by red and blue for positively and negatively charged particles, respectively. Due to the deflection of charge carriers by the magnetic field (Lorentz drift), the cluster shape is not symmetric in β . For a flat detector module that measures 16 mm in y , particles of similar momentum leave markedly different cluster shapes at different y_0 positions on the module. Figures 4b-c show the average cluster shapes at the extreme edges of the module.

The cluster y -size is defined as the number of pixel rows in which non-zero net charge has been deposited after 4 nanoseconds. The dependence of the cluster y -size on both the charged particle p_T and y_0 is shown in Figure 5. The decrease in y -size from the left to right edge of the sensor is due to Lorentz drift. In order to study the potential gain from timing

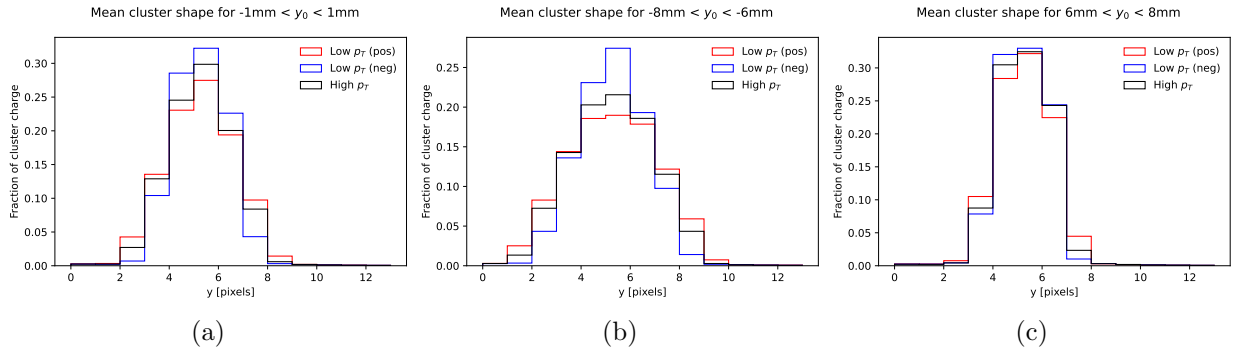


Figure 4: Distribution of y -profile at different y_0 : (a) $-1 < y_0 < 1$ mm, (b) $-8 < y_0 < -6$ mm, and (c) $6 < y_0 < 8$ mm. Shown separately for positively charged low p_T particles (red), negatively charged low p_T particles (blue) and high p_T particles of both signs (black).

information, the simulation also records the induced and collected charge in each pixel at 200 picosecond time intervals.

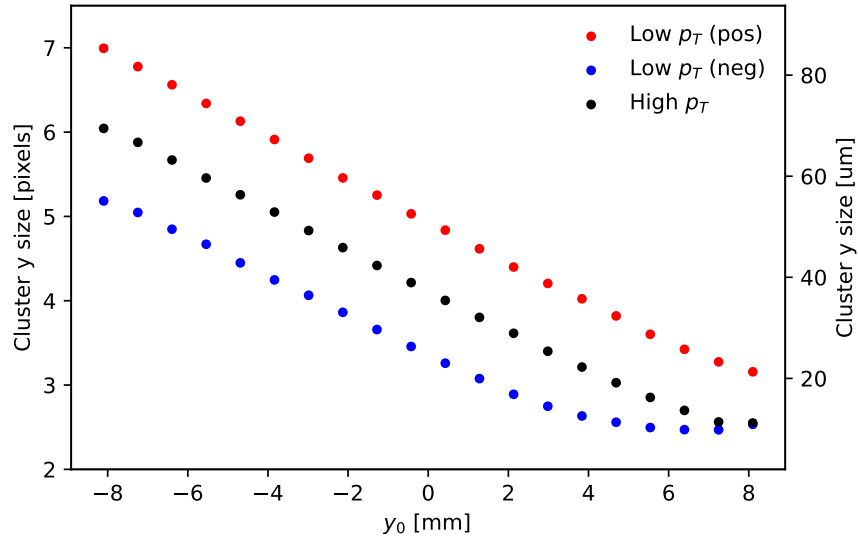


Figure 5: Cluster y -size vs. y_0 for different values of particle charge and p_T . The decrease in cluster size from the left to right side of the sensor plane is due to Lorentz drift.

2.2. Untracked data

The simulated dataset described above is derived from clusters in the CMS detector that are combined with signatures in other detector layers to form particle tracks. In an example 2022 CMS data taking run, only about 40% of clusters in the innermost layer of the CMS pixel detector are associated with tracks in this way. The remaining 60% of clusters, designated

as *untracked*, can result from a variety of sources, including very low p_T particles, radiation related backgrounds, or detector effects.

Such untracked clusters are difficult to model, but critical to include in an estimate data reduction. A dedicated dataset has therefore been generated for this purpose based on untracked clusters in CMS data. The untracked clusters from CMS data are used as a proxy for backgrounds that should be rejected at a general collider experiment and are not being used to tune a specific pixel upgrade project. For each untracked cluster measured in CMS, we assume that it was created by a charged pion with energy 1 GeV traversing the detector. The incidence angles α and β of the particle are approximated based on the cluster size:

$$\text{size}_x = \frac{t}{\Delta_x} |\cot \alpha| + 1 \quad (1)$$

$$\text{size}_y = \frac{t}{\Delta_y} \left| \cot \beta + \frac{y_0}{R} \right| + 1 \quad (2)$$

Here, t represents the sensor thickness and Δ_x and Δ_y the pixel pitch in the x and y directions. The degeneracy in the incidence angle due to the absolute value is broken by using measured correlations between β and y_0 and α and z to assign values to each cluster. The (x, y) position where the particle traverses the sensor mid-plane is assumed to be uniform across the central 3×3 pixel array. The response of our futuristic pixel detector to charged pions with the corresponding properties is then simulated.

The resulting description of untracked clusters is approximate, and a more precise description can be derived in the future using data from beam tests.

3. Pixel cluster filtering benchmark and algorithm design

The vast majority of particles produced at the LHC correspond to low-energy hadronic activity that is not pertinent to the studies of high-energy collisions. This is exacerbated by the increased rate of pileup at the HL-LHC compared to previous LHC runs. Rejection of clusters consistent with low p_T particles at the data source, within the pixelated area of a readout chip, would save bandwidth by reducing the amount of data that must be transferred off-chip.

This study focuses on filtering out tracks created by low momentum particles, and therefore uses only the y -profile information which is most relevant for track curvature in the magnetic field. In future studies, we will look to exploit more of the cluster shape information for further data reduction.

3.1. Single cluster p_T filtering algorithm

A neural network classifier is designed to identify clusters initiated by high p_T charged particles. The high p_T signal class contains tracks with $p_T > 200$ MeV. Because oppositely charged particle tracks curve in opposite directions in the magnetic field, the resulting clusters are very different in shape, as shown in Figure 4. Therefore two background classes are

defined, corresponding to positively and negatively charged particles with $p_T < 200$ MeV. The choice of p_T threshold in the training class definition affects the physics performance and could be adjusted depending on the physics goals. In this study, we find that a 200 MeV threshold in the training data results in a flat signal efficiency for track $p_T > 2$ GeV, which we consider to be the useful range for physics analysis.

The simulated dataset of 800,000 clusters is used for training (no untracked data). The simulated dataset is split into a training set (80%) and a test set (20%) to be used for evaluation of the algorithm's performance. While the true p_T distribution is shown in Fig. 1, the training sample provided to the network contains the same number of clusters for each p_T class.

All models were implemented in TensorFlow (v2.10.0) [17] using the Keras API [18]. Neural network trainings were run for 200 epochs where early stopping was used if the loss function showed no improvement after 20 epochs. A batch size of 1024 was used in all models. The Adam optimizer [19] with a learning rate of 0.001 was used in conjunction with the Keras Sparse Categorical Cross entropy loss function in all models. The models are trained with three output categories: positively charged and $p_T < 200$ MeV; negatively charged and $p_T < 200$ MeV; and $p_T > 200$ MeV, both positively and negatively charged. A softmax activation was used in all models to generate classification probabilities between 0 and 1, and each cluster is assigned the classification label corresponding to the highest probability.

The charge cluster shape along the axis parallel to the magnetic field direction (sensor x) is assumed to be largely uncorrelated with track p_T . The projection of the cluster shape onto the sensor y -axis (y -profile) is therefore used as the training input to the classifier. Three training setups are developed corresponding to input features of different complexity in order to demonstrate how additional information improves p_T discrimination:

- Cluster y -size: number of pixel rows with nonzero charge deposited after 4 nanoseconds
- Cluster y -profile: the amount of charge collected in each row of pixels after 4 nanoseconds
- Cluster y -profile with timing: the amount of charge collected in each row of pixels evaluated at eight intervals of 200 picoseconds

In each case, the neural network can be trained using the cluster position on a flat module (y_0) as an additional input feature. Alternatively, separate networks can be trained for each region in y_0 (see Section 4.3). Each model is summarized below.

Model 1: cluster y -size. Two input features are used in Model 1: the cluster y -size and position y_0 . The model consists of one dense layer with 128 neurons and 771 parameters. This model provides a test of performance with minimal information provided to the neural network.

Model 2: cluster y -profile. The cluster y -profile model has fourteen input features: cluster y -profile (thirteen features corresponding to thirteen pixel rows) and the y_0 position (1 feature).

This model consists of one dense layer with 128 neurons and 2307 parameters.

Model 3: cluster y -profile with timing information. The third and most complex model takes as input the cluster y -profile distribution at eight time slices (13×8 features) and the y_0 position (1 feature). The first eight time slices contain the most useful information, as most charge deposition occurs at the beginning of the cluster time evolution. This model uses a convolutional neural network (CNN) to pass a time-lapse picture of the cluster charge to the network. The cluster y -profile inputs were passed through two two-dimensional convolutional layers (Conv2D), with 16 and 64 filters, respectively, using ReLU activations to introduce non-linearity [20]. The shape of the kernels was 3×3 , and strides was 1×1 . The output of the Conv2D layers was flattened and concatenated with the y_0 input. This was then passed through a dense layer with 32 neurons, and using dropout of 0.1. The final model contains 83,331 parameters.

The classifier acceptance is defined as the fraction of clusters that the network selects as $p_T > 200$ MeV. A comparison of the three models in terms of the acceptance is shown as a function of the true p_T in Figure 6. Table 1 compares two figures of merit for each model:

- signal efficiency: the fraction of clusters with $p_T > 2$ GeV that are classified as high p_T
- background rejection: the fraction of clusters with $p_T < 2$ GeV that are classified as low p_T

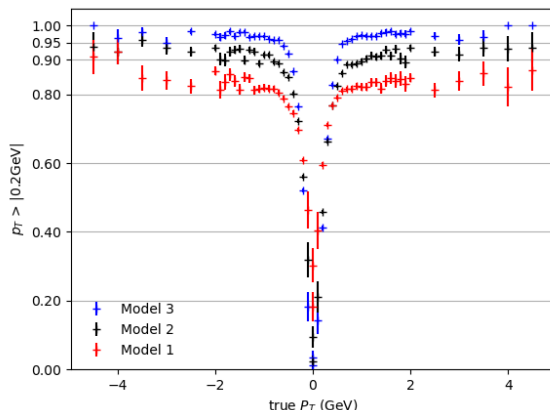


Figure 6: Classifier acceptance as a function of p_T for three models with different input features. Positive and negative values of p_T represent the performance on clusters initiated by particles of positive and negative charge, respectively.

Model 1 (y -size and y_0) has the simplest architecture and achieves the highest data reduction rate. However, the information contained in only two features is insufficient for achieving a high signal efficiency, and this model selects less than 85% of tracks with $p_T > 2$ GeV. Model 2 (cluster y -profile) achieves an accuracy of 93.3% for tracks with $p_T > 2$

Model	Sig. efficiency	Bkg. rejection
Model 1	84.8 %	26.6 %
Model 2	93.3 %	25.1 %
Model 3	97.6 %	21.7 %

Table 1: Comparison of model performance in terms of signal efficiency and background rejection.

GeV, and remains sufficiently compact for implementation on-ASIC. The inclusion of timing information in Model 3 achieves an additional 4% gain in the signal efficiency and is the most accurate overall. However, the extraction of time-sliced charge information presents challenges to the chip architecture that merit further study but remain outside the scope of this work. *Model 2 is therefore chosen as the baseline model for hardware implementation.*

3.2. Estimate of overall reduction in bandwidth

The full dataset that will be seen by our detector is a combination of the simulated dataset described in Section 2.1, the untracked dataset described in Section 2.2, and the single pixel clusters that are not associated with tracks. Due to the small pixel size, all simulated clusters resulting from the traversal of a charged particle consist of at least two pixels. We can therefore assume that single pixel clusters will be rejected out of hand.

The first column of Table 2 shows the expected contribution from each component sample based on a 2022 CMS data run. The second column shows the fraction of clusters in each sample that are rejected by the baseline model (Model 2 in Section 3.1). The fraction of simulated clusters rejected is corrected to account for the CMS tracking efficiency.

	Fraction of dataset	Rejection rate
Simulated tracks	40%	36.3%
Multi-pixel untracked	55%	61.9%
Single pixels	5%	100%

Table 2: Breakdown of the total dataset seen by the detector and the rejection rate achieved on each subsample.

Because the classifier training is agnostic to the multi-pixel untracked dataset, we consider the 61.9% rejection achieved on this sample to be a conservative lower bound. Using this lower bound, the expected fraction of clusters rejected by the classifier is 53.5%. Assuming that 100% rejection can be achieved on multi-pixel untracked clusters gives an upper bound on the fraction of clusters rejected of 74.5%.

Charge clusters vary in size, and the size of the data read out per cluster is proportional

to the number of pixels contained in the cluster. The overall reduction in bandwidth is therefore calculated by weighting each cluster by the number of pixels it contains. The result is a bandwidth savings of 54.4-75.4%.

3.3. Model quantization

The training inputs of the models discussed thus far are the full-precision y -profile distribution (the exact number of electrons simulated in each pixel) and a single-precision floating point value for the y_0 position. However, in order for the p_T filtering algorithm to satisfy the power and area requirements for on-ASIC implementation, the charge collected per pixel must be described by the output of an analog-to-digital converter (ADC). The baseline quantized model uses a 2-bit quantization of y -profile with each output corresponding to a range of collected charge, as summarized in Table 3. The optimization of this input quantization and the size of the ADC is discussed later in this Section. The y_0 coordinate is reduced to a 6-bit input by dividing the range of possible y_0 values into 64 equal bins of $250 \mu m$ width.

ADC output	Charge interval [e^-]
00	< 400
01	400 – 1600
10	1600 – 2400
11	> 2400

Table 3: Mapping between 2-bit ADC output and collected charge.

For implementation on-ASIC, the neural network weights must also be quantized to a precision of a few bits without significant loss in performance. The QKeras library [21, 22] is used to perform quantization-aware training, enabling an early evaluation of the impact of low bit precision on the model’s performance. This allows an initial assessment of the trade-offs between accuracy and resource utilization before finalizing a design for the ASIC implementation (see Sec 4.3). The quantized baseline model consists of a quantized dense layer and a quantized representation of a ReLU activation function. Additionally, a Batch Normalization (BN) layer that provides a regularization effect during training, akin to dropout, was incorporated to prevent overfitting. A softmax activation generates classification probabilities between 0 and 1, and each cluster is assigned the classification label corresponding to the highest probability. The final quantized model has 2819 parameters. The number of bits used for network weights and activation layers were treated as hyper-parameters of the model, and chosen to maximize the signal efficiency. Figure 7 compares the performance for selected combinations of the quantization hyper-parameters following a hyperparameter search.

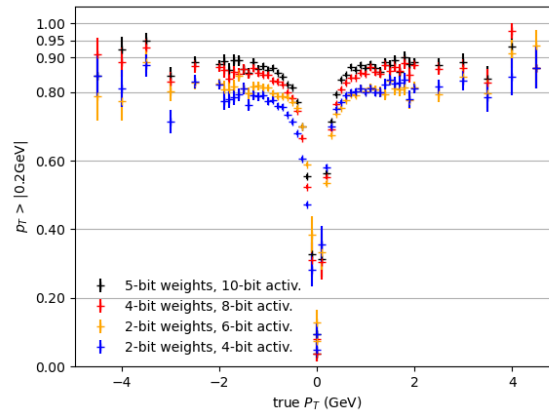


Figure 7: Classifier acceptance vs. p_T for Model 2, with each curve corresponding to a different choice of quantization hyper-parameters. Positive and negative values of p_T represent the performance on clusters initiated by particles of positive and negative charge, respectively.

The model achieving best performance uses five-bit weights and ten-bit activations and is illustrated in Figure 8. However, the model using four-bit weights and eight-bit activations gives only slightly lower efficiency, and is ultimately selected for hardware implementation given resource constraints (see Section 4.3).

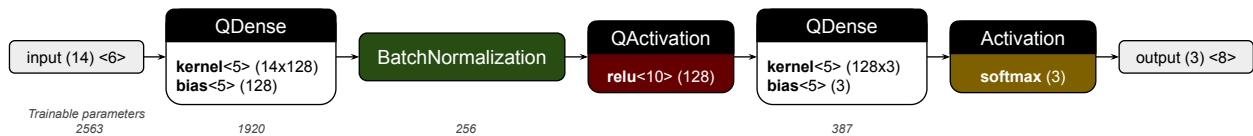


Figure 8: Baseline quantized model consisting of 14 inputs (13 bins of y -profile plus y_0), three outputs, and two hidden layers containing 128 and three neurons each. For each layer, the dimension is reported in round brackets and the bit-width of the fixed-point representation in angular brackets.

The full precision baseline model accurately classifies 93.3% of tracks with $p_T > 2$ GeV as high p_T . Quantizing the collected charge and y_0 inputs reduced this accuracy to 88.8%. The quantization of the neural network weights and activations via QKeras led to a further drop in accuracy of about 1.5%, suggesting that the quantization via QKeras was efficient. Table 4 compares the performance of the full precision model, the model with quantized input features only, and the model with quantized input features and weights.

Model	Sig. efficiency	Bkg. rejection
Full precision	93.3 %	25.1 %
Quantized inputs	88.8 %	25.8 %
Quantized weights & inputs	87.3 %	28.2 %

Table 4: Comparison of the baseline model performance at different stages of the quantization in terms of signal efficiency and background rejection.

Noise threshold Because the p_T filtering algorithm is sensitive to cluster shape, noise hits can be rejected based on both shape and magnitude of collected charge. This allows for the noise threshold (the boundary between ADC output 00 and 01) to be set lower than in present-day pixel detectors, where a typical threshold of ~ 1500 electrons is selected to suppress noise.

Table 5 shows the signal efficiency and background rejection for several example noise thresholds. Setting lower thresholds tends to improve performance as it allows more information to be passed to the neural network classifier. However, because the input y -profile distribution represents a sum over pixel columns, the classifier is not especially sensitive to this threshold. The 400 electron threshold was chosen for the baseline model as it provided good performance and could serve as a realistic threshold for hardware.

Threshold [e^-]	Sig. efficiency	Bkg. rejection
400	87.7 %	27.0 %
600	86.0 %	28.2 %
800	86.0 %	27.1 %

Table 5: Signal efficiency and background rejection for quantized Model 2 using different values of the noise threshold.

Number of bits in the ADC The baseline quantized model uses a 2-bit ADC where the output corresponds to the charge intervals in Table 3. The model was also trained with 1, 2, 3, and 4 bits in the ADC using multiple mappings of charge intervals to ADC value. Figure 9 shows a comparison of the best performing quantized model for each number of bits in the ADC. Details of the alternative mappings are included in Appendix B.

Because the 2-bit ADC provides many advantages for low-power implementation, with minimal loss of performance in the p_T filter, this option is selected for the final chip.

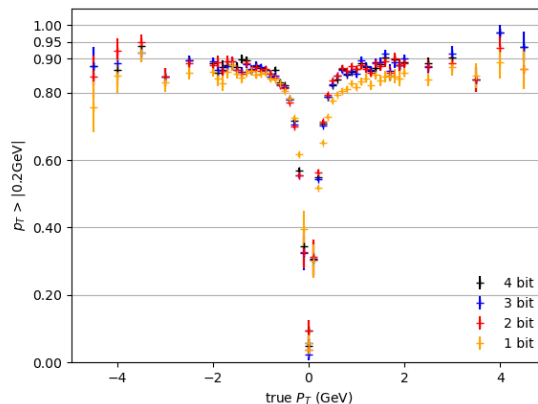


Figure 9: Classifier acceptance vs. p_T for the quantized Model 2 using different numbers of bits in the ADC. Positive and negative values of p_T represent the performance on clusters initiated by particles of positive and negative charge, respectively.

4. On-chip Implementation

Hybrid pixel detectors consist of silicon sensors bonded to a pixelated readout integrated circuit (ROIC). The ROIC is designed in a high performance 28 nm CMOS process. It amplifies and digitizes the signal produced by charged particles traversing the detector and selectively transmits data off-chip for further analysis. In this section, we explore implementation of the baseline algorithm within the pixelated region of the ROIC to enable data-filtering at source, i.e. the rejection of data clusters from low momentum particles.

Each ROIC pixel contains an analog frontend consisting of a charge sensitive preamplifier and three auto-zero comparators for a 2-bit flash-type ADC operating synchronously at 40 MHz. This makes it possible for all hits to be accurately recorded, even when two hits occur in consecutive bunch crossings, without off-time registration of events. A more detailed description of the 2-bit ADC which provides the input to the algorithm is described in Ref. [23].

4.1. ROIC mapping to sensor geometry

A 2×2 array of ROIC pixels (corresponding to $50 \times 50 \mu m$) is laid out so that the four analog frontends create an island, which is surrounded by the digital logic of the ADC and the filtering algorithm. This geometry is shown in Figure 10, where the pink (blue) region represents the analog (digital) region of the ROIC pixel. This layout, which is a common technique used in pixel detectors, maximizes efficiency in terms of area [24, 25].

Each 2×2 array of ROIC pixels corresponds to a 1×4 array of the sensor pixels described in Section 2. The sensor pixels are shown in yellow in Figure 10, and the ROIC-to-sensor pixel mapping links ROIC pixels A, D, B, C to sensor pixels 1, 2, 3, 4 respectively.

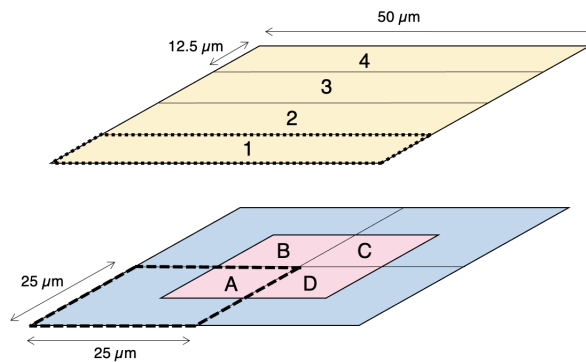


Figure 10: A diagram of the mapping of sensor pixels (yellow) to ROIC pixels (analog portion in pink, digital in blue).

The clusters used for algorithm development discussed in previous sections are simulated in a 13×21 array of sensor pixels. However, to simplify the digital implementation, we assume a 16×16 array of sensor pixels for the ROIC. The size of each sensor pixel remains unchanged with a dimension of $50 \times 12.5 \mu m$. A *super-pixel* of 16×16 sensor pixels will be bonded and remapped to a matrix of 4×32 frontend electronics pixels as shown in Figure 10. This does not change the performance of the algorithm except in the case of very wide clusters (in the bending plane). However, it is still straightforward to validate the algorithm digital implementation.

4.2. Design space optimization

Following the selection of a baseline quantized model in Sec. 3.3, an additional architectural exploration was performed that fully accounts for hardware constraints. During this stage, five major design decisions were made in order to compress the model without significant loss of performance:

- (i) the BN layer is folded into the Dense layer
- (ii) the number of neurons is reduced
- (iii) the number of bits in the network weights and activations is reduced
- (iv) the final activation is changed from Softmax to Argmax.
- (v) y_0 is not directly used as an input to the model. Instead we employ a region-specific implementation.

Each of these design choices is described in detail below.

First, we folded the BN layer into the Dense layer. This optimization technique combines the BN parameters with the Dense layer's weights and biases, effectively reducing computational costs during inference while maintaining the model's performance [26]. The

resulting model has 2,307 trainable parameters, which is a decrease from 2,563 trainable parameters in the original model.

Second, we examined the trade-off between accuracy and area by considering the reduction in the number of neurons and bit precision. Reducing the number of neurons in a neural network can lead to decreased accuracy due to diminished model capacity, resulting in the loss of vital information, underfitting, and limited expressiveness. However, in terms of hardware implementation, fewer neurons create a simpler data path, thereby reducing the required area. In our exploration, we sought to balance network complexity and hardware area while striving to maintain the accuracy of the original network outlined in Sec. 3.3. The optimal model was found to reduce the number of neurons of the first dense layer from 128 to 58.

Third, we reduced the number of bits in the weights and activations by 2 bits each. Design decisions ii (reduction in number of neurons) and iii (reduction of the bit-width of the fixed point representation) produced a hardware implementation with a third of the area originally necessary with a drop in signal efficiency of only 3.5%.

To further reduce the computational complexity of the model without sacrificing the overall predictive capability, we utilized an Argmax function as the final layer (decision iv) instead of the conventional Softmax activation function. By directly identifying the class with the highest score using the Argmax function, we bypass the need to calculate the

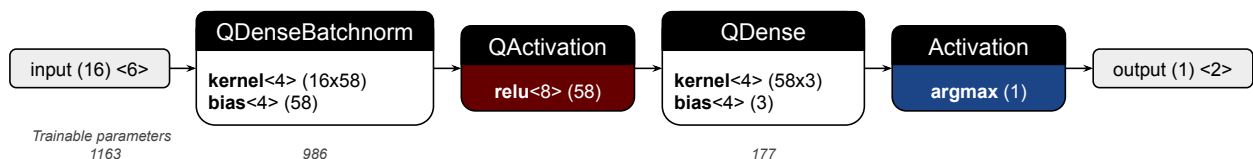


Figure 11: Final model architecture with reduced bit-precision and trainable parameters. For each layer, we report the dimensions in round brackets and the bit-width of the fixed-point representation in angular brackets.

Figure 12 shows the signal efficiency and accuracy of our final architecture trained in different ranges of y_0 . A decrease in accuracy is observed at the edge of the sensor ($y_0 > 6\text{mm}$), where Lorentz drift effects are the greatest and the cluster size decreases

to an average of only two pixels, making the shapes of low and high p_T tracks nearly indistinguishable (recall Fig. 5).

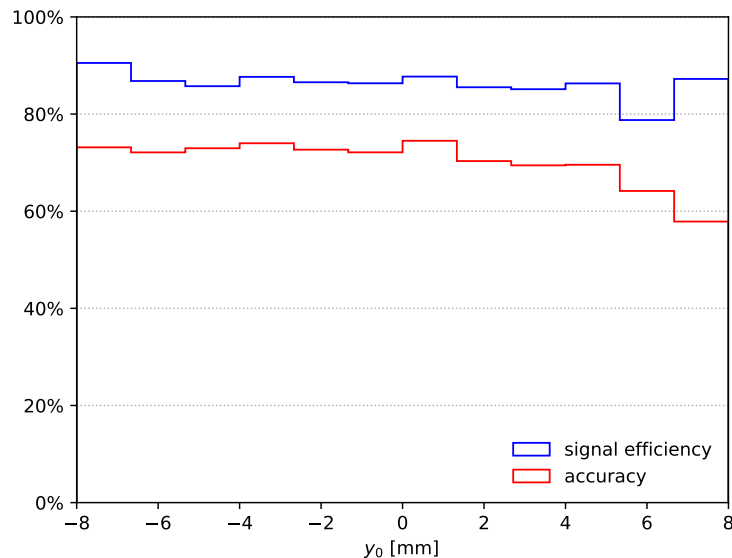


Figure 12: Signal efficiency and accuracy as a percentage in different y_0 regions for the final architecture to be implemented on-ASIC.

The final optimized model has 1,163 trainable parameters, which is a reduction of 55% with respect to the original model.

4.3. Design of algorithm implementation with *hls4ml*

To translate the algorithm from a quantized graph representation into an optimal hardware implementation, we use the *hls4ml* workflow, an open-source Python framework for co-design and translating machine learning algorithms into hardware implementations [12, 27]. The *hls4ml* workflow begins with a trained model from a conventional machine learning framework such as TensorFlow or PyTorch, or a quantized model from QKeras. Designers can use *hls4ml* to optimize not only numerical precision, but also the hardware implementation’s parallelism, striking a balance between area, performance, and power consumption based on the system constraints [28]. Subsequently, *hls4ml* translates the model into C++ code for Siemens Catapult HLS [29]. The HLS tool generates a hardware description at the register-transfer level (RTL) for the traditional ASIC flow. We have opted to fully parallelize the hardware logic, resulting in a combinational implementation of our models from HLS. This choice is driven by the desire to minimize the latency of the neural network. We integrated the resulting RTL design into the system alongside registers and data movers.

QKeras allows the designers to define and train models with specified bit precision for weights and activation functions, but lacks the capability to configure the bit precision of internal multiply-accumulators for different network layers, as it relies on floating-point

representations. When converting the model to a C++ specification for HLS using `hls4ml`, the designers automatically import the fixed-point precision for weights and activations from QKeras. Additionally, they must manually specify the accumulator bit precision for each layer to prevent overflow and loss of precision during fixed-point arithmetic operations. A careful choice of bit precision is crucial for multiply-accumulators, and `hls4ml` offers two potential solutions: a dynamic and a static approach. The dynamic approach requires monitoring the bit precision during simulation and adjusting the fixed-point format to avoid overflow and loss of precision. If the result exceeds the representable range or loses significant precision, an overflow or loss of precision has occurred. Designers must then adjust the fixed-point format by increasing the number of integer bits or fractional bits and repeat the simulation. The static approach determines the required bit precision without simulation, based on the given fixed-point precision for weights and activations. In `hls4ml`, an optimizer pass calculates the appropriate number of integer bits n and fractional bits m for a multiply-accumulator $Qn.m$ based on the bit precision of the two operands $Qn_1.m_1$ and $Qn_2.m_2$. For addition operations, it allocates $n = \max(n_1, n_2) + 1$ integer bits and $m = \max(m_1, m_2)$ fractional bits; for multiplications, it allocates $n = n_1 + n_2$ integer bits and $m = m_1 + m_2$ fractional bits. It is worth noting that a simulation-based approach tends to be dependent on the specific training results used for the simulation, and employing different weights from a subsequent training campaign may still result in overflow or loss of precision. In contrast, a static approach is more conservative and may require additional area to accommodate worst-case scenarios, but it is guaranteed to be error-free. In our exploration, we adopted the described static approach.

A summary of the hardware design space exploration is shown in Fig. 13. Here we show the original 128-node hidden layer implementation and then the relative performance (accuracy) and digital area as we vary the size of the hidden dimension and the bit precision of the neural network computation. We find that reducing precision and number of hidden nodes reduces the area by 67% with a reduction of 5% in accuracy. We leave it to future work to continue optimization of the algorithm while this version of the algorithm is sufficient for digital implementation.

4.4. ROIC integrated sensing and edge computing

Finally, the ML algorithm must be integrated into the array of ROIC pixels. The front-end of the ROIC must amplify and digitize the event's signal at 40 MSPS (Mega Samples Per Second) so the neural network can make an inference within a single bunch crossing. The architecture used is based on a synchronous flash ADC [23] since the conventional Time Over Threshold (TOT) architecture would require multiple clock cycles to digitize an event.

The input to the algorithm, the y -profile distribution, is generated by summing the two-bit ADC outputs across all columns. The algorithm accepts 16 inputs of size six bits each, corresponding to the y -profile distribution (13 inputs) padded with zeros for the implementation (see Section 4.3). Ultimately, the two-bit output of the algorithm

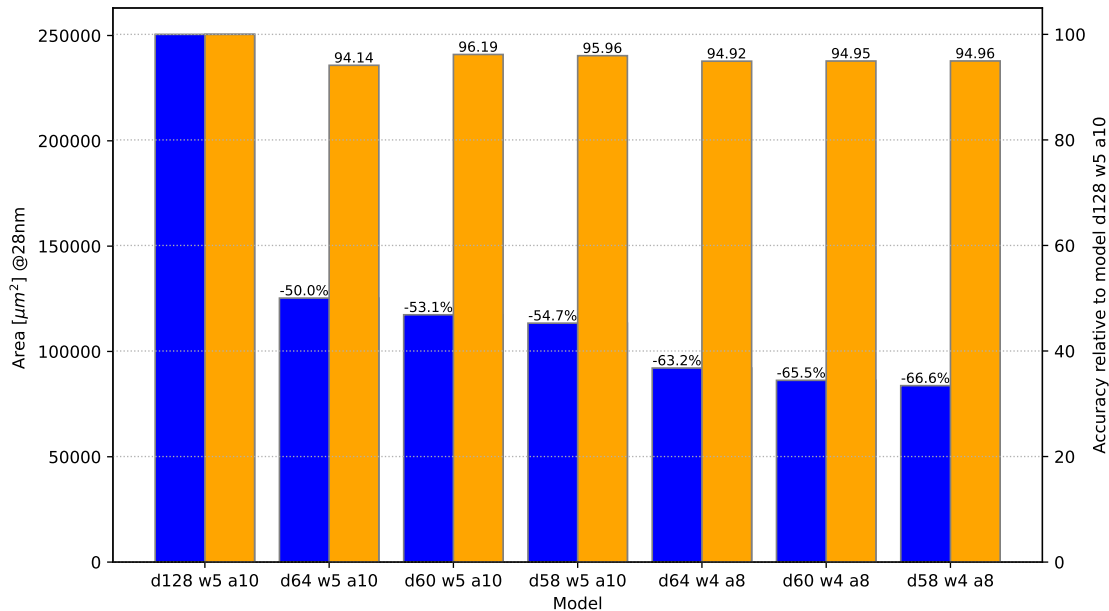


Figure 13: Comparison of the algorithm accuracy and area in 28nm CMOS for different model configurations. The model name corresponds to d{number of neurons in the first hidden layer} w{weight bit-width} a{activation bit-width}.

classifies the clusters as a negatively charged low momentum track, a positively charged low momentum track, or a high momentum track. The sum over the ADCs and their subsequent input into the NN algorithm is depicted in Fig. 14. At the top of the figure, we also illustrate that the algorithm is reconfigurable by introducing new weights and biases into a fixed architecture. This allows the algorithm to be adapted to different regions of the detector and changing detector conditions (due to radiation damage, for example).

The physical layout of the super-pixel is shown in Figure 15. The size of the digitally implemented super-pixel is $889 \mu\text{m} \times 222 \mu\text{m}$. The green areas correspond to the analog circuit islands, while the red contains the digital logic. Given our design, the registers require a one-time setup and leakage at low temperatures is deemed insignificant, thus the combinatorial logic is projected to account for the majority of the power utilization. Based on the technology power models, the anticipated power consumption of the digital logic is roughly $300 \mu\text{W}$, measured when toggling 50% of the inputs every clock cycle (25 nanoseconds).

5. Conclusions and Outlook

High granularity silicon pixel sensors are at the heart of energy frontier particle physics collider experiments and provide the highest spatial granularity measurements for charged particles. At an LHC collision rate of 40 MHz, these detectors create massive amounts of data. Our goal in this proof-of-concept study is to explore the potential for on-sensor data

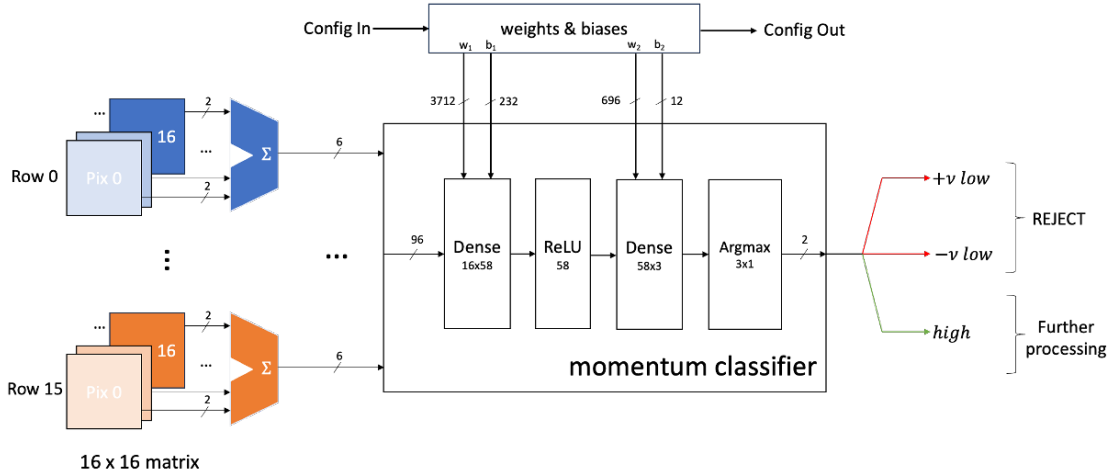


Figure 14: Data flow through the digital implementation of the algorithm from the summed ADC bits (on the left) through the neural network and the final classification layer. At the top of the diagram we illustrate the reconfigurability of the weights and biases in the algorithm stored in memory.

filtering to reduce the data rate from pixel sensors so that they could be read out at 40 MHz and their data could be used in online trigger systems. To accomplish this task, we explore the lossy data compression task of filtering out clusters in the pixel detector originating from low momentum charged particle tracks by deploying on-sensor ML techniques. This study sets a first baseline for what is possible using current technologies, and follow up studies will explore emerging technologies which could improve computational efficiency, physics performance, and additional data reduction techniques.

This study makes a number of novel contributions to this task:

- We introduce a new public dataset which can be used for the benchmark task of pixel on-sensor data reduction. It includes pixel sensor charge deposition information for realistic track distributions as well as information about the charge deposition time structure which can potentially be used in emerging computing architectures.
- We develop and compare a number of algorithmic approaches from simple cluster size information to cluster distributions to time-evolved cluster distributions and present their performance for the cluster filtering task. This shows that for the simple cluster distribution inputs, we can filter out, conservatively, 54% of the pixel sensor data while maintaining a $>90\%$ efficiency for clusters from tracks with $p_T > 2 \text{ GeV}$. We then optimize the algorithm for circuit implementation by exploring quantization of both the charge deposition inputs and the neural network parameters.
- As a first design for implementation on chip of these algorithms, we synthesize and integrate the algorithm which takes as input the cluster distribution (y -profile). The ML algorithm is integrated with a 2-bit flash ADC. We synthesize and place the design using

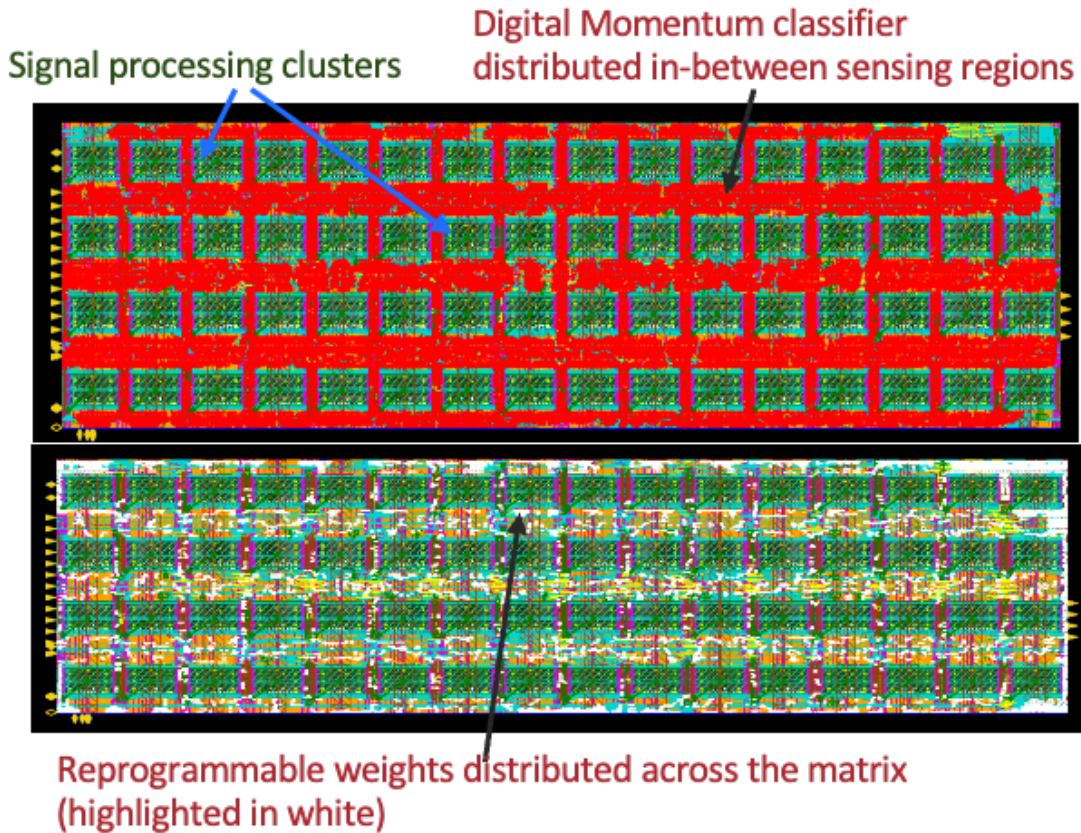


Figure 15: Physical layout of the 16×16 pixel array digital implementation. The top view shows the analog islands in green and the digital implementation of the neural network in red. The bottom view shows different digital circuit elements highlighting the reprogrammable weight storage in white.

low-power 28 nm CMOS technology and emulate and verify the bitwise performance. The design is expected to operate at less than $300 \mu\text{W}$ with an area of less than 0.2 mm^2 .

While this study demonstrates significant strides, there are a number of areas where we can improve performance and realism, which we leave to future studies. We classify them roughly into algorithmic and microelectronics advancements.

Algorithms: One straightforward extension to our studies to improve the data reduction performance of our approach is to train dedicated classification of untracked clusters. Beyond that, one other very important technique for reducing pixel data is to expand the task to featurize the cluster data after filtering. We can transform raw cluster data into physics features, such as position and angle, and their related uncertainties. That will be especially useful in downstream systems to decrease data bandwidth and computations and system complexity by reducing tracking combinatorics. Algorithmically, we have not yet taken

full advantage of the cluster time-evolution which holds additional information. This can, perhaps, be naturally captured using neuromorphic approaches which can treat the pixels as spikes in an imaging system [30, 31]. Neuromorphic computing has been demonstrated as a promising technology for edge applications [32]. Spiking Neural Networks (SNN) and their inherent temporal dynamics have been shown to be suitable for processing sparse streaming data and for deployment on low-power neuromorphic platforms closely integrated with the sensors [33–35]. In other work, we have also demonstrated an initial analysis of the neuromorphic approach for smart-pixels that has yielded models with fewer parameters than in DNNs [36]. Beyond these ideas for improvement, increasing realism will be important in future studies (for example, cases where multiple nearby clusters require arbitration).

Microelectronics: To fully exploit the algorithmic approaches in this work and future work discussed above, we must continue to explore methods for improving computational efficiency. We mention here a non-exhaustive list of potential approaches for emerging microelectronics technologies [37]. Analog computations are much more efficient than digital approaches for neural networks, and they could be particularly well-suited to sensor data, but are more challenging to design and simulate, and are less reliable. Novel emerging technologies for efficient neuromorphic compute such as memristors offer promising advances for power and speed as well. Finally, we also mention the potential of 3D stacking as a way to improve computational area without compromising on system constraints.

It is likely that a combination of these additional ideas is needed to realize a “smart pixel” sensor that can meet particle physics experimental needs. However, having such a sensor would be a transformative technology for particle physics and demonstrating this capability would enable advances in many other scientific domains as well.

Acknowledgements

This work was completed using computing resources at the Fermilab Elastic Analysis Facility (EAF). We thank Burt Holzman for computing support.

We acknowledge the Fast Machine Learning collective as an open community of multi-domain experts and collaborators.

We acknowledge the CMS collaboration for their collection of untracked pixel clusters that serve as a valuable background dataset.

DB, JD, GDG, FF, LG, JH, RL, BP, GP, CS and NT are supported by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the Department of Energy (DOE), Office of Science, Office of High Energy Physics. JD, FF, BP, GP, and NT are also supported by the DOE Early Career Research Program. NT is also supported by the DOE Office of Science, Office of Advanced Scientific Computing Research under the “Real-time Data Reduction Codesign at the Extreme Edge for Science” Project (DE-FOA-0002501). AB is supported through NSF-PHY award 2013007. MS is supported by NSF-PHY award 2012584. CM is supported by NSF-PHY award 2208803. KD is supported in part by

the Neubauer Family Foundation Program for Assistant Professors and the University of Chicago. AY and SK are supported by the DOE Office of Science Research Program for Microelectronics Codesign (sponsored by ASCR, BES, HEP, NP, and FES) through the Abisko Project. MSN is supported through NSF cooperative agreement OAC-2117997, the DOE Office of Science, Office of High Energy Physics, under Contract No. DE-SC0023365, and the Discovery Partners Institute under the "Democratizing AI Hardware with an Open-Source AI-Chip Design Toolkit" Project.

References

- [1] "The ATLAS Experiment at the CERN Large Hadron Collider". In: *JINST* 3 (2008). Also published by CERN Geneva in 2010, S08003. DOI: 10.1088/1748-0221/3/08/S08003. URL: <https://cds.cern.ch/record/1129811>.
- [2] "The CMS experiment at the CERN LHC. The Compact Muon Solenoid experiment". In: *JINST* 3 (2008). Also published by CERN Geneva in 2010, S08004. DOI: 10.1088/1748-0221/3/08/S08004. URL: <https://cds.cern.ch/record/1129810>.
- [3] *Technical Design Report for the ATLAS Inner Tracker Pixel Detector*. Tech. rep. Geneva: CERN, 2017. DOI: 10.17181/CERN.F0ZZ.ZP3Q. URL: <https://cds.cern.ch/record/2285585>.
- [4] A Dominguez et al. *CMS Technical Design Report for the Pixel Detector Upgrade*. Tech. rep. Additional contacts: Jeffrey Spalding, Fermilab, Jeffrey.Spalding@cern.ch Didier Contardo, Universite Claude Bernard-Lyon I, didier.claude.contardo@cern.ch. 2012. URL: <https://cds.cern.ch/record/1481838>.
- [5] Matthew J Strassler and Kathryn M Zurek. "Echoes of a hidden valley at hadron colliders". In: *Physics Letters B* 651.5-6 (2007), pp. 374–379.
- [6] Simon Knapen, Jessie Shelton, and Dong Xu. "Perturbative benchmark models for a dark shower search program". In: *Physical Review D* 103.11 (2021), p. 115013.
- [7] Simon Knapen et al. "Triggering soft bombs at the LHC". In: *Journal of High Energy Physics* 2017.8 (2017), pp. 1–33.
- [8] M Garcia-Sciveres and X Wang. "Data compression considerations for detectors with local intelligence". In: *Journal of Instrumentation* 9.10 (Oct. 2014), p. C10011. DOI: 10.1088/1748-0221/9/10/C10011. URL: <https://dx.doi.org/10.1088/1748-0221/9/10/C10011>.
- [9] Maurice Garcia-Sciveres and Xinkang Wang. "Data encoding efficiency in binary strip detector readout". In: *Journal of Instrumentation* 9.04 (2014), P04021.
- [10] Maurice Garcia-Sciveres and Xinkang Wang. "Data encoding efficiency in pixel detector readout with charge information". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 815 (2016), pp. 18–22.

- [11] Patrick J Fox et al. “Beyond 4d tracking: using cluster shapes for track seeding”. In: *Journal of Instrumentation* 16.05 (2021), P05001.
- [12] Javier Duarte et al. “Fast inference of deep neural networks in FPGAs for particle physics”. In: *JINST* 13.07 (2018), P07027. DOI: 10.1088/1748-0221/13/07/P07027. arXiv: 1804.06913 [physics.ins-det].
- [13] FastML Team. *fastmachinelearning/hls4ml*. Version v0.7.1. 2023. DOI: 10.5281/zenodo.1201549. URL: <https://github.com/fastmachinelearning/hls4ml>.
- [14] Giuseppe Di Guglielmo et al. “A reconfigurable neural network ASIC for detector front-end data compression at the HL-LHC”. In: *IEEE Transactions on Nuclear Science* 68.8 (2021), pp. 2179–2186.
- [15] Morris Swartz and Jenet Dickinson. *Smart pixel dataset*. Version 1. Nov. 2022. DOI: 10.5281/zenodo.7331128. URL: <https://doi.org/10.5281/zenodo.7331128>.
- [16] Morris Swartz. *A Detailed Simulation of the CMS Pixel Sensor*. Tech. rep. Geneva: CERN, 2002. URL: <https://cds.cern.ch/record/687440>.
- [17] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: <https://www.tensorflow.org/>.
- [18] Francois Chollet et al. *Keras*. 2015. URL: <https://github.com/fchollet/keras>.
- [19] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980. URL: <https://arxiv.org/abs/1412.6980>.
- [20] Abien Fred Agarap. “Deep learning using rectified linear units (relu)”. In: *arXiv preprint arXiv:1803.08375* (2018).
- [21] Claudionor N. Coelho et al. “Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors”. In: *Nature Machine Intelligence* 3.8 (June 2021), pp. 675–686. DOI: 10.1038/s42256-021-00356-5. URL: <https://doi.org/10.1038/s42256-021-00356-5>.
- [22] Erwei Wang et al. *Enabling Binary Neural Network Training on the Edge*. 2021. DOI: 10.48550/ARXIV.2102.04270. URL: <https://arxiv.org/abs/2102.04270>.
- [23] Benjamin Parpillon, Amit Trivedi, and Farah Fahim. “Readout IC with 40 MSPS in-pixel ADC for future vertex detector upgrades of Large Hadron Collider”. In: *2023 IEEE International Symposium on Circuits and Systems (ISCAS)* (2023). DOI: 10.1109/ISCAS46773.2023.10182033. URL: <https://ieeexplore.ieee.org/document/10182033>.
- [24] Adam Quinn et al. “A Cryogenic Readout IC with 100 KSPS in-Pixel ADC for Skipper CCD-in-CMOS Sensors”. In: (Feb. 2023). URL: <https://www.osti.gov/biblio/1924325>.

- [25] RD53 collaboration. *RD53B users guide: Introduction to RD53B pixel chip architecture, features and recommendations for use in pixel detector systems*. URL: [\url{https://cds.cern.ch/record/2754251/files/users_guide_v11.pdf}](https://cds.cern.ch/record/2754251/files/users_guide_v11.pdf) (visited on 05/10/2021).
- [26] Benoit Jacob et al. “Quantization and training of neural networks for efficient integer-arithmetic-only inference”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2704–2713.
- [27] FastML Team. *fastmachinelearning/hls4ml*. 2021. DOI: 10.5281/zenodo.1201549. URL: <https://github.com/fastmachinelearning/hls4ml>.
- [28] Farah Fahim et al. *hls4ml: An open-source codesign workflow to empower scientific low-power machine learning devices*. 2021. arXiv: 2103.05579.
- [29] Siemens. *Catapult HLS*. <https://eda.sw.siemens.com/en-US/ic/ic-design/high-level-synthesis-and-verification-platfor>.
- [30] Catherine D Schuman et al. “Neuromorphic computing for temporal scientific data classification”. In: *Proceedings of the Neuromorphic Computing Symposium*. 2017, pp. 1–6.
- [31] Guillermo Gallego et al. “Event-Based Vision: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.1 (2022), pp. 154–180. DOI: 10.1109/TPAMI.2020.3008413.
- [32] Catherine D Schuman et al. “Opportunities for neuromorphic computing algorithms and applications”. In: *Nature Computational Science* 2.1 (2022), pp. 10–19.
- [33] Robert Patton et al. “Neuromorphic Computing for Scientific Applications”. In: *2022 IEEE/ACM Redefining Scalability for Diversely Heterogeneous Architectures Workshop (RSDHA)*. IEEE. 2022, pp. 22–28.
- [34] Tom Birkoben et al. “A spiking and adapting tactile sensor for neuromorphic applications”. In: *Scientific reports* 10.1 (2020), p. 17260.
- [35] Dominique J Kösters et al. “Benchmarking energy consumption and latency for neuromorphic computing in condensed matter and particle physics”. In: *APL Machine Learning* 1.1 (2023), p. 016101.
- [36] Shruti R. Kulkarni et al. “On-Sensor Data Filtering Using Neuromorphic Computing for High Energy Physics Experiments”. In: *Proceedings of the 2023 International Conference on Neuromorphic Systems*. ICONS ’23. Santa Fe, NM, USA: Association for Computing Machinery, 2023. ISBN: 9798400701757. DOI: 10.1145/3589737.3605976. URL: <https://doi.org/10.1145/3589737.3605976>.
- [37] Allison McCarn Deiana et al. “Applications and Techniques for Fast Machine Learning in Science”. In: *Front. Big Data* 5 (2022), p. 787421. DOI: 10.3389/fdata.2022.787421. arXiv: 2110.13041 [cs.LG].

Appendix A. Validation of the dataset

This appendix contains additional validation plots of the Smart Pixels dataset [15]. Note that this dataset contains no corrections for losses due to tracking efficiency at low p_T .

The distribution of simulated incidence angles is shown in Figure A1. Larger values of $\cot \alpha$ are achieved compared to $\cot \beta$ due to the position of the sensor at various points along the axis of a cylindrical detector barrel, as shown in Figure A2.

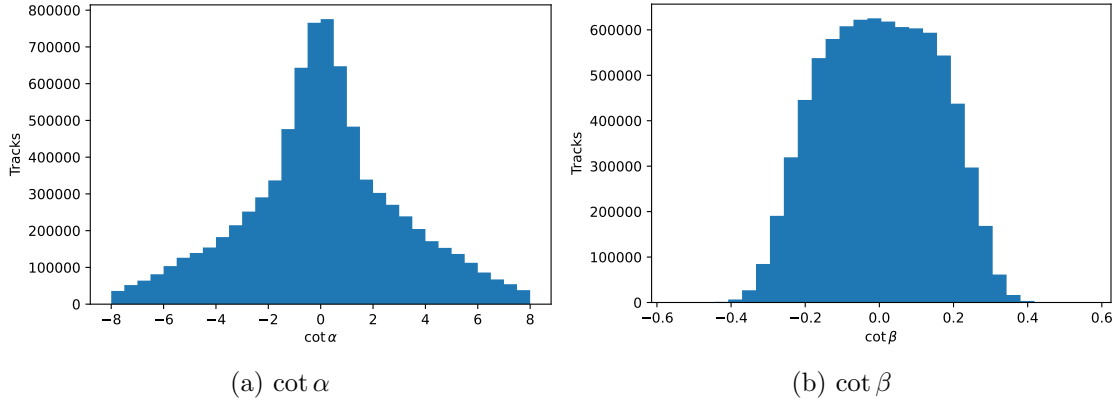


Figure A1: Simulated particle angle of incidence in the (a) $x - z$ and (b) $y - z$ plane (the bending plane of the magnetic field).

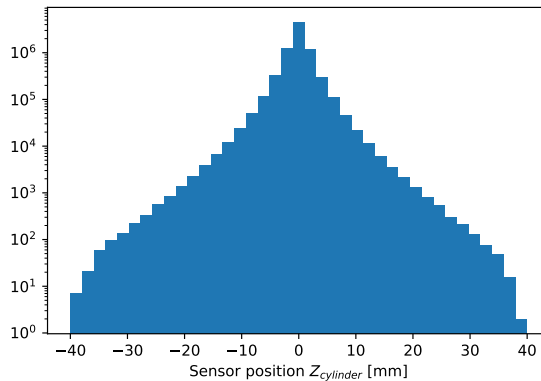


Figure A2: The position of the sensor along the axis of the cylindrical barrel layer, where 0 indicates the particle's point of origin.

The distribution of impact position on the sensor mid-plane is shown in Figure A3a, and is uniformly distributed across the central 3×3 pixel array. The distribution of the y_0 position of each cluster is shown in Figure A3b.

The relationship between the particle p_T and its incident angle in the bending plane of the magnetic field β is shown in Figure A4. The expected dependence of β on p_T and y_0 can

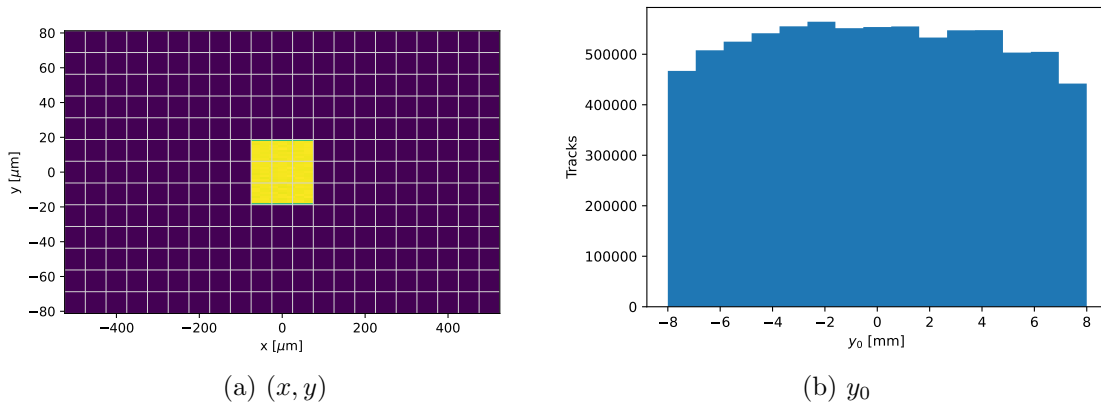


Figure A3: (a) The position at which the simulated particle crosses the sensor midplane is distributed uniformly across the central 3×3 pixel array. (b) The y_0 position is approximately uniform across the 1.6cm long sensor.

be derived analytically:

$$\beta = \pi/2 - \Delta\phi - \arctan(y_0/R) \quad (\text{A.1})$$

where R denotes the radial position of the sensor. The p_T dependence is embedded in the quantity $\Delta\phi$, which denotes the change in the direction of the charged particle due to the magnetic field:

$$\sin(\Delta\phi) = qRB/(2p_T) \quad (\text{A.2})$$

$$(\text{A.3})$$

The expected curves are overlaid on the simulated data distribution for the central and most extreme values of y_0 .

The relationship between the cluster size and incident angle is shown in FigA5a and b for α and β respectively. As expected, the cluster size is proportional to the cotangent of the incident angle.

Finally, the time evolution of two example charge clusters is shown in Figures A6-A7.

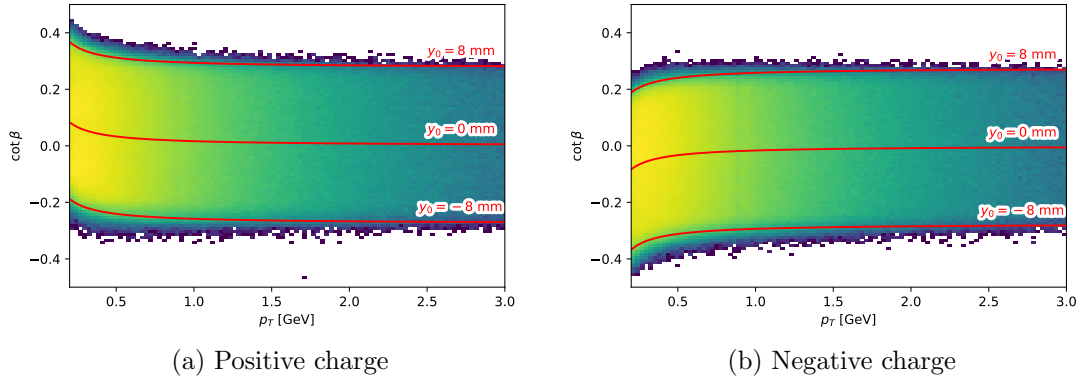


Figure A4: Relationship between $\cot \beta$ and p_T for particles of different electric charge. Red lines denote the relationship at a fixed y_0 position.

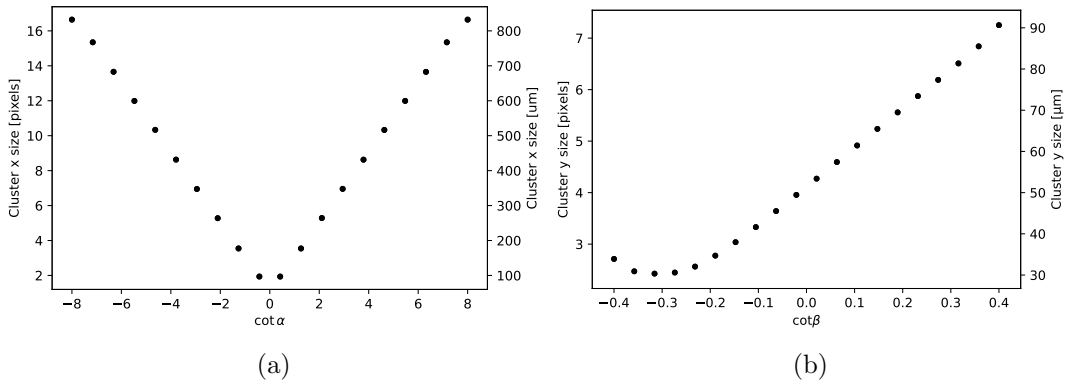


Figure A5: Dependence of the cluster size on the incident angle. Panel (a) shows the x size vs. angle α in the $x - z$ plane, while (b) shows the y size vs. angle β in the $y - z$ plane. In the bending plane of the magnetic field (b), the minimum cluster size occurs at $\cot \beta < 0$ due to Lorentz drift.

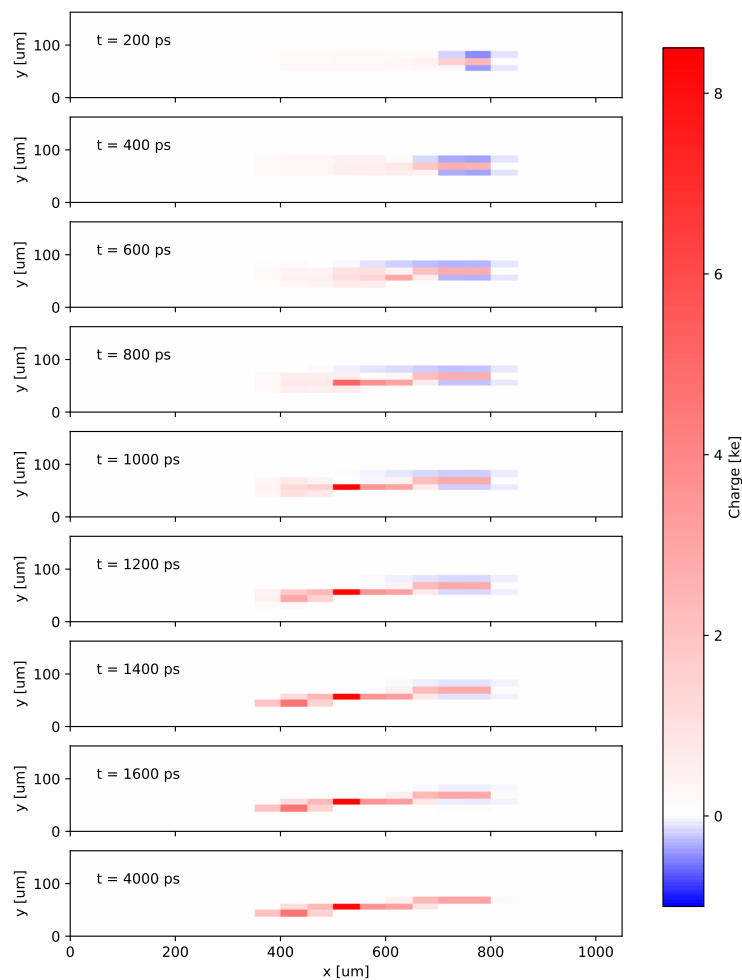


Figure A6: Time evolution of a charge cluster in the simulated 21×13 pixel array. The incident particle has $y_0 = 2.3$ mm and $p_T = 1.9$ GeV. The color scale represents the number of electrons collected after the time denoted. Blue indicates induced negative charge in the pixel.

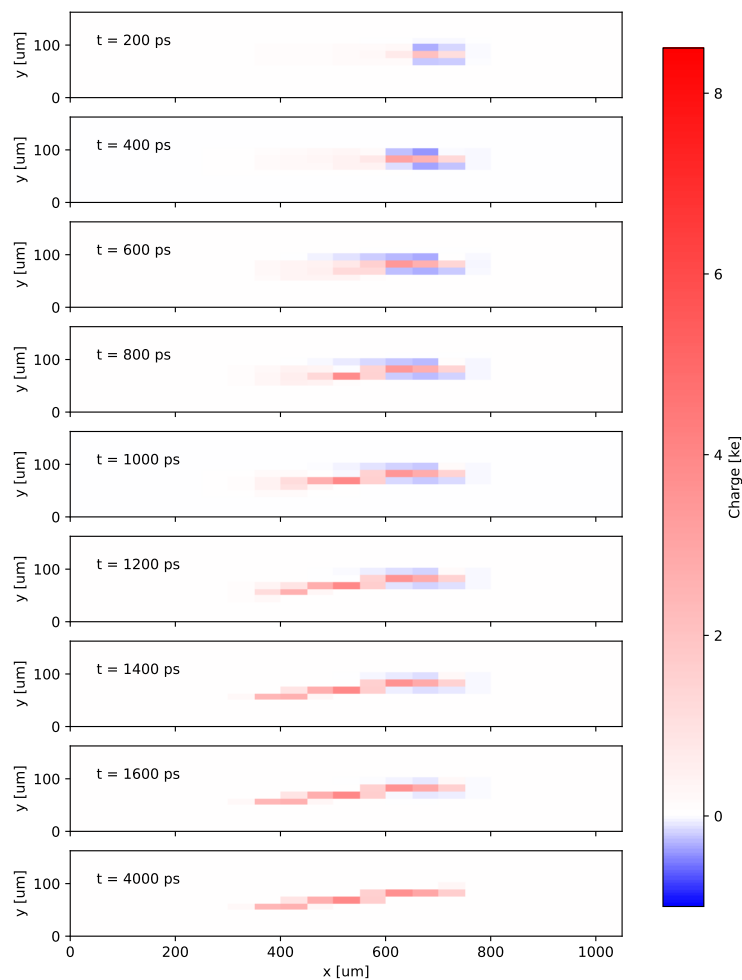


Figure A7: Time evolution of a charge cluster in the simulated 21×13 pixel array. The incident particle has $y_0 = 2.3$ mm and $p_T = 135$ MeV. The color scale represents the number of electrons collected after the time denoted. Blue indicates induced negative charge in the pixel.

Appendix B. Choice of ADC mapping

The mapping of charge intervals to ADC value was explored for an ADC with 1, 2, 3, and 4 bits. Three 2-bit ADC quantizations are shown in Table B1. Intervals A correspond to the selected baseline shown in Table 3. Intervals B prioritize granularity at low charge, while Intervals C place bin boundaries farther apart than the baseline. Figure B1 shows the acceptance vs. p_T of the baseline model trained with each set of intervals.

ADC output	Intervals A [e^-]	Intervals B [e^-]	Intervals C [e^-]
00	< 400	< 400	< 400
01	400 – 1600	400 – 800	400 – 2500
10	1600 – 2400	800 – 1200	2500 – 5000
11	> 2400	> 1200	> 5000

Table B1: Mapping between 2-bit ADC output and collected charge.

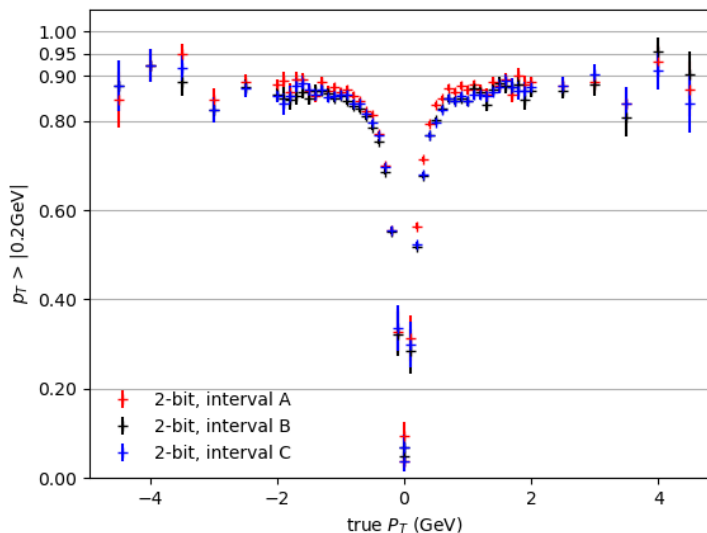


Figure B1: Classifier acceptance vs. track p_T using different 2-bit ADC quantization. Positive and negative values of p_T represent the performance on clusters initiated by particles of positive and negative charge, respectively.

The charge intervals studied for 3-bit ADC models are shown in Table B2, and Figure B2 shows the acceptance vs. p_T of the baseline model trained with each set of 3-bit intervals.

For a 4-bit ADC, the following charge intervals were tested:

- **Intervals G:** 400 - 3200 e in intervals of 200 e, plus underflow and overflow

ADC output	Intervals D [e^-]	Intervals E [e^-]	Intervals F [e^-]
000	< 400	< 400	< 400
001	400 – 1600	400 – 800	400 – 600
010	1600 – 2400	800 – 1200	600 – 800
011	2400 – 4000	1200 – 1600	800 – 1000
100	4000 – 6000	1600 – 2000	1000 – 1200
101	6000 – 8000	2000 – 24000	1200 – 1400
110	8000 – 10,000	2400 – 2800	1400 – 1600
111	> 10,000	> 2800	> 1600

Table B2: Example mappings between 3-bit ADC output and collected charge.

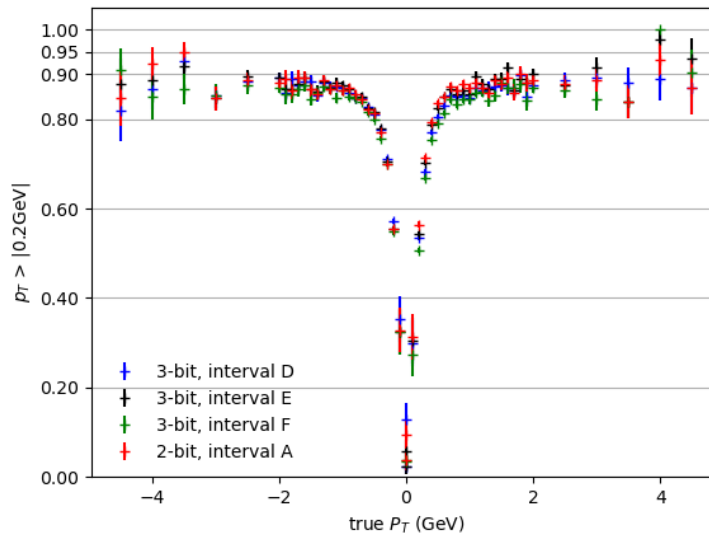


Figure B2: Classifier acceptance vs. track p_T using different 3-bit ADC quantization. Positive and negative values of p_T represent the performance on clusters initiated by particles of positive and negative charge, respectively.

- **Intervals H:** (high granularity at low charges) 400 - 1800 e in intervals of 100 e, plus underflow and overflow
- **Intervals I:** (coarse granularity at low charges) 400-6000 in intervals of 400 e, plus underflow and overflow

The acceptance using each set of 4-bit intervals is shown in Figure B1 as a function of p_T . The 4 bit quantization did not show a significant increase in performance compared to smaller ADC, perhaps because the intervals between electron thresholds do not scale well with integer addition.

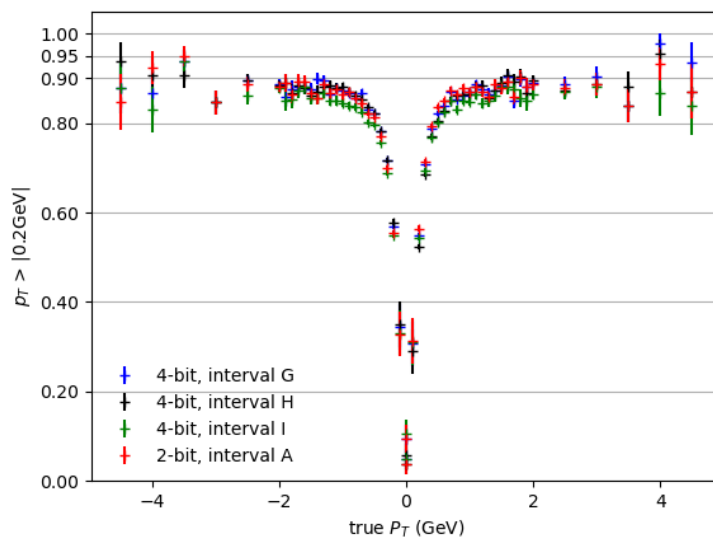


Figure B3: Acceptance vs. track p_T using different 4-bit ADC quantization. Positive and negative values of p_T represent the performance on clusters initiated by particles of positive and negative charge, respectively.