Machine Learning for Efficient Image Filtering at PETRA-III

Thirathep N. Phiankham, Khon Kaen University, Thailand
Supervisor: Vijay Kartik, DESY

September 3, 2024

Abstract

This project explores the use of K-means and DBSCAN clustering algorithms for labeling PETRA-III detector datasets, followed by training Multi-Layer Perceptron (MLP) models for classification. While DBSCAN excelled in identifying complex data structures, K-means was preferred for its ability to define the number of clusters. Two MLP models with different hidden layer configurations were tested to evaluate the impact on classification accuracy. The results indicated that neither model achieved the desired accuracy, highlighting the need for more advanced neural network designs and improved data preprocessing. This study provides insights for future enhancements in clustering and classification techniques.

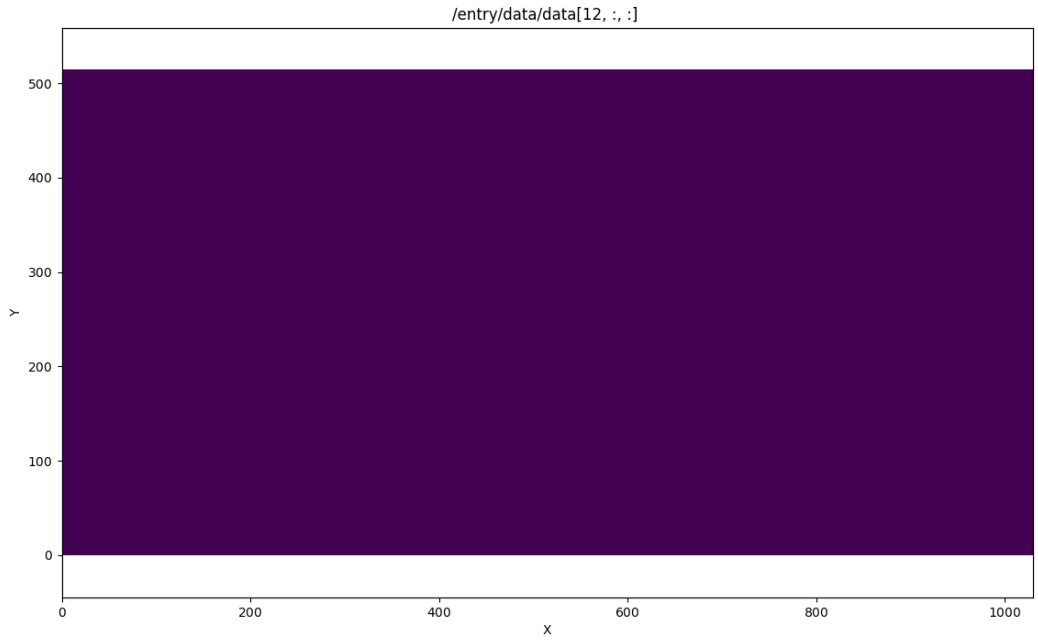
Contents

Introduction

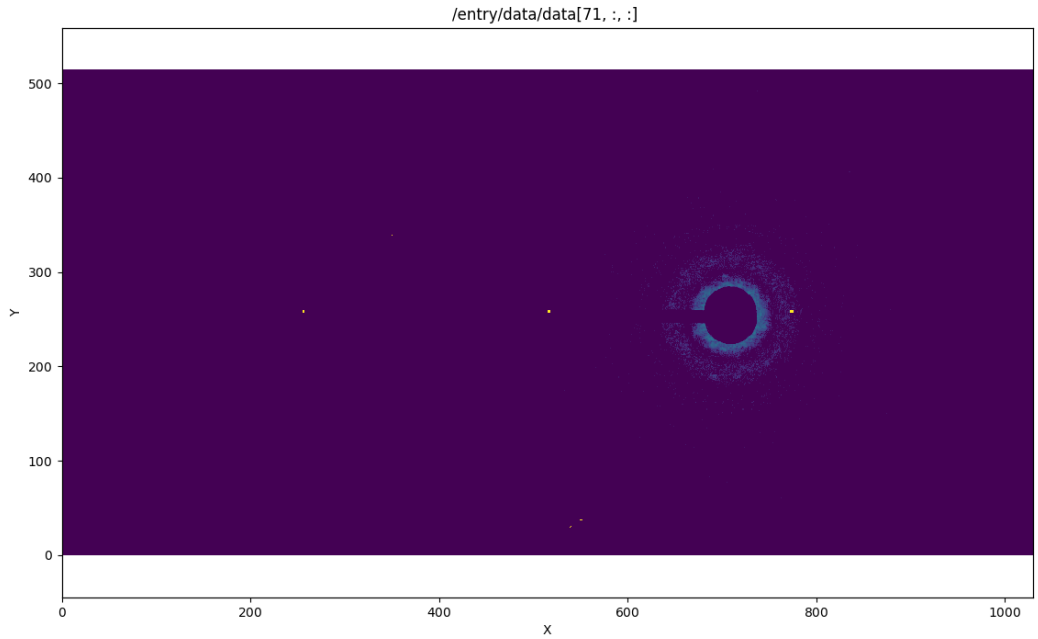
PETRA-III, one of the world's most advanced synchrotron radiation facilities, plays a crucial role in a wide array of scientific research fields, including materials science, chemistry, and biology [1]. The high-resolution imaging data produced by PETRA-III is essential for researchers to make accurate and insightful discoveries. However, the quality of this imaging data is paramount, as any inconsistencies can lead to erroneous interpretations or require significant post-processing efforts.

One common issue affecting the usability of images at PETRA-III is related to the improper opening of the shutter at the early stages of data acquisition. When the shutter does not open wide enough, the beamline cannot receive the full flux required, resulting in underexposed or otherwise faulty images. These unusable images, if not filtered out, can compromise the quality of the data analysis and diminish the overall efficiency of experiments conducted at the facility.

The primary objective of this project is to develop a machine learning-based system that can automatically detect and filter out these unusable images, ensuring that only high-quality data is utilized in subsequent analysis. By leveraging the power of machine learning, this project aims to enhance the efficiency of image processing at PETRA-III, reducing the time and effort required for manual data cleaning and improving the accuracy of experimental results. This report details this machine learning solution's development, implementation, and evaluation, highlighting its potential impact on the PETRA-III facility's operations.



(a) Examples of unusable images from the PETRA-III dataset



(b) Examples of usable images from the PETRA-III dataset

Figure 1: Sample images from the PETRA-III dataset illustrating which images were selected for removal and which were retained before analysis.

Theory

In this section, we will delve into the background theory of machine learning. We will mention about machine learning we used in this project. This will include foundational knowledge of machine learning, key algorithms, and how they relate to image filtering in the context of PETRA-III.

Introduction to Machine Learning

Machine learning (ML) is a branch of artificial intelligence (AI) that focuses on developing algorithms and statistical models enabling computers to perform tasks without explicit instructions [7]. Instead, these systems learn from data, identifying patterns, making decisions, and improving their performance over time. This capability to learn and adapt is what differentiates machine learning from traditional programming, where explicit rules are defined for each task.

Definition and Scope

At its core, machine learning involves the development of models that can make predictions or decisions based on input data [9]. These models are built through a process of training, where the algorithm is exposed to large amounts of data and adjusts its internal parameters to minimize errors in its predictions. Machine learning is widely applied in various fields, including computer vision, natural language processing, and predictive analytics, due to its ability to handle complex and high-dimensional data.

The scope of machine learning is broad, encompassing various subfields and techniques. These can be categorized primarily into three learning paradigms:

- **Supervised Learning**: Involves training a model on a labeled dataset, where the correct output is provided for each input example. The model learns to map inputs to the desired outputs, enabling it to make predictions on new, unseen data.
- **Unsupervised Learning**: Deals with data that lacks explicit labels. The model attempts to identify underlying structures, such as clusters or associations, within the data without guidance on the correct output.

- **Reinforcement Learning**: Focuses on training agents to make decisions through interactions with an environment, learning from the consequences of their actions to maximize some notion of cumulative reward.

Machine Learning in Image Processing

Machine learning has revolutionized the field of image processing, enabling the development of sophisticated systems that can automatically analyze and interpret visual data. Traditionally, image processing relied on manually crafted algorithms to perform tasks such as edge detection, segmentation, and object recognition. These techniques, while effective in some scenarios, often struggle with complex images where variability and noise can obscure important features.

Machine learning, particularly deep learning, has overcome many of these limitations by allowing models to learn directly from raw image data. In image processing, machine learning algorithms can be trained to perform tasks such as:

- **Classification**: Assigning an image to a specific category or class based on its content.
- **Detection**: Identifying and locating objects or regions of interest within an image.
- **Filtering**: Enhancing or modifying an image to remove noise or irrelevant data, which is particularly relevant to the project at PETRA-III.

The ability of machine learning models to learn from large datasets and generalize to new images makes them particularly powerful in settings where manual processing would be infeasible or too time-consuming. These models are especially useful in machine-learning environments, such as synchrotron facilities like PETRA-III, where thousands of images need to be processed rapidly and accurately.

This theoretical foundation of machine learning provides the basis for developing advanced image filtering techniques that can automatically identify

and discard unusable images, thus enhancing the overall quality of data available for scientific research at PETRA-III.

In the next section, We will describe the specific machine learning tools and techniques we utilized in this project, detailing how each was applied to the task of filtering unusable images at PETRA-III. This includes both unsupervised learning methods, such as K-means and DBSCAN, as well as a supervised learning approach using a Multilayer Perceptron (MLP).

Unsupervised Learning Tool

Unsupervised learning algorithms are designed to identify patterns or structures in data without the need for labeled examples. In the context of this project, unsupervised methods were employed to explore the inherent structure of the image data, potentially identifying clusters of unusable images caused by insufficient shutter opening.

K-means

K-means [8] is a widely used clustering algorithm that partitions data into K distinct clusters based on their features. The algorithm iteratively assigns each data point to the cluster with the nearest mean and then recalculates the cluster centroids until convergence.

The objective of the K-means algorithm is to minimize the within-cluster sum of squares (WCSS), which is defined as:

$$J = \sum_{i}^{j} \sum_{x \in C_i} |\sqrt{x - \mu_i}|^2 \tag{1}$$

where:

J is the within-cluster sum of squares (WCSS),

K is the number of cluster,

C_i is the set of data points assigned to cluster i,

x represent a data point,

μ_i is the cluster point of the cluster i.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN [3] is a density-based clustering algorithm that identifies clusters by grouping together points that are densely packed and separating points that are isolated in low-density regions as outliers. Unlike K-means, DBSCAN does not require specifying the number of clusters in advance and can detect clusters of arbitrary shapes.

DBSCAN has two key parameters:

- **Epsilon (ϵ)**: This parameter defines the maximum distance between two samples for them to be considered as neighbors. It essentially sets the radius of the neighborhood around each point. A smaller ϵ will result in a higher number of smaller clusters, while a larger ϵ may merge points into larger clusters.
- **MinPts**: This parameter specifies the minimum number of points required to form a dense region (or cluster). It helps distinguish between core points (which have at least MinPts neighbors within ϵ) and border or noise points. A higher MinPts value generally leads to fewer, larger clusters and more points being classified as noise.

Supervised Learning Tool

Supervised learning algorithms learn from labeled data to make predictions on new, unseen data. In this project, a Multilayer Perceptron (MLP) was used to classify images as usable or unusable based on the training data provided.

Multilayer Perceptron (MLP)

MLP [5] is a type of feedforward artificial neural network composed of multiple layers of neurons, including an input layer, one or more hidden layers, and an output layer. MLPs are capable of modeling complex, non-linear relationships in the data by learning to map input features to the desired output

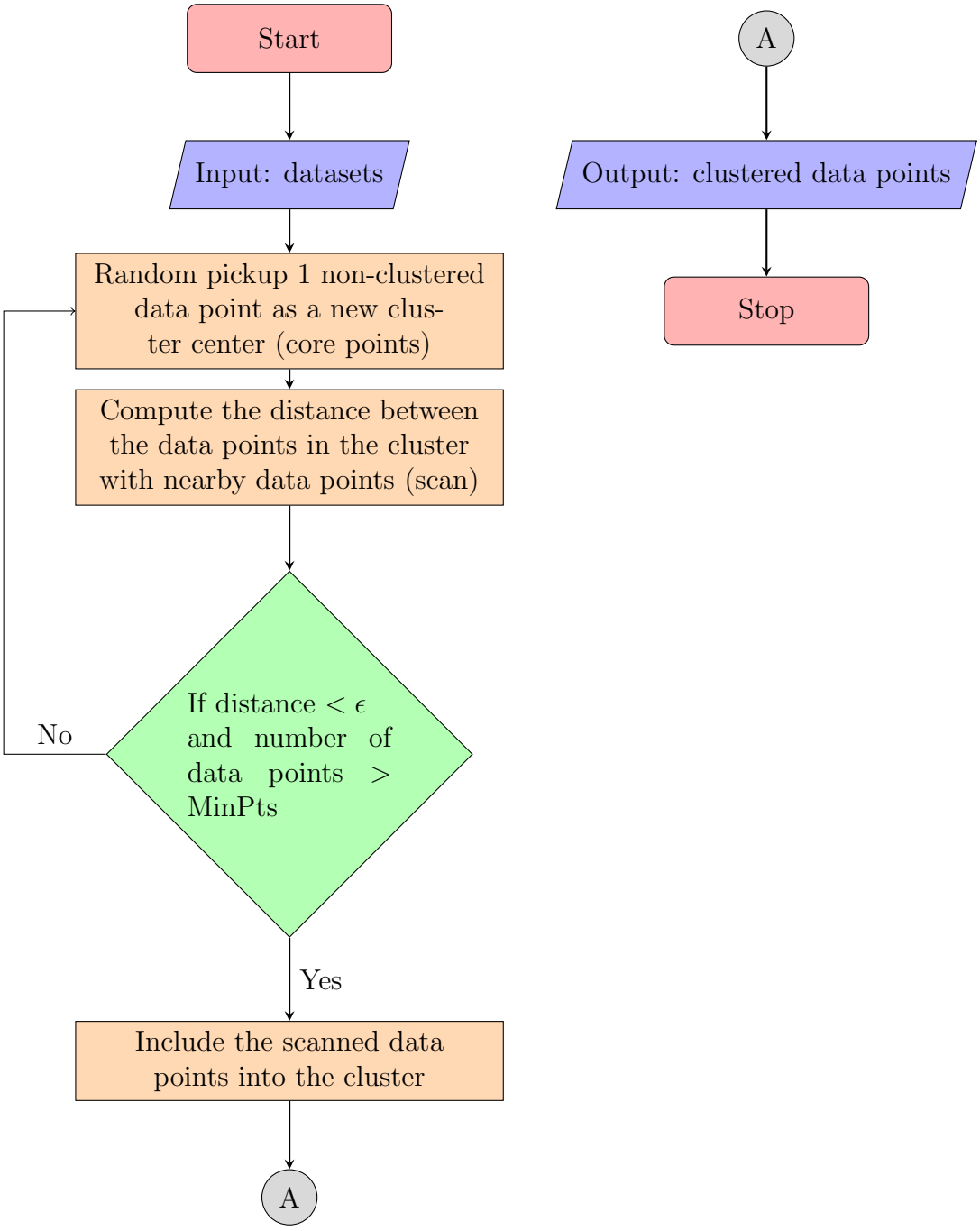


Figure 2: Flowchart of DBSCAN algorithm

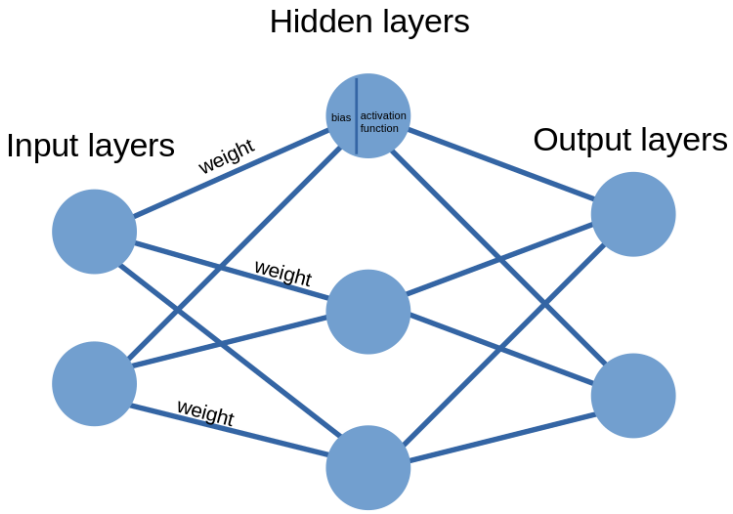


Figure 3: Multilayers Perceptron diagram

through backpropagation. Each neuron performs a weighted sum of its inputs, adds a bias term, and then applies an activation function to introduce non-linearity. The output of a neuron y is given by:

$$y = \sum_{i=1}^{n} (w_i x_i) + b \tag{2}$$

where:

n is the number of the input neurons,

w_i is the weight,

x_i is the input value,

b is the bias.

In MLP, there are three key components: the input layer, hidden layers, and output layer:

- **Input layer**: This layer contains neurons that represent the features of the input data. The number of neurons equals the dimensionality of the input.

- **Hidden layers**: These layers are intermediate between the input and output layers. Each hidden layer transforms the data using weighted connections and activation functions, enabling the network to learn complex patterns. The depth (number of hidden layers) and width (number of neurons per layer) are crucial in determining the network's capacity to model intricate relationships in the data.
- **Output layer**: The final layer produces the predictions.

Implementation

Clustering

In this project, we utilized K-means and DBSCAN for data labeling through clustering. We compared the results from both methods to determine which best suited our needs. Ultimately, we chose K-means due to its ability to specify the number of clusters using the K parameter. Although DBSCAN provided potentially better-fitting clusters, it lacked consistency in determining the number of clusters across different datasets. Since we needed to classify our data into exactly two classes, K-means offered the stability and control we required.

One of the challenges we faced was that the initial label in some clustered datasets was not always 0. Since our Multi-Layer Perceptron (MLP) model required the labels to be consistently arranged, we needed a way to address this issue. To solve this, we implemented a condition that checks the first label of each dataset. If the first label is 0, the data is left unchanged. However, if the first label is 1, we swapped all the labels in the dataset to ensure consistency. The solution was straightforward but effective in maintaining the required label order for the MLP model.

```
if kmeans_labels[0] == 1:
kmeans_labels = 1 - kmeans_labels
```

This approach allowed me to correctly swap the labels into the desired order.

Classification

In the classification process, we trained a Multi-Layer Perceptron (MLP) model using data labeled by either K-means or DBSCAN clustering algorithms. For this project, we specifically used the labels generated by K-means to train the MLP. To assess the impact of the number of nodes on model accuracy, we trained two different models. The first model consisted of three hidden layers with 512, 256, and 128 nodes, respectively. In contrast, the second model had hidden layers with 256, 128, and 64 nodes, respectively. Below, we provide details on the activation functions, loss criterion, and optimization methods used in these models to enhance performance and ensure accurate classification.

Activation Function

- **ReLU**: Introduces non-linearity by outputting the input directly if positive, otherwise outputting zero.

$$f(x) = max(0, x) \tag{3}$$

- **Softmax**: Converts logits into a probability distribution, ensuring that the output values sum to 1 across multiple classes.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{4}$$

where:

σ is the Softmax,

$\vec{z}$ is the input vector,

e^{z_i} is the standard exponential function for input vector,

K is the number of classes in the multi-class classifier,

e^{z_j} is the standard exponential function for output vector

Criterion

In this project, we used the ***Cross-Entropy loss*** [4] function as the criterion for training our classification model. ***Cross-Entropy loss***, also known as log loss, measures the difference between the predicted probability distribution and the actual distribution of the target classes. Specifically, it quantifies how well the predicted probabilities align with the true class labels. The ***Cross-Entropy loss*** is particularly well-suited for classification problems, as it penalizes incorrect predictions more heavily, especially when the model is confident but wrong. By minimizing this loss during training, we effectively guide the model to produce probability estimates that are as close as possible to the true class probabilities, thereby improving its classification performance.

The Cross-Entropy loss function is defined as:

$$\text{Cross-Entropy} = -\sum_{i=1}^{n} y_i \log(p_i) \tag{5}$$

where:

n is the number of classes,

y_i is the true probability distribution of class i,

p_i is the predicted probability of class i.

Optimization

In this project, we employed the ***Adam*** [6] optimization algorithm, an extension of *Stochastic Gradient Descent (SGD)* [10], to enhance the training process of our model. ***Adam***, which stands for ***Adaptive Moment Estimation***, combines the advantages of two other popular optimization techniques: *AdaGrad* [2] and *RMSProp*. It maintains a moving average of both the gradients and their squared values, adjusting the learning rate adaptively for each parameter. This approach helps to accelerate convergence and improve the efficiency of training, especially in cases where gradients can be noisy or vary significantly.

In this project, we used the ***scikit-learn*** library to manage both the criterion and optimization aspects of our model. ***scikit-learn*** is a versatile and widely-used tool in Python for machine learning. It provides straightforward and reliable methods for implementing loss functions and optimization algorithms. By integrating ***scikit-learn*** into our workflow, we were able to easily set up and fine-tune these critical components, which helped us streamline the training process and ensure the accuracy of our model.

Results & Discussion

Comparison of K-means and DBSCAN

In Figure 4, we illustrate the key differences between the clustering results of K-means and DBSCAN. On the left side of the figure, you can see that both methods produced similar cluster separations. However, the right side reveals a more nuanced comparison. K-means clustered the data points into two groups with equal density, which did not align with our expectations. In contrast, DBSCAN provided a more nuanced clustering result, better capturing the structure of the data. Nonetheless, DBSCAN's limitation is that it does not allow us to specify the number of clusters in advance, which is a significant drawback for our purposes. Given these considerations, we chose to focus more on K-means for this project, despite its limitations, due to its ability to define the number of clusters explicitly.

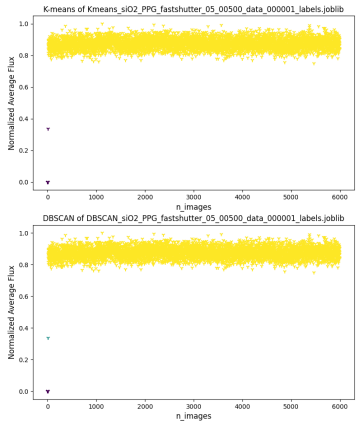
Confusion Matrix

The confusion matrix in Figure 5 highlights some critical insights into the model's performance. It shows that the model's accuracy is insufficient for real-world applications. The matrix displays the predicted labels (on the x-axis) versus the true labels (on the y-axis). Notably, predictions for the label equal to 1 are quite accurate, whereas the model performs poorly for another label, yielding unreliable results.

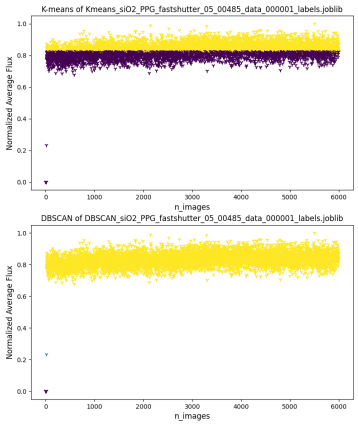
The left side of the matrix provides a clearer picture of these issues, suggesting that the higher number of nodes in the MLP model might be contributing to better performance. In contrast, the right side of the matrix shows significantly worse predictions compared to the left. A closer examination reveals that the right side had half as many nodes in the first hidden layer compared to the left side. This discrepancy suggests that the number of nodes in the hidden layers may play a crucial role in model performance and could serve as a basis for future improvements and refinements in our model.

Comparing Visualizations of Average Flux

In Figure 6, we observe the differences in the visualization of average flux for datasets obtained from the PETRA-III detector. The primary distinction

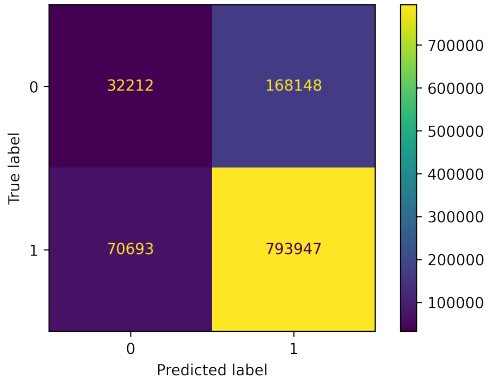


(a) Well-formed clusters generated by K-means (top) and DBSCAN (bottom)

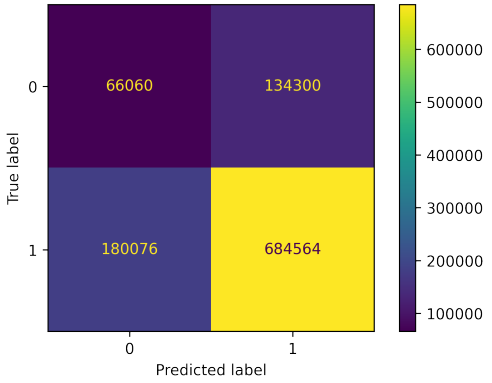


(b) Unexpected clustering results from K-means (top) and DBSCAN (bottom)

Figure 4: Comparison of K-means and DBSCAN Clustering on PETRA-III Average Flux Data. This figure contrasts the clustering results of K-means and DBSCAN algorithms applied to average flux measurements from PETRA-III detector datasets. Yellow points denote the data to retain, whereas purple points indicate the data to exclude, highlighting the differences in clustering patterns and effectiveness between the two methods.



(a) Confusion Matrix for a Model with a 512-Neuron First Hidden Layer



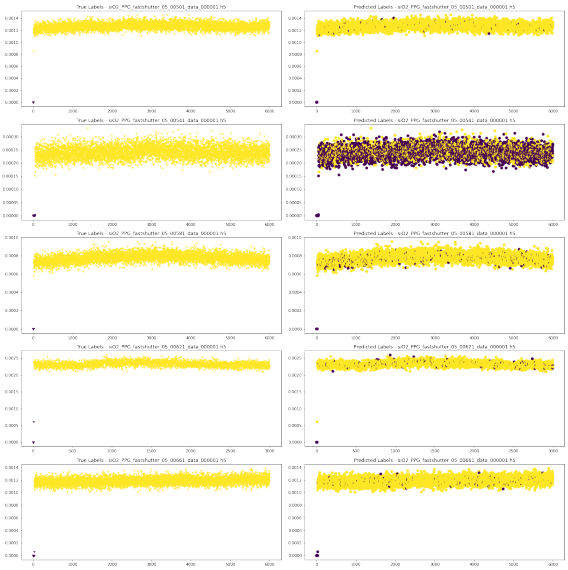
(b) Confusion Matrix for a Model with a 256-Neuron First Hidden Layer

Figure 5: Confusion matrix of the trained MLP model

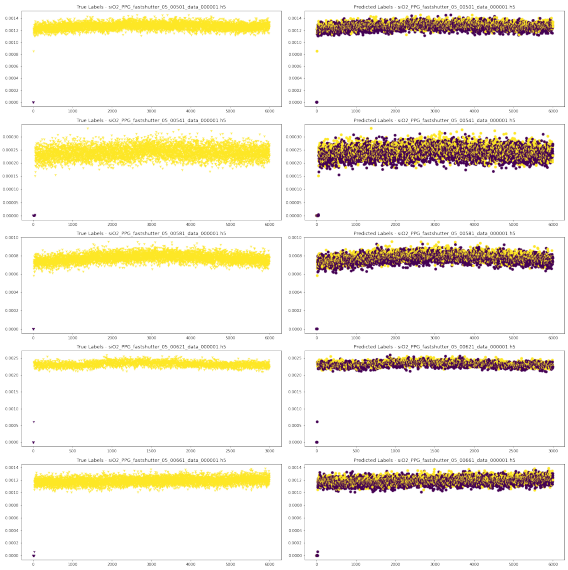
between the visualizations on the left and right sides lies in the number of nodes used in the first hidden layer of the MLP model. The model on the left was trained with 512 nodes in the first hidden layer, whereas the model on the right was trained with 256 nodes.

These differences in node configuration significantly impact the model's performance. Although the predictions from the model with 512 nodes on the left appear closer to the desired outcome, they still fall short of achieving the level of accuracy required. The model on the right, with 256 nodes, performs even worse, highlighting the sensitivity of the MLP model to the number of nodes in its architecture.

The visualizations further underscore the conclusion that neither model succeeds in meeting the main objective of accurately predicting the average flux. The disparities between the two models suggest that simply adjusting the number of nodes is insufficient for resolving the underlying issues. This insight strengthens our understanding that the current model configurations are inadequate, prompting the need for further refinement or alternative approaches to achieve reliable results.



(a) Comparison of K-means Clustering and MLP with a 512-Neuron First Hidden Layer



(b) Comparison of K-means Clustering and MLP with a 256-Neuron First Hidden Layer

Figure 6: Visualization of MLP Model Predictions for Average Flux in Images from a single PETRA-III dataset

Conclusions

This project set out to explore the effectiveness of using K-means and DBSCAN clustering algorithms to label datasets from the PETRA-III detector, followed by training Multi-Layer Perceptron (MLP) models for classification purposes. Through the course of this work, several important insights were gained regarding the strengths and limitations of these approaches.

The initial phase of the project involved clustering the data using K-means and DBSCAN, with a focus on understanding how each algorithm handled the unique characteristics of the PETRA-III datasets. DBSCAN proved effective in identifying clusters of arbitrary shapes and handling noise in the data, making it a robust choice for complex datasets. However, its inability to specify the number of clusters beforehand presented a significant limitation, especially given the project's requirement to classify data into a predefined number of classes. K-means, while less flexible in handling irregular cluster shapes, allowed us to define the number of clusters, which made it a more practical choice for our classification task.

In the MLP model training part, with the data labeled by K-means, we trained two different MLP models to evaluate how variations in the number of nodes in the hidden layers would impact classification accuracy. The first model, with 512, 256, and 128 nodes in its hidden layers, was designed to test a more complex architecture, while the second model, with 256, 128, and 64 nodes, offered a simpler configuration. The results indicated that while increasing the number of nodes in the hidden layers did improve the model's performance to some extent, it was still insufficient to achieve the level of accuracy required for reliable real-world application. The comparison between the two models underscored the importance of not just the number of nodes, but also other factors such as model depth, data preprocessing, and potential overfitting, which may have influenced the outcomes.

In the performance evaluation part, the confusion matrix provided a detailed view of the model's performance across different classes. It revealed that although the model with more nodes showed a slight improvement, it still struggled with misclassifications, especially in classes where data was less representative. This performance gap highlighted the limitations of the current MLP configurations and pointed to the need for further refinement. The

visualizations of average flux further supported these findings, showing that neither model configuration was capable of consistently producing accurate classifications, particularly under varying conditions.

The project concluded that while the approach of using K-means for labeling and MLP for classification was theoretically sound, in practice, the models did not meet the desired level of accuracy. The main challenges identified were the need for a more sophisticated MLP architecture, better handling of class imbalances, and possibly the exploration of hybrid models that combine the strengths of both K-means and DBSCAN.

Future work should focus on experimenting with deeper and more complex neural networks, incorporating additional preprocessing steps, and potentially exploring alternative clustering methods that balance the strengths of K-means and DBSCAN.

References

[1] PETRA III - Deutsches Elektronen-Synchrotron DESY — desy.de. `https://www.desy.de/research/facilities__projects/petra_iii/index_eng.html`. [Accessed 03-09-2024].

[2] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[3] Martin Ester, Hans-Peter Kriegel, Jorg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[4] I. J. Good. Rational decisions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 14(1):107–114, 1952.

[5] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[7] John R. Koza, Forrest H. Bennett, David Andre, and Martin A. Keane. *Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming*, pages 151–170. Springer Netherlands, Dordrecht, 1996.

[8] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[9] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[10] Herbert E. Robbins. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.