LHC Jets

Roman Sokolov
zcapsok@ucl.ac.uk

April 3, 2024

Abstract

This project will discuss the usage of neural networks which distinguish all-hadronic decays at LHC, specifically focusing on five distinct classes: Gluon, Quark, W-Boson, Z-Boson, and Top. The primary objective of this mini-project is to develop and evaluate a machine learning classifier utilising data from the HLS4ML dataset. The study will investigate various ML models which use reconstructed JetImages and various jet features of proton-proton collisions at CERN, offering an interesting prospects of machine learning application in the field of Nuclear and Particle Physics, inspiring future researches to utilise this powerful tool.

Contents

1 Introduction

Jets are a cascade of particles produced through the process of particle acceleration and collision (e.g proton-proton collision). When particles collide, immense energy involved leads to creation of new particles (including quarks and gluons) which quickly undergo hadronization process (where they combine with other quarks and gluons to form a bound state known as hadrons). Due to the nature of strong force,which governs interactions between quarks and gluons, the resulting hadrons are often produced in collimated jets of particles travelling approximately in the same direction. These jets are observed using detectors surrounding the collision points, this will be described later in the section. These are reconstructed by algorithms that combine information from different detector components and then cluster into jets using physics motivated sequential recombination algorithms.[3] Studying these jets provides insights into properties of particles produced during collision allowing us to study fundamental forces and fundamental blocks of matter.

Jet classification / tagging, refers to the process of categorising and labelling jets produced in high-energy particle collision. The goal of tagging is to identify the type of particle that initiated a given cascade, such as quarks (q), gluons (g) , W & Z Bosons and top quarks (t). Machine learning techniques are often employed for jet classification as they can efficiently learn complex patterns and features from data to distinguish between different types of jets. Jet classification plays a crucial role in various areas of particle physics research, including search for new particles, the measurement of standard model parameters and investigation of fundamental interaction and symmetries. In this mini-project, I will structure and compare different neural networks for the Task 1, Task 2 and extension described later in the report and find the most efficient one.

2 Aims/Tasks and Extensions

For this mini project my main two tasks which needed to be completed are:

1. To develop a machine learning classifier that can separate the five classes of event listed above (Gluon, Quark, W-Boson, Z-Boson, Top Quark) using only the combined jetImage image.

2. Test machine learning classifier and investigate how the efficiency of the classifier depends on the jet feature variables shown above. And which classes of events are the easiest to separate.

These two tasks will have additional extensions as I will not only develop one classifier model, but I will create 3 different classifying models, such as: NN, CNN and PCA and I will compare performance of the models trained on original "raw" data to trained on augmented data.

3 Data Set Description

The data used for this mini-project was taken from an open source dataset called HLS4ML from Zenodo. This dataset is a collection of high-pT jets from simulations of proton-proton collisions at LHC. HLS4ML (High-Level Synthesis for Machine Learning) is a framework designed to accelerate the deployment of machine learning models onto field programmable gate arrays (FPGA) using

high-level synthesis techniques. HLS4ML is beneficial for applications where low-latency, high-throughput and energy efficiency are critical, for example real time data processing in high energy physics experiments. This mini-project is based on data base consisting of simulated jets with energy of $p_t \approx 1$ TeV, originating from gluons,quarks, W and Z bosons with top quark and produced in $\sqrt{s} = 13$TeV proton-proton collision [3]

Figure 1: Average and example 100 x 100 jet images from detectors at CERN LHC which will be used for training and testing machine learning algorithms from the hls4ml dataset.

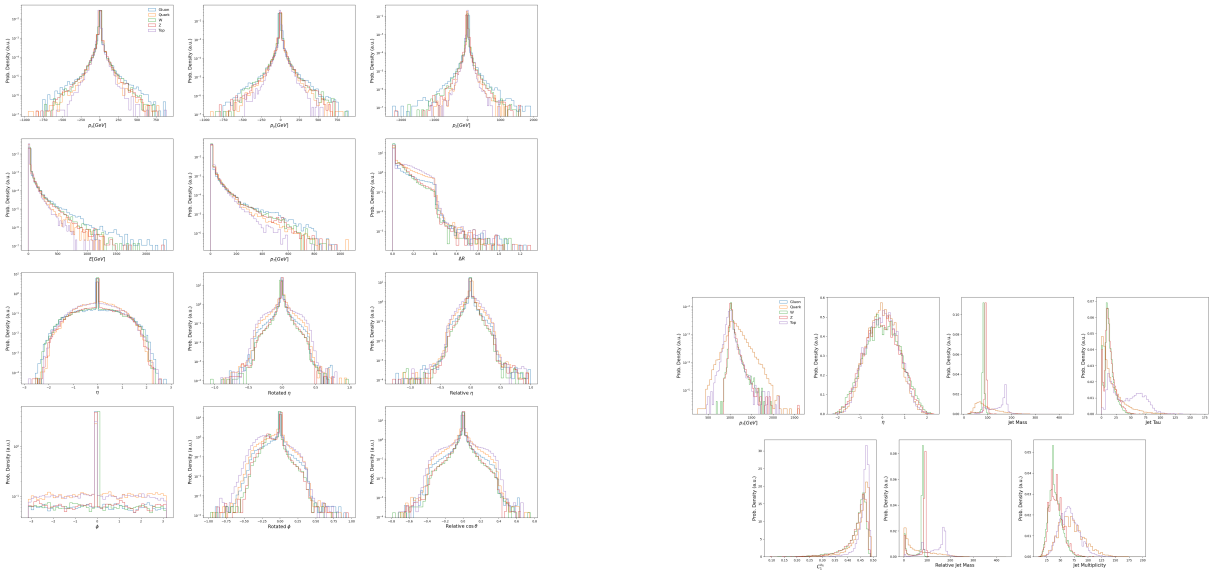

Figure 2: (Left) Example of distribution of 7 features of 100 particles. (Right) Example of distribution of kinematic features of 100 particles in each jet from the hls4ml dataset.

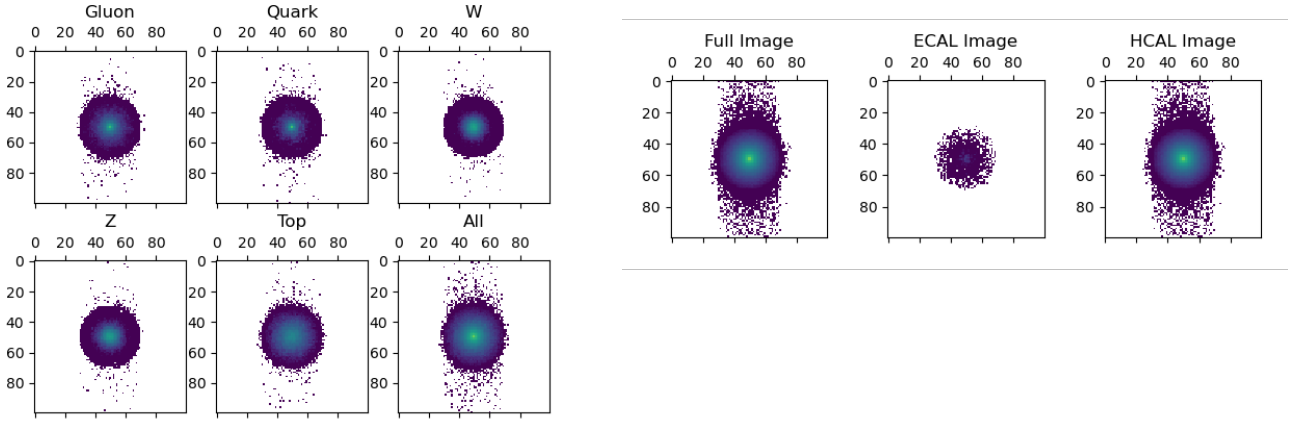

Figure 3: (Left) Average and (right) example 100 x 100 image from the hls4ml database for five jet classes used in this mini-project: Gluon (g), Quarks (q), W boson (w), Z boson (z), top quark (t). The temperature map represents the amount of transverse momentum collected in each pixel of image with units in GeV from the hls4ml dataset.

4 Data Pre-Processing

Data pre-processing can have a huge impact on the performance of machine learning models for the classification tasks. After reading [2] several pre-processing techniques have been selected as they have the most effect on the ML models. And some of these techniques will be applied for Task 1 (classification using images) and Task 2 (classification using jet features values).

4.1 Rotation

Usually used for images. Randomly rotated images with probability (70%) in a dataset will be used, as rotation can be useful in removing random aspects of a decay angle relative to the eta-phi coordinate system [1]. But there are challenges, as it can alter jet mass, thus potentially impacting the classification performance. Also, any rotation other than factor of 90 cannot be performed exactly and interpolation needed to be used [2], but for simplicity factors of 90 are used.

4.2 Reflection

In theory, should not affect the physics of the jet, but this transformation can be used to ensure positive phi contains the half of the jet with more energy, for instance due to radiation emission [2] Random rotations to the dataset with probability (70%) will be applied.

4.3 Normalisation

4.3.1 Min-Max Normalisation

Min-max or scaling features is a technique used to re-scale numerical features within a specific range. This normalisation is commonly used in ML to bring all feature values into similar scale, preventing certain features from dominating the learning process due to differences in their original ranges.

$$x_{normalised} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

Where x_{min} is minimum pixel intensity or energy value and x_{max} is maximum pixel intensity or energy value from the particle.

4.3.2 L^2 Normalisation

L^2 Normalisation or vector normalisation is technique commonly used in ML, including image processing tasks to normalise feature vectors. It is desire that the sum of squared pixel intensities or energy values is equal to one. It is implemented by dividing each pixels or particle intensity by L^2. It is computed as the square root of the sum of the squares of all data in the array or image. [1]

$$x_{normalised} = \frac{x}{\sqrt{\sum_{i=1}^{n} x_i^2}} \tag{2}$$

where x is energy value (or pixel intensity in image) from particle in the jet, and n is number of particles/measurements or (pixels in an image).

4.4 Gaussian Noise

ML is often added to input data or model parameters to introduce randomness and robustness to the learning process. Generated from gaussian distribution which is defined by its mean mu and standard deviation sigma. The noise values are sampled randomly. Adding noise to images helps in improving the generalisation ability of ML models by exposing them to variations and perturbations in input data, it prevents overfitting and encourages models to learn more robustly.

$$f_x(\alpha) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\alpha-\mu)^2}{2\sigma^2}} \tag{3}$$

Where α is positive real number, μ is the mean of the distribution and σ is standard deviation.

5 Methodology for Task 1

For task one, I will investigate how efficiently different neural networks classify raw and augmented images. Overall, the data augmentation which I will be using for my models training (FNN,CNN,PCA) will be separated into 3 tests:

1. Raw Data and Augmented data without noise
2. Raw Data and Augmented data with noise
3. Normalisation of raw data using methods mentioned in section before for raw data

Later, I will compare the probability of classifying particles using a confusion matrix as it provides a comprehensive summary of the predictions made by the model compared to the actual labels in the dataset. It is usually used for understanding types of errors made by the model and assessing it effectiveness across different classes (which are encoded with different labels), such as: Gluons ($g \rightarrow 0$), Quarks ($q \rightarrow 1$), W Boson ($W \rightarrow 2$), Z Boson ($Z \rightarrow 3$), Top Quark ($q \rightarrow 4$).

In a confusion matrix, there are 4 classes of results (TP, TN, FP, FN) which helps to identify the performance of the model:

- **True Positive** (TP) - number of instances correctly predicted and belonging to positive class.
- **True Negative** (TN) - number of instances correctly predicted and not belonging to positive class.
- **False Positive** (FP) - number of instances incorrectly predicted and belonging to positive class.
- **False Negative** (FN) - number of instances incorrectly predicted and not belonging to positive class

Overall, confusion matrices are very useful for evaluating and diagnosing the performance of classification models

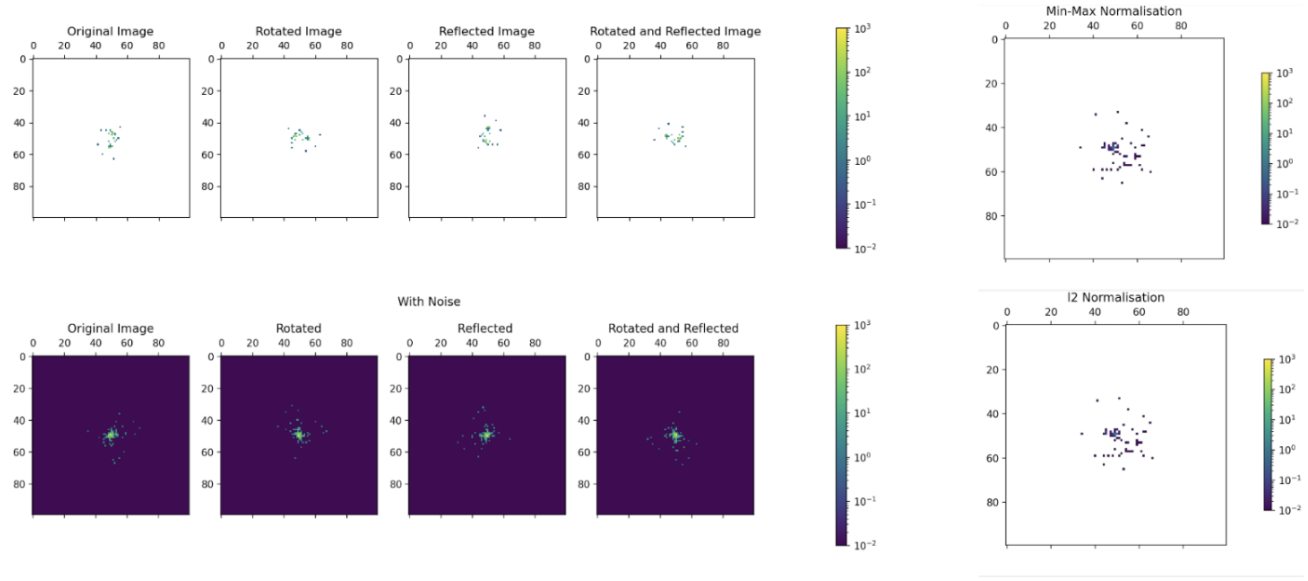

Figure 4: (Left) Illustration of Data Augmentation in the form of rotation, reflection and reflection with rotation (Top-left). Also, this is repeated with noise (Bottom-left). (Right) Example of Normalisation of image using two methods mentioned in the section before.

5.1 ML Models Description

Below, I am explaining each model and why I have decided to use them for this task. In order to have "fair" comparisons between each other, the structure of each model's dense layers, optimization, dropout, batchsize, epochs, and activation were the same, except only the initial evaluation (for example CNN convolution and max pooling layers). Later in the section, I am plotting all 3 models together and comparing their training, validation and testing data. To improve the performance of neural networks, I have decided to first: separate data into training 60%, validation 20% and testing 20%, for the ease of visualisation training and overfitting validation data was created. Second, I have randomised the datasets and also shuffled the training and validation data in order to reduce dependence of certain data in certain places of the dataset. This has shown to increase training performance by several percent for each neural network. Instead of plotting separately rotated, reflected and together reflected + rotated I have decided to just plot original and rot + ref graphs for simplicity for this report.

5.1.1 FNN (Feedforward Neural Network)

Is a type of neural network where connections between nodes do not form cycles. FNN are usually used for a variety of tasks, such as including regression, classification and function approximation. I have decided to use this model to compare it with the other two as they are all used for classification. I still need to flatten the image for NN to learn, so it will be a good comparison to the other networks.

5.1.2 CNN (Convolutional Neural Network)

Is a type of network particularly suited for analysing visual data. It uses convolutional layers to automatically and adaptively learn spatial hierarchies of features from the input images. Usually recognise patterns directly from pixel images with minimal preprocessing but, as for this section, I am only using one convolutional layer (in the code I will talk more about using multi-convolutional layers CNNs and compare their efficiency.)

5.1.3 PCA (Principal Component Analysis)

Is a type of statistical technique used for dimensionality reduction. It defines the principal components in the data which are directions of maximum variance. It is done by projecting original data onto a new coordinate system, helping with reducing the amount of data while maintaining most important information. I have decided to use this to see how selecting the most important information about the images affects the results. I have tested the efficiency of the number of PCA used for training and found that selecting the first 2000 PCA gave accuracy of 73.5%. Thus, for the rest of the project I will be using 2000 PCA of each jet image for training the neural network.

5.2 Analysis

5.2.1 NNs Trained on Raw Data

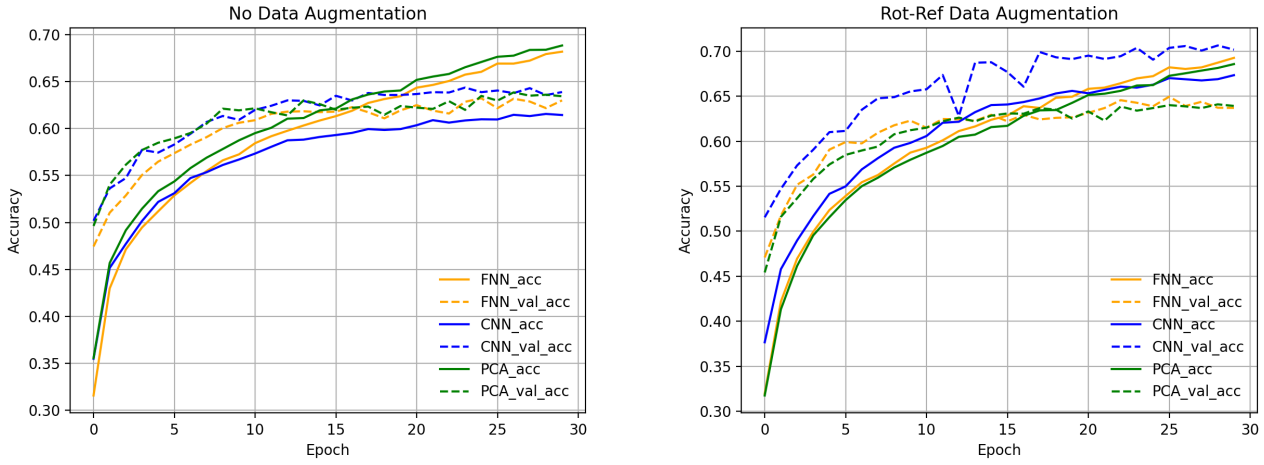

Figure 5: Illustration of FNN (orange), CNN (blue) and PCA (green) neural networks being trained on raw data (left graph) and rotated + reflected data (right)

From the graph 5.2.3 and data 5.2.1 above we can see that overall CNN has better performance and has no overfitting for raw data with no noise while FNN and PCA have lower performance (not significantly) but do have large overfitting compared to CNN.

Models Evaluation for Raw Data			
End Parameter	FNN	CNN	PCA
Training	0.6819	0.6144	0.6884
Validation	0.6300	0.6389	0.6344
Testing	0.6216	0.6884	0.6187

Table 1: Models Performance on Raw Data with no Noise

Models Evaluation for Augmented Data			
End Parameter	FNN	CNN	PCA
Training	0.6926	0.6734	0.6858
Validation	0.6370	0.7017	0.6392
Testing	0.6332	0.6940	0.6387

Table 2: Models Performance on Augmented Data with no Noise

From the graph 5.2.3 and 5.2.1 data above, CNN still performs the best. Augmented image increase CNN testing performance by 6.76% together with validation 6,28% and training 5,9%. Still, no overfitting is present compared to FNN and PCA models where performance of testing and validation hasn't been increased much or at all. Overall, augmentation of image largely affects CNN model increasing its classification performance of particles.

From figure 5.2.1 above we can see the classification performance of different NN models. Different particles have different effects on the model. Taking a close look:

- **Gluon** best classified by CNN (57%) and FNN (56%) and worst by PCA (0.045). The difference is not much between CNN and FNN but large with PCA. PCA mistakes gluon for quark with (51%) prediction which is very fatal for the neural network. Solving this by further introducing rotation and reflection may affect this (mentioned later)

- **Quark** best classified by PCA (74%) and CNN (71%) and worst by FNN (55%). The difference between two NNs compared to FNN is huge. FNN mistakes quark for gluon by (27%), this may be solved by introducing image augmentation (mentioned later in the section)

- **W Boson** best classified by CNN (83%) then similar by PCA (78%) and NN (77%). The difference of CNN is large compared to the other two NNs. There are no large false predictions for this particle.

- **Z Boson** classified equally by all 3 neural networks. And all three had false predictions. Z boson is often mistaken for W boson FNN with 19%, CNN together with PCA with 24% which is very confusing why such is happening. This will be explored later in the sections.

- **Top Quark** best classified by CNN (82%) then by PCA (79%) and worst by FNN (68%). This particle was confused for FNN with gluon by 15% and for PCA with Z boson by 11%.

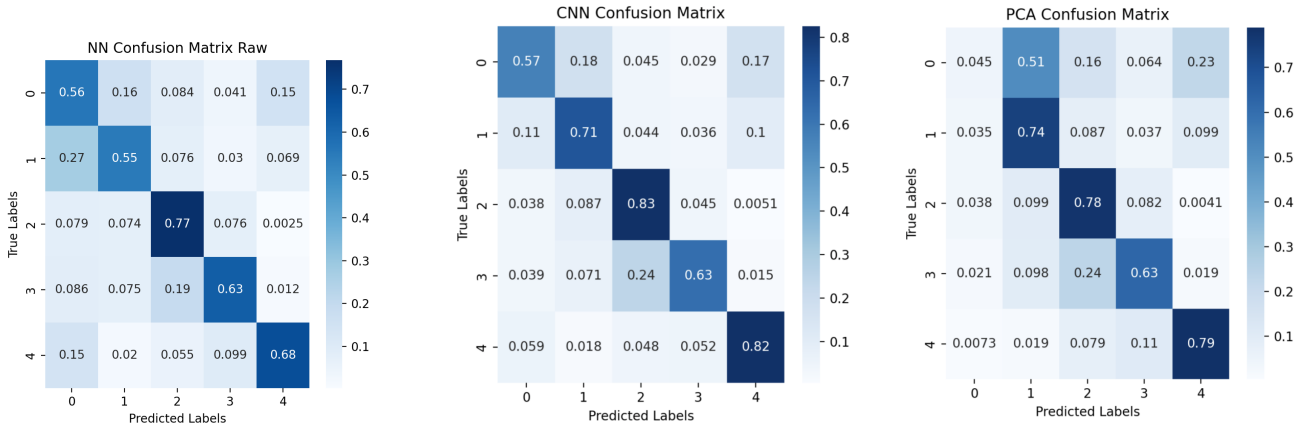

Figure 6: Illustration of confusion matrices for 3 different neural networks

Overall, most hardest particle to classify is Z bosons and Gluon probably due to similarity of W and Z boson and gluon small transverse momentum (intensity of the pixel), and best neural network which has highest efficiency in particle classification is CNN with highest percentage of true positive results and lowest true negative cases.

5.2.2 FNN Confusion Matrices with Data Augmentation

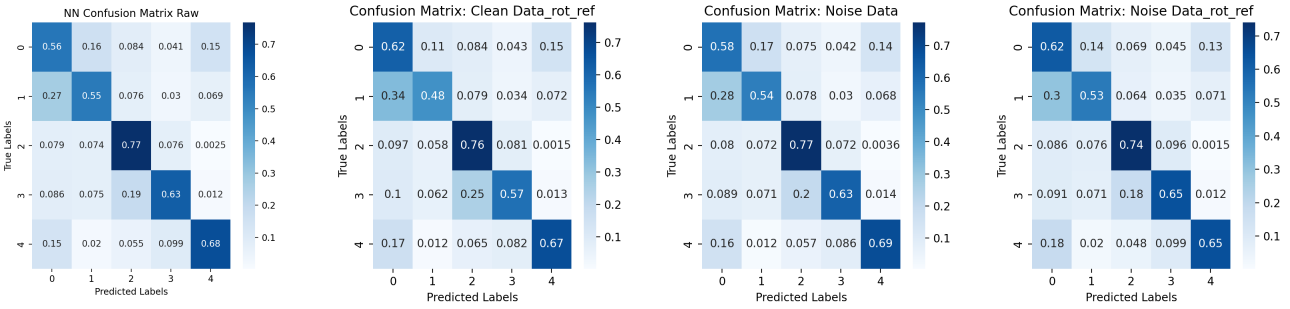

Figure 7: Illustration of confusion matrices of FNN neural network for 4 different data types

From the 5.2.2 there is shown a pattern that introducing data augmentation in the form of rotation and reflection worsened the classification for the FNN model. And addition of noise to FNN hasn't made much of the benefit, actually it introduced slightly more errors. For future, need to do the same analysis for CNN as it showed higher results when introducing noise to the images (but not rotation and reflection).

5.2.3 MLs Trained on Noise

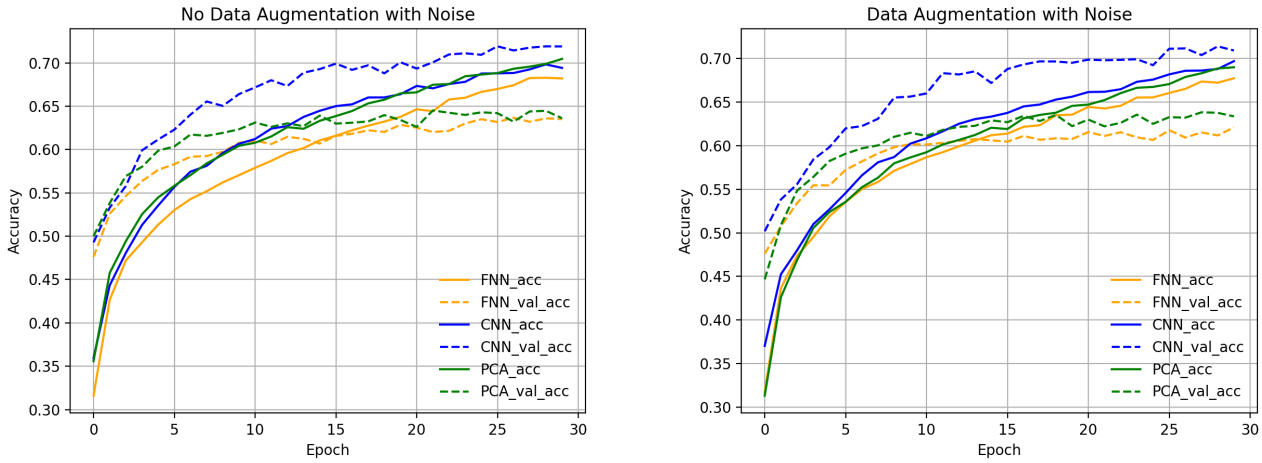

Figure 8: Illustration of FNN (orange), CNN (blue) and PCA (green) neural networks being trained on raw with noise data (left graph) and rotated + reflected data (right) with noise.

Models Evaluation for Noise Data			
End Parameter	FNN	CNN	PCA
Training	0.6821	0.6942	0.7047
Validation	0.6352	0.7190	0.6361
Testing	0.6328	0.7137	0.6314

Table 3: Table to test captions and labels.

From the graph and data above, is shown thatbest performing neural network which is trained on data with noise is CNN again, having highest testing performance of 71,37% which is large compared to other two networks. Also, CNN doesn't have overfitting , suggesting that training on this sort of data is beneficial for the network. Comparing CNN training on data with no noise 62,64% there is a massive jump of 8,73% in performance testing but not so much in training and validations. For FNN and PCA networks there is no much difference, they still perform the same compared to data without noise. In the end, adding noise to the data only beneficial to CNN network.

From the graph and data above, it is shown that adding augmentation in the form of rotation and reflection to data with noise worsens the performance of all neural networks compared to NNs without noise and rotref and just with noise, making addition of this augmentation useless. FNN and PCA still have overfitting and CNN is quite close to it, but overall CNN performs better than the other two networks.

Models Evaluation for Noise Augmented Data			
End Parameter	FNN	CNN	PCA
Training	0.6772	0.6969	0.6898
Validation	0.6209	0.7091	0.6335
Testing	0.6133	0.6967	0.6238

Table 4: Table to test captions and labels.

5.2.4 MLs Trained on Normalised Data

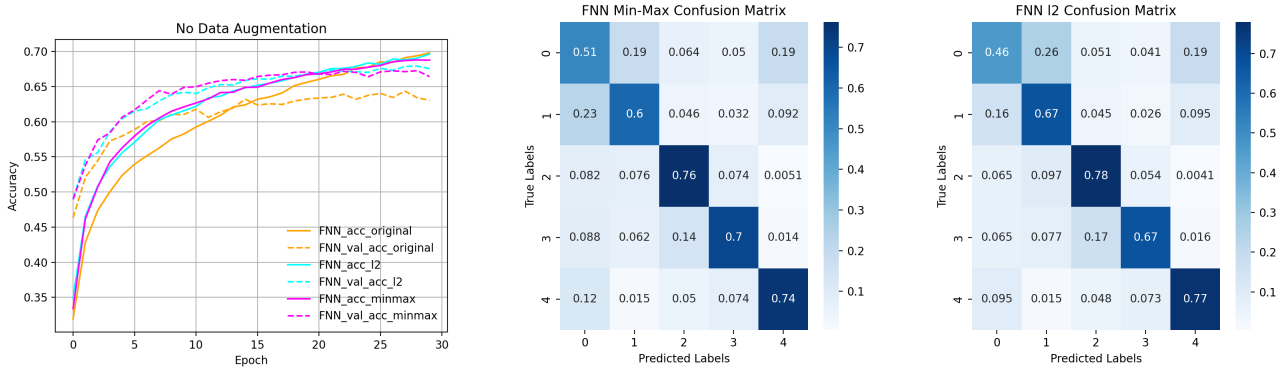

Figure 9: (Left) Illustration of FNN network trained on Normalised data. (Cyan) vector normalised data with accuracy: (0.6684). (Magenta) min-max normalised data with accuracy: (0.6614) . (Orange) Original data (with no augmentation) with accuracy: (0.6330). (Right) respective confusion matrices for each FNN trained model.

FNN Model Performance			
End Parameter	Original	Min-Max	l^2
Training	0.6913	0.6902	0.6958
Validation	0.6466	0.6662	0.6801
Testing	0.6330	0.6614	0.6684

Table 5: Table to test captions and labels.

Looking at the figures and data above, it is shown that introducing normalisation to images increased the performance of the FNN network by roughly 3,5% which is not much but significant. However overfitting is still present. Vector normalisation performs better than min-max normalisation as it is shown by validation data. Looking at confusion matrices, introduction of normalisation worsened identification of Gluon from 56% to 46% and 51% but increased all other probabilities and decreased false positive results.Overall, for FNN network normalisation is beneficial when it comes to image classification.

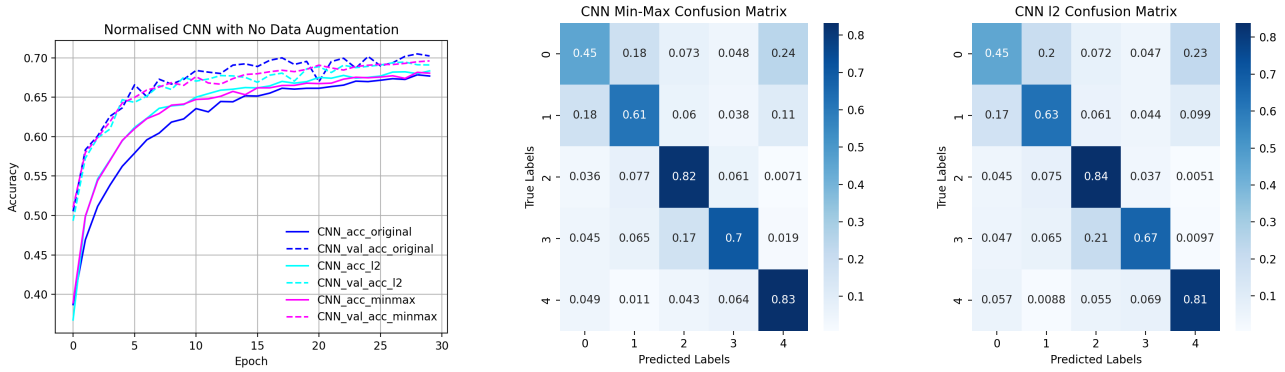

Figure 10: (Left) Illustration of CNN network trained on Normalised data. (Cyan) vector normalised data. (Pink) min-max normalised data . (Blue) Original data (with no augmentation). (Right) respective confusion matrices for each CNN trained model.

CNN Model Performance			
End Parameter	Original	Min-Max	l^2
Training	0.6768	0.6804	0.6833
Validation	0.7023	0.6960	0.6909
Testing	0.6907	0.6832	0.6791

Table 6: Table to test captions and labels.

Introducing normalisation to the CNN model only made the performance of it worse. Looking at the graph in 5.2.4 and data above we can see that there is still no overfitting but, the testing performance of globally normalised data using two techniques worsened performance compared to original data. Looking at confusion matrices from before, introduction of normalisation worsened identification of Gluon and Quark while increasing identification of Z boson and keeping others similar. Overall, normalisation for CNN network is not beneficial.

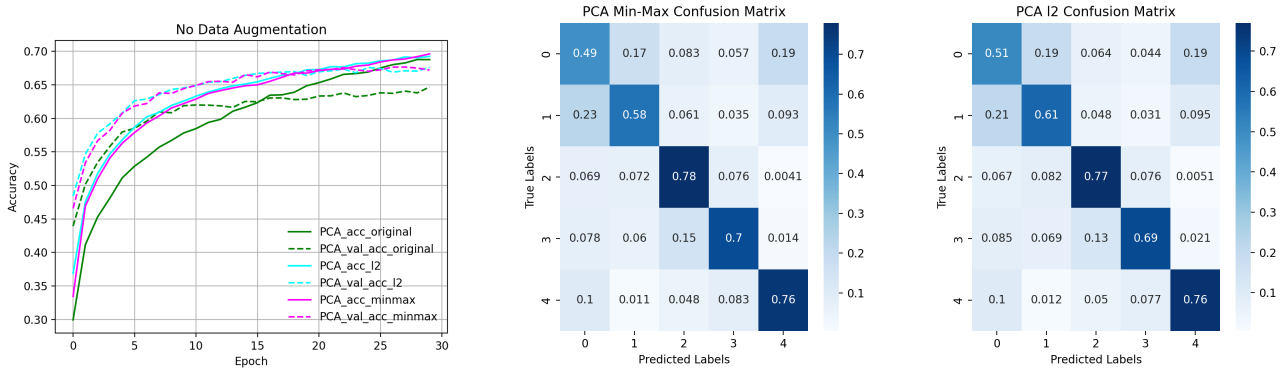

Figure 11: (Left) Illustration of PCA network trained on Normalised data. (Cyan) vector normalised data . (Pink) min-max normalised data . (Green) Original data (with no augmentation). (Right) respective confusion matrices for each PCA trained model.

PCA Model Performance			
End Parameter	Original	Min-Max	l^2
Training	0.6876	0.6961	0.6924
Validation	0.6467	0.6720	0.6762
Testing	0.6322	0.6614	0.6679

Table 7: Table to test captions and labels.

Introducing normalisation to PCA network increases the performance and improves the identification of Gluon as it is shown from 5.2.4, also from the data above using vector normalisation solves the overfitting problem which was consistent in original and min-max normalised data. It reduces the false positive particle identification, increases Z boson probability but decreases Quark, top quark identification. Overall, normalisation for the PCA model is beneficial and especially using vector normalisation which increases network performance by 3,57% and solves overfitting.

6 Methodology for Task 2

For this task, I have chosen to use FNN but I have also included Random Tree and Guassian Bayes functions which were also compared with each other (mentioned in the code). The jet features consists of 59 rows of data of which only 53 chosen as they were representing key jet features picked up by the detectors. I have decided to use all 53 of them to see how well FNN will classify them. I have hot encoded my labelling data like in the previous tasks and made massive array of jet feature which was then fed into FNN raw and augmented, by introducing Normalisation and Noise function (mentioned in section before) to see how well the FNN network will perform with data augmentation. I have used similar data splitting tacktick of separate data into training

60%, validation 20% and testing 20% and random sampling of data and shuffling during training to increase the performance of the model.

6.1 Analysis

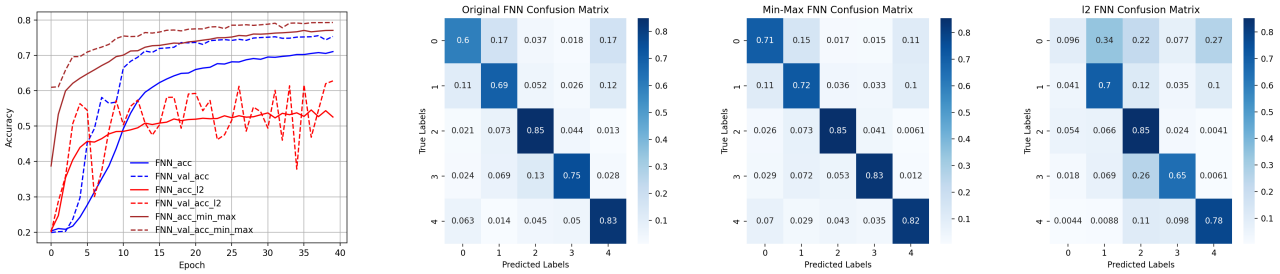

Figure 12: (Left) Illustration of training FNN model on raw, min-max and vector normalised data. (Right) representation of probabilities predicted by the FNN using a confusion matrix.

FNN Model Performance			
End Parameter	Original	Min-Max	l^2
Training	0.	0.7706	0.5256
Validation	0.7543	0.7929	0.6277
Testing	0.7206	0.7865	0.6129

Table 8: Table to test captions and labels.

From the graph and data above, we can see that adding min-max normalisation to the data has increased the performance of the FNN overall with no overfitting. Vector normalisation performs the worst, and looking at the confusion matrix it is shown that gluon particles are not identified at all, this method of normalisation introduces overfitting also, which is overall not beneficial. On the other hand, min-max normalisation improves classification of Gluon, Quark, Z boson and reduces the false positive errors in the matrix. Overall, using min-max normalisation of this type of data is highly preferred.

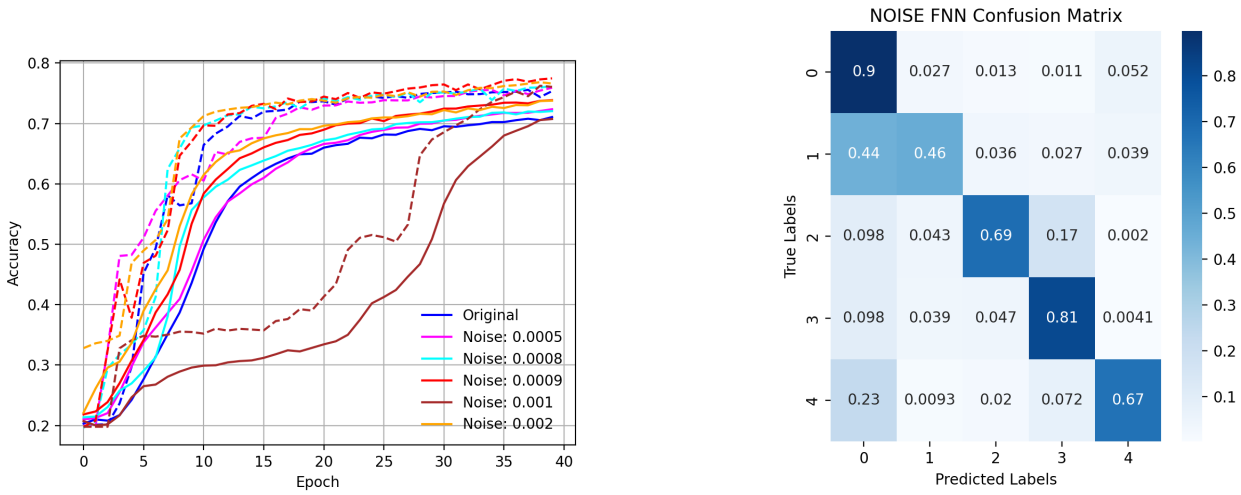

Figure 13: (Left) Illustration of model performance during training (thick line) and data validation (dotted line). (Right) Confusion matrix represented by the best performing model trained on noise = 0.0009.

FNN Noise Trained Model Performance						
Lines	Blue	Magenta	Cyan	Red	Brown	Orange
Training	0.0000	0.7235	0.7203	0.7385	0.7073	0.7377
Validation	0.7543	0.7203	0.7619	0.7746	0.7605	0.7660
Testing	0.7206	0.7294	0.6676	0.7048	0.7337	0.7162

Table 9: Table to test captions and labels.

Introduction of noise had a very strange effect as it is shown in figure and data above. Starting from noise level of 0.0005 and increasing up to 0.002 had a gradual decrease in performance, especially on noise level 0.001 where the start of the training was very poor but in the end catched up with other training. Introducing noise is not as beneficial as min-max normalising data, so it is not preferred. For future research, normalisation of data and then adding noise can be tested to see how well model performs

7 Appendix

7.1 Alternative Models

7.1.1 TASK 1

FNN Extra Models				
Parameters	Model 1	Model 2	Model 3	Model 4
IA func	ReLU	sigmoid	ReLU	tanh
Dropout	0.1	0.2	0.1	0.4
FA func	linear	sigmoid	softmax	softmax
Optimizer	adam	adamax	nadam	adamdelta
Batchsize	500	100	800	250
Accuracy	0.6369	0.6561	0.6339	0.2860
Time	1s	1s	1s	1s

Table 10: Different model parameters for FNN investigated for task 1.

CNN Extra Models				
Parameters	Model 1	Model 2	Model 3	Model 4
Conv2D	5x5	2x2	2x2	5x5
Max Pool.	3x3	3x3	3x3	3x3
Filter	10	10	20	10
IA func	ReLU	ReLU	ReLU	ReLU
Dropout	0.1	0.2	0.4	0.2
FA func	linear	linear	linear	linear
Optimizer	adamax	adamax	nadam	adam
Batchsize	500	250	100	500
Accuracy	0.7109	0.6425	0.6204	0.6925
Time	49s	24s	50s	51s

Table 11: Different model parameters for CNN investigated for task 1.

7.1.2 TASK 2

Different types of model parameters were used. From the table there can be seen a trend that using ELU activation function increases the accuracy dramatically. Also, making the last dense layer activation function to be the same as initial also decreased the accuracy a bit (compared to linear activation) and also introduced errors, and had to reload the model several times for it to work and get proper values.

Extra Models				
Parameters	Model 1	Model 2	Model 3	Model 4
IA func	ELU	tanh	ELU	ReLU
Dropout	0.05	0.05	0.2	0.15
FA func	ELU	tanh	Linear	Linear
Optimizer	adam	adam	nadam	adam
Loss func	SCC	SCC	SCC	SCC
Batchsize	500	100	50	50
Epochs	40	40	30	30
Accuracy	0.7849	0.6916	0.7946	0.7342
Training	6ms	3ms	3ms	3ms
Time	1s	1s	1s	1s

Table 12: Different model parameters for FNN investigated for task 2.

References

[1] Abdalrahman Mohamed Elsheikh Eltahir. "Impacts of Data Pre-processing on Jet Images Classification using Deep Leanring". In: *CERN Summer Student Program* (2023).

[2] Michael Kagan. "Image-Based Jet Analysis". In: *arXiv:2012.09719* (2020).

[3] Eric A. Moreno et al. "JEDI-net: a jet identification algorithm based on interaction networks". In: *arXiv:1908.05318* (2019).