university of groningen

faculty of science and engineering

Bachelor's Project in Physics

Convolutional Neural Network for Noise Reduction and Particle Tracking Applications

Andrey Mikus

supervised by
dr. JG Messchendorp
dr. M Kavatsyuk

July 9, 2021

Abstract

This thesis investigates the applications of a convolution neural network, CNN, for noise reduction and track identification of e^+e^- collision data taken from the drift chamber of BESIII. The CNN was originally developed and tested by Harmjan de Vries in his Master's thesis. The noise reduction capabilities were further explored in a follow-up Bachelor's thesis by Ignacio Graña. The network was shown to be effective in filtering noise and classifying individual particle tracks for Monte Carlo data simulated for e^+e^- collisions in the BESIII experiment. In this thesis, the CNN was trained and further tested for five different event topologies.. The network was successful in filtering noise from events containing up to 20 particle tracks. The feasibility of track identification, broken down into individual track labeling, charge classification and pair recognition was demonstrated. The effect of broken layers on the performance was also examined. The performance of the network for track identification decreases significantly for events with a large number of tracks. We observed that the network is particularly sensitive to the underlying features directly linked to the curvature of tracks, and, therefore, to the transverse momentum and electric charge of the particles. Tracks can be identified successfully in the case they are distinct in those features and among each other. For pair recognition, the opening angle between the tracks appears to be a key feature for discrimination.

Contents

1 Introduction

Modern, state of the art experiments in particle physics, such as the upcoming PANDA experiment at the Facility for Antiproton and Ion Research (FAIR), generate enormous amounts of data. A typical interaction rate in the PANDA experiment is in the magnitude of 10^7 events/s, which translates to 200 GB/s [1]. At this rate, the storage facilities ($\sim$ 3 PB/year), would be exhausted in less than 5 hours. A real-time event selection algorithm, able to reconstruct particle tracks and reduce the pile-up of data, is necessary to circumvent the storage limitation. Machine learning, or more specifically deep learning, is a class of algorithms potentially well-suited for this task.

Deep learning algorithms are widely used in many areas of knowledge, from the natural sciences to mathematics to economics. Common applications include speech recognition, image classification and predictive analytics. In the recent years, the performance and complexity of deep learning algorithms have risen dramatically, largely due to the advances in computational power and the advent of big data. Deep learning methods are based on artificial neural networks, or simply neural networks, that vaguely resemble biological neural networks in animal brains. A Convolutional Neural Network, CNN, is a class of neural networks most suitable for analyzing visual imagery. CNNs have been very successful for biomedical image segmentation tasks, in particular those based on the U-Net architecture. [2–4].

The CNN studied in this thesis was developed by Harmjan de Vries in his Master's thesis titled *Convolutional Neural Network for Reducing Noise and Detecting Tracks in the BES-III Main Drift Chamber* [5], hereinafter referred to as *the Master's thesis*. It uses the U-Net architecture to analyze images containing particle tracks from a collision. The CNN was trained on data generated with Monte Carlo simulations of e^+e^- collisions in the BESIII experiment. Both, noise reduction and track recognition algorithms were shown to be very effective for the decay channel $e^+e^- \to \Psi(2S) \to J/\Psi\, \pi^+\pi^- \to e^+e^-\, \pi^+\pi^-$.

The Master's thesis was followed up by Ignacio Fernández Graña in his Bachelor's thesis titled *Deep Learning for Particle Tracking* [6], hereinafter referred to as *the Bachelor's thesis*. Ignacio investigated the limits of the noise reduction algorithm for the same decay channel, by introducing moderate to large amount of noise to the image. The network maintained a high performance score, despite track-to-noise ratios up to 1:8. However, it was also shown to be ineffective at filtering a single track unrelated to the observed event and at labeling tracks in incomplete events.

In this thesis, the applications and limits of the network are further explored. In an attempt to generalize to other types of events, the network is trained and tested on a number of decay channels, both simple and more complex. The ability of the network to filter noise from high image density events, like those with up to 20 tracks, is evaluated. The track identification algorithm is split into three components: individual track labeling, charge identification and pair identification, in order to assess its capabilities on a variety of tasks. Charge identification will be useful in experiments where the particles are to be treated separately based on their electrical charge. It may also be used in conjunction with individual track labeling, where some tracks are preemptively discarded based on charge. Pair identification is the next step after individual track labeling, in which one exploits the correlations between two oppositely charged

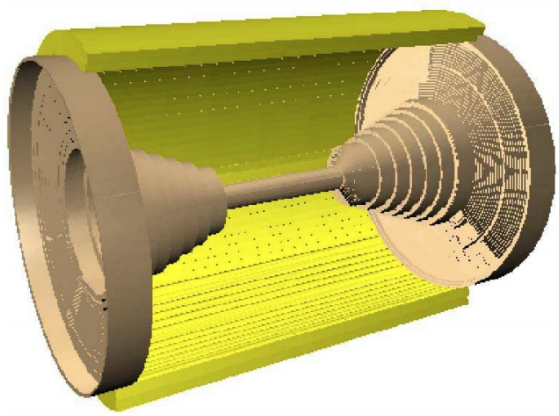
tracks. It provides insight into which tracks belong to which part of the decay chain. Lastly, the effect of broken wire layers is investigated, by testing trained network weights on incomplete events.

2 BESIII detector

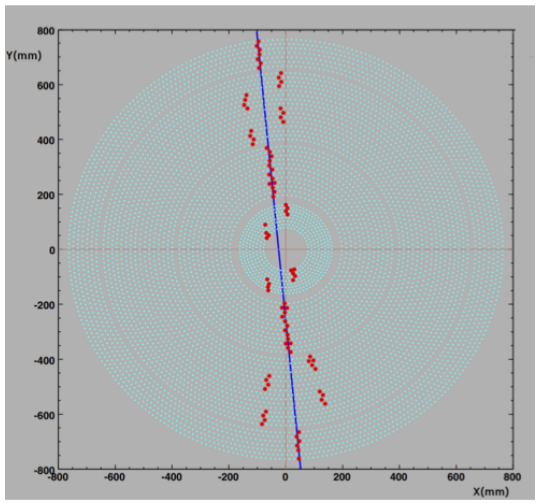
2.1 Working mechanism

The Beijing Spectrometer III (BESIII) is a particle physics experiment conducated at the Institute of High Energy Physics. The experiment began collecting data in 2008 with the intent to study the physics of charm and "charmonium", a meson consisting of a charm quark and a charm antiquark [7]. The discovery of charmonium has had far-reaching consequences for high-energy physics, as it is an important tool for the study of forces between quarks in the context of quantum chromodynamics in the non-perturbative regime [8]. The BESIII experiment studies the decay channels of charmonium produced in e^+e^- collisions in the energy range of 2.0-4.9 GeV.

The innermost tracker in the BESIII experiment is the Main Drift chamber (MDC), pictured in figure 1a. The MDC is 2.4 meters long and contains 6796 tungsten signal wires arranged in 43 layers. It is located inside a 1 T superconducting solenoid magnet, which curves the paths of charged particles. The extent of the curvature of the particle track is used to calculate the momentum of the particle. A hit on the wire is recorded whenever the ions from the ionization of the helium and propane gas mixture by a travelling charged particle drift to the nearest wire. The data corresponding to wire hits can be stored as a 2D image, representing the transverse cross section of the MDC, as depicted in figure 1b. In this thesis, the data is generated via Monte Carlo simulations of a number of different decay channels of charmonium with distinct event topologies.



(a) Main Drift chamber (MDC) of the BESIII [7].



(b) Example of cosmic ray detection in the MDC. Red dots are measured hits, blue line is true path of particle and teal dots are the wires [9].

Figure 1: BESIII Main drift chamber cross-section and mechanism of particle detection.

2.2 Relevant decay channels

Both in the Master's and the Bachelor's theses, the decay channel $e^+e^- \to \Psi(2S) \to J/\Psi\pi^+\pi^- \to e^+e^-\,\pi^+\pi^-$ was the only process studied for the purpose of investigating noise reduction and track identification. The $\Psi(2S)$ resonance is an excited charmonium state, with a rest mass of 3.68 GeV/c^2. The aim of this thesis is to address the performance of the CNN for different decay topologies with varying complexity. We therefore simulated several other decay chains of the $\Psi(2S)$ resonance, which allows us to perform a thorough and systematic investigation.

The decay channels that were studied in this thesis are listed below. The text in bold indicates the shorthand name assigned to the channel to be used throughout the thesis.

1. **pipiee** – $e^+e^- \to \Psi(2S) \to J/\Psi\ \pi^+\pi^- \to e^+e^-\,\pi^+\pi^-$.
   The decay channel studied in the Master's and the Bachelor's theses. The J/Ψ is also an excited charmonium state with a rest mass of 3.097 GeV/c^2.

2. **npipi** – $e^+e^- \to \Psi(2S) \to n(\pi^+\pi^-)$.
   n is the number of $\pi^+\pi^-$ multiplicities resulting in $2n$ tracks. This channel is selected to investigate the feasibility of the network to recognize and classify events with a large number of tracks and to filter noise in high-density events. The channel does not produce a cascade of excited state and particles, therefore the particles have equal momentum distributions.

3. **2pipi alt** – $e^+e^- \to \Psi(2S) \to J/\Psi\ \pi^+\pi^- \to \pi^+\pi^-\,\pi^+\pi^-$.
   The reaction is a cascade process in which the transition $\pi^+\pi^-$ pair has a much lower momentum than the $\pi^+\pi^-$ pair decaying from the J/Ψ state. It has an event topology very similar to that of the *pipiee* reaction, which makes it useful for supporting the conclusions drawn from studying *pipiee*.

4. **3pairs** – $e^+e^- \to \Psi(2S) \to K^+K^-\,\pi^+\pi^-\,\pi^+\pi^-$.
   The decay chain is broken down into:

   - $\Psi(2S) \to J/\Psi\ \pi^+\pi^-$
   - $J/\Psi \to \phi\,\pi^+\pi^-$
   - $\phi \to K^+K^-$

   The decay chain involves intermediate unstable resonances, namely J/Ψ and ϕ ($s\bar{s}$), that decay into 3 distinct long-lived final state pairs. The ϕ meson is a vector meson with a rest mass of 1.02 GeV/c^2. The distinct final states make this reaction useful for exploring the pair and track identification algorithms.

5. **4pairs** – $e^+e^- \to \Psi(2S) \to \pi^+\pi^-\,K^+K^-\,K^+\pi^-\,\pi^+\pi^-$.
   The decay chain is broken down into:

   - $\Psi(2S) \to J/\Psi\ \pi^+\pi^-$
   - $J/\Psi \to \phi\,K_S\,K^+\pi^-$
   - $\phi \to K^+K^-$
   - $K_S \to \pi^+\pi^-$

It is similar to the *3pairs* reaction, only with an additional intermediate K_S (neutral kaon) resonance with a rest mass of 0.498 GeV/c^2. The K_S resonance has $c\tau$ equal to a couple of centimeters, meaning the decay occurs away from the interaction vertex. This brings another level of complexity to the topology, which makes this decay an interesting candidate for investigation. The decay result is 4 long-lived pairs.

3 Deep Learning

3.1 Overview

Deep learning is a subset of machine learning based on the methods of artificial neural networks, or, simply, neural networks. A neural network is a series of algorithms based on a collection of connected nodes, called neurons. Each neuron outputs a single value, which is computed by some non-linear function, known as the *activation function*, of the sum of its inputs. The connections between neurons typically have *weights* that represent the strength of the connection. These weights are adjusted during the learning process to improve the quality of predictions. Neurons are usually aggregated in a series of layers: one input layer, one output layer and zero or more hidden layers. Each layer of the network takes the outputs of the previous layer as inputs, which increases the complexity of detected features as the algorithm propagates through the layers. The hidden layers are not visible to external systems, hence the name.

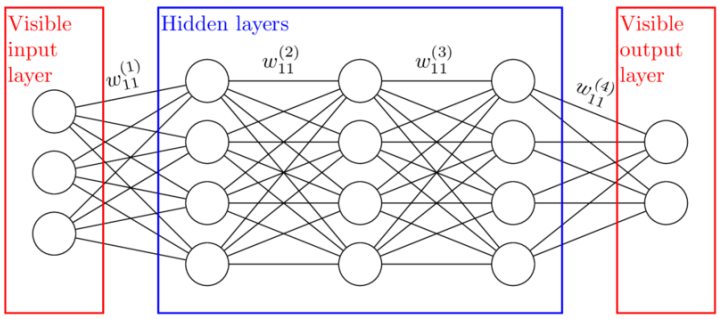


Figure 2: The schematic of a neural network with three hidden layers.

Artificial neural networks learn via supervised learning. In supervised learning, the network is presented with processing examples, the *training set*, containing a known input and target output. The network compares its predictions to the target output by determining the difference, the error value, between them. Subsequently, using this error value, the weights of the network are adjusted according to a learning process, such as *gradient descent*. The performance of the network is evaluated using a separate dataset, called the *testing set*. The testing set is kept separately from the training set to avoid overestimating the performance by *overfitting* the model. Neural networks are excellent tools for approximating functions regardless of their complexity. However, studying the structure of the network does not provide insights on the form of the function being approximated. Therefore, neural networks are sometimes described as "black-box models".

3.2 Learning process

The learning process of a neural network consists of two stages: forward propagation and backpropagation. In forward propagation, the input data is transformed successively by propagating through the network layers, until it reaches the output layer where the prediction is made. During backpropagation, the direction of the data flow is reversed. The network efficiently computes the best adjustments needed for each weight to minimize the error value, starting at the last layer.

3.2.1 Forward propagation

Forward propagation is a feature of feedforward neural networks, networks for which the connections between the neurons are not cyclical. The input data propagates forward, from the input layer, through the hidden layers (if any) and to the output layer. During forward propagation, the output of a neuron is calculated by taking the weighted sum of all inputs and applying a non-linear *activation function*. Without the non-linear activation function, the neural network simply acts as a linear transformation, which drastically reduces the number of problems it can solve.

The activation functions used in the architecture of the neural network investigated in this thesis are:

1. ReLU (Rectified Linear Unit) function:

$$R(z) = max(0, z). \tag{1}$$

   The ReLU function has been shown to yield a higher performance than other activation functions for networks with many layers [10]. It is used in all hidden layers of the neural network.

2. Sigmoid function:

$$\sigma(z) = \frac{1}{1 - e^{-z}}. \tag{2}$$

   The sigmoid function is used in the output layer to constrain the predictions between 0 and 1.

At the end of forward propagation, the network computes the *loss function*, which outputs a real number describing how "far" the prediction is from the target output. The goal of the neural network is to minimize this loss function.

3.2.2 Backpropagation

Backpropagation, short for backward propagation of errors, is an algorithm for training feedforward neural networks. It allows to efficiently compute the gradient of the loss function with respect to the weights. The efficiency of backpropagation makes it feasible to use gradient methods, such as gradient descent, for updating the weights. In gradient descent, the loss function is minimized by updating the parameters in the direction of its gradient.

There are three primary types of gradient descent: batch, mini batch and stochastic. In batch gradient descent, the parameters of the network are updated after one iteration through the

training set, meaning that the entire training set is used to compute the gradient. In mini-batch gradient descent, the training set is divided into batches of some fixed sized. The parameters are updated after each batch has been processed. Lastly, stochastic gradient descent is simply an extreme instance of mini-batch gradient descent with a batch size of 1. For batch gradient descent, the network must process the entire training set before making an update to the parameters, which is inefficient. Stochastic gradient descent can also be inefficient due to the frequency of parameter updates and may also lead to overfitting. In this project, mini-batch training with a batch size of 50 is implemented, as was successfully implemented in the Master's thesis.

The network uses the Adam optimization algorithm, which is an extension to stochastic gradient descent. It was first proposed in 2016 [11] and since then has been shown to compare favourably to other stochastic optimization methods [12]. The method computes individual learning rates for each network parameter and adapts them during training. This accelerates the gradient descent of the loss function towards the minimum, as compared to other fixed learning rate algorithms. For a more detailed discussion, see section 3.4 of the Master's thesis [5].

3.3 Overfitting and underfitting

The main goal of a neural network is to learn to generalize well to new data. Neural network models are generally trained on as much training data as possible to cover as many variations of data as possible. This allows the model to make good predictions on data it has never previously encountered. Overfitting and underfitting refer to issues that the model might suffer from, hindering its ability to generalize.

Overfitting is the scenario where the network has trained sufficiently well to minimize the loss function and to make good predictions on the training set, but performs poorly for the testing set. Essentially, the model has "memorized" the features of the training set, which makes it less applicable and generalizable to the testing set and new data. Overfitting most commonly occurs either due to the limited variety in the training dataset or due to overtraining the model. The techniques used to combat overfitting are called *regularization techniques*. These include L1 and L2 regularization methods, early stopping and dropout [13].

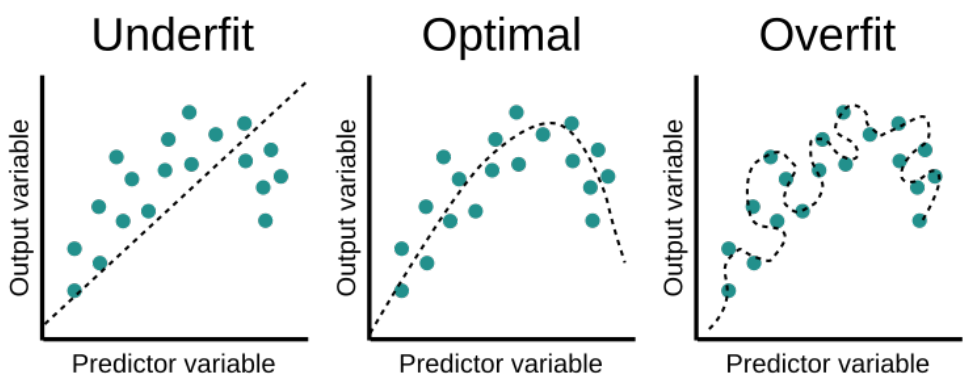


Figure 3: A representation of an underfitting, optimal fitting and overfitting.

Underfitting is the scenario where the model cannot make good predictions on neither the training set nor the testing set. It usually means that the network was not sufficiently trained or that the model is not complex enough to accurately identify the dataset's relevant features. Solutions to underfitting include revising the architecture, adding more hidden layers, or simply, increasing the duration of training.

3.4 Convolutional Neural Networks

A convolutional neural network (CNN) is a class of deep neural networks specialized in analyzing visual imagery. CNNs are similar to ordinary neural networks, in the sense that they are comprised of the same building blocks: neurons with trainable weights and biases. They have a loss function that is calculated in the output layer and they use forward propagation and backpropagation to adjust the weights and improve the quality of predictions. CNNs exploit the nature of 2D images by arranging the neurons in three dimensions: width, height and depth. The neurons in a layer are only connected to a small region of the preceding layer, which drastically reduces the number of trainable parameters, decreases the complexity of the model and prevents overfitting.

3.4.1 Convolutional layers

Convolutional layers are the core building blocks of convolutional neural networks. The layer performs a linear operation, called a convolution, that is an element-wise multiplication of an $n_f \times n_f$ filter (or kernel) by $n_f \times n_f$ patches of an $n \times n$ input, which is then summed to produce a single value. The filter is a two-dimensional array of weights that the network aims to optimize in training. The filter is intentionally smaller than the input ($n_f < n$), such that it may be systematically applied to each overlapping $n_f \times n_f$ patch of the input. The result of this operation is a two-dimensional *feature*, or *segmentation, map* that describes the detected features of the input. A convolution operation is depicted in figure 4.

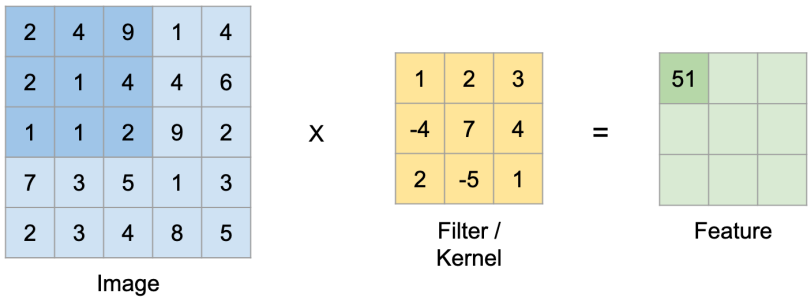


Figure 4: A convolution operation between a 3×3 filter and a 5×5 input, depicting an element-wise multiplication of the filter with the first filter-sized patch of the input.

A convolution operation decreases the size of the image. "Same" padding calculates the dimensions required to preserve the size of the image. In this project, a "same" zero-padding is applied. Another variable parameter is the stride s, which is the distance the filter shifts across

the input image after each successive application. In this project stride is kept at a fixed value of 1.

3.4.2 Pooling

Pooling is a downsampling operation, typically applied after a convolutional layer. Max pooling is the most commonly used type of pooling operation. It divides the input into $n \times n$ subregions and selects the maximum value from the current pooling view, which preserves the detected features. Other types of pooling include average pooling, which averages the values of the current view, and global pooling, which reduces the dimensions of a feature map to a single value. In this project, a 2×2 max pool is used to extract the most important features of the feature map and reduce the dimensions of the input image by 2. A 2×2 max pooling operation is depicted in figure 5.

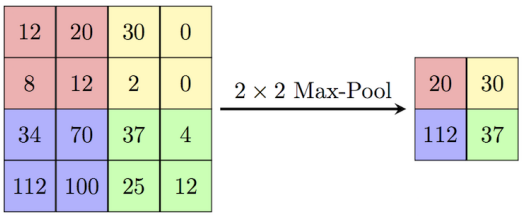


Figure 5: A 2×2 max pooling operation applied to a 4×4 input.

3.4.3 Batch Normalization

Batch normalization is a technique first proposed in 2015 [14]. It has the effect of greatly accelerating the learning process. In some cases, batch normalization may even improve the performance of the network by behaving as a regularization technique, which helps prevent overfitting. The reasons behind its effectiveness are still a subject of discussion. Originally believed to mitigate the problem of internal covariant shift, some scholars have recently come out with studies supporting alternative reasons [15]. In this project, batch normalization is applied after each convolutional layer.

4 Hardware and software specification

4.1 Deep learning framework

The convolutional neural network was originally written using the open-source software library Keras, which provides a simple, but flexible, Python interface for neural networks. Tensorflow acts as the backend for Keras, handling low-level operations such as tensor products and convolutions. Keras offers a declarative interface for composing neural networks, which allows for quick implementations of prototypes. However, this comes at a cost of computational speed, with Keras ranking consistently slower compared to other frameworks such as PyTorch or Caffe [16, 17]. Additional code for input/output processing and plotting was written and executed in Python 3.6.10.

4.2 Server specifications

The network was trained on a server with 48 Intel Xeon E5-2650 v4 @ 2.20GHz CPUs and 100GB of RAM. The server did not have a GPU, which offers massive training speed improvements for convolutional neural networks. In some cases, training the model is 4-5 times faster [18] on a GPU than on a CPU. The main reason is that GPUs are bandwidth optimized, meaning that they are able to perform heavy matrix multiplication and convolution operations more efficiently.

5 Network design

5.1 Overview

In this thesis, no changes are made to the network architecture developed in the Master's thesis, except for the number of output segmentation maps, which depends on the training algorithm. The noise reduction algorithm will produce one feature map - corresponding to the image with reduced noise. For the track identification algorithms, the number of feature maps will vary depending on the number of tracks or pairs. For instance, for charge identification, the number of feature maps will be two - representing positive and negative charges.

5.2 U-Net architecture

The network developed in the Master's thesis is based on the U-Net architecture. The U-Net CNN was first proposed in 2014 [2] and specifically developed for biomedical image segmentation. The U-Net has become the most prominent CNN architecture in the field, due to its capabilities for fast and precise segmentation of images [3,4]. The resemblance between tracking detector images and biomedical images was the motivation behind implementing the U-Net, and in fact, it outperformed the standard CNN during architecture comparison in the Master's thesis.

The U-Net consists of a contracting part and an expanding part. In the contracting part, the image is repeatedly convolved using a 5×5 filter with "same" padding and $s = 1$ stride, followed by a ReLU activation and batch normalization, and finally by a 2×2 max pooling operation. The dimensions of the image are halved and the number of filters is doubled after every max pooling layer, beginning with 24 filters in the first layer. In the expanding part, the image is convolved with a filter having the same properties as in the contracting part, followed by a 2×2 up-sampling operation, and finally by a 2×2 convolution with "same" padding and $s = 1$ stride. The up-sampling operation repeats each value in the image to a 2×2 grid, doubling its size. The operations in both the contracting part and expanding parts repeat four times. The last layer is a convolutional layer with a 1×1 filter, "same" padding and $s = 1$ stride, followed by a sigmoid activation function to produce the output segmentation map(s). The architecture is illustrated in figure 6.

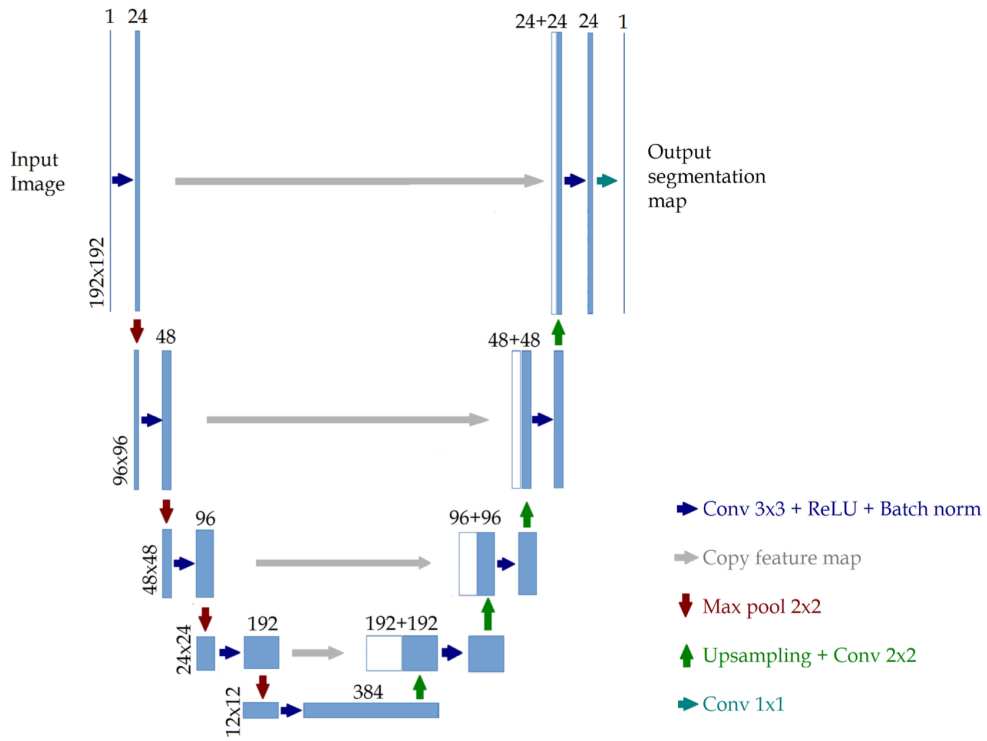


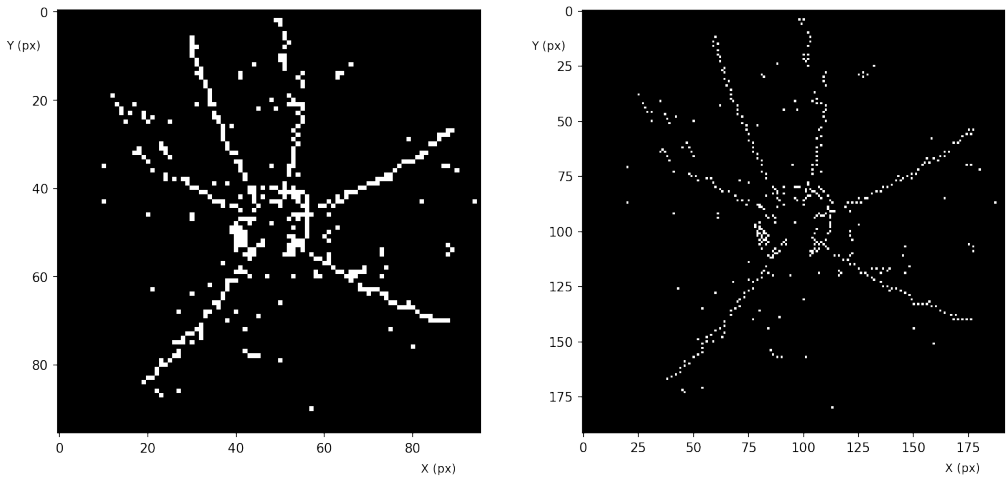
Figure 6: U-Net architecture for noise reduction. The number of output segmentation maps varies for track identification, depending on the algorithm used.

5.3 Data processing

5.3.1 Binning

Data binning is a pre-processing technique where the data points that fall within a certain interval (a bin) are replaced by a representative value, usually the central value, of the interval. In the context of the two-dimensional image data used as input to the CNN, binning is the process of converting the raw event data, represented by x and y coordinates of the measured wire hits, to an $n \times n$ pixels binary image, where 1's represent wire hits. As devised in the Master's thesis, 192×192 pixels are the ideal dimensions for the input image. It is calculated such that it is impossible to bin several wire hits into one pixel. The number $192 = 3 \cdot 2^6$, which makes it possible to apply 2×2 max pooling at most 6 times. Other candidates are 128×128, 96×96 and 64×64 pixel images.

In this thesis, we will mostly be working with 96×96 pixel images, which are 4 times smaller than the 192×192 pixel images. This is mainly due to the fact that working with a smaller image dramatically increases speed of training and reduces RAM usage, allowing us to carry out more experiments and train several models at once. Even at a reduced resolution, the network is sufficiently able to pick up on the key features of the tracks. A performance comparison between 96×96 and 192×192 pixel images for noise reduction in events with a large number of tracks and high levels of noise is made in section 6. A visual comparison between the resolutions is illustrated in figure 7.



(a) 96 × 96 pixels image (b) 192 × 192 pixels image

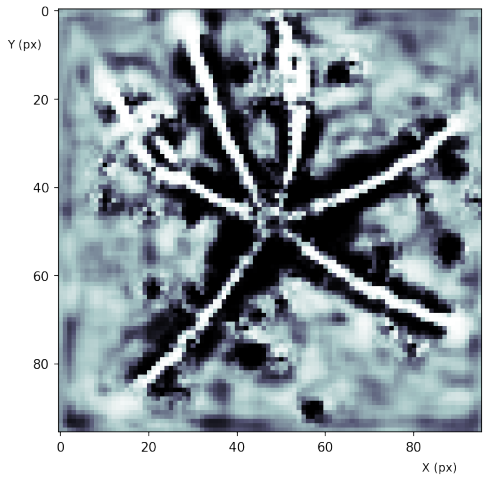
Figure 7: Example of a *3pipi* event at two different resolutions.

5.3.2 Input processing

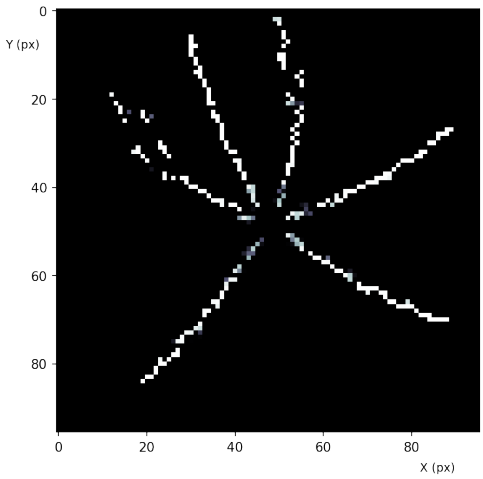
The next step after binning event data to a two-dimensional image is to label the pixels according to their type. For the noise reduction algorithm, a pixel represents either the tracks or the noise to be removed. For the track identification algorithms, a pixel represents either the relevant track(s) or the track(s) to be removed. The pixels to preserve are assigned a value of 1 and the pixels to remove are assigned a value of -1 in the target image(s). This approach to labeling pixels allows the undesirable pixels to be easily discarded with a ReLU function in both the loss functions and the performance metrics. Examples of input images and the corresponding target images after processing are shown in figures 9, 11 and 14.

5.3.3 Output processing

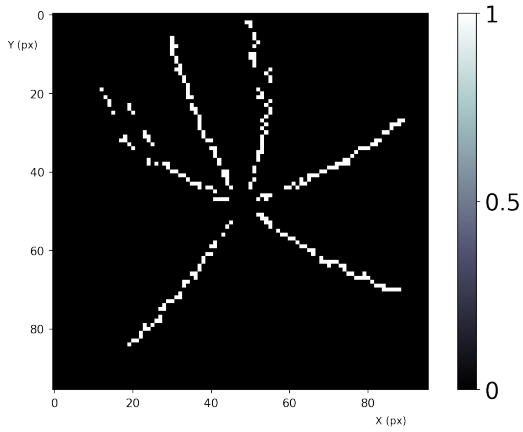
Output processing consists of three stages. Firstly, the network outputs a segmentation map or several segmentation maps in the case of track identification. Since the activation function of the output layer is a sigmoid function, the segmentation map is comprised of pixels with values between 0 to 1, which represent the 'confidence score' or the probability that the pixel is being predicted correctly. Then, the segmentation map is filtered by removing all the points that were not part of the true image. In the last stage, the performance of the network is evaluated by iterating over threshold values between 0 and 1 in steps of 0.01, in search of the value that yields the best performance. All the pixels with a confidence score below the threshold are discarded. The values of the remaining pixels are set to 1, such that the values of the pixels in the image are either 0 or 1. The three stages of output processing are illustrated in figure 8.



(a) Output of the network - a segmentation map.



(b) Filtering points not part of the input image.



(c) Applying a threshold value of 0.44.

Figure 8: The three stages of output processing for a *3pipi* event after noise reduction.

5.4 Loss functions and performance metrics

It is critical to select appropriate loss functions and performance metrics that accurately reflect the underlying algorithm. As stated previously in section 3.2, neural networks learn by minimizing the loss function. However, the loss function is not a good indicator of the network's performance, because it does not take the value of the chosen threshold into account. All performance metrics are therefore functions of the threshold. The performance metrics chosen in this thesis range in value from 0 to 1, where 1 is the highest possible score.

5.4.1 Noise reduction

The loss function for the noise reduction algorithm is a mean squared error function:

$$L(y, \hat{y}) = \frac{\sum_{ij,n} (\hat{y}_{ij} - x_{ij} y_{ij})^2}{\sum_{ij,n} x_{ij}}, \tag{3}$$

where $\hat{y}$ is the target output, y is the prediction and x is the input. The sum is taken for all pixels i, j of the image, for all n batches. Each predicted pixel y_{ij} is multiplied by the corresponding input pixel x_{ij} to discard any pixels in the prediction that were not part of the input image, i.e where $x_{ij} = 0$.

The performance metric for noise reduction is the classification accuracy:

$$\text{accuracy} = \frac{\text{number of correctly predicted data points}}{\text{total number of data points}}, \tag{4}$$

simply referred to as accuracy.

5.4.2 Track identification

The loss function for track identification

$$L(y, \hat{y}) = 1 - \frac{\sum_{ij,n} 2\hat{y}_{ij} - y'_{ij}}{\sum_{ij,n}(2\hat{y}_{ij}y'_{ij} + \hat{y}_{ij}(1 - y'_{ij}) + y'_{ij}(1 - \hat{y}_{ij}))}, \tag{5}$$

where y'_{ij} is short for $x_{ij}y_{ij}$, is directly based on the definition of the performance metric F1 score:

$$\text{F1 score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \tag{6}$$

where precision and recall are defined as:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}; \quad \text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}. \tag{7}$$

The precision is the number of true positives divided by the number of all points identified as positive (correctly or incorrectly), whereas the recall is the number of true positives divided by the number of all points that should have been identified as positive. The F1 score is a harmonic mean of the precision and the recall, meaning that both precision and recall are assigned an equal weight. A more generic F_β score allows to assign custom weights to the precision and recall, but for the purposes of this thesis, the F1 score is a sufficient metric.

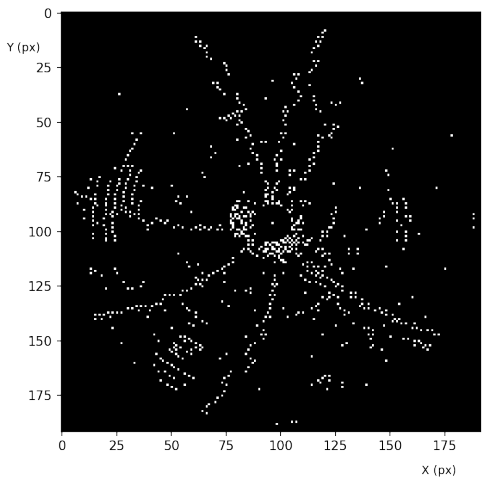
6 Results

6.1 Noise reduction

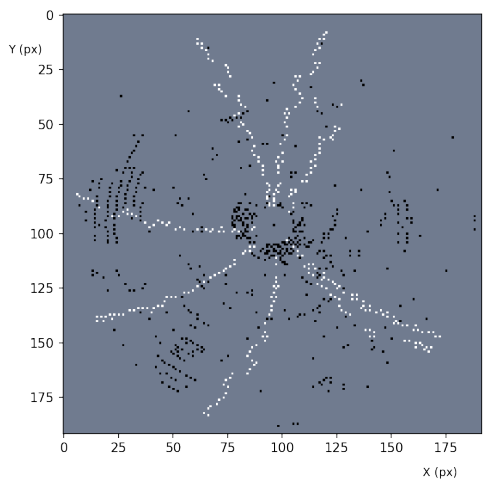
The performance of the noise reduction algorithm has already been covered extensively in the Master's and the Bachelor's theses. Track-to-noise ratios ranging from 1:1 up to 1:8 were studied for *pipiee* events. The model was tested on 1,000 images after being trained on a training set of 10,000 images for 25 epochs. For a ratio of 1:1, a maximum accuracy in the testing set of 97.1% was found, with a slight drop to 95.2% for a ratio of 1:8. It is unnecessary to increase the fraction of noise beyond the noise ratio of 1:8, since that would correspond to a very unrealistic scenario.

Instead of further experimenting with the track-to-noise ratios, the noise reduction algorithm was applied to events with a larger number of tracks, namely the *npipi* reaction for $n = \{3, 4, 5\}$, i.e. 6, 8 and 10 tracks. Generating *npipi* data beyond $n = 5$ is very resource-consuming and in practice, such a reaction has an extremely small probability of ever occurring. To produce images with even more tracks, events from the *pipiee* dataset were programmatically stacked on top of each other. For stacked *pipiee* events, the number of tracks $= 4s$, where s is the number of stacks. $s = \{3, 4, 5\}$ resulting in events with 12, 16 and 20 tracks. The noise ratios were taken directly from the *pipiee* dataset, and are therefore relative to 4 tracks. For example, at 1:8 noise ratio, an image with 6 tracks will have the exact same noise data points as the one with 20 tracks.

Combining the *npipi* events and the stacked *pipiee* events, the model was trained and tested on images with 6, 8, 10, 12, 16 and 20 tracks for 1:2, 1:4 and 1:8 noise ratios. The 1:1 noise ratio was omitted to focus on the higher, more interesting ratios. Furthermore, the performance was compared between images processed to a resolution of 96×96 and 192×192 pixels, both with a noise ratio of 1:8.



(a) Input image. Wire hits are labeled with white pixels.



(b) Target image. Track points are labeled with white pixels. Noise to be removed is labeled with black pixels.

Figure 9: Example of *4pipi* event with 1:4 noise ratio as fed to the CNN.

The performance results of the noise study are summarized in table 1. Further details can be found in Appendix A.

	6 tracks	8 tracks	10 tracks	12 tracks	16 tracks	20 tracks
Ratio 1:2 (96×96)	94.9	94.3	94.0	94.0	94.0	93.9
Ratio 1:4 (96×96)	93.9	92.5	91.8	91.6	91.0	90.6
Ratio 1:8 (96×96)	93.2	91.5	89.9	89.6	88.3	87.2
Ratio 1:8 (192×192)	94.4	93.1	91.8	91.5	90.3	89.5

Table 1: Maximum accuracies in the testing set (%) for each number of tracks and track-to-noise ratios.

Clearly, both the number of tracks and the amount of noise have an inverse relationship with the performance of the neural network. The highest accuracy in the testing set, 94.9%, was achieved for 6 tracks with noise ratio 1:2 and the lowest, 87.2%, for 20 tracks with noise ratio 1:8 (for image resolution of 96×96 pixels). Intuitively, for a large number of tracks and high levels of noise, the network has to predict more data points, which increases the room for error and reduces the overall accuracy. Moreover, if the image is too clustered with data points, it becomes more and more difficult to distinguish track from noise.

Furthermore, it is evident that the maximum accuracy in the testing set improves when the image size increases from 96×96 to 192×192 pixels, for all events for noise ratio 1:8. The impact of increasing the image resolution on computational performance is explained in section 5.3.1.

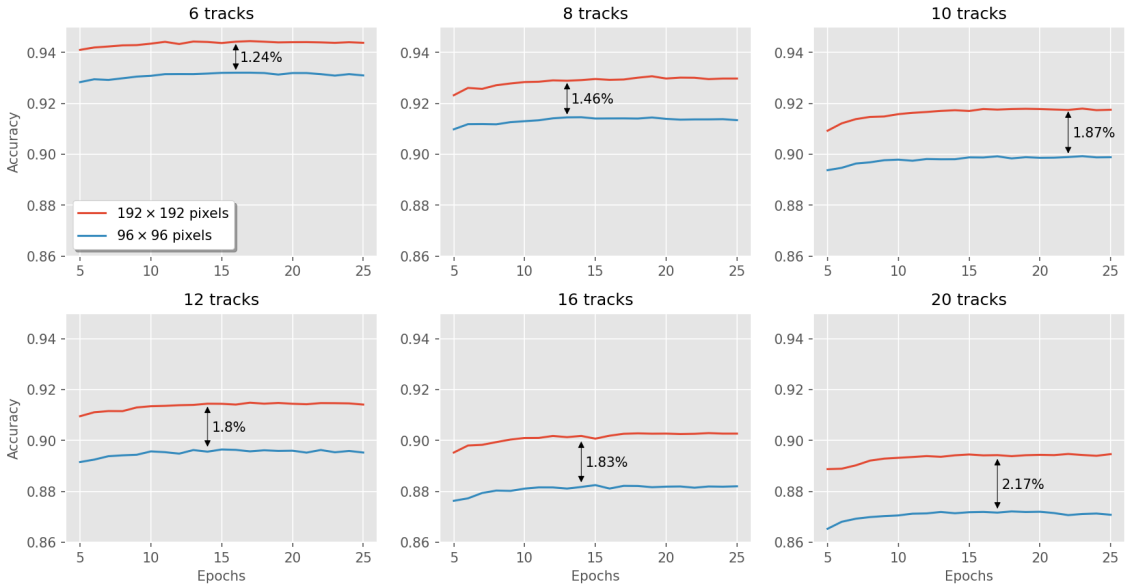


Figure 10: Comparing accuracies in the testing set between image resolutions for a noise ratio of 1:8.

Figure 10 illustrates the effect of the image resolution on the accuracy of the testing dataset. The black arrow indicates the difference between the maximum accuracy for images with 96×96

and 192×192 pixels at the same epoch. This difference increases with an increasing number of tracks, with a maximum of 2.17% for 20 tracks and a minimum of 1.24% for 6 tracks. A reason for this behavior is that a larger image size renders more defined tracks, which makes them easier to differentiate from noise. This becomes increasingly important for events with many particle tracks.

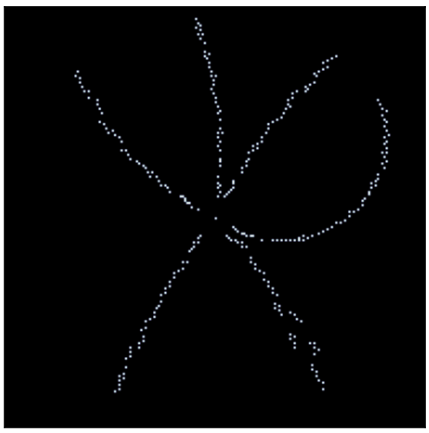
6.2 Track identification

The track identification section of this thesis is broken down into three parts: individual track labeling, charge identification and pair identification. The key interests lie in evaluating the ability of the network to label tracks in events with more than four tracks. Also, in testing the feasibility of applying the network to label charged particles based on their charge and recognize particles produced in pairs.

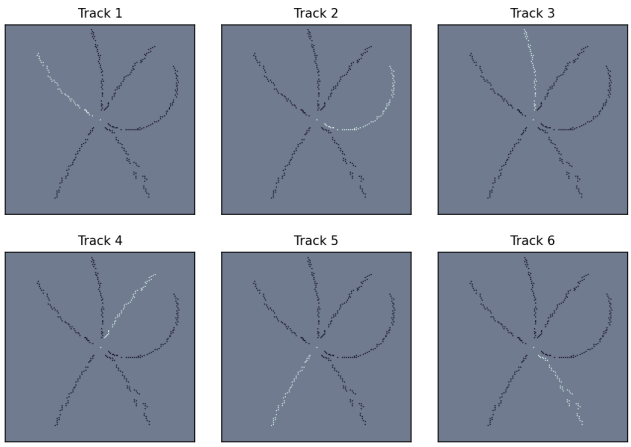
Prior to training, the datasets are cleared of all noise. For the purposes of this section, it can be assumed that the recorded events either did not contain any measurable noise or the noise was removed, for example via a neural network, during input processing. Removing the effect of noise reduces the number of factors needed to take into account to make appropriate conclusions.

6.2.1 Individual track labeling

The previous works, described in the Master's and the Bachelor's theses, focused on individual track labeling for a specific reaction, namely *pipiee*. In this work, we extend these studies by making a comparison with other reactions. With this, we aim to conclude on the applicability of the architecture to analyze a broader spectrum of channels.



(a) Input image. Wire hits are labeled with white pixels.

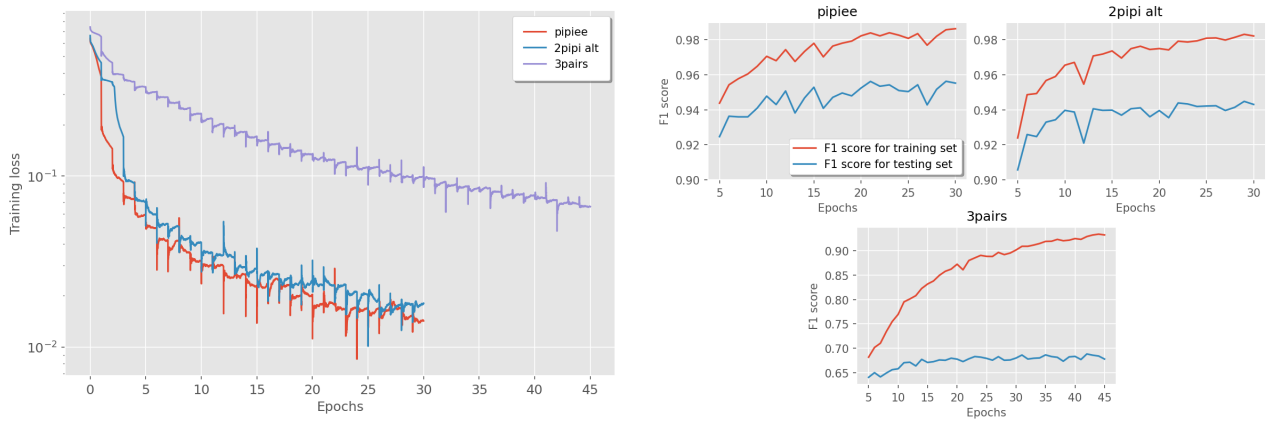


(b) Target images, corresponding to individual tracks. The track of interest is labeled with white pixels. Other tracks are labeled with black pixels.

Figure 11: Example of *3pairs* event as fed to the CNN.

The track labeling algorithm was tested on all the relevant decay channels listed in section 2.2. The progression of the training loss and F1 score by epoch is shown in figure 12.

The best F1 scores in the testing dataset for *pipiee*, *2pipi alt* and *3pairs* are 95.6%, 94.5% and 68.8%, and occur after epoch 28, 28 and 41, respectively. Topologically, *pipiee* and *2pipi alt* are very similar, hence it is not surprising to see comparable F1 scores. On the other hand, the *3pairs* reaction is distinctly different, having 6 tracks and thus a more complex output consisting of 6 feature maps, instead of 4. The corresponding F1 score in the testing set (68.8%) is significantly lower than for *pipiee* and *2pipi alt*.



(a) Training loss vs. epoch

(b) F1 score vs. epoch

Figure 12: Individual track labeling training loss and F1 score by epoch for *pipiee*, *2pipi alt* and *3pairs* reactions.

Evidently from the training loss and the F1 scores progression, the *pipiee* and *2pipi alt* events trained more quickly and effectively than *3pairs*. The F1 score in the testings set for *3pairs* plateaus at around 69.0%. Although the best F1 score in the testing set occurred at epoch 41, the second and third highest scores occurred at epochs 34 and 30 respectively. This is a good example of overfitting. The F1 score in the training set continued to grow, seemingly unbounded, long after the F1 score for the testing set has plateaued. It means that the model has started to extract features specific only to the training set, that bear no relevance and are not applicable to the testing set.

The results are only given for *pipiee*, *2pipi alt* and *3pairs*. This is due to the fact that the network was unable to identify the relevant features necessary to make reasonable predictions for all the other reactions. The network learns that the best, most accurate prediction it can make is to simply return the input image for each output image. Any results are therefore unrepresentative of the actual performance of the network. The reasons for this observation are discussed later in the section.

Empirically, the incompetence of the network means that the training loss plateaus at some value y and that the model has encountered a local minimum in the loss function during gradient descent. Breaking free from this local minimum to reach the global loss minimum would either require adjusting the learning rate [19] or enriching the data with higher-level features in order to make the tracks distinguishable from one another. Unfortunately, both of these potential solutions are out of the scope of this thesis. The training loss plateau is further en-

countered in sections 6.2.2 and 6.2.3.

As stated in the overview of section 3, a neural network is a black box. It is impossible to determine the weight that the network assigns to a particular feature. However, by simply looking at the images and the performance of the network, a prediction about the most relevant features can be made. Most notably, these are the radius of track curvature and the particle charge. The radius, R (meters), is proportional to the transverse momentum, p_T (GeV/c), via

$$p_T = B \times q \times R, \tag{8}$$

where $B = 1\ T$ is the strength of the magnetic field and q is the charge in units of e. Thus, the particle transverse momentum dictates the bending radius of the track and the particle charge dictates the direction of the curvature of the track. Positively charged particles are bent to the right, negatively - to the left.

The transverse momentum distributions for *pipiee*, *2pipi alt*, *3pairs* and *2pipi* are illustrated in figure 13. The distributions are represented by boxplots. Boxplots display variations in samples of a statistical population. It is a standardized way of displaying the minimum, maximum, sample median and the first and third quartiles of a distribution. The whiskers at the ends depict the minimum and the maximum of the distribution, excluding the outliers. The median of the distribution is given by the orange line. The first and third quartiles are the medians of the lower half of the dataset and the upper half of the dataset, respectively. They are represented by the lower and upper edges of the rectangles. Boxplots are also utilized to illustrate distributions later the thesis.

Note that the two pions in the *pipiee* reaction have an identical momentum distribution, but differ in charge. Therefore, they are distinguishable from each other via the direction of their curvature. The electron/positron particles are similarly distinguishable. The same logic applies to the *2pipi alt* reaction. On the other hand, for the *3pairs* reaction, there is a considerable distribution overlap between the particles of the second pion pair and those of the kaon pair. The lower F1 score for the *3pairs* reaction can be attributed to this distribution overlap. Finally, based on charge and p_T alone, it is impossible to distinguish one π^+ particle from another in the *2pipi* reaction. To the network, they have an identical topology. This explains our earlier observation that the network fails to label the individual tracks for the *2pipi* reaction.

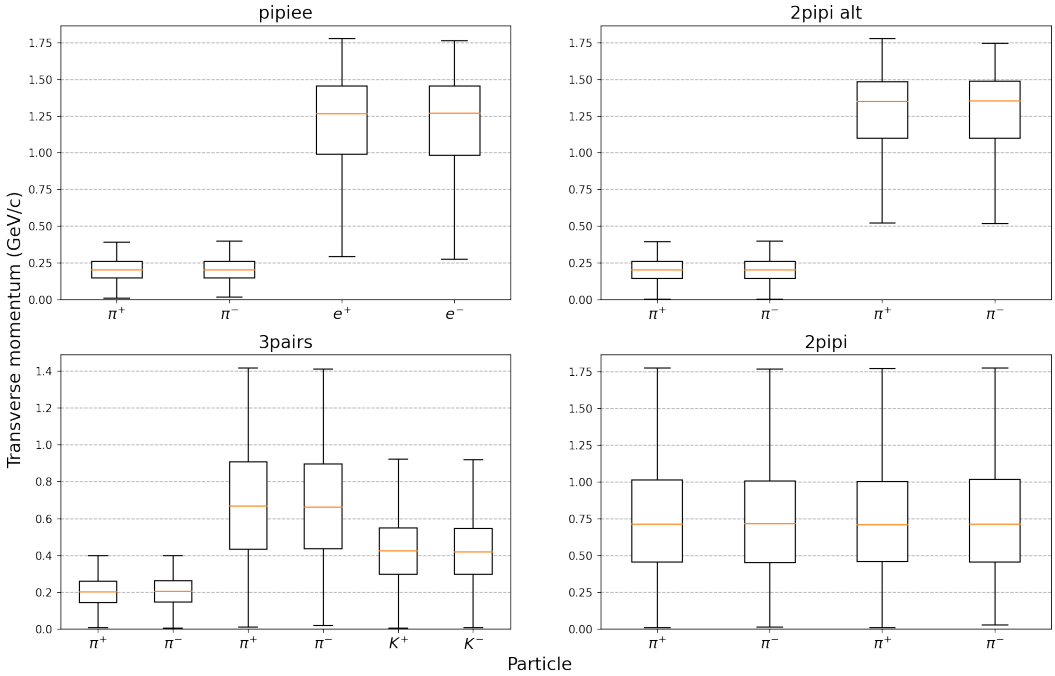
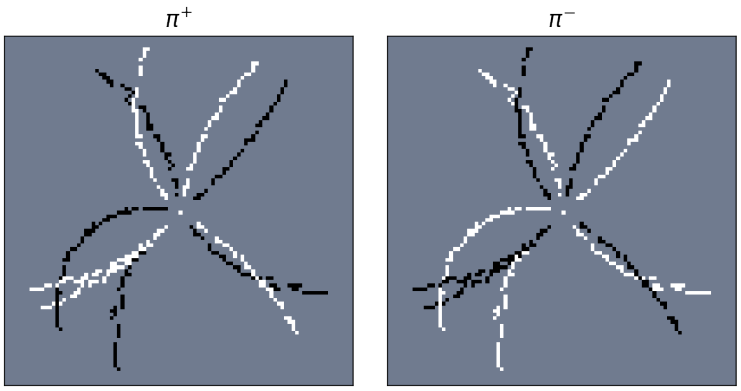


Figure 13: Boxplots showing the absolute transverse momenta distributions of particles in the training set for various reactions.

6.2.2 Charge identification

The charge identification algorithm implies decomposing the input image into two target images corresponding to positively and negatively charged particle tracks (see figure 14). As in the individual track labeling section, the algorithm was trained and tested on all the relevant reactions listed in section 2.2.

(a) Input image. Wire hits are labeled with white pixels.

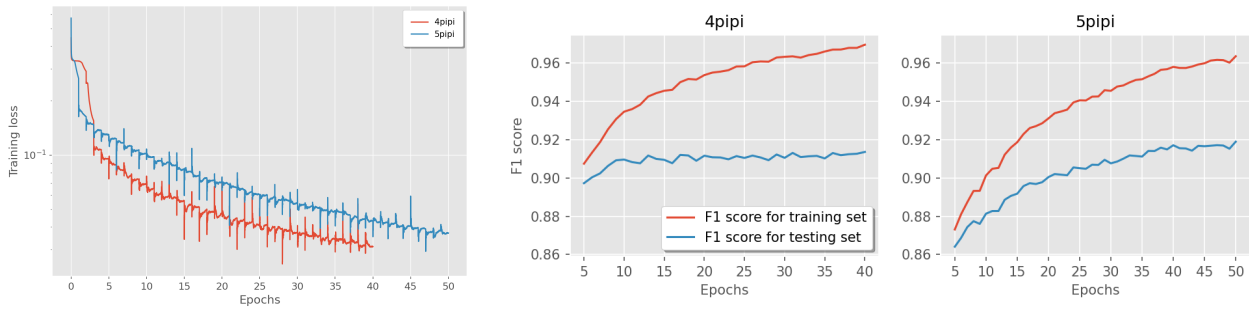


(b) Target images, corresponding to positive and negative charge. The tracks of interest are labeled with white pixels. Other tracks are labeled with black pixels.

Figure 14: Example of *5pipi* event as fed to the CNN for charge identification.

The training loss and F1 score after each epoch is shown in figure 15. Evidently from plot of the training loss at each epoch, the training loss for the *4pipi* reaction is significantly lower than for the *5pipi* reaction. The *5pipi* reaction also took longer to train, reaching a plateau in the F1

score of the testing set after 50 epochs. The obvious reason for this lies in the complexity of the topology, which is more complex for *5pipi* (10 tracks) than for *4pipi* (8 tracks). Nevertheless, the maximum F1 scores of the testing set are comparable - 91.4% and 91.9% for *4pipi* and *5pipi*, respectively. It indicates that, for charge identification, the topological complexity has an apparent effect on the training speed and little to no effect on the performance.



(a) Training loss vs. epoch

(b) F1 score vs. epoch

Figure 15: Charge identification training loss and F1 score by epoch for *4pipi* and *5pipi* reactions.

Similarly to the previous section, the results are only given for *4pipi* and *5pipi*. For the rest of the reactions the network was unable to extract relevant features and a local minimum in the loss function was reached. It may seem surprising that the charge identification algorithm worked for *4pipi*, but not, for instance, for *3pipi* or *2pipi*. After all, the *npipi* events are similar topologically, only with a varying number of tracks. To understand the reasons behind this peculiarity, it helps to observe the transverse momenta distribution of π^+ particles in figure 16.

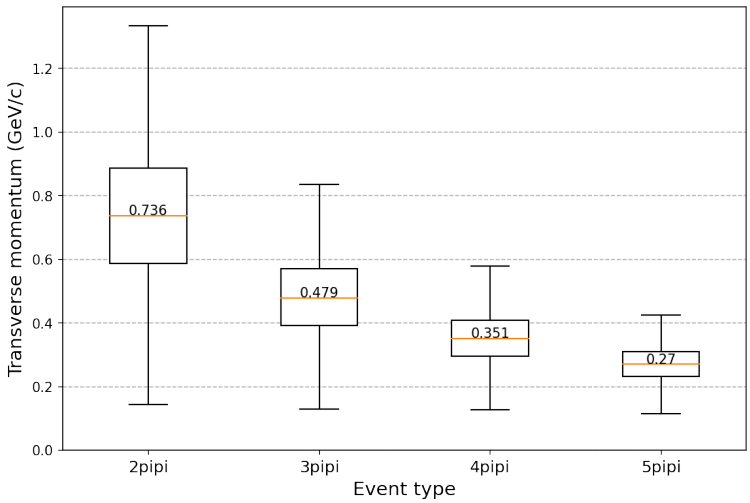


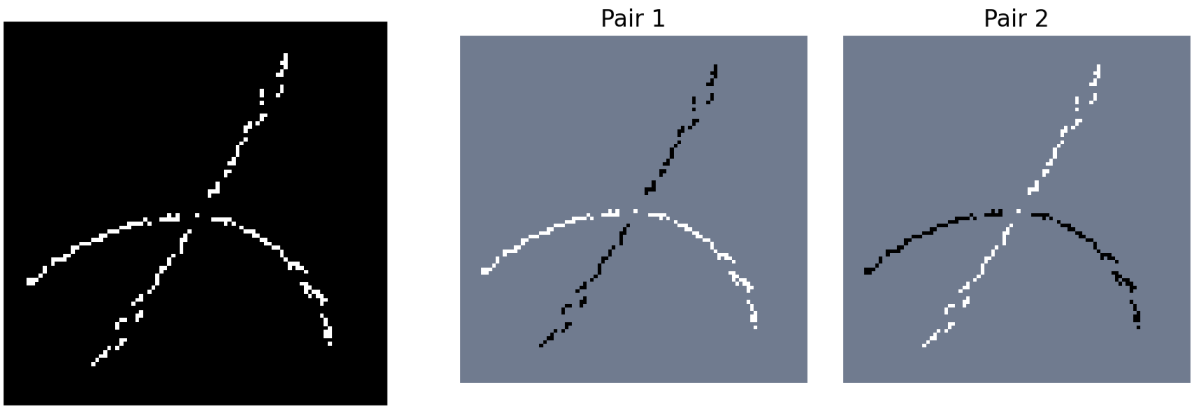
Figure 16: Boxplots showing the absolute transverse momenta distributions of π^+ particles in the training set for various *npipi* events.

Clearly, the transverse momenta of π^+ particles produced in the *npipi* reactions decrease with increasing n. This is an obvious consequence of the constant center-of-mass energy of 3.68 GeV being distributed over n number of tracks. From eq. 8, $p_T \propto R$ and thus particles with lower transverse momentum have a more defined curvature. However, the reasoning that the network

is able to assess the direction of the curvature for a track, only if it is below a certain p_T threshold is flawed. We note that the network was capable of identifying the charge of the e^+ and e^- particles for the individual track labeling algorithm. The median transverse momentum for these particles was around 1.26 GeV/c, which is higher than the median transverse momentum of 0.736 GeV/c for the pions in *2pipi*. The bending radius of the tracks may play a role, but it cannot be the only reason for the observed results. The reason is likely to be a combination of factors, such as the simplicity of topologies, steep local minima in the loss function that are difficult to escape, or other obscure factors hidden inside the black box that is the neural network.

6.2.3 Pair identification

The last of the track identification algorithms tested is pair identification. Here, the input image is decomposed into images equal to the number of pairs produced in the reaction (see figure 17). The network was trained on all the relevant interactions except for *npipi*, because the pions in *npipi* do not come in distinct pairs.



(a) Input image. Wire hits are labeled with white pixels.

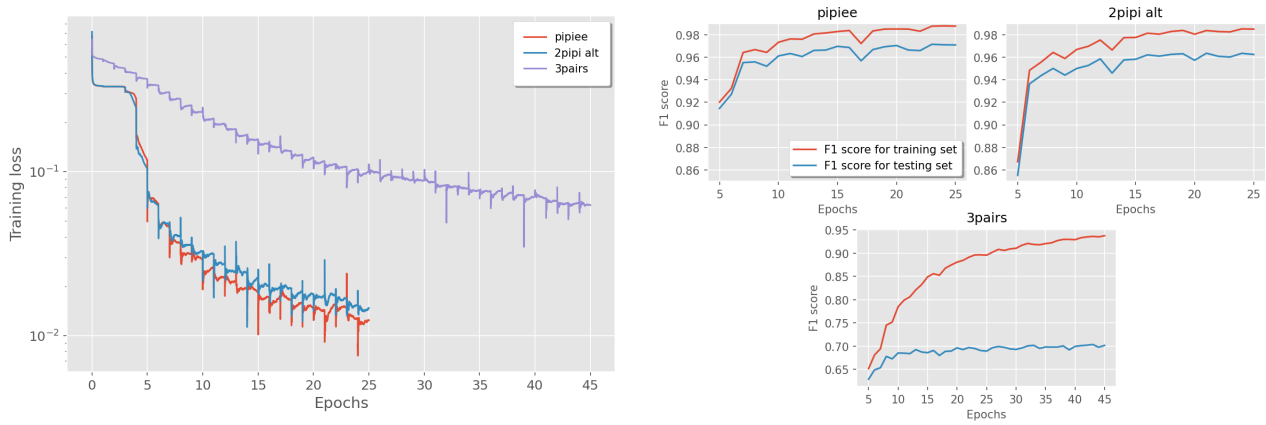
(b) Target images, corresponding to pairs. The tracks of interest are labeled with white pixels. Other tracks are labeled with black pixels.

Figure 17: Example of *2pipi alt* event as fed to the CNN for pair identification.

The training loss and F1 score by epoch is shown in figure 18. The *3pairs* reaction was trained for 45 epochs, as opposed to 25 for *pipiee* and *2pipi alt*, due to slower training. However, the F1 score in the testing set stabilized around epoch 30 after which the model was being overfitted, similarly to what was observed in section 6.2.1. The best F1 scores in the testing set for *pipiee*, *2pipi alt* and *3pairs* are 97.1%, 96.3% and 70.4% attained after epochs 23, 21 and 43, respectively.

It is interesting to note that the training loss curves for *pipiee* and *2pipi alt* exhibit a stagnancy up to epoch 3, followed by a rapid decline between epoch 3 and epoch 5. This behavior is also apparent in the sharp rise of accuracy between epoch 5 and 6. A possible explanation for this phenomenon could be that a local minimum was initially encountered in the loss function. Soon after, the model managed to extract some higher-level features from the images and break out from the local minimum.

The model was unable to identify the pairs in the *4pairs* reaction, stabilizing at a local minimum of the loss function. The result can be attributed to two different factors: the complexity of the *4pairs* topology and the opening angle distributions of paired particles (see discussion below). These factors also explain the low F1 score calculated for the *3pairs* pairs, in comparison to *pipiee* and *2pipi alt*. The effect of the topological complexity has been discussed previously and the consequences are rather simple. The model takes longer to train and the overall performance after training is relatively low. The opening angle distributions are arguably a more important factor for this type of track identification.



(a) Training loss vs. epoch

(b) F1 score vs. epoch

Figure 18: Pair identification training loss and F1 score by epoch for *pipiee*, *2pipi alt* and *3pairs* reactions.

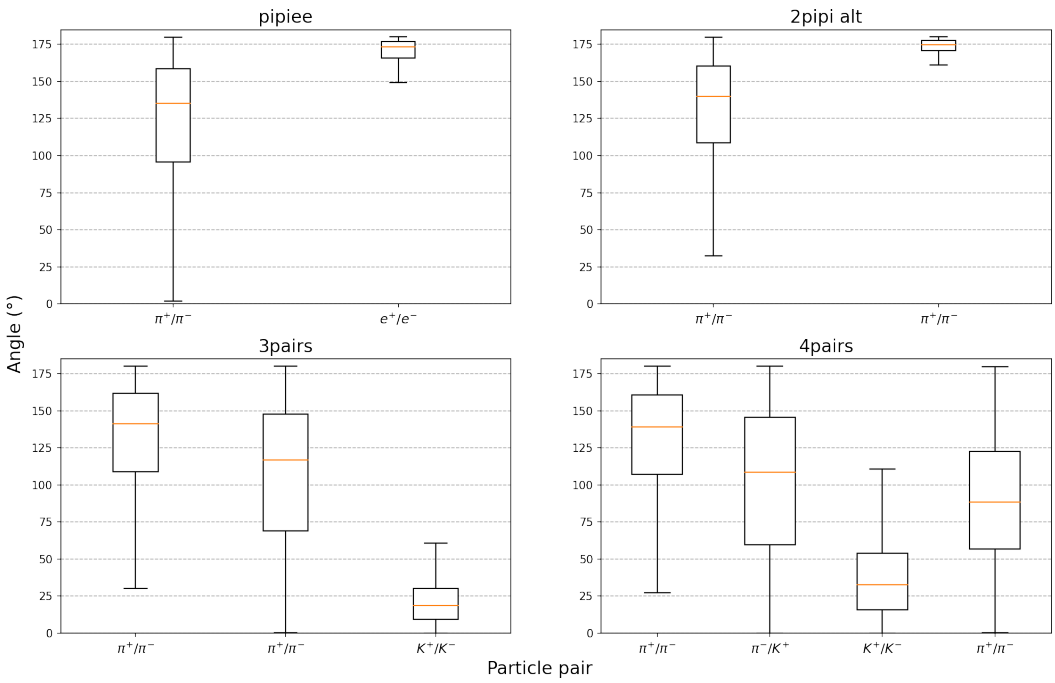


Figure 19: Boxplots showing the opening angle distributions of pairs in the training set for various reactions.

The opening angle is the angle between two particle tracks at the moment of their creation. Formally, it is the angle between two transverse momentum vectors $\mathbf{p_1}$ and $\mathbf{p_2}$, and is calculated via the definition of the dot product:

$$\cos\theta = \frac{\mathbf{p_1} \cdot \mathbf{p_2}}{|p_1||p_2|}$$

The hypothesis is that in order to classify two particle tracks as a pair, the model must identify higher-level features related to the pair as a whole, in addition to p_T distributions of individual tracks. The opening angle is one such high-level feature. The model is more likely to identify a pair if the particles' opening angles follow a unique pattern. For example, if the particles are generally expelled at angles $> 170°$, they may be interpreted as one long uninterrupted track. On the other hand, at a small angle $< 10°$, the V-pattern resembles a sharp edge at the origin. Boxplot diagrams of opening angle distributions for all considered reactions are depicted in figure 19.

The distributions of pairs produced in *pipiee* and *2pipi alt* are very similar. Both reactions have a distinguishable pair with a narrow distribution and a median value of 173.4° and 174.9° respectively. The other pair has a much wider distribution, and thus it is difficult to say whether a particularly distinct pattern is formed. In the *3pairs* reaction, the K^+K^- pair forms a prominent V-pattern, whereas the remaining two $\pi^+\pi^-$ pairs are less recognizable. Considering the relatively low F1 score, 70.4%, it can be deduced that the model may wrongly attribute pion tracks to the pair they do not belong to. Finally, for the *4pairs* reaction, an extra pair introduces another level of difficulty and the necessity to find meaningful patterns. The K^+K^- pair is noticeably different from the rest, characterized by a narrow distribution with a 32.7° median, but nowhere near the extent of the e^+e^- pair in the *pipiee* reaction, for example. Coupled with a complex topology, the outcome of the training is a local minimum in the loss function and the inability of the network to classify pairs.

6.3 Effect of broken layers

The model weights obtained from both noise reduction and track identification algorithms were evaluated on the testing set in which a number of layers were selected as broken. A broken layer means that any hit measured by the wires belonging to that layer is discarded from the event image. Two different schemes for discarding layers were implemented. The layers were either discarded randomly or starting from the middle outwards. The maximum number of discarded layers was 35, around 80% of the total number of layers. Going beyond that, the event no longer has any resemblance to the original image. In reality, more than 50% of broken layers is already absurd. At that point, it would be more important to determine and eliminate the cause of the broken layers, rather than to work around it.

An interesting metric to look at is the maximum number of layers that may be broken before the performance decreases below a certain percentage of base performance. The base performance is the performance of the network with 0 broken layers. The metric is defined such that the smaller the number is, the greater the impact of broken layers on performance is.

6.3.1 Noise reduction

The layers to be discarded were selected randomly before each model prediction. Judging by the data in table 2, the effect of broken layers is slightly amplified for higher noise ratios. On average, across all events, the maximum number of broken layers tolerated in order to maintain an accuracy at 98% of base accuracy, for noise ratio of 1:2, is 7.3. For the noise ratio of 1:8, that number is 5.7. Similarly, to maintain an accuracy at 95% of base accuracy, the numbers for noise ratios of 1:2 and 1:8 are 11.5 and 9.5 respectively. The abundance of noise increases the demand for continuity in the tracks, which is interrupted by the missing layers.

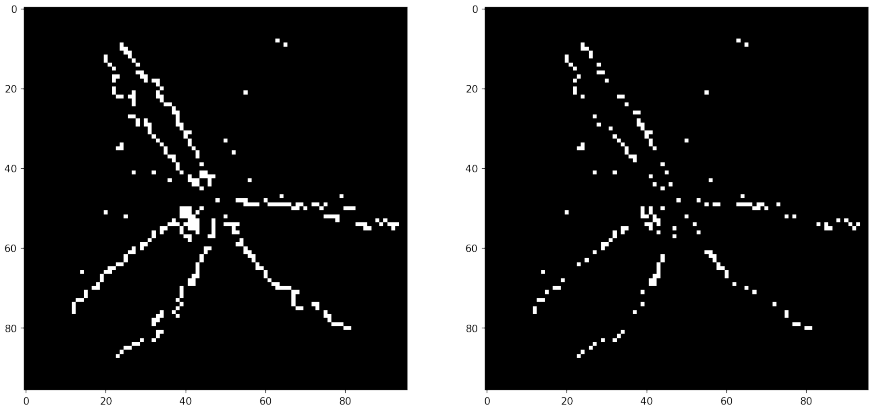


Figure 20: Example of *3pipi* event. Left: input image. Right: input image with 20 randomly broken layers. Wire hits are labeled with white pixels.

Furthermore, broken layers play a bigger role for higher resolution images. The comparison between the two resolutions can be seen in table 3. The results are given for the noise ratio of 1:8. To maintain 98% of base accuracy, the network tolerated, on average, a maximum of 5.7 broken layers for a resolution of 96×96 pixels and 4.7 for 192×192 pixels.

Noise ratio	1:2						1:4						1:8					
Tracks	6	8	10	12	16	20	6	8	10	12	16	20	6	8	10	12	16	20
99%	6	4	5	4	5	9	4	5	3	2	3	3	6	5	5	3	3	3
98%	8	8	7	6	5	10	8	6	7	7	4	5	7	6	6	5	5	5
95%	13	9	11	10	13	13	14	9	10	12	10	9	11	10	8	9	9	10
90%	15	15	17	14	13	15	14	13	13	16	14	15	15	13	13	14	13	13

Table 2: Maximum numbers of broken layers, for which the accuracy is greater than x% of base accuracy. Given for each event at noise ratios of 1:2, 1:4 and 1:8.

Resolution (pixels)	96×96						192×192					
Tracks	6	8	10	12	16	20	6	8	10	12	16	20
99%	6	5	5	3	3	3	4	5	4	3	2	3
98%	7	6	6	5	5	5	6	5	4	6	4	3
95%	11	10	8	9	9	10	11	8	8	9	9	8
90%	15	13	13	14	13	13	14	11	11	10	11	10

Table 3: Maximum numbers of broken layers, for which the accuracy is greater than x% of base accuracy. Given for each event for resolutions of 96×96 and 192×192 pixels.

The plots of accuracy at each number of randomly broken layers can be seen in Appendix B.

6.3.2 Track identification

For the track identification algorithms, in addition to selecting broken layers at random, layers were systematically discarded from the middle outwards. It is interesting to compare the performance of the network for both of these cases, particularly the response to a lack of continuity created by removing the middle layers. The plots of F1 score at each number of broken layers can be seen in Appendix C.

Type	Individual		Charge		Pair	
Event	*pipiee*	*2pipi alt*	*4pipi*	*5pipi*	*pipiee*	*2pipi alt*
99%	8	8	7	5	8	6
98%	11	10	11	8	13	11
95%	13	15	19	14	16	17
90%	20	20	19	21	23	26

Table 4: Maximum numbers of randomly broken layers, for which the F1 score is greater than x% of base score. Given for each track identification algorithm.

Type	Individual		Charge		Pair	
Event	*pipiee*	*2pipi alt*	*4pipi*	*5pipi*	*pipiee*	*2pipi alt*
99%	5	5	4	3	7	6
98%	7	7	6	4	10	9
95%	10	10	8	6	15	14
90%	14	13	13	9	19	18

Table 5: Maximum numbers of layers systematically broken from the middle, for which the F1 score is greater than x% of base score. Given for each track identification algorithm.

Clearly from tables 4 and 5, the negative impact of broken layers is greater for layers missing from the middle rather than randomly. The most extreme example is the *4pipi* event after charge identification. For example, the maximum number of layers missing to maintain 95% of base accuracy is 19 for randomly broken layers and 8 for layers missing from the middle. As concluded previously in section 6.2.2, the network relies on the curvature of the track to classify charge. Removing middle layers breaks continuity of the tracks, masking their apparent curvatures. On the other hand, for randomly broken layers, the general shape of the tracks is maintained.

The pair identification algorithm has the best performance for broken middle layers. This is likely due to the hypothesis proposed in section 6.2.3, that the most relevant feature of pairs is their opening angle. The opening angle is presented in the first couple of layers of the image, so the removal of layers from the middle is not very effective at disturbing the pattern.

7 Conclusion

A Convolutional Neural Network previously developed by a Master's student [5] was extended to five different particle interactions and four algorithms. The purpose of the investigation was to explore the limits of the network architecture and to discover new areas of application. The network was trained and tested on Monte Carlo simulated data of collision events in the BESIII experiment. The performance of the network was evaluated for the noise reduction and track identification algorithms. The track identification algorithm was broken down into three parts: individual track labeling, charge identification and pair identification. Furthermore, the effect of broken layers on the performance of the network was examined. All the algorithms were trained on 10,000 events and tested on 2,000 events.

The noise reduction algorithm was trained for 25 epochs. It was applied to events with 6, 8, 10, 12, 16 and 20 tracks for 1:2, 1:4 and 1:8 noise ratios. The 12-, 16- and 20-track events were generated by stacking a 4-track events several times on one image, simulating high interaction rate conditions of the BESIII experiment. The network performed quite well. As expected, the best accuracy score, 94.9%, was attained for events with 6 tracks and a 1:2 signal-to-noise ratio. The worst accuracy score, 87.2%, was attained for events with 20 tracks and a 1:8 signal-to-noise ratio. By increasing the image resolution from 96×96 pixels to 192×192 pixels for events with a 1:8 signal-to-noise ratio, the accuracy improved for all events. The increase in accuracy was marginally greater for events with a larger number of tracks. Therefore, depending on the requirements and events in question, image resolution may be sacrificed in favor of computational performance.

The individual track labeling algorithm was trained for 30 epochs, with the exception of some reactions for which the learning process was slower and required more epochs. The most prominent features of the particles were found to be transverse momentum, p_T, and charge. The transverse momentum together with the charge dictate the bending radius and the direction of the track's curvature. In order for the network to perform well, each particle track in the event must have a unique topology, i.e a unique combination of p_T and charge. The network performed significantly worse for an event with 6 tracks, attaining an F1 score of 68.8%, as a result of a considerable overlap in p_T of some particle tracks. Due to the limited number of unique combinations of p_T and charge, the performance of the network further deteriorates with an increasing number of tracks.

Based on the data provided to the network, the only way to classify tracks by their charge is to detect the direction of their curvature. The radius of the curvature is directly proportional to the transverse momentum, meaning that particles with a lower momentum have a more apparent curvature. Relatively good F1 scores, 91.2% and 90.9%, were attained during charge classification of pions with median transverse momenta ≤ 0.351 GeV/c. The network was unable to classify pions with relatively high momenta by charge. This result is attributed to a number of factors, not limited to the transverse momentum of the particles, such as steep local minima in the loss functions and other factors concealed within the hidden layers of the network.

To categorize particle tracks as pairs, the opening angle distribution was determined to be a higher-level feature the network was most sensitive to. A narrow distribution implies a very specific pattern in the pair and allows the network to treat it as a single entity. The network

was very successful in identifying pairs in two 2-pair reactions, achieving 97.1% and 96.3% F1 scores. However, for the reaction producing 3 pairs, the network scored a less impressive 70.4%. There was a significant overlap in the opening angle distributions, resulting in incorrect classification. Due to the conservation of momentum, the opening angle distributions become wider and less definitive as the number of pairs increases, which results in a performance drop.

Investigating the impact of broken layers on the performance has shown that the network can operate within 99% of its base performance score for 3 to 8 broken layers. For some events, the number of broken layers may reach up to 15 and the network would still operate within 95% of its base accuracy or F1 score. This thesis demonstrates that the network is robust for a realistic number of broken layers. Also, depending on the user's tolerance for error, the findings of this thesis can be used as a framework to determine the acceptable number of broken layers for their needs.

As it stands, the network and the data have several limitations that negatively impact performance. The performance of the network depends on some specific conditions, like the unique topologies of particle tracks and narrow distributions of opening angles for pair identification. Hence, its effectiveness decreases for events with a large number of tracks, where attaining such conditions becomes increasingly difficult. Possible solutions to these limitations could be to utilize other pieces of data available from the tracking detector, such as the drift time and the energy loss, or to introduce data from other detectors, e.g from an electromagnetic calorimeter, as input to the network.

Bibliography

[1] A. Herten, "GPU-based Online Tracking for the PANDA Experiment," in *GPU Computing in High-Energy Physics*, 6 2015.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[3] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal, "The importance of skip connections in biomedical image segmentation," 2016.

[4] N. Ibtehaz and M. S. Rahman, "Multiresunet: Rethinking the u-net architecture for multimodal biomedical image segmentation," *Neural Networks*, vol. 121, p. 74–87, Jan 2020.

[5] H. de Vries, "Convolutional neural network for reducing noise and detecting tracks in the bes-iii main drift chamber," 2019.

[6] I. F. Graña, "Deep learning for particle tracking," 2020.

[7] D. Asner *et al.*, "Physics at bes-iii," 2008.

[8] G. Barucca *et al.*, "Panda phase one," *The European Physical Journal A*, vol. 57, p. 184, Jun 2021.

[9] C. Chen *et al.*, "The besiii drift chamber," in *2007 IEEE Nuclear Science Symposium Conference Record*, vol. 3, pp. 1844–1846, 2007.

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[12] S. Ruder, "An overview of gradient descent optimization algorithms," 2017.

[13] N. Ismoilov and S.-B. Jang, "A comparison of regularization techniques in deep neural networks," *Symmetry*, vol. 10, p. 648, 11 2018.

[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[15] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, "How does batch normalization help optimization?," 2019.

[16] S. Shi, Q. Wang, P. Xu, and X. Chu, "Benchmarking state-of-the-art deep learning software tools," 2017.

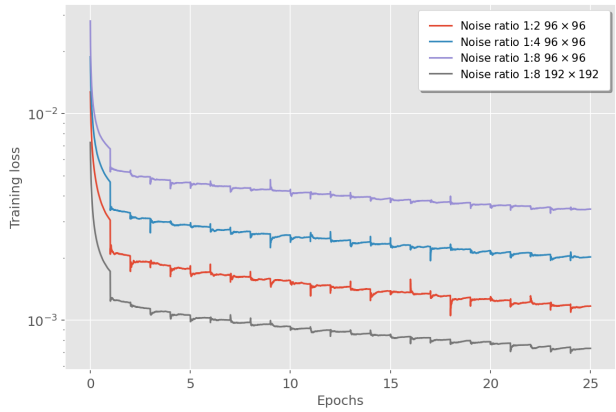
[17] S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah, "Comparative study of deep learning software frameworks," 2016.

[18] E. BUBER and B. DIRI, "Performance analysis and cpu vs gpu comparison for deep learning," in *2018 6th International Conference on Control Engineering Information Technology (CEIT)*, pp. 1–6, 2018.

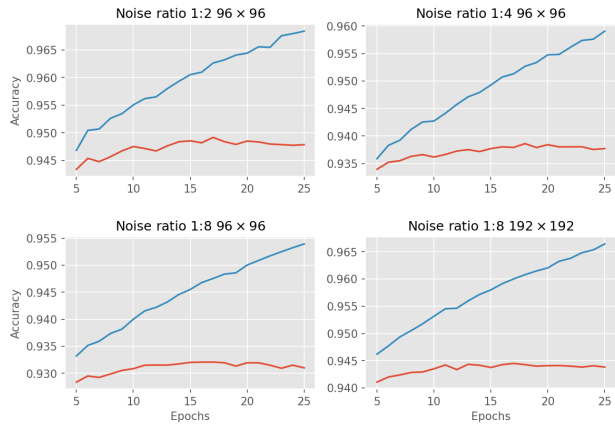
[19] C. Versloot, "Getting out of loss plateaus by adjusting learning rates," Feb 2020.

Appendices

A Noise reduction training loss and performance plots

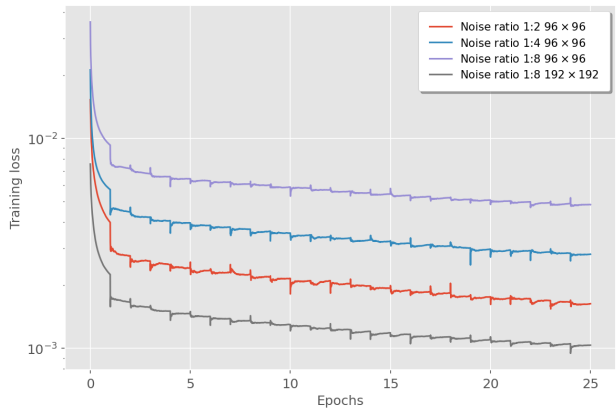


(a) Training loss vs. epoch

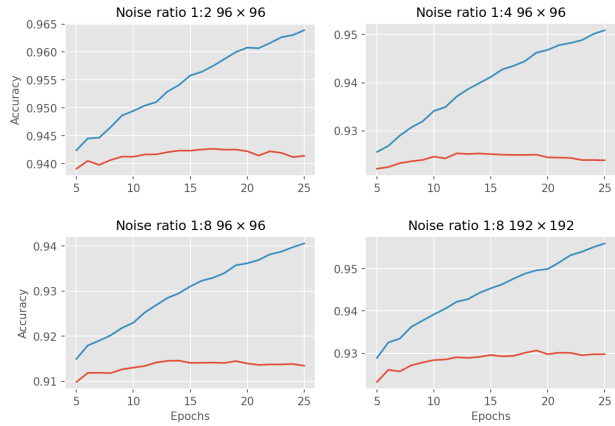


(b) Accuracy vs. epoch. Blue is training accuracy. Red is testing accuracy.

Figure 21: Noise reduction training loss and performance by epoch for *3pipi* event.

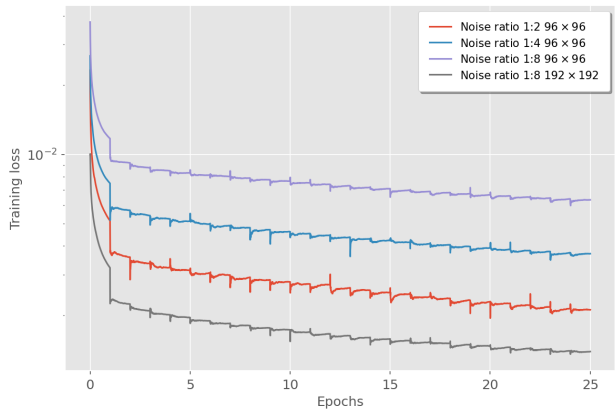


(a) Training loss vs. epoch

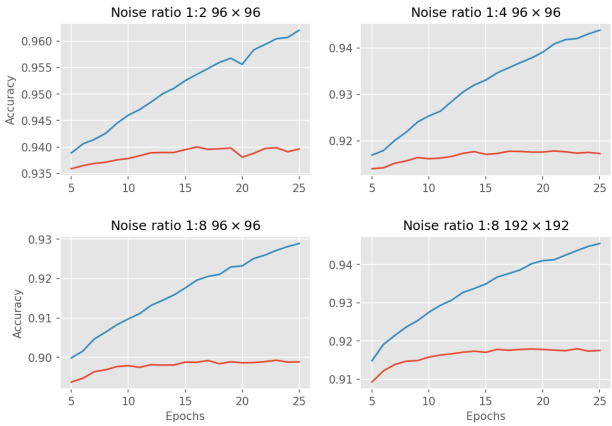


(b) Accuracy vs. epoch. Blue is training accuracy. Red is testing accuracy.

Figure 22: Noise reduction training loss and performance by epoch for *4pipi* event.

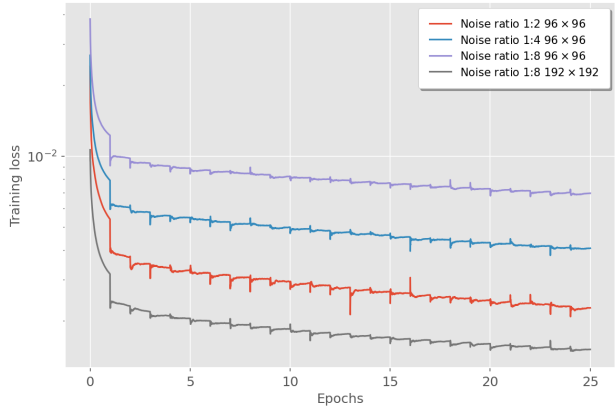


(a) Training loss vs. epoch

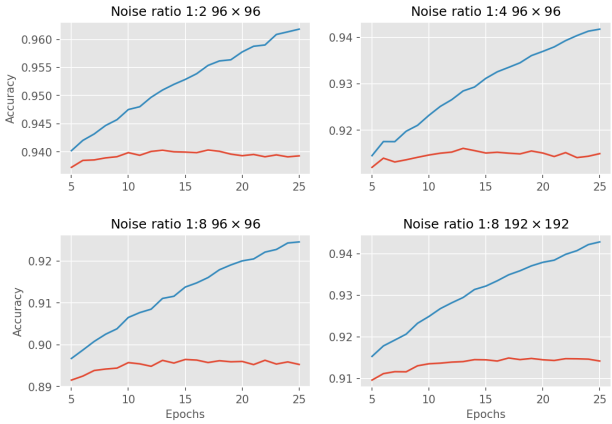


(b) Accuracy vs. epoch. Blue is training accuracy. Red is testing accuracy.

Figure 23: Noise reduction training loss and performance by epoch for *5pipi* event.

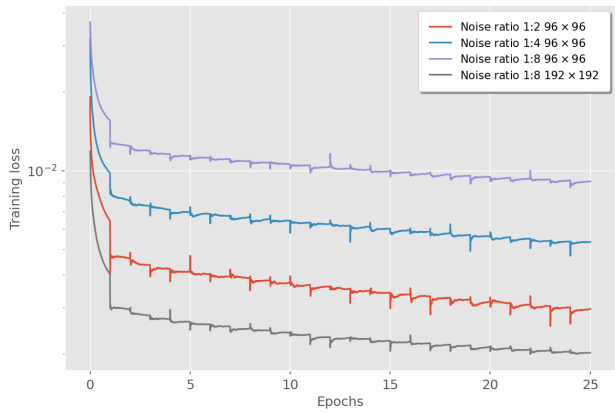


(a) Training loss vs. epoch

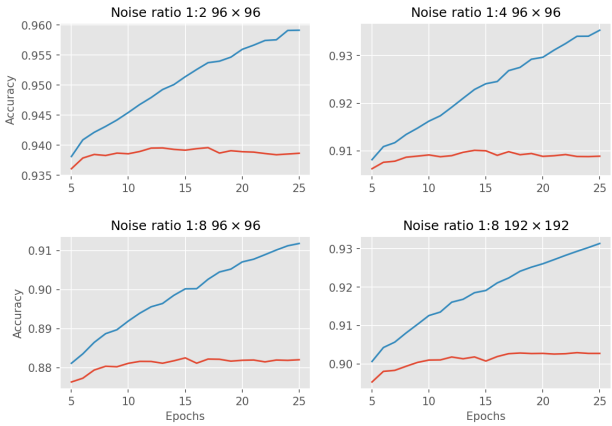


(b) Accuracy vs. epoch. Blue is training accuracy. Red is testing accuracy.

Figure 24: Noise reduction training loss and performance by epoch for *pipiee* event, overlapping 3 times.

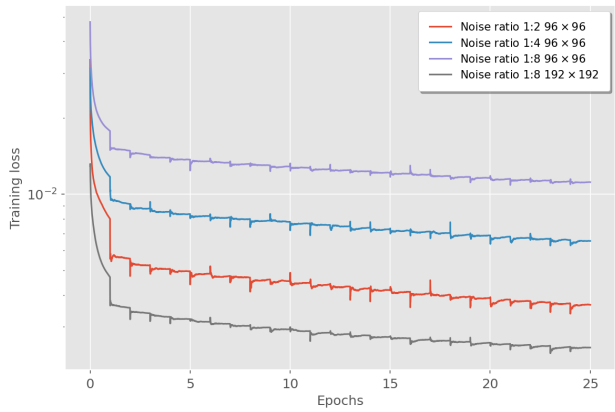


(a) Training loss vs. epoch

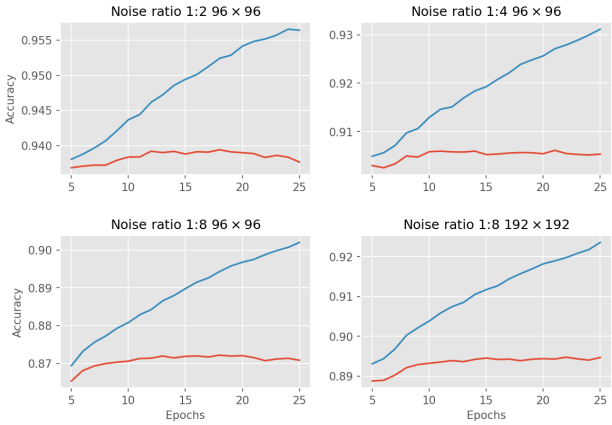


(b) Accuracy vs. epoch. Blue is training accuracy. Red is testing accuracy.

Figure 25: Noise reduction training loss and performance by epoch for *pipiee* event, overlapping 4 times.



(a) Training loss vs. epoch



(b) Accuracy vs. epoch. Blue is training accuracy. Red is testing accuracy.

Figure 26: Noise reduction training loss and performance by epoch for *pipiee* event, overlapping 5 times.

B Noise reduction broken layers accuracy plots

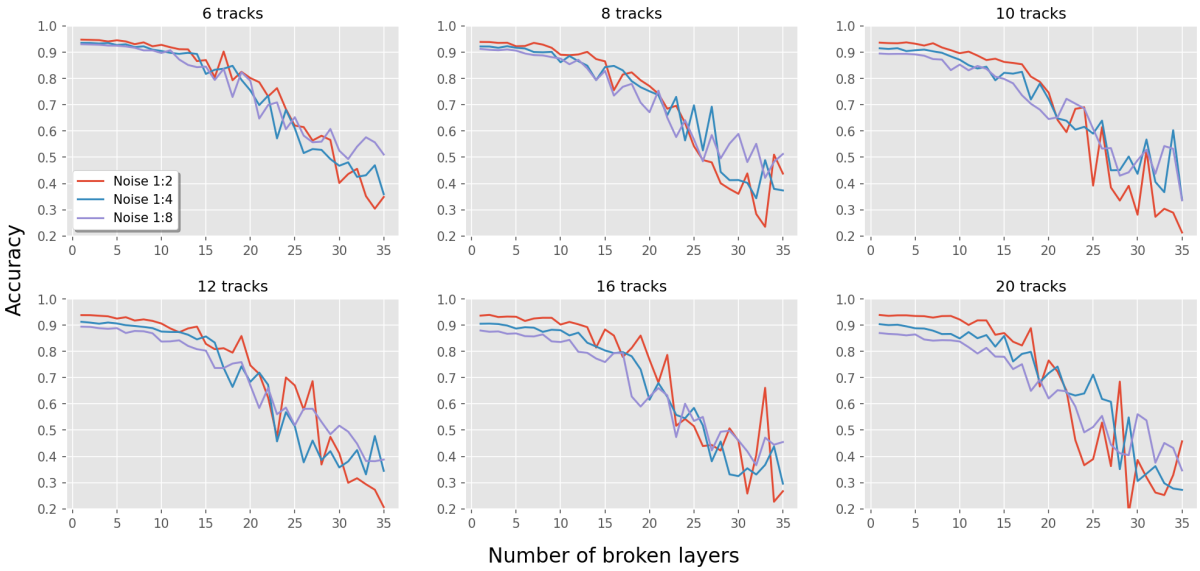


Figure 27: Noise reduction testing set accuracies at each number of randomly broken layers. Comparison between different noise ratios.

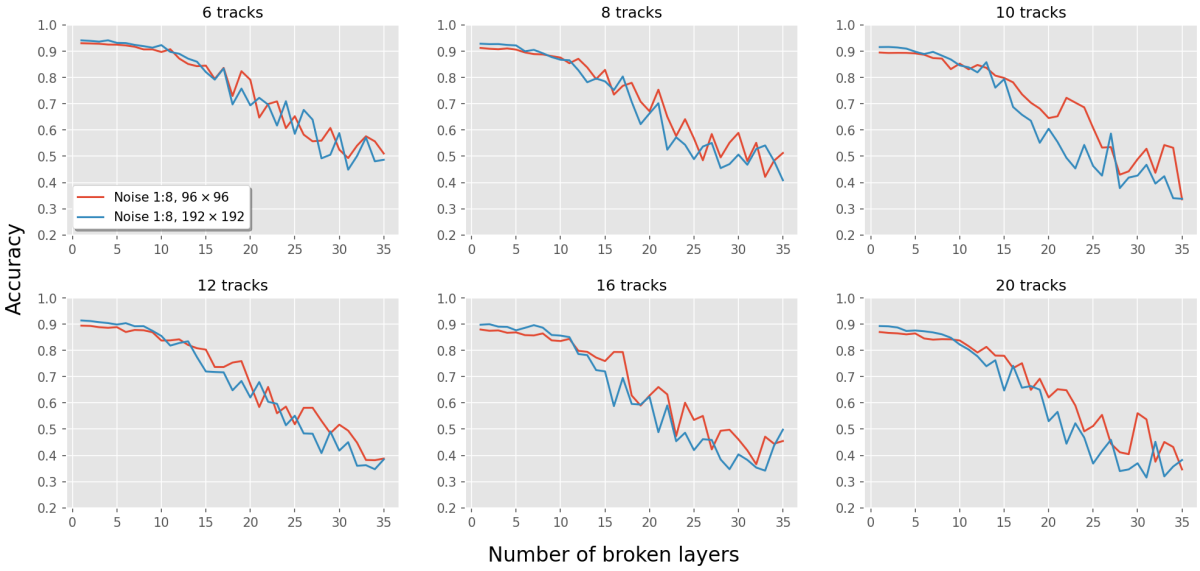


Figure 28: Noise reduction testing set accuracies at each number of randomly broken layers. Comparison between 96×96 and 192×192 resolutions.

C Track identification broken layers F1 score plots

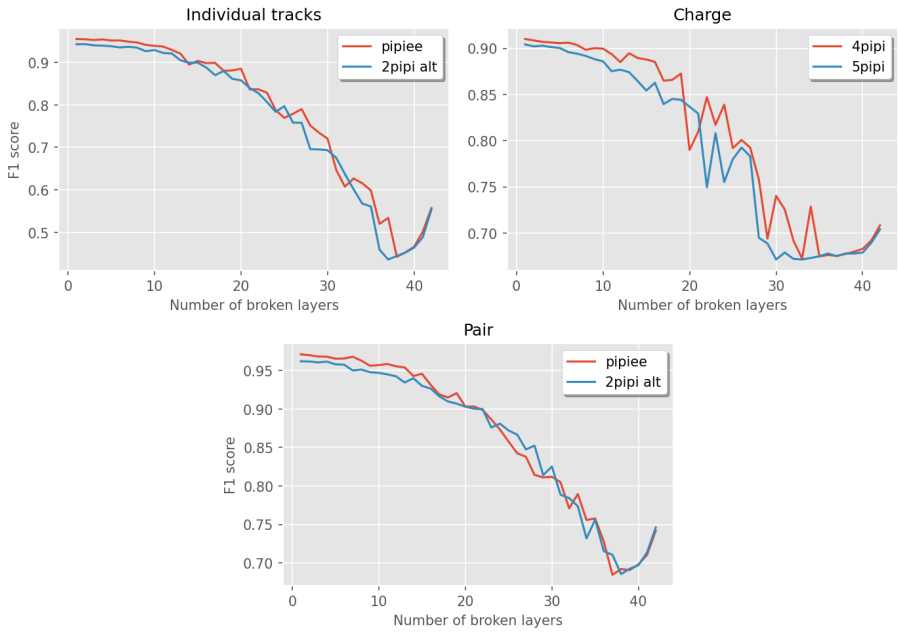


Figure 29: Track identification testing set F1 scores at each number of randomly broken layers.

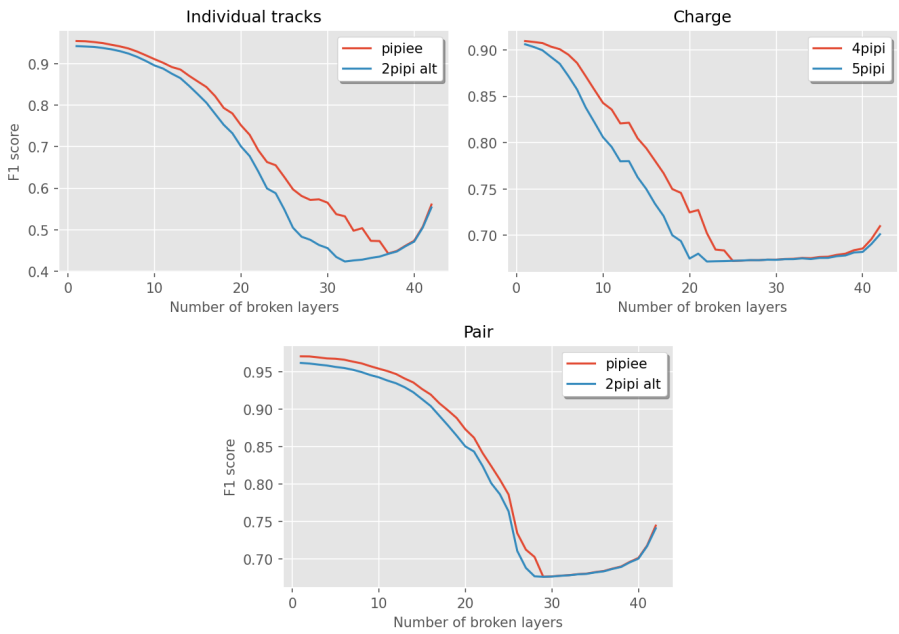


Figure 30: Track identification testing set F1 scores at each number of layers systematically broken from the middle.