Determining Optimal Configurations for Deep Fully Connected Neural Networks to Improve Image Reconstruction in Proton Radiotherapy

REU Site: Online Interdisciplinary Big Data Analytics in Science and Engineering

Alina M. Ali[1], David C. Lashbrooke Jr.[2], Rodrigo Yepez-Lopez[3], Sokhna A. York[4], Carlos A. Barajas[5], Matthias K. Gobbert[5], Jerimy C. Polf[6]

[1]Department of Mathematics and Statistics, University of Houston–Downtown
[2]Department of Mathematics and Department of Statistics, Purdue University
[3]Department of Computer Science, American University
[4]Center for Data, Mathematics, and Computational Sciences, Goucher College
[5]Department of Mathematics and Statistics, University of Maryland, Baltimore County
[6]Department of Radiation Oncology, University of Maryland School of Medicine

Technical Report HPCF–2021–12, hpcf.umbc.edu > Publications

Abstract

Proton therapy is a unique form of radiotherapy that utilizes protons to treat cancer by irradiating cancerous tumors while avoiding unnecessary radiation exposure to surrounding healthy tissues. Real-time imaging of prompt gamma rays can be used as a tool to make this form of therapy more effective. The use of Compton cameras is one proposed method for the real-time imaging of prompt gamma rays that are emitted by the proton beams as they travel through a patient's body. The non-zero time resolution of the Compton camera, during which all interactions are recorded as occurring simultaneously, causes the reconstructed images to be noisy and insufficiently detailed to evaluate the proton delivery for the patient. Deep Learning has been a promising method used to remove and correct the different problems existing within the Compton Camera's data. Previous papers have demonstrated the effectiveness of using deep fully connected networks to correct improperly detected gamma interactions within the data. More thorough hyperparameter studies than in previous works show that using a combination of larger batch sizes, higher neurons per layer, and higher layer counts tend to produce better performing networks, which show promise in reducing the complexity of previous network architectures. We also experiment with recurrent neural networks to test the viability of this type of architecture and report initial results.

1 Introduction

Because of its many advantages, proton therapy has been increasingly growing in popularity as a form of cancer treatment. Most types of radiation work with a similar objective to damage the cellular DNA of target cancer cells that reside in the nucleus of every cell. Although X-ray therapy is able to deliver dosage at the tumor site, the radiation continues to travel through the body until it exits the other side. This may potentially cause harm to healthy surrounding tissues and organs that are unnecessarily exposed to radiation. By contrast, proton therapy has the advantage to deliver radiation dosage directly at the tumor site without travelling further posterior into the body. This advantageous characteristic of proton therapy is called the Bragg Peak [12]. The Bragg Peak makes it possible to spare any surrounding healthy tissues from unnecessary radiation damage. In order to take full advantage of all of the perks that proton therapy has to offer, we must have an efficient technique to image the prompt gamma rays in real-time as they travel through the patient's body.

The Compton Camera is one promising technique to get real-time imaging of the proton beam, by detecting prompt gamma rays emitted along the path of the beam [9]. Unfortunately, some of the data produced by the Compton camera is not always usable due to the non-zero time resolution of the Compton camera, during which all interactions are recorded as occurring simultaneously. The use of the raw data by the Compton Camera with modern reconstruction algorithms yield noisy and insufficiently detailed images to evaluate the proton delivery for the patient.

Fully connected neural networks have shown promise in their ability to detect interaction for true triples in [13]. They note that the neural network has comparable accuracy to classical Compton camera sequencing techniques. They go on to highlight that there are superior methods to neural networks and the classical method but they are too computationally intensive for any real-world use case. A different usage of neural networks was seen in [10] where the authors simply trained a network to determine whether data was “good” or “bad” and then used the “good” data for reconstruction. They true triples and false events and noticed that the network had respectable accuracy and improved the good data to bad ratio that can be used for reconstruction. Both of these examples use simpler Monte-Carlo simulations, shallow neural networks, and do not consider how neural networks can play a role in a broader clinical application. When dosage rates change, the amount of data pollution increases greatly to the point that there is more unusable data than there is usable data [9].

This work’s specific purpose differs greatly from some of the authors previous work done on this topic. The earliest form of work was seen in [5] where they started training a neural network to order a single interaction with the hope of expanding the neural network to do multi-task classification for interaction ordering. Inevitably they settled on just a single-task classification method using ordering permutations and published a portion of those results in [3]. Then in [4] the neural network structure was changed from a shallower but wide design to a long but thin design and also trained using a python generator which feeds one input type (double/triple/double-to-triple/false) at a time in an attempt to improve accuracy without changing the neural network structure. It is important to note that up to this point all the doubles-to-triples, false triples, and false doubles were being generated during a preprocessing step rather than coming directly from a Monte-Carlo simulation. In [1] several changes had been made to the training process, data generation, preprocessing, and network design. Doubles and Triples had been separated into their own categories and a specialized neural network was trained for each. All data was generated using a Monte-Carlo simulation instead of stitched together during preprocessing. A residual block structure was implemented to allow for a deep fully connected neural network instead of shallow one. The results for the doubles only neural network was published in [8] and the results for the true triples plus doubles-to-triples was published in [2]. The final aim of [1] was to create a neural network which was capable of quickly and accurately ordering interactions for triples and doubles which also detecting false couplings. The ability to reorder and detect false couplings is known to improve reconstruction quality and can allow for Compton cameras to be used in a clinical setting. The clinical viability and impact of the neural network on reconstructions was published in [11]. At this point we have seen that deep neural networks have great success when ordering interactions and detecting false couplings. In this document we are attempting to create a more compact neural network than is seen in [1]. We define a “compact network” as a network with similar or superior performance but contains fewer total parameters and fewer neurons the previous networks with similar function. The fewer neurons the network has, the computationally cheaper it is to use, and the faster the network can classify records.

Section 6 contains the results of a basline hyperparameter study with 288 studies using 1024 epochs, which identifies and presents in more detail the results from 6 particularly promising studies. Additionally, results from 2 longer studies with 4096 epochs and 10% dropout rate are presented. These studies all use a more “compact network” that is cheaper for studies to run, with comparable results as the previously used fully connected network.

In Section 7, based on these observations, we conduct experiments with (i) different normalizers, (ii) different optimizers, and (iii) initial tests with recurrent neural networks, (RNN). These studies do not yield significantly better results than the original studies. In particular, more overall improvements are still needed for clinical use and we are currently experimenting with recurrent neural networks to test the viability of this type of architecture for this application.

The remainder of this report is organized as follows: Section 2 contains more in depth information about the distinct advantages that proton therapy provides versus other forms of radiotherapy. Section 3 describes how Compton Camera imaging can be used to improve the restraints of proton therapy and how the misordered interactions and the presence of false events can limit the usage of the Compton Camera. Section 4 gives a background on neural networks, as well as explaining the network's conception. Section 6 describes 288 different hyperparameters studies, some promising results, and how the network performed on different beam energies. Section 7 describes the several additional studies that we are carrying out, this includes normalization studies, optimizer studies and studies using a recurrent neural network (RNN). Section 8 discusses the impact, shortcomings, and overall successes associated with our goals and results as well as possible future work.

2 Proton Therapy

Radiation therapy is a form of cancer treatment that uses high doses of radiation that act to kill cancer cells and ultimately the tumor. X-ray therapy is one common technique used for cancer treatment. In this type of therapy, the majority of the radiation dosage is delivered upon entering the body. Because of this, the tumor does not receive as high of a concentrated dose as it should. In addition, X-rays will continue to travel posterior into the human body until it exits out the other side. This is not ideal as there is no need for extra radiation exposure within the body. Proton therapy on the other hand is more efficient in this manner. Rather than depositing the majority of the dosage at the entry site, proton therapy works to deposit the majority of the dosage at the tumor site itself, thus making the process more effective. Proton therapy also has an advantage over X-ray therapy in the sense that the proton beam travels no further posterior into the body than the site of the tumor allowing for minimal exposure to surrounding tissue.

In Figure 2.1, we observe two horizontal cross-sections of the chest that show a tumor located anterior to the heart. Figure 2.1 (a) an example of X-ray treatment which clearly shows unnecessary radiation being spread to healthy organs and tissue around the tumor. X-ray treatment in this scenario can be very dangerous as radiation exposure to the heart can cause life threatening prob-

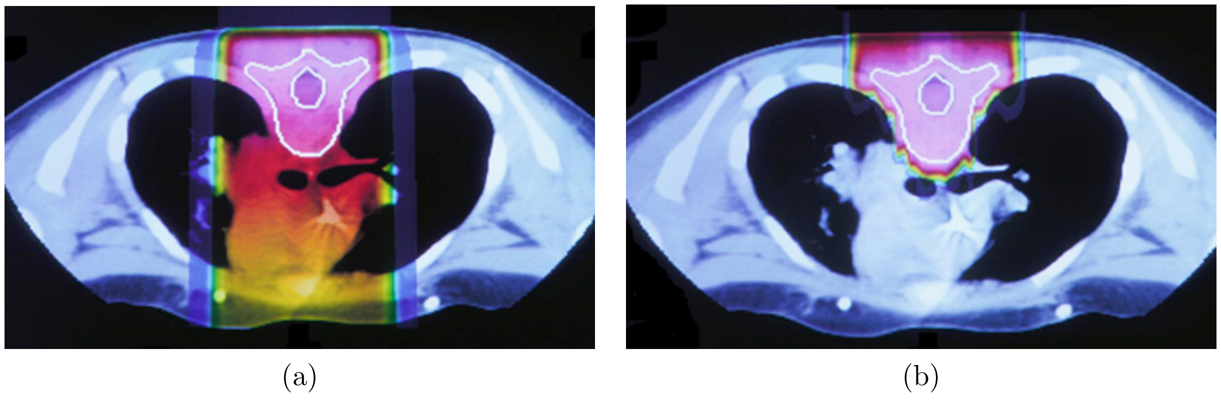
(a) (b)

Figure 2.1: X-ray treatment (a) as compared to proton beam treatment (b).

lems. These problems include inflammation of the pericardium, severe damage to the myocardium, congestive heart failure, and premature coronary artery disease. To avoid such exposure to the heart, one may take advantage of the benefits of proton radiotherapy. Proton radiotherapy shown in Figure 2.1 (b) acts to limit unnecessary radiation exposure to surrounding healthy organs and tissue. The major benefit of proton therapy can be clearly visualized in this image. The dosage of radiation goes no further posterior into the body than the site of the tumor which again limits the potential for unnecessary radiation exposure.

The proton therapy delivery process is quite unique. The protons start their journey inside of a vacuum tube that connects to a linear accelerator that leads to a cyclic particle accelerator called a synchrotron. This is where the protons stay until their energies reach a point at which they can reach any given depth inside the body of the patient. Once this energy level is reached, protons navigate through a beam-transport system that consists of many magnets that guide the protons to their target location. There are two tools involved in this guiding process. The nozzle-like aperture, acts to shape the beam of protons while the compensator acts to shape the protons into a 3-D shape which allows the protons to travel as deep as they need to into the body to reach the tumor. Once the proton beam has reached the depth of the tumor, the beam deposits the majority of its energy into the tumor.

Depending on the size of the tumor, the beam may have to kill the tumor cells layer by layer. When delivering a dosage to a tumor, the professional who is treating the patient will create what is called a safety margin. This safety margin enlarges the treatment area to ensure that all parts of the tumor are guaranteed to receive dosage. The safety margin is needed to account for slight movements in the patient during treatment as well as sligthly different positioning of the patient from one treatment to the next over several weeks.

In Figure 2.2, we observe a tumor outlined in green, the heart outlined in pink, and the safety margin outlined in orange. Because the tumor is resting right up against the heart, the safety margin and path must be carefully determined. In Figure 2.2 (a) a path is presented in such a way that exploits proton therapy's advantage of not sending radiation all the way through the body. Although the depth of the beam's penetration can be controlled, the safety margin created in this path includes necessarily a small portion of the heart. Therefore, we must assume that the heart will more than likely receive a small dosage of radiation. To stay clear of radiating the heart,

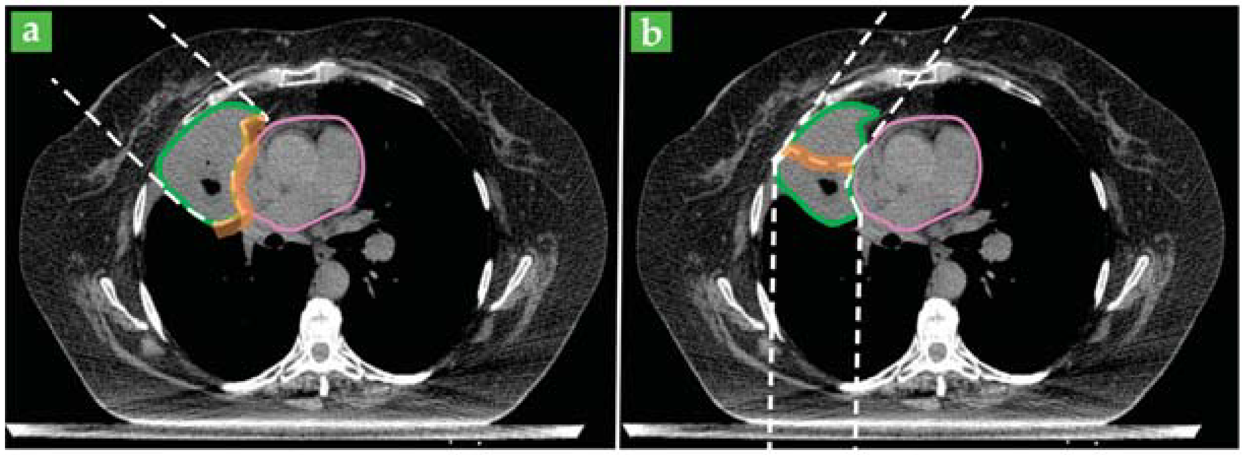

Figure 2.2: (a) Optimal proton beam trajectory. (b) Suboptimal trajectory necessary to protect heart.

Figure 2.2 (b) suggests a more suboptimal path in which two proton beams are used to allow for a path that avoids the heart. However, this path does expose surrounding healthy lung tissue to radiation. Figure 2.2 (b) is currently the preferred method for treatment as although the lungs do receive exposure to radiation, the heart does not.

If real-time information on the trajectory of the proton beam through the patient's body were available during a treatment, the safety margin could be smaller and the optimal path provided in Figure 2.2 (a) could be used. The use of Compton cameras is one proposed method for the real-time imaging of prompt gamma rays that are emitted by the proton beams as they travel through the body.

3 Compton Camera

3.1 Background on Compton Camera

Compton Cameras are multistage detectors that generate images of gamma rays through Compton scattering, [11]. As the protons penetrate the human body, they interact with atoms in the body, leading to prompt gamma rays emission. As those gamma rays exit the body, some of them collide with the modules in the Compton Camera. Modules of the camera then measure the energy deposited by the prompt gamma and its position as it passes through different detection stages of the camera. For each Compton scatter the camera records x-, y-, z-coordinates and the energy level of the scatter. The readout of interactions in a single period is called an event. The raw output data from the camera for each interaction is in the form (e_i, x_i, y_i, z_i) where $i = 1, 2, 3$, and e_i is the energy level.

Image reconstruction algorithms exist that can recover the path of the proton beam from the Compton camera data. The Compton camera's capability to reconstruct full 3D images of the proton beam range could be used with the patient's CT to compare the planned treatment dose and make adjustments. Radiotherapy treatment requires a conformity between the treatment plan and the treatment delivery, making sure that patient's bone and soft tissue landmarks are aligned as they were at the time of treatment planning [12]. Having a patient change position, wiggle, scratch, look the other way, or any other subtle movement could cause disruption in the treatment plan. By obtaining reliable information regarding the patient from the reconstructed images, clinicians have the opportunity to better ensure that the entire tumor receives the exact dose as planned while making sure surrounding healthy tissues are safe.

3.2 Image Reconstruction and Representation of Scattering Events

During image reconstruction, existing algorithms require that the interactions within an event happen sequentially as they occur. The modules residing in the camera that record interactions have a non-zero time-resolution. This means that the time it takes for the camera to record interactions is slower than the time it takes gamma rays to pass through the camera. This results in the interactions being detected as happening simultaneously. Reconstruction algorithms assume that interactions happen from a single prompt gamma ray source, yet the Compton camera may combine several single scatters into one event. This noisy data makes the reconstruction more difficult, because the image reconstruction algorithms have set prerequisites that are not met by the Compton Camera data output. Usually, at lower energy levels, the raw data recorded by the Compton Camera can still be directly used with adept algorithms for reconstruction. However, it has been observed that at higher energy levels, such as those used during proton radiotherapy, the raw data has proven to be ineffective for image reconstruction [8]. At higher energy levels, the proton beams tend to emit increased numbers of prompt gamma rays, which then increases the likelihood for the camera to record false events, making the reconstructed images ineffective and noisy.

The scattering events can be classified into five groups: True Triples, True Doubles, False Triples, False Doubles, and Double to Triple. A True Double and True Triple implies that the gamma-ray interacted exactly with the camera twice or three times, respectively. These are the events needed for the reconstruction of the images. A False Double event consists of two separate gamma rays interacting simultaneously with the camera, thus recorded as a True Double. Similarly, a True Triple encloses three separate gamma rays that interacting with the camera at the same time, thus falsely recorded by the module of the camera. These two False events need to be removed from the data, as they cause noise and corruption. Lastly, the Double to Triple occurs when two interactions were made by a single gamma ray and another interaction made by a different gamma-ray. The last interaction should be removed from the event for a better reconstruction process.

4 Deep Learning

4.1 Introduction to Deep Learning

Deep learning is a subfield of Machine Learning. It uses a successive multi-layer structure, thus the "deep" in the name, called neural network. Neural networks serve the purpose of helping the model learn by identifying patterns present in the data and classifying information based on the learned patterns. A fully connected deep neural network (DNN) contains a series of fully connected layers that work to connect each node or "neuron" in one layer to each in the next layer. Figure 4.1 exhibits the architecture of a fully-connected network. A DNN is composed of an input layer, which takes in the data, hidden layers that specifically transforms the data using some function and an output layer that returns a specific format of the transformed data. Normally, a data set is divided into two parts, a training set and a testing set. Within the training set, the data is divided into training data and validation data. The training data are used to find an optimal set of connection weights, the test data are used to choose the best network configuration, and once an optimal network has been found, a validation set is required in order to test the true generalization ability of the model [6]. When the entire training data goes through the network is called an epoch.

We run the data through the network for thousands of epochs, in order to improve its accuracy. There is a way to evaluate the network after each epoch to track the network's learning ability throughout the learning process. Moreover, deep learning models do not evaluate the entire data in the model at once, rather they separate it in different batches. One of the biggest challenges in machine learning is the dynamics between generalization and optimization. Optimization refers to the process of adjusting a model to get the best performance possible on the training data, whereas generalization is how the network performs on data it had never seen before. At the beginning of the training phase, generalization and optimization are correlated and the network has not learned all the patterns in the training data yet. This phenomenon is referred to as underfitting. After a certain number of epochs, the network's generalization stagnates and there is no improvement, this is called overfitting [6]. It performs well on the training data, but does not do as well on new data that it has not trained on rather the network starts to memorize the data. For this application, the network should have the ability to transform each scatter event in a way that it orders the interactions originated from the same prompt gamma ray in the correct order.

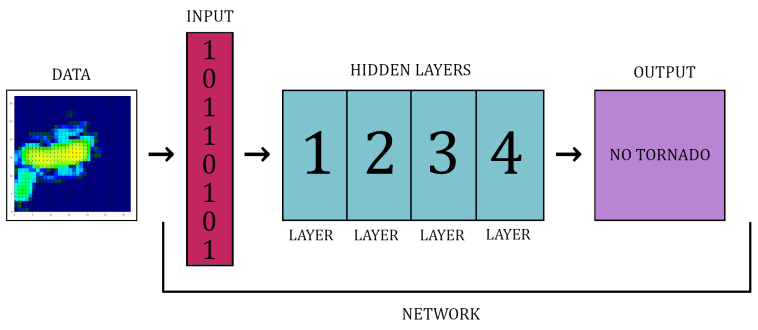

Figure 4.1: A fully connected neural network architecture.

4.2 Fully Connected Residual Blocks

Neural networks, especially fully connected ones, break down once they start becoming notably deep and complex [1]. The issues encountered with deep networks is that the values become smaller and smaller until getting to zero and like non-zero values as you go deeper and deeper. During back propagation we start to see the gradient becoming like-zero causing little to no update to existing weights which causes learning stagnation. This phenomenon is discussed more in details in [7] where they further explain the process. The takeaway from [7] is that they create ResNet, a network built from "residual blocks" as a solution to this issue. A visual representation of a residual block can be seen in Figure 4.2. Consider some record x. We pass it as an input to a small group of n layers with their own activators. The result of the layer output we can call y. Finally we concatenate x and y, using addition. This addition operation helps push non-zero values through the forward propagation process which helps keep input data to each block new and non-zero. This also helps prevent vanishing gradients during the back propagation process. We use this residual block for our deep connected network. Our residual blocks only use fully connected layers with post-activation concatenation for the classification of prompt gamma events. The fully connected residual blocks allows us to create a thin yet super deep fully connected neural network which avoids problems, mentioned above.

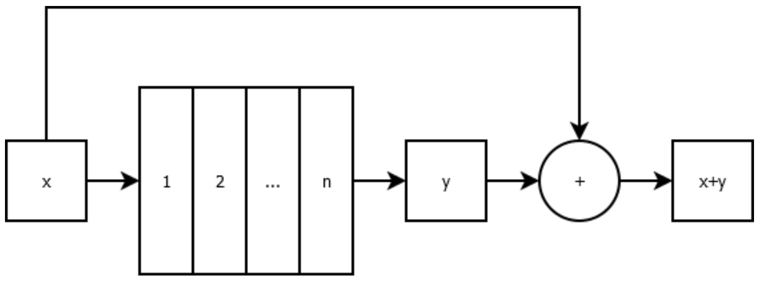

Figure 4.2: Our fully connected residual block takes an input and passes it through n layers eventually adding it to the output of the n layers

4.3 Recurrent Neural Network

Up to this point we have only done experiments which trained and evaluated the performance of fully connected neural networks. Fully connected have performed well but there are more specialized neural network configurations currently in use like recurrent neural networks. These specialized networks can leverage additional properties of the input data that would otherwise be lost when using a fully connected network. Recurrent neural networks (RNN) are able to incorporate the literal ordering of sequenced data. Another perspective is that any time data a future data entry relies on or relates to previous entries then some type of RNN would preferred. If one wanted to predict tomorrow's rainfall for some region on Earth based on the past several years worth of rainfall data then a RNN would be the preferred network choice. Another example is when trying to understand language base data like speech or written word. Language, written and spoken, rely on a structure ordering of words in order for proper communication of an idea. When dealing with language processing RNNs, or similar architectures, are the preferred choice for their ability to understand orderings in sequences [6].

Each event in our data consists of 2 or 3 randomly ordered interactions. These interactions have a correct ordering which a fully connected network has succeeded in detecting. It should be possible for a RNN to take in a single event as a collection of three interactions and determine what the correct order of the sequence is. A similar language problem would be to feed a RNN a jumbled sentence and asking it rebuild the original sentence. This idea coupled with the previous successes of fully connected networks gives us confidence that a RNN should be capable of performing this same task. The major drawbacks is that RNNs are very sensitive and prone to vanishing values and exploding gradient problems. They require a lot of tuning for optimal performance. For this reason we will be only conducting some preliminary viability tests on two different types of RNNs: LSTM and GRU. Additional details about LSTMs, GRUs, and RNNs can be found in [6].

5 Hardware and Software

We used the Graphic Processing Unit (GPU) clusters in the taki system in the UMBC High Performance Computing Facility (www.hpcf.umbc.edu) for our hyperparameter studies. For the studies that have 256 neurons or 256 layers we use the gpu2018 partition. This 1 GPU node has four NVIDIA Tesla V100 GPUs (5120 computational cores over 84 SMs, 16 GB onboard memory) connected by NVLink, two 18-core Intel Skylake CPUs, and 384 GB of memory (12×32 GB DDR4 at 2666 MT/s). For all other studies that has fewer than 256 neurons or layers, we use the gpu2013 partition which has 18 hybrid CPU/GPU nodes, each with two NVIDIA K20 GPUs (2496 computational cores over 13 SMs, 4 GB onboard memory), two 8-core Intel E5-2650v2 Ivy Bridge CPUs (2.6 GHz clock speed, 20 MB L3 cache, 4 memory channels), and 64 GB of memory (8×8 GB DDR3). For the gpu2018 we use 1 GPU per job, a time limit of 4 to 5 hours, and 35 GB of memory. For the gpu2013 we use 2 GPUs per job, a time limit of 4 to 16 hours, and MaxMemPerNode memory. the neural network backbone [8]. The training was done using Keras' Model.fit method on two NVIDIA K20 GPUs

This network was built using Tensorflow v2.4.0 (www.tensorflow.org) with the bundled Keras module8. We also used scikit-learn v0.22.1 (https://scikit-learn.org/stable/) to preprocess and normalize the data. Moreover pandas v1.0.4 (https://pandas.pydata.org/) and numpy v1.19.5 (www.numpy.org) were also used to help preprocess the data. Finaly we used the matplotlib v3.2.1 (www.matplotlib.org) library to graph our results.

6 Results

For our studies, we trained the neural network on a data set that was generated using a Monte Carlo simulation and that consisted of 1,821,255 records and 15 features. These features represent spatial coordinates, Euclidean distance, and energy deposition for each interaction. An interaction is a grouping of three spatial coordinates and an energy level. Each row is either a triple, double-to-triple, or a false triple and consists of three interactions each. Our training data set only consisted of True Triples, Double-to-Triple scatter, and False events. Furthermore, when testing the neural network we used datasets that used 150MeV (Mega electron Volt) beams with three different dosage rates: 20kMU (kilo Monitor Unit), 100kMU, and 180kMU. The larger kMU values correspond to more intense dosage rates.

6.1 Baseline Hyperparameter Studies

A total of 288 studies were run based on a combination of hyperparameters. In each of our studies, the validation rate, the number of epochs, and the residual block size were held constant at 0.2, 1024, and 8 respectively, to act as a control. The normalization method used also stayed consistent throughout all 288 studies. The energy values were normalized using a power transformer (Yeo-Johsnon), while the spatial coordinates were normalized using `MaxAbsScaler` from the sklearn library. Other hyperparameters such as dropout rate, number of neurons per layer, number of layers, and batch size were changed for each study. The values used for the changing variables are as follows:

- Drop out rate: 0, 0.1, 0.4
- Number of neurons per layer: 32, 64, 128, 256
- Number of layers: 8, 16, 32, 64, 128, 256
- Batch size: 1024, 2048, 4096, 8192

Thus, the total number of studies is $(3)(6)(4)(4) = 288$.

After all 288 studies were complete, each study that reported a peak validation accuracy greater than 0.76 was considered to be promising. The six promising studies includes studies 84, 90, 162, 167, 168, and 238 in Subsections 6.1.1 through 6.1.6, respectively. Each subsection contains a *hyperparameters table*, a *training and validation plot*, and three *confusion matrices* of the MCDE model test1 at 20kMU, 100kMU, and 180kMU dosage rates. The *hyperparameter tables* are Tables 6.1, 6.2, 6.3, 6.4, 6.5, 6.6 and list validation rate, dropout rate, neurons per layers, number of layers, batch size, and epochs. The *training and validation plots* are provided in Figures 6.1, 6.5, 6.9, 6.13, 6.17, 6.21. In these training and validation plots, the accuracy, denoted as a decimal percent, is plotted against the number of epochs. The peak validation accuracy is displayed at the top of each plot in addition to the total wall clock time the study took to run. The blue line represents the training accuracy and the orange line represents the validation accuracy. The three *confusion matrices* for each study are provided in Figures 6.2, 6.3, 6.4 for study 84, Figures 6.6, 6.7, 6.8 for study 90, Figures 6.10, 6.11, 6.12 for study 162, Figures 6.14, 6.15, 6.16 for study 167, Figures 6.18, 6.19, 6.20 for study 168, and Figures 6.22, 6.23, 6.24 for study 238. In these confusion matrices, the left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. These percentages sum to 100% in each row. Each cell is colored from white to dark green proportional to the maximum value in the entire matrix. The darkest entry in each row is the dominant classification of the input class. Within the matrices, there are thirteen different labels: 123, 132, 213, 231, 312, 321, 124, 214, 134, 314, 234, 324, and 444. Label 123 represents the case where all three interactions are correctly ordered and no adjustments need to be made. Label 132 represents the case where the third interaction should happen second and the second interaction should happen third. Label 213 represents the circumstance the second interaction should happen first, the first should occur second and the third is correctly labeled. Label 231 represents the scenario where the third interaction should be first, the second interaction should be third and the first interaction should be second. The label 312 represents the case where the first interaction should be third, the second interaction first whereas the last interaction should be second. The label 321 is the scenario where the first interaction should happen third, the second is labeled correctly, and the third interaction should happen first. The label 124 indicates that the first two interactions are a correctly ordered double and the third interaction should be disposed of. The label 214 indicates that the first two interactions are a incorrectly ordered double and the third interaction should be disposed of. The label 234 indicates that the last two interactions are a correctly ordered double and the first interaction should be disposed of. The label 324 indicates that the last two interactions are a incorrectly ordered double and the first interaction should be disposed of. The label 134 indicates that the first and third interactions are a correctly ordered double and the second interaction should be disposed of. The label 314 indicates that the first and third interactions are a incorrectly ordered double and the second interaction should be disposed of. The thirteenth label 444, is a false event, where the three interactions should be thrown away.

6.1.1 Study 84

This study is one of the promising of the 288 baseline hyperparameter studies. In Figure 6.1, we trained a deep fully connected network using the hyperparameters shown in the Table 6.1. We can see that both the training and validation accuracies start at 40% at 0 epochs, then proceed to increase significantly in the first few epochs, where both accuracies increase to 60%. Then, the slope becomes flatter as the network struggles to learn more information about the data.

Figure 6.2 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.1. It classifies the MCDE model test1 150MeV 20kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. We can observe that the maximum classification in each class is the input class itself in dark green. The accuracies for the true triples range from 66.3% to 74.6%., the double-to-triples' accuracies range from 62.2% to 71.5%, and the false events are $\approx$ 63%. Which means that when given any input class the will classify it the majority of the time. However there is still some discussion to be had about the dominant misclassifications of events. For some true triples the second highest classification percent is the double-to-triple version of itself. For instance, if we look at input class 132, the second highest at the 134 in the top column. This means that the network is able to classify correctly the first two interactions, but might have a harder time distinguishing the last interaction from a falsely coupled single. The other dominant misclassification case is where the second highest percentage is still a true triple but not correctly ordered. Finally we observe that the true triples are rarely classified as false events. For the double-to-triple data, we see that the second-highest ordering is the reverse ordered double-to-triple. This suggest that the network is able to correctly identify the interaction that does not belong in the event. However, it fails to order them once it takes out the single interaction that is not part of the pair. The third most likely misclassification of the doubles-to-triples are false events, which means that the network will drop just those events from the data. This is helpful for the reconstruction phase as it decrease noise in the data even if the data could have been used if correctly classified. There is still a large amount of doubles-to-triples that are still incorrectly labeled as true triples that will leads to noise in the reconstruction. For the false events we notice that second and third highest misclassification percentages are doubles-to-triples. When this happens we are unintentionally adding pure noise to the reconstruction algorithm leading to poorer results.

The conclusions made about Figure 6.2 also hold true for Figure 6.3 except for its respective percentages. Figure 6.1 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.1, which classifies the MCDE model test1 150MeV 100kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column.

The conclusions made about Figure 6.2 also hold true for Figure 6.4 except for its respective percentages. Figure 6.4 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.1, which classifies the MCDE model test1 150MeV 180kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. Even though the percentages in Figures 6.2, 6.3 and 6.4 are different, the trends and conclusions drawn from the network's performance remain the same.

Validation Rate	Dropout Rate	Neurons per Layer	Layers	Batch Size	Epochs
0.2	0	64	256	2048	1024

Table 6.1: Hyperparameters for study 84

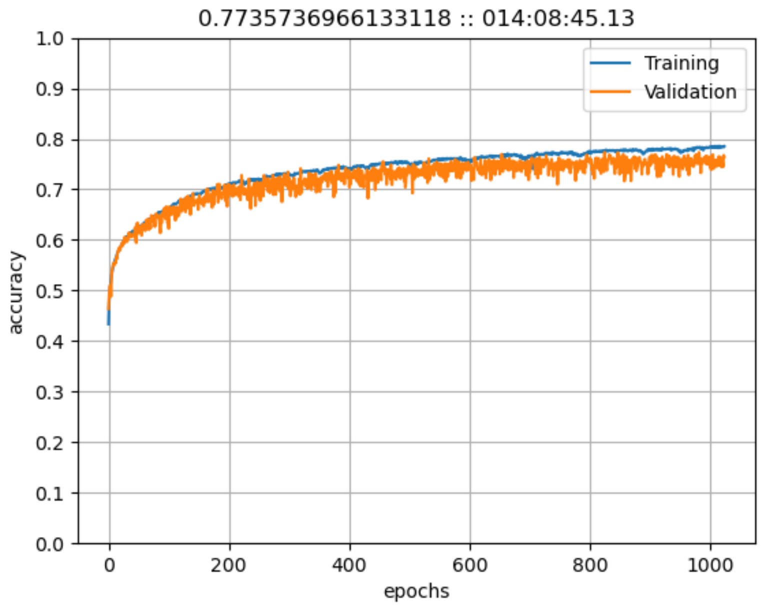

Figure 6.1: Training and validation plot for Study 84 using a fully connected network

We have seen that the networks ability to classify events of all input classes is good enough to warrant further experimentation but not nearly as high as we suspect they could be. The

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	66.3	8.1	2.1	3.4	3.0	2.7	8.1	0.6	0.1	0.1	3.5	1.4	0.6
132	3.8	71.2	2.6	2.2	2.9	3.1	0.3	0.1	7.8	0.8	1.4	3.3	0.5
213	3.3	3.5	70.6	3.4	2.0	2.9	1.1	6.7	4.4	1.2	0.3	0.0	0.6
231	1.5	3.2	4.8	71.1	3.0	5.4	0.1	0.4	1.6	2.5	4.8	1.1	0.3
312	2.7	2.7	2.3	2.7	74.6	4.0	3.0	1.5	0.9	5.1	0.1	0.2	0.3
321	2.5	3.3	2.8	2.2	5.9	72.1	1.4	2.8	0.0	0.3	0.7	5.6	0.3
124	3.5	0.4	0.6	0.1	3.2	2.1	71.5	9.3	0.9	0.4	0.4	1.7	5.8
214	0.8	0.3	4.5	0.3	2.3	3.3	13.6	66.8	0.5	1.2	0.8	0.5	5.0
134	0.4	4.0	3.7	2.5	0.4	0.1	1.0	0.6	71.3	8.8	1.8	0.5	5.0
314	0.1	0.8	2.1	5.1	6.8	0.4	0.3	1.4	8.9	66.5	0.6	0.9	6.1
234	2.6	2.4	0.3	7.6	0.1	1.5	0.8	1.1	1.3	0.7	62.2	13.8	5.7
324	1.3	4.6	0.2	0.8	0.2	8.0	1.1	0.6	0.9	0.6	8.9	67.6	5.2
444	0.6	0.3	0.9	0.6	0.6	0.3	6.3	6.6	4.1	5.0	8.5	6.3	59.9

Figure 6.2: Confusion Matrix for Study 84 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 20kMU beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	66.8	6.1	2.7	2.7	2.9	3.8	7.3	0.5	0.3	0.1	4.6	1.9	0.4
132	3.6	71.2	3.1	1.9	2.1	1.8	0.2	0.0	9.6	1.2	1.3	3.4	0.5
213	3.9	3.6	68.1	4.8	3.3	2.5	1.1	5.9	4.7	1.3	0.4	0.0	0.4
231	1.3	3.0	4.8	73.0	1.8	5.6	0.3	0.3	1.7	2.0	4.4	1.0	0.6
312	1.9	2.3	2.7	2.7	76.9	4.2	2.9	1.1	0.4	4.4	0.0	0.4	0.1
321	2.1	3.4	3.8	2.5	7.3	70.7	1.3	3.7	0.0	0.2	0.6	4.1	0.4
124	5.3	0.4	0.8	0.1	3.4	2.4	70.8	8.9	0.6	0.4	0.4	1.5	5.0
214	0.8	0.1	4.8	0.5	1.7	4.0	14.0	65.5	0.8	1.1	0.9	0.7	5.1
134	0.2	4.9	3.0	2.1	0.7	0.1	0.5	0.7	70.0	10.4	1.6	0.4	5.4
314	0.1	0.5	1.6	4.3	6.1	0.2	0.3	1.4	10.0	68.1	0.6	1.1	5.8
234	2.8	2.2	0.4	6.2	0.2	2.0	1.1	0.7	1.4	0.7	63.4	13.6	5.4
324	1.2	4.6	0.1	1.3	0.5	6.7	1.3	0.7	0.8	0.5	8.4	67.4	6.6
444	0.5	0.3	0.2	0.6	0.6	0.6	5.5	5.7	5.6	5.0	4.6	5.5	65.4

Figure 6.3: Confusion Matrix for Study 84 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 100kMU beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	69.7	7.5	2.9	1.2	2.4	1.7	7.9	0.2	0.2	0.2	3.4	2.2	0.5
132	4.6	72.1	2.4	1.2	2.2	2.6	0.0	0.2	7.9	0.7	1.4	3.6	1.0
213	2.9	1.4	73.3	4.6	3.1	0.7	1.9	6.0	3.6	1.4	0.2	0.2	0.5
231	1.4	3.1	4.1	72.3	3.4	5.3	0.2	0.0	1.2	2.9	5.1	1.0	0.0
312	2.9	3.1	1.0	2.2	74.9	3.9	3.1	1.4	1.0	5.1	0.0	0.2	1.2
321	2.9	3.6	2.7	1.0	4.6	72.3	1.7	5.5	0.0	0.2	0.7	4.3	0.5
124	4.4	0.5	0.3	0.2	3.6	1.9	71.0	9.5	1.0	0.6	0.6	0.7	5.8
214	1.0	0.1	4.2	0.7	2.1	3.2	14.0	67.0	0.5	1.4	0.7	0.3	5.0
134	0.4	5.2	2.7	2.0	1.3	0.0	0.7	0.6	72.4	8.0	1.5	0.7	4.4
314	0.2	0.6	1.4	4.2	6.4	0.5	0.4	1.3	9.1	68.5	0.3	0.7	6.4
234	2.8	2.1	0.4	5.6	0.3	1.4	0.6	0.4	1.5	0.6	65.1	12.8	6.4
324	1.4	4.3	0.1	1.0	0.3	6.8	2.1	0.3	0.6	0.6	9.7	66.2	6.4
444	0.5	1.0	0.3	0.6	0.3	0.7	5.8	5.7	5.1	5.7	6.3	4.4	63.5

Figure 6.4: Confusion Matrix for Study 84 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 180kMU beam.

percentages in Figure 6.2, 6.3, 6.4 are still 5% to 10% worse than the more complex networks seen in [1] for all input classes. The positive side is that while our network is worse it is significantly faster and more memory efficient since it uses fewer layers and neurons per layer than the more complex networks. Additionally neither our network nor more complex networks meet the 80% to 90% real-world use threshold.

6.1.2 Study 90

In Figure 6.5, we trained a deep fully connected network using the hyperparameters shown in the Table 6.2. The values for the validation and the dropout rate remain the same, however, we changed the other parameters. In Figure 6.5, we can observe a steep jump in the first few epochs. Then, we have a flatter slope steadily increasing over the 1000 epochs. We can observe that both validation and training are still increasing at 1024 epochs, thus might yield better results if given sufficient time. Figure 6.6 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.2, which classifies the MCDE model test1 150MeV 20kMU beam.

The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. We can observe that the maximum classification in each class is the input class itself in dark green. In Figure 6.6, we can notice that the accuracies for the triples range from 60% to 74%, the Double-to-Triples' accuracies range from 60% to 71% and finally the false event falls into 57%.

We can observe that the second-highest percentage for the true triples are classified as unordered true triples, meaning that the network has a hard time detecting the order at which the events happened. Sometimes, they are also classified as their corresponding Double-to-Triples, meaning that the network can correctly classify the first two interactions, but thinks the last one belong to another event.

For the Double-to-Triples, we see that the second-highest percentage are also Double-to-Triples, showing that the network is successfully able to detect that there is a pair and a separate interaction. However, it fails to recognize the correct ordering of the events.

In the case of the false events, we see that apart from the ones that are correctly detected as false events, most of them are classified as Double-to-Triples. This means that there is a false double being passed in the network. Even though, the network is able to majoritarily classify the events as they are, we still have some data that incorrectly classified, which will cause noise in the reconstruction process. The accuracies are still comparable to that of larger and more complex networks.

The conclusions made about Figure 6.6 also hold true for Figure 6.7 except for its respective percentages. Figure 6.7 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.2, which classifies the MCDE model test1 150MeV 100kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column.

The conclusions made about Figure 6.6 also hold true for Figure 6.8 except for its respective percentages. Figure 6.8 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.2, which classifies the MCDE model test1 150MeV 100kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column.

Validation Rate	Dropout Rate	Neurons per Layer	Layers	Batch Size	Epochs
0.2	0	64	256	4096	1024

Table 6.2: Hyperparameters for study 90

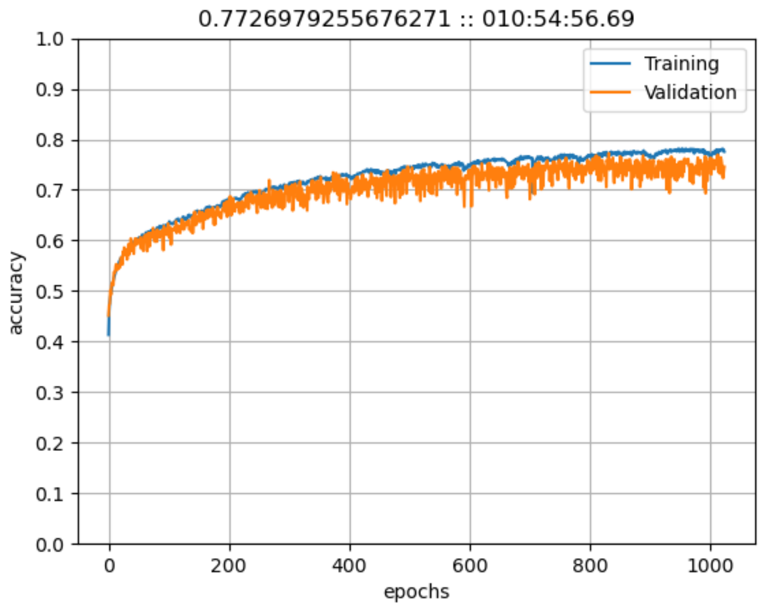

Figure 6.5: Training and validation plot for Study 90 using a fully connected network

6.1.3 Study 162

In Figure 6.9, we trained a deep fully connected network using the hyperparameters shown in the Table 6.3. The values for the validation and the dropout rate remain the same, however, we changed the other parameters. In Figure 6.9, we can observe a steep jump in the first few epochs. Then, we have the training plot increasing at higher rate than the validation plot. Still, both of them are still increasing meaning that the network is continuing to learn.

The conclusions made about Figure 6.2 also hold true for Figure 6.10 except for its respective percentages. Figure 6.10 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.3, which classifies the MCDE model test1 150MeV 20kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. We can observe that the maximum classification in each class is the input class itself in dark green.

The conclusions made about Figure 6.2 also hold true for Figure 6.11 except for its respective percentages. Figure 6.11 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.3, which classifies the MCDE model test1 150MeV 100kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. We can observe that the maximum classification in each class is the input class itself in dark green.

The conclusions made about Figure 6.2 also hold true for Figure 6.12 except for its respective percentages. Figure 6.12 is a confusion matrix created from our fully connected network, whose

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	73.4	4.5	1.9	3.2	2.4	1.9	6.0	0.7	0.3	0.0	4.3	1.1	0.4
132	9.8	67.5	2.2	3.1	2.0	1.9	0.2	0.1	6.8	0.5	3.1	2.3	0.4
213	7.2	3.6	66.5	5.0	2.2	1.8	1.1	5.3	4.5	1.7	0.4	0.1	0.5
231	3.4	3.2	6.1	71.6	2.0	1.5	0.0	0.2	2.0	2.3	6.8	0.4	0.4
312	4.0	5.5	3.0	3.4	68.5	3.5	3.1	1.4	1.2	5.6	0.1	0.4	0.2
321	4.6	2.5	3.5	6.6	9.1	61.6	1.8	2.3	0.0	0.4	1.5	5.6	0.6
124	5.1	0.4	1.3	0.2	3.7	1.8	66.1	11.3	1.1	0.8	1.0	2.0	5.2
214	1.6	0.3	5.3	0.3	2.8	2.8	15.5	62.5	0.5	2.0	1.1	0.7	4.5
134	0.4	5.6	3.7	2.1	0.4	0.2	0.4	0.5	69.8	9.8	1.8	0.8	4.5
314	0.0	1.0	2.4	5.7	6.4	0.5	0.7	1.1	13.2	62.8	0.4	1.0	4.8
234	3.2	2.2	0.2	6.0	0.2	0.5	0.8	1.0	2.0	0.6	70.3	7.6	5.5
324	1.9	3.4	0.0	1.3	0.3	6.1	0.9	0.5	0.5	0.8	19.3	60.2	4.9
444	0.9	1.3	1.3	0.6	0.0	0.6	5.3	7.5	3.8	5.0	9.1	7.8	56.7

Figure 6.6: Confusion Matrix for Study 90 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 20kMU beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	74.1	4.2	2.0	3.9	2.0	1.3	5.5	1.0	0.1	0.1	3.9	1.6	0.2
132	10.0	67.0	1.9	3.4	2.2	1.0	0.1	0.0	7.9	1.0	3.0	2.0	0.4
213	8.0	4.3	63.6	6.5	1.8	1.8	1.2	4.7	5.0	1.7	0.5	0.0	0.7
231	2.9	2.0	6.0	76.2	1.4	1.4	0.1	0.0	2.2	2.0	5.3	0.2	0.1
312	3.7	5.6	3.7	2.3	70.3	3.8	2.7	1.3	0.8	5.1	0.1	0.3	0.3
321	4.7	3.1	4.8	5.4	9.7	59.8	1.5	3.6	0.1	0.4	1.5	5.2	0.2
124	7.3	0.4	1.0	0.2	3.1	2.2	68.7	8.8	0.9	0.3	0.8	1.3	5.0
214	1.4	0.1	5.4	0.3	2.2	3.3	15.4	61.3	0.8	2.2	1.1	0.9	5.7
134	0.1	5.8	2.9	2.2	0.4	0.2	0.6	0.7	70.0	9.7	1.9	0.5	4.8
314	0.1	1.0	2.0	3.7	4.7	0.4	0.4	1.3	15.7	63.8	0.5	1.1	5.3
234	3.0	2.2	0.6	6.6	0.1	0.4	0.7	0.6	1.3	0.5	72.1	7.7	4.3
324	1.6	3.4	0.2	1.6	0.3	5.3	1.0	0.7	0.9	0.3	19.4	60.2	5.1
444	0.6	0.7	0.5	0.7	0.5	0.7	4.4	5.8	6.0	7.5	6.9	5.7	59.8

Figure 6.7: Confusion Matrix for Study 90 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 100kMU beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	70.9	6.7	1.7	3.4	3.1	1.7	6.0	0.5	0.2	0.0	4.1	1.4	0.2
132	10.1	69.2	1.4	2.6	1.7	1.9	0.5	0.0	5.3	0.7	2.6	3.1	0.7
213	7.5	2.2	66.0	6.0	1.4	1.7	1.4	7.0	4.6	1.4	0.5	0.0	0.2
231	3.9	3.4	7.2	68.9	2.9	1.0	0.0	0.5	2.4	2.4	6.3	0.7	0.5
312	4.8	7.0	1.9	1.4	68.9	3.1	2.9	1.7	1.9	4.3	0.2	0.0	1.7
321	4.3	2.2	4.6	8.2	8.9	61.2	1.2	2.2	0.0	0.0	1.0	5.5	0.7
124	7.6	0.6	0.4	0.2	2.6	1.8	67.0	10.5	1.4	0.7	0.8	1.3	5.2
214	1.7	0.1	5.0	0.5	2.0	3.0	16.0	61.7	0.5	1.9	1.0	0.4	6.2
134	0.2	6.2	2.9	2.0	0.6	0.1	0.5	0.8	72.8	7.8	1.9	0.8	3.4
314	0.1	0.9	1.7	4.1	5.9	0.3	0.3	1.4	14.5	63.9	0.8	0.8	5.4
234	3.5	1.8	0.5	6.0	0.4	1.1	0.2	0.2	1.2	0.5	73.1	7.0	4.5
324	2.4	3.9	0.1	1.0	0.2	6.4	1.3	0.3	0.6	0.8	19.8	57.7	5.6
444	0.6	1.1	0.2	0.7	0.3	0.2	4.9	5.9	6.2	6.3	8.7	6.2	58.8

Figure 6.8: Confusion Matrix for Study 90 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 180kMU beam.

parameters are displayed in Table 6.3, which classifies the MCDE model test1 150MeV 180kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. We can observe that the maximum classification in each class is the input class itself in dark green.

The accuracies are low ranging from 55% to 72% showing that the network might have started to memorize the data because the training accuracy is higher than the testing accuracies. As mentioned before, we might still have many unusable data slip in the reconstruction, thus polluting it and causing noise.

Validation Rate	Dropout Rate	Neurons per Layer	Layers	Batch Size	Epochs
0.2	0	128	256	4096	1024

Table 6.3: Hyperparameters for study 162

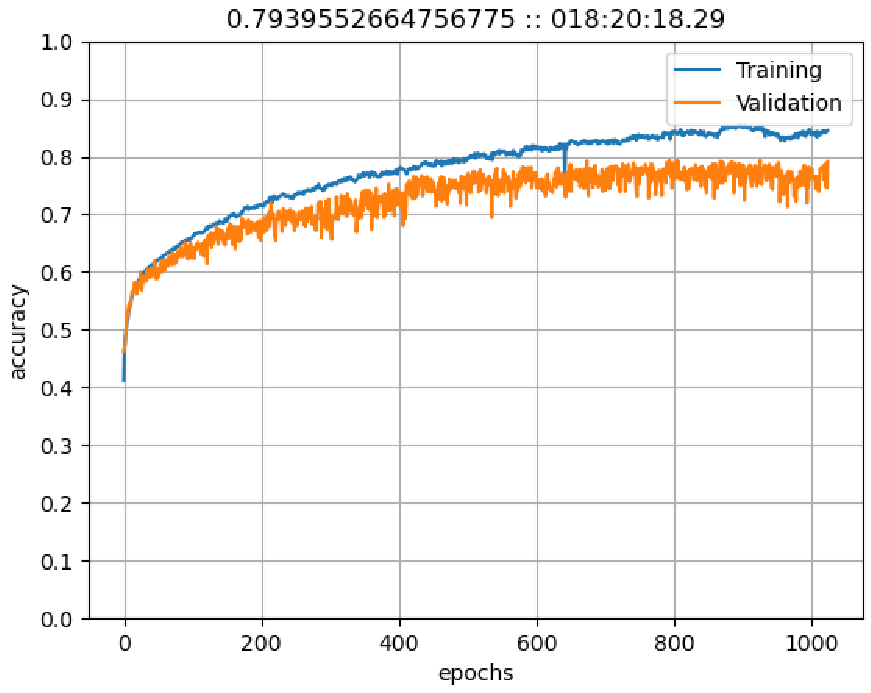

Figure 6.9: Training and validation plot for Study 162 using a fully connected network

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	65.4	6.4	4.5	2.9	3.6	3.2	5.6	1.2	0.4	0.1	4.6	1.9	0.2
132	4.7	67.6	3.3	2.7	3.0	3.4	0.3	0.1	7.4	0.9	2.1	4.0	0.5
213	2.2	2.9	70.9	4.5	1.7	3.3	1.1	5.1	5.6	1.9	0.4	0.1	0.4
231	3.2	3.0	6.5	66.9	2.4	3.8	0.1	0.2	2.1	3.9	6.7	1.0	0.4
312	2.5	3.2	4.1	3.1	67.1	6.3	2.5	1.6	1.2	7.3	0.1	0.6	0.4
321	2.7	2.7	4.5	3.1	5.2	71.7	0.9	2.3	0.1	0.4	0.9	5.1	0.3
124	6.1	0.5	1.9	0.1	4.5	3.5	60.9	12.8	0.9	0.4	0.6	1.8	6.1
214	1.2	0.3	8.1	0.2	2.6	6.3	12.2	59.5	0.9	2.1	0.7	0.5	5.5
134	0.3	5.4	3.8	1.5	0.7	0.2	0.5	0.4	69.8	10.0	1.6	0.9	4.9
314	0.1	1.1	2.6	4.5	6.6	0.6	0.6	1.2	14.1	62.1	0.3	1.3	4.7
234	3.2	2.0	0.2	6.9	0.3	1.4	0.6	0.9	1.7	1.5	63.0	12.1	6.4
324	1.5	3.7	0.1	0.5	0.2	8.2	1.8	0.3	0.4	1.2	11.6	65.6	5.1
444	0.9	0.6	0.9	0.6	0.3	0.9	6.6	5.6	7.2	5.6	6.9	8.5	55.2

Figure 6.10: Confusion Matrix for Study 162 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 20K beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	66.2	5.7	4.0	4.0	3.1	2.2	5.0	0.8	0.3	0.0	4.5	3.8	0.4
132	4.4	68.2	3.2	2.5	2.9	2.5	0.1	0.1	7.4	1.5	2.0	4.6	0.7
213	3.7	3.1	67.7	6.3	1.5	3.1	0.8	4.6	5.6	2.2	0.8	0.2	0.3
231	3.0	1.9	7.0	68.5	1.9	2.8	0.2	0.1	2.9	3.8	6.4	0.9	0.5
312	2.1	3.7	3.9	2.3	67.5	7.7	2.5	1.2	1.3	6.8	0.0	0.6	0.3
321	2.6	3.1	5.3	2.8	5.8	70.3	1.2	2.8	0.1	0.4	0.7	4.7	0.3
124	7.4	0.4	0.8	0.1	4.6	2.6	63.1	11.8	0.8	0.9	0.6	2.0	4.9
214	1.0	0.1	8.3	0.4	2.5	5.3	12.0	59.6	1.4	1.9	0.9	0.8	5.7
134	0.4	5.3	3.4	2.1	0.4	0.1	0.5	0.4	69.3	10.6	1.7	0.7	5.2
314	0.1	0.9	1.9	4.2	5.7	0.4	0.5	1.0	13.5	63.7	0.7	1.6	5.8
234	2.8	1.6	0.5	7.1	0.1	1.2	0.8	0.5	2.1	1.1	65.7	11.1	5.2
324	1.7	3.4	0.1	1.3	0.5	8.6	1.7	0.5	0.8	1.0	11.2	63.0	6.2
444	1.0	0.5	0.6	0.8	0.5	0.9	5.3	5.2	7.7	7.3	6.0	5.5	58.7

Figure 6.11: Confusion Matrix for Study 162 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 100K beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	65.1	6.2	4.1	2.2	3.8	4.1	5.5	1.2	0.2	0.0	4.8	2.6	0.0
132	4.1	71.9	1.4	1.7	2.4	3.1	0.2	0.0	6.5	0.7	2.2	4.3	1.4
213	2.4	3.4	71.6	4.8	1.7	2.7	1.7	5.1	4.1	2.4	0.0	0.0	0.2
231	3.6	4.3	8.0	62.9	2.4	4.6	0.2	0.5	1.9	4.6	5.5	1.2	0.2
312	2.9	3.9	1.9	1.4	67.0	8.4	1.9	1.9	1.2	7.7	0.0	1.2	0.5
321	2.2	2.4	5.1	1.7	4.3	72.3	3.1	3.6	0.0	0.2	1.0	3.9	0.2
124	6.8	0.4	1.0	0.1	4.7	3.6	61.0	12.9	1.3	0.6	1.1	1.4	5.2
214	1.5	0.0	7.8	0.7	2.0	5.4	12.7	60.4	1.0	2.0	0.7	0.8	5.1
134	0.5	5.5	3.1	1.7	0.7	0.1	0.5	0.5	70.7	9.9	1.4	0.8	4.8
314	0.2	0.5	1.7	3.1	7.1	1.0	0.5	1.2	14.1	63.9	0.5	0.6	5.7
234	3.2	2.0	0.9	6.9	0.3	1.1	0.4	0.5	1.4	1.0	64.5	11.8	6.0
324	1.8	3.9	0.1	1.1	0.4	8.0	2.0	0.5	0.6	1.4	12.3	62.0	5.9
444	0.7	0.8	0.5	0.4	0.3	0.9	5.4	4.7	7.7	7.5	7.2	5.9	57.9

Figure 6.12: Confusion Matrix for Study 162 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 180K beam.

6.1.4 Study 167

In Figure 6.13, we trained a deep fully connected network using the hyperparameters shown in the Table 6.4. The values for the validation and the dropout rate remain the same, however, we changed the other parameters. In Figure 6.13, we can observe a steep jump in the first few epochs. Then, we have both of the training and validation plots increasing meaning that the network is continuing to learn.

The conclusions made about Figure 6.2 also hold true for Figure 6.14 except for its respective percentages.

Figure 6.14 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.4, which classifies the MCDE model test1 150MeV 20kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. We can observe that the maximum classification in each class is the input class itself in dark green.

We can make the same conclusions from Figure 6.2 for Figure 6.15 except for its respective percentages.

Figure 6.15 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.4, which classifies the MCDE model test1 150MeV 100kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. We can observe that the maximum classification in each class is the input class itself in dark green.

We can make the same conclusions from Figure 6.2 for Figure 6.16 except for its respective percentages.

Figure 6.16 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.4, which classifies the MCDE model test1 150MeV 100kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. We can observe that the maximum classification in each class is the input class itself in dark green.

Validation Rate	Dropout Rate	Neurons per Layer	Layers	Batch Size	Epochs
0.2	0	128	128	8192	1024

Table 6.4: Hyperparameters for study 167

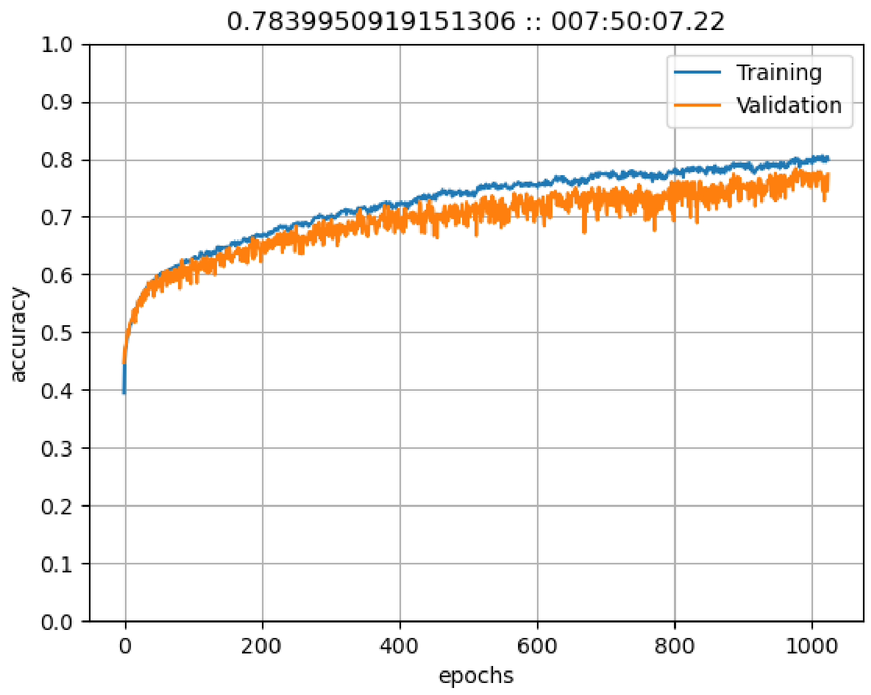

Figure 6.13: Training and validation plot for Study 167 using a fully connected network

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	62.9	5.0	5.4	3.3	2.2	4.3	6.6	1.8	0.6	0.1	5.1	2.0	0.7
132	6.4	61.7	1.8	3.3	9.0	3.4	0.2	0.1	6.1	1.3	2.5	3.7	0.5
213	2.9	3.1	62.2	8.8	2.8	4.3	0.9	6.4	5.7	1.8	0.5	0.2	0.4
231	3.1	3.0	4.0	70.0	2.8	2.6	0.1	0.2	2.7	2.7	7.9	0.7	0.2
312	2.5	3.0	3.2	2.7	67.9	9.4	2.5	2.1	1.3	4.4	0.2	0.4	0.4
321	3.0	2.6	3.2	5.0	4.6	69.1	1.7	2.5	0.2	0.5	1.4	5.6	0.5
124	4.3	0.7	1.3	0.2	3.8	3.8	57.7	17.3	1.2	0.7	0.4	1.8	6.8
214	0.8	0.2	4.4	0.6	2.4	4.8	12.0	65.7	0.8	1.5	0.6	0.8	5.5
134	0.4	5.0	2.4	2.8	1.9	0.3	0.5	0.5	66.1	13.2	1.6	0.9	4.3
314	0.1	1.0	1.4	4.0	7.8	0.6	0.8	1.2	14.6	62.5	0.4	0.9	4.8
234	3.3	2.0	0.1	6.0	0.1	0.9	0.8	1.1	2.1	0.9	65.5	11.8	5.2
324	1.5	4.2	0.1	0.9	0.3	7.3	1.3	0.4	1.1	0.8	14.0	62.6	5.5
444	0.6	1.6	0.9	0.6	0.0	0.0	5.3	7.5	7.5	4.4	6.9	8.5	56.1

Figure 6.14: Confusion Matrix for Study 167 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 20K beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	61.7	4.8	5.8	3.8	1.9	3.7	6.4	1.3	0.6	0.0	7.3	2.1	0.5
132	6.0	61.9	1.9	3.1	7.8	1.9	0.2	0.0	8.6	1.5	2.9	4.0	0.2
213	4.0	2.7	59.4	9.7	3.8	4.3	0.9	6.5	5.3	1.9	0.7	0.2	0.5
231	3.5	1.8	3.8	71.1	2.5	1.9	0.1	0.3	3.8	2.8	7.9	0.6	0.0
312	2.3	2.8	2.7	3.0	70.9	9.0	2.0	1.1	1.0	3.9	0.4	0.4	0.5
321	3.2	3.1	3.2	4.2	5.2	68.3	1.7	3.8	0.2	0.4	1.5	4.7	0.5
124	5.5	0.4	1.0	0.0	3.1	2.9	60.2	16.4	1.3	0.5	0.5	1.7	6.4
214	1.0	0.2	4.4	0.3	1.9	4.2	10.7	66.7	0.7	1.6	0.9	0.7	6.6
134	0.2	4.5	2.1	2.5	1.7	0.1	0.4	0.8	67.6	13.9	1.3	0.4	4.6
314	0.0	0.5	1.8	3.9	7.5	0.4	0.5	1.3	15.1	61.3	0.9	1.5	5.2
234	3.1	1.9	0.5	5.7	0.1	1.0	0.5	0.5	2.0	0.4	66.8	11.8	5.6
324	1.3	3.2	0.2	1.4	0.4	7.0	1.1	0.7	1.1	0.5	15.5	61.2	6.4
444	0.6	0.4	0.3	0.5	0.9	0.5	4.8	5.5	7.5	5.7	7.3	6.0	59.9

Figure 6.15: Confusion Matrix for Study 167 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 100K beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	66.3	6.0	4.6	3.6	2.4	2.4	5.3	2.2	0.2	0.0	3.8	2.9	0.2
132	5.5	64.9	1.9	3.1	6.5	2.4	0.5	0.2	7.2	0.7	2.2	3.6	1.2
213	2.9	2.9	62.9	8.9	3.4	3.1	0.7	7.7	4.6	2.4	0.5	0.0	0.0
231	3.4	3.1	5.5	67.0	4.3	2.9	0.2	0.0	1.9	2.9	7.5	0.7	0.5
312	1.4	3.1	2.7	2.4	66.5	11.1	3.4	1.7	1.4	5.1	0.0	0.2	1.0
321	4.1	3.4	3.4	3.4	3.9	68.0	1.7	4.3	0.0	0.0	0.7	6.0	1.2
124	5.0	0.6	0.9	0.1	2.9	2.9	59.1	16.6	1.6	0.6	1.1	1.4	7.4
214	1.2	0.1	4.5	0.3	1.6	4.3	11.0	67.1	0.7	1.1	0.7	0.5	6.8
134	0.2	5.2	2.1	2.0	2.1	0.1	0.3	0.8	67.9	13.2	1.7	1.1	3.4
314	0.2	0.7	1.3	3.8	7.7	0.4	0.5	1.9	14.5	62.1	0.6	0.7	5.6
234	3.2	2.0	0.3	4.4	0.2	0.9	0.5	0.6	2.1	0.6	70.5	10.3	4.4
324	2.1	3.3	0.0	1.2	0.4	7.3	1.2	0.2	0.6	0.5	15.6	60.0	7.7
444	0.5	0.7	0.6	0.6	0.5	0.5	5.1	5.8	7.4	5.7	7.8	6.2	58.6

Figure 6.16: Confusion Matrix for Study 167 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 180K beam.

6.1.5 Study 168

In Figure 6.17, we trained a deep fully connected network using the hyperparameters shown in the Table 6.5. The values for the validation and the dropout rate remain the same, however, we changed the other parameters. In Figure 6.17, we can observe a steep jump in the first few epochs. Then, we have both of the training and validation plots increasing meaning that the network is continuing to learn.

The conclusions made about Figure 6.2 also hold true for Figure 6.18 except for its respective percentages.

Figure 6.18 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.5, which classifies the MCDE model test1 150MeV 20kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. We can observe that the maximum classification in each class is the input class itself in dark green.

We can make the same conclusions from Figure 6.2 for Figure 6.19 except for its respective percentages.

Figure 6.19 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.5, which classifies the MCDE model test1 150MeV 100kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. We can observe that the maximum classification in each class is the input class itself in dark green.

We can make the same conclusions from Figure 6.2 for Figure 6.20 except for its respective percentages.

Figure 6.20 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.5, which classifies the MCDE model test1 150MeV 100kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. We can observe that the maximum classification in each class is the input class itself in dark green.

Figure 6.18, Figure 6.19, Figure 6.20 are confusion matrices of classification accuracies. The network is correctly classifying of all events between 52% to 67% for 20KMU, 51% to 68% for 100kMU, and 52% to 69% for 180kMU. The accuracy is low ranging from 52% to 69% showing comparable results to that of more complex networks.

Validation Rate	Dropout Rate	Neurons per Layer	Layers	Batch Size	Epochs
0.2	0	128	256	8192	1024

Table 6.5: Hyperparameters for study 168

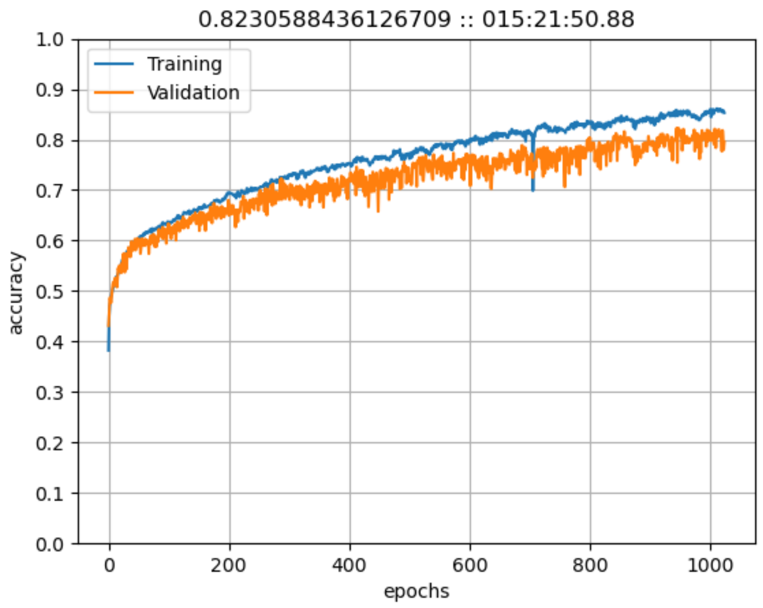

Figure 6.17: Training and validation plot for Study 168 using a fully connected network

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	63.4	7.4	7.9	2.2	2.2	3.7	6.9	0.9	0.3	0.1	3.9	1.0	0.2
132	8.6	63.4	2.7	2.1	6.5	3.4	0.5	0.1	5.6	1.4	2.5	2.7	0.5
213	3.8	4.0	67.6	7.1	2.1	3.2	1.8	4.9	3.1	1.9	0.2	0.0	0.4
231	3.8	4.6	7.0	61.4	3.0	5.6	0.2	0.4	2.1	3.0	7.6	0.8	0.4
312	2.8	5.5	4.6	2.0	62.5	9.0	4.2	2.1	1.1	5.7	0.0	0.2	0.3
321	4.5	2.1	4.1	3.9	4.7	67.4	3.0	3.1	0.1	0.5	2.3	3.9	0.4
124	7.0	0.4	2.5	0.1	3.0	2.1	65.3	11.5	0.8	0.7	0.5	1.3	4.7
214	1.4	0.3	9.3	0.2	2.5	4.7	18.3	56.1	0.3	1.5	0.8	0.3	4.3
134	0.6	6.2	3.2	2.5	1.6	0.3	1.3	0.4	57.2	18.9	2.5	0.9	4.5
314	0.1	1.6	2.2	4.5	6.6	0.5	1.1	1.6	11.6	65.1	0.6	0.9	3.7
234	5.0	2.5	0.3	5.1	0.1	1.0	0.7	1.4	1.8	1.2	67.1	9.8	3.8
324	2.7	3.8	0.1	1.1	0.3	7.6	2.3	0.6	0.6	1.5	23.3	51.8	4.4
444	0.9	0.6	0.3	0.3	0.3	0.0	9.7	6.0	5.3	6.3	8.5	7.5	54.2

Figure 6.18: Confusion Matrix for Study 168 using fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 20K beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	64.8	7.5	7.6	1.5	2.0	3.5	6.8	0.5	0.2	0.0	4.5	1.1	0.0
132	8.4	66.8	2.1	1.5	5.7	2.2	0.4	0.2	6.0	1.0	2.3	2.9	0.3
213	5.0	4.1	64.1	8.9	2.2	3.0	1.0	4.4	3.9	2.1	0.6	0.1	0.5
231	4.4	3.6	7.7	63.0	1.4	5.4	0.4	0.5	2.9	2.7	7.0	0.8	0.2
312	2.9	5.2	4.5	1.6	63.4	9.1	4.1	2.0	1.3	5.4	0.1	0.2	0.1
321	3.5	2.9	5.7	3.9	5.5	65.9	3.1	3.8	0.1	0.2	2.1	3.0	0.4
124	6.8	0.6	2.0	0.0	3.0	2.2	67.7	11.1	0.7	0.4	0.6	1.0	4.0
214	1.7	0.1	8.3	0.2	1.5	3.7	17.4	58.1	0.7	1.6	0.8	0.4	5.4
134	0.5	5.9	5.0	2.0	1.3	0.4	1.0	0.6	57.1	19.9	2.2	0.5	3.7
314	0.0	1.3	2.6	3.6	6.7	0.2	1.2	1.6	12.3	64.8	0.8	0.8	4.0
234	5.0	2.3	0.8	5.6	0.2	1.2	1.0	0.7	1.4	0.8	68.5	8.5	3.8
324	2.8	3.1	0.2	1.0	0.6	8.8	2.3	0.7	0.6	0.9	22.4	51.5	5.1
444	1.4	0.5	0.5	0.6	0.8	0.6	8.0	5.6	6.8	8.3	6.6	5.3	55.0

Figure 6.19: Confusion Matrix for Study 168 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 100K beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	63.5	8.2	7.2	0.7	1.4	4.8	7.7	0.7	0.0	0.5	4.1	1.2	0.0
132	9.9	63.5	1.9	0.5	6.0	3.6	1.0	0.2	5.3	1.9	2.4	3.1	0.7
213	3.9	2.4	69.2	6.7	2.4	2.2	1.7	5.5	2.7	2.9	0.0	0.0	0.5
231	3.6	4.1	8.4	62.4	3.1	4.1	0.2	1.7	1.4	1.9	7.2	1.2	0.5
312	2.4	5.3	3.9	1.2	61.0	12.3	4.6	1.9	1.2	5.5	0.2	0.5	0.0
321	5.5	2.7	4.8	2.9	5.1	65.8	3.9	3.1	0.0	0.0	2.4	3.1	0.7
124	7.3	0.5	2.1	0.1	2.8	2.9	66.7	10.5	0.9	0.5	0.4	0.6	4.8
214	1.4	0.0	7.8	0.2	2.1	3.7	16.7	60.0	0.6	1.7	0.5	0.4	4.9
134	0.5	7.6	3.2	1.9	2.0	0.0	1.2	0.6	58.7	18.1	1.9	0.6	3.7
314	0.1	1.0	1.7	3.2	7.5	0.3	1.1	2.0	11.5	66.1	0.5	0.4	4.6
234	5.7	2.1	0.6	4.0	0.3	1.9	1.1	0.8	0.8	1.0	69.3	8.6	3.7
324	2.7	4.1	0.1	0.9	0.4	7.9	3.3	0.7	0.7	0.9	23.0	50.0	5.3
444	1.2	1.0	0.5	0.5	0.3	0.5	9.0	5.3	6.3	8.6	8.5	6.0	52.2

Figure 6.20: Confusion Matrix for Study 168 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 100K beam.

6.1.6 Study 238

In Figure 6.13, we trained a deep fully connected network using the hyperparameters shown in the Table 6.4. The values for the validation and the dropout rate remain the same, however, we changed the other parameters. In Figure 6.13, we can observe a steep jump in the first few epochs. Then, we have both of the training and validation plots increasing meaning that the network is continuing to learn.

The conclusions made about Figure 6.2 also hold true for Figure 6.22 except for its respective percentages.

Figure 6.22 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.6, which classifies the MCDE model test1 150MeV 20kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. We can observe that the maximum classification in each class is the input class itself in dark green.

We can make the same conclusions from Figure 6.2 for Figure 6.23 except for its respective percentages.

Figure 6.23 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.6, which classifies the MCDE model test1 150MeV 100kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. We can observe that the maximum classification in each class is the input class itself in dark green.

We can make the same conclusions from Figure 6.2 for Figure 6.24 except for its respective percentages.

Figure 6.24 is a confusion matrix created from our fully connected network, whose parameters are displayed in Table 6.6, which classifies the MCDE model test1 150MeV 180kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. We can observe that the maximum classification in each class is the input class itself in dark green.

Validation Rate	Dropout Rate	Neurons per Layer	Layers	Batch Size	Epochs
0.2	0	256	64	8192	1024

Table 6.6: Hyperparameters for above study

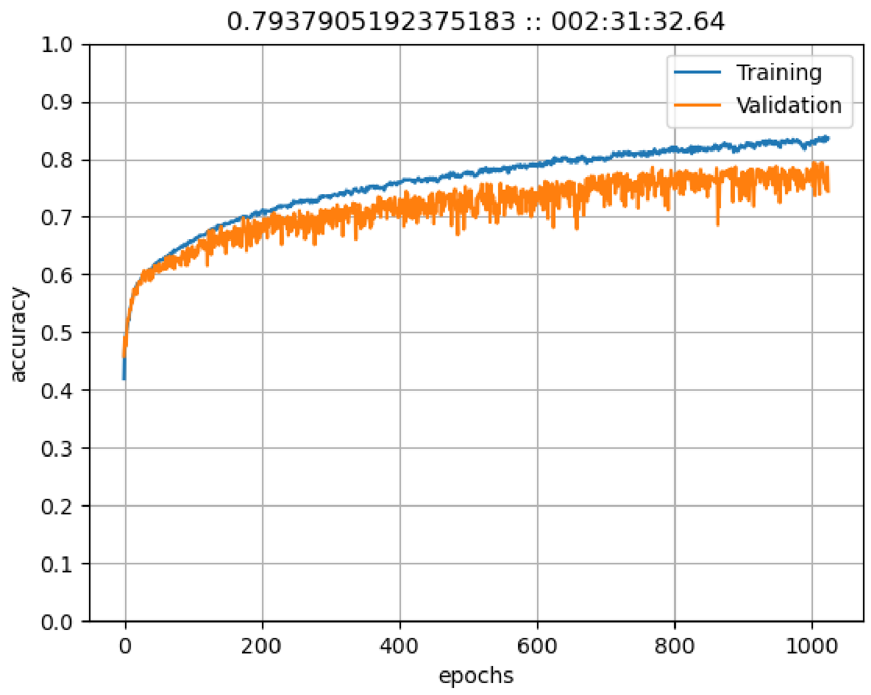

Figure 6.21: Training and validation plot for Study 238 using a fully connected network

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	62.8	6.8	8.6	2.2	2.4	1.5	5.2	2.9	0.3	0.1	4.9	1.3	1.1
132	7.8	65.9	2.2	2.2	4.8	2.1	0.4	0.1	5.6	1.6	3.0	3.4	0.9
213	3.6	3.8	66.8	5.8	2.6	1.4	1.0	7.4	4.1	2.1	0.4	0.2	1.0
231	2.8	4.3	5.8	67.5	2.2	2.5	0.0	0.6	1.9	3.1	7.6	1.1	0.5
312	4.2	4.1	3.3	3.1	65.4	5.0	3.1	2.9	0.9	6.9	0.1	0.5	0.5
321	4.8	3.9	4.2	11.0	6.8	53.2	1.9	4.6	0.0	0.5	3.1	5.1	0.9
124	4.6	0.4	1.5	0.2	3.5	1.6	49.8	27.7	0.5	0.6	0.6	1.3	7.8
214	1.1	0.3	4.9	0.4	2.3	2.1	9.8	70.7	0.4	1.0	0.7	0.6	5.8
134	0.5	5.9	2.9	2.1	1.1	0.1	0.4	0.9	58.2	17.4	1.6	1.2	7.6
314	0.1	1.5	2.0	5.0	4.9	0.5	0.9	1.6	12.0	64.5	0.6	0.6	5.9
234	3.2	2.3	0.1	6.8	0.1	0.5	0.4	1.0	1.3	0.9	65.5	8.9	9.0
324	1.7	3.7	0.1	2.3	0.3	4.1	1.4	0.7	0.6	0.9	31.0	45.4	7.8
444	0.6	0.9	0.3	0.6	0.0	0.6	5.0	7.2	3.1	4.1	4.4	8.2	64.9

Figure 6.22: Confusion Matrix for Study 238 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 020K beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	63.9	6.4	8.2	2.5	1.6	1.6	5.2	2.6	0.3	0.0	5.4	1.0	1.2
132	6.7	65.1	2.9	2.5	4.6	1.7	0.2	0.0	6.7	2.8	2.8	3.3	0.8
213	4.3	4.0	62.4	7.6	2.6	1.5	1.0	7.2	5.7	1.8	0.7	0.1	1.1
231	2.8	4.1	5.9	69.9	1.7	2.0	0.0	0.4	2.6	3.0	6.2	0.6	0.8
312	3.5	3.8	3.4	3.1	67.0	6.3	2.0	2.6	0.7	6.5	0.2	0.7	0.2
321	5.0	4.1	5.6	10.8	7.8	51.4	1.9	5.3	0.1	0.4	2.6	4.3	0.7
124	6.3	0.2	1.1	0.1	3.7	1.8	52.8	24.2	0.5	0.5	0.4	1.3	7.0
214	1.2	0.1	4.9	0.2	1.9	2.6	8.8	70.5	0.5	1.2	0.8	0.8	6.5
134	0.4	6.1	2.6	2.1	1.3	0.0	0.7	0.7	57.0	19.2	1.8	0.5	7.7
314	0.1	0.9	2.5	3.9	5.0	0.1	0.3	1.6	11.6	65.8	0.5	1.4	6.1
234	3.0	2.8	0.5	6.5	0.1	0.5	0.6	0.9	1.2	0.5	66.0	8.8	8.5
324	2.2	3.2	0.2	2.7	0.5	3.5	1.1	0.7	0.6	0.8	29.2	46.1	9.1
444	0.6	0.5	0.1	0.9	0.6	0.2	4.7	6.4	4.1	5.3	4.4	4.2	68.0

Figure 6.23: Confusion Matrix for study 238 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 100K beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	66.3	7.2	5.5	2.9	1.4	1.2	4.3	2.6	0.7	0.0	6.2	1.2	0.2
132	7.7	70.0	1.7	1.2	4.6	1.9	0.0	0.2	4.6	1.2	2.6	3.4	1.0
213	3.1	3.9	64.8	6.5	2.4	1.0	1.2	8.4	4.3	2.9	0.5	0.0	1.0
231	2.7	5.1	8.2	63.6	3.1	2.2	0.2	0.7	1.9	2.9	6.7	1.4	1.2
312	3.6	4.8	1.9	2.7	65.3	6.3	1.9	3.9	0.5	8.0	0.0	0.5	0.7
321	4.8	3.6	3.4	8.2	7.2	55.2	2.7	7.2	0.0	0.0	2.9	3.6	1.2
124	5.2	0.5	1.3	0.1	2.5	1.7	50.8	26.9	0.5	0.3	0.3	1.6	8.3
214	1.2	0.1	4.8	0.6	1.5	2.9	10.2	69.5	0.3	1.2	1.1	0.4	6.2
134	0.2	6.5	2.7	2.7	1.5	0.2	0.7	0.8	58.5	17.1	1.2	0.9	7.1
314	0.1	0.8	1.2	3.3	4.6	0.4	0.5	1.9	11.4	67.3	0.6	1.0	7.0
234	3.8	1.8	0.5	4.9	0.2	0.9	0.6	0.9	1.0	0.2	67.8	7.6	9.9
324	2.1	3.5	0.1	1.9	0.4	3.9	1.4	0.6	0.2	1.0	30.2	44.9	9.8
444	0.5	0.6	0.3	0.7	0.4	0.3	5.4	5.5	4.1	5.7	6.0	5.1	65.4

Figure 6.24: Confusion Matrix for study 238 using fully connected network trained on triples, double to triples, and false data from a 150MeV over 1024 epochs. The testing data used is from the MCDE model test1 150MeV 180K beam.

6.2 Reducing the Hyperparameter Space to Enable Elongated Studies

In this section we used the results from Section 6.1 to pick a reduced set of hyperparameters for searching while using a larger amount of epochs. The idea is that we pick our best performers and give them more time to learn and possibly plateau. This should give us additional insights into what configurations are optimal for learning with a more compact network. Each of the studies are presented with a hyperparameter table, training and validation plot along with three confusion matrices that corresponds to different dosage rates of 20kMU, 100kMU, 180kMU.

A total of 12 studies were run based on a our reduced combination of hyperparameters. In each of our studies, the validation rate, the number of epochs, and the residual block size were held constant at 0.2, 4096, and 8 respectively, to act as a control. The normalization method used also stayed consistent with Section 6.1. Our varied hyperparameters such as dropout rate, number of neurons per layer, number of layers, and batch size were reduced in scope changed for each study. The values used for the changing variables are as follows:

- Drop out rate: 0.1
- Number of neurons per layer: 64, 128
- Number of layers: 64, 128
- Batch size: 2048, 4096, 8192

Thus, the total number of studies is (1)(2)(2)(3) = 12. For consistency when we refer to a study number we will be using the numbering scheme from the original 288 studies. The two most promising studies were studies 167 and 168 in Subsections 6.2.1 and 6.2.2, respectively. Each subsection contains a *hyperparameters table*, a *training and validation plot*, and three *confusion matrices* of the MCDE model test1 at 20kMU, 100kMU, and 180kMU dosage rates. The *hyperparameter tables* are Tables 6.7 and 6.8, and list validation rate, dropout rate, neurons per layers, number of layers, batch size, and epochs. The *training and validation plots* are provided in Figures 6.25, 6.29. In these training and validation plots, the accuracy, denoted as a decimal percent, is plotted against the number of epochs. The peak validation accuracy is displayed at the top of each plot in addition to the total wall clock time the study took to run. The blue line represents the training accuracy and the orange line represents the validation accuracy.

The three *confusion matrices* for each study are provided in Figures 6.26, 6.27, 6.28 for study 167, Figures 6.30, 6.27, 6.32 for study 168. In these confusion matrices, the left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. These percentages sum to 100% in each row. Each cell is colored from white to dark green proportional to the maximum value in the entire matrix. The darkest entry in each row is the dominant classification of the input class. Within the matrices, there are thirteen different labels: 123, 132, 213, 231, 312, 321, 124, 214, 134, 314, 234, 324, and 444. Label 123 represents the case where all three interactions are correctly ordered and no adjustments need to be made. Label 132 represents the case where the third interaction should happen second and the second interaction should happen third. Label 213 represents the circumstance the second interaction should happen first, the first should occur second and the third is correctly labeled. Label 231 represents the scenario where the third interaction should be first, the second interaction should be third and the first interaction should be second. The label 312 represents the case where the first interaction should be third, the second interaction first whereas the last interaction should be second. The label 321 is the scenario where the first interaction should happen third, the second is labeled correctly, and the third interaction should happen first. The label 124 indicates that the first two interactions are a correctly ordered double and the third interaction should be disposed of. The label 214 indicates that the first two interactions are a incorrectly ordered double and the third interaction should be disposed of. The label 234 indicates that the last two interactions are a correctly ordered double and the first interaction should be disposed of. The label 324 indicates that the last two interactions are a incorrectly ordered double and the first interaction should be disposed of. The label 134 indicates that the first and third interactions are a correctly ordered double and the second interaction should be disposed of. The label 314 indicates that the first and third interactions are a incorrectly ordered double and the second interaction should be disposed of. The thirteenth label 444, is a false event, where the three interactions should be thrown away.

6.2.1 Study 167 with 4096 Epochs

Figure 6.25 is a training and validation plot for our fully connected network, whose parameters are displayed in Table 6.7. As we can see at 0 epochs we have a validation accuracy of a little less than 40%, then we have a sudden spike to a little over 60% at around 100 epochs. After that, the network seems to learn steadily with a slight increase over 3000 epochs, as it drops more neurons. There are sudden dips around 3000 epochs, where the training data is smaller than the validation data. Figure 6.26 is a confusion matrix created from our fully connected networkclassifying the MCDE model test1 150MeV 20kMU beam. The left most column is the correct input class and the percentage in the other columns represents the amount of data put into the class at the top of the column. We can observe that the maximum classification in each class is the input class itself in dark green. The accuracies for the true triples range from 70% to 78%, the doubles-to-triples' accuracies range from 62% to 75%, and the false events are $\approx 66\%$. Which means that when given any input class the will classify it the majority of the time. However there is still some discussion to be had about the dominant misclassifications of events. For the true triples, the matrix shows that the second-highest classification accuracies are either a true triple, or the corresponding double-to-triple event. This means that the network recognizes true events but struggles to reorder the events correctly; other times, it is able to identify the ordering of the first two interactions but improperly decouples the third interaction. Both these cases could cause pollution in the data used for reconstruction. We also observe that the true triples are barely classified as false events. For the doubles-to-triples, we can see that the second-highest percentage is another double-to-triple event. In most cases we see that it correctly determines the falsely coupled interaction but cannot determine the correct the order of the remaining pair. Even so there are some still some cases where doubles-to-triples are also classified as true triples. The presence of this extra interaction will cause noise during reconstruction. We observe that the false events are classified correctly most of the time but occasionally they are classified as doubles-to-triples. Any false data that leaks into the reconstruction data will always cause noise during reconstruction.

This study is one of the promising of the 288 baseline hyperparameter studies. In Figure 6.1, we trained a deep fully connected network using the hyperparameters shown in the Table 6.1. We can see that both the training and validation accuracies start at 40% at 0 epochs, then proceed to increase significantly in the first few epochs, where both accuracies increase to 60%. Then, the slope becomes flatter as the network struggles to learn more information about the data.

By choosing a study whose dropout rate was greater than 0 we fixed the discrepancy between validation and testing accuracy seen in Section 6.1. Even when only using 0.1 dropout, just a small amount of dropout has brought our confusion matrices' accuracies much closer to our validation accuracy seen in the training and validation plots. If we had picked something larger we may have not seen any favorable results within 4096 epochs. The triples now have comparable accuracy to a more complex network but the doubles-to-triples and false data are still 1% to 7% worse than the more complex network. This is an improvement over our previously trained networks which used no dropout rate. Our training time for these studies are high but we cannot locate bugs within our code or TensorFlow. The results are unaffected by this increase in training time. Currently this network configuration still does not have the classification performance for real-world use but is with more tuning could approach the lower bounds.

Validation Rate	Dropout Rate	Neurons per Layer	Layers	Batch Size	Epochs
0.2	0.1	128	128	8192	4096

Table 6.7: Hyperparameters for study 167

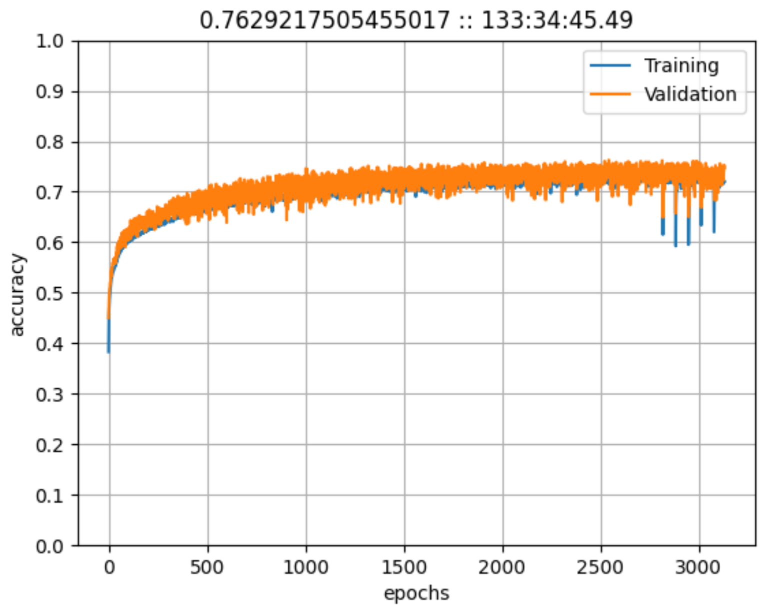

Figure 6.25: Training and validation plot for Study 167 using a fully connected network

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	77.0	6.4	1.4	1.9	1.9	2.0	5.0	0.4	0.1	0.0	2.5	1.0	0.3
132	3.8	72.8	1.9	2.4	6.4	2.7	0.2	0.0	5.1	1.0	1.0	2.6	0.2
213	3.1	2.5	71.3	9.1	1.8	2.5	0.5	5.0	2.1	1.7	0.1	0.0	0.2
231	2.5	2.6	2.9	73.5	2.4	8.0	0.0	0.1	0.9	2.1	3.4	1.4	0.2
312	3.4	1.8	1.9	2.9	77.1	3.2	2.3	1.3	0.6	5.3	0.0	0.1	0.1
321	2.6	3.0	3.1	2.1	3.4	77.8	0.6	2.1	0.0	0.1	0.6	4.3	0.3
124	5.0	0.4	1.0	0.1	3.7	2.1	68.9	11.1	0.6	0.6	0.4	1.3	4.7
214	0.8	0.3	5.4	0.6	1.8	4.0	7.0	74.3	0.3	1.0	0.3	0.4	3.8
134	0.7	4.5	2.6	2.8	1.1	0.2	0.6	0.2	63.8	18.1	1.3	0.8	3.3
314	0.1	0.7	1.8	5.0	6.1	0.4	0.6	1.2	6.8	72.1	0.2	0.8	4.2
234	3.0	2.3	0.1	6.5	0.1	1.5	0.5	0.8	1.1	0.6	62.3	16.7	4.5
324	1.5	4.5	0.1	0.6	0.3	7.2	0.9	0.4	0.5	0.8	7.8	72.0	3.5
444	0.6	0.6	0.3	0.9	0.0	0.6	4.1	5.3	4.7	3.8	5.6	7.5	65.8

Figure 6.26: Confusion Matrix for new Study 167 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 4096 epochs. The testing data used is from the MCDE model test1 150MeV 20K beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	77.0	7.2	1.8	1.7	1.5	1.7	3.9	0.8	0.0	0.0	2.5	1.6	0.2
132	3.5	73.1	1.5	2.2	6.3	1.5	0.0	0.0	6.5	1.3	0.9	2.7	0.3
213	4.3	1.9	66.6	12.5	2.3	1.9	0.7	5.0	2.3	1.9	0.2	0.1	0.1
231	1.8	2.6	2.5	76.5	1.7	8.5	0.1	0.0	1.1	1.5	2.5	1.1	0.1
312	2.9	1.3	2.8	2.8	78.0	3.8	2.0	1.0	0.4	4.8	0.0	0.1	0.1
321	2.6	3.2	3.6	2.2	3.6	76.7	0.8	3.1	0.0	0.2	0.3	3.5	0.3
124	6.7	0.3	0.6	0.0	3.2	2.1	71.0	9.8	0.7	0.4	0.4	1.0	3.7
214	1.0	0.1	5.6	0.6	1.6	4.6	7.2	73.0	0.5	0.9	0.3	0.3	4.4
134	0.3	5.5	2.8	2.2	0.8	0.0	0.5	0.5	65.3	16.3	1.3	0.2	4.2
314	0.1	0.7	1.6	4.2	5.6	0.3	0.2	0.7	7.3	73.7	0.4	0.9	4.5
234	3.9	2.4	0.5	5.2	0.2	1.9	0.4	0.5	0.8	0.2	61.0	18.1	4.7
324	1.3	2.8	0.1	1.0	0.5	7.1	0.6	0.5	0.6	0.5	7.1	72.4	5.5
444	0.8	0.3	0.3	0.8	0.6	0.8	4.1	4.7	5.5	5.0	4.1	4.4	68.6

Figure 6.27: Confusion Matrix for new Study 167 using a fully connected network trained on triples, double to triples,and false data from a 150MeV over 4096 epochs. The testing data used is from the MCDE model test1 150MeV 100K beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	75.5	8.7	1.7	1.4	1.9	1.9	4.3	0.5	0.0	0.0	2.6	1.4	0.0
132	2.4	76.9	1.9	1.7	5.3	1.7	0.0	0.0	4.1	0.5	0.7	4.3	0.5
213	2.4	2.2	73.5	9.2	1.4	1.0	1.0	6.0	2.2	1.2	0.0	0.0	0.0
231	2.2	2.7	2.9	71.8	3.4	8.7	0.0	0.2	1.0	2.2	3.6	1.2	0.2
312	2.9	2.4	1.4	1.9	77.8	3.4	2.4	1.0	0.2	5.8	0.0	0.5	0.2
321	1.7	2.7	3.6	1.9	3.6	78.6	1.2	3.1	0.0	0.0	1.0	2.7	0.0
124	6.3	0.4	0.3	0.2	3.3	2.2	70.1	10.2	0.5	0.0	0.4	1.0	5.2
214	1.3	0.0	4.8	0.6	1.4	4.4	6.6	74.5	0.4	1.2	0.2	0.5	4.3
134	0.4	4.7	2.2	2.9	1.7	0.1	0.6	0.4	64.7	17.4	1.3	0.6	3.0
314	0.2	0.5	1.0	4.0	5.8	0.4	0.2	1.2	7.4	73.8	0.2	0.8	4.5
234	3.1	2.0	0.3	4.8	0.3	2.0	0.1	0.5	0.7	0.4	62.5	18.3	4.9
324	2.3	3.6	0.1	0.7	0.2	6.6	1.0	0.2	0.6	0.3	7.0	72.1	5.2
444	0.5	1.0	0.2	0.7	0.4	0.5	4.4	4.8	5.6	5.1	6.2	5.0	65.5

Figure 6.28: Confusion Matrix for new Study 167 using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 4096 epochs. The testing data used is from the MCDE model test1 150MeV 180K beam.

6.2.2 Expansion of Study YYY

The hyperparameter values for the new Study YYY can be seen in the Table 6.8 Using this set of parameters, we found that both the training and the validation plots are increasing over the 4096 epochs. The peak validation accuarcy is 0.749. Figure 6.30, Figure 6.31, Figure 6.32 are confusion matrices of classification accuracies. The network is correctly classifying of all events between 67% to 77% for 20KMU, 70% to 77% for 100kMU, and 68% to 79% for 180kMU. The accuracy is low ranging from 67% to 79% and needs improvements. The network performs well on the validation data but not as well on the testing data.

Validation Rate	Dropout Rate	Neurons per Layer	Layers	Batch Size	Epochs
0.2	0.1	64	128	8192	4096

Table 6.8: Hyperparameters for the expansion of study YYY

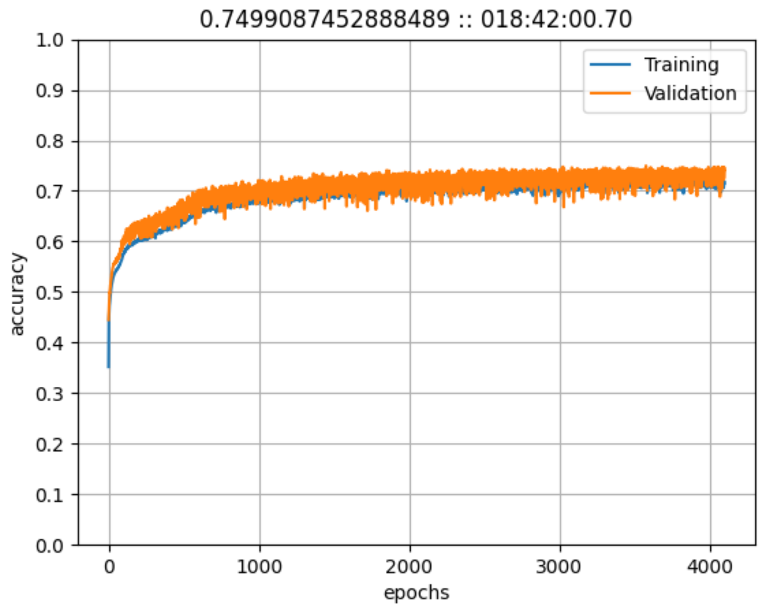

Figure 6.29: Training and validation plot for the expansion of study YYY using a fully connected network

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	74.7	4.9	2.1	2.2	2.3	2.0	6.4	0.5	0.1	0.0	3.1	1.2	0.3
132	4.7	74.4	2.3	2.0	2.4	2.6	0.2	0.0	6.1	0.4	1.2	3.1	0.4
213	2.2	2.9	77.0	3.7	1.4	2.3	0.7	5.3	2.9	1.2	0.1	0.0	0.2
231	2.9	2.3	4.3	74.8	2.8	3.6	0.0	0.2	0.9	2.3	4.9	0.8	0.2
312	3.2	2.4	1.9	2.3	75.3	4.7	2.7	1.3	0.6	5.4	0.0	0.0	0.1
321	2.2	3.2	3.5	2.4	3.1	76.4	1.0	2.2	0.0	0.2	0.6	5.0	0.3
124	4.0	0.3	0.9	0.3	3.7	1.8	70.3	11.0	0.5	0.4	0.4	1.0	5.5
214	0.8	0.1	4.7	0.3	1.8	3.7	7.3	74.3	0.3	1.4	0.6	0.4	4.4
134	0.5	4.8	3.7	2.0	0.5	0.1	0.8	0.3	72.7	8.6	1.3	0.7	4.1
314	0.0	0.6	2.4	4.7	6.6	0.4	0.6	0.9	6.6	71.5	0.2	0.8	4.8
234	3.1	1.5	0.1	6.5	0.1	1.1	0.6	0.9	1.3	0.7	70.3	8.9	4.9
324	1.3	3.5	0.1	0.5	0.4	6.3	1.2	0.3	0.4	0.8	8.8	72.9	3.5
444	0.3	0.9	0.3	0.9	0.0	0.3	5.0	5.0	3.4	4.1	6.0	6.0	67.7

Figure 6.30: Confusion Matrix for new Study YYY using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 4096 epochs. The testing data used is from the MCDE model test1 150MeV 20K beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	77.2	4.7	2.0	2.2	1.5	1.5	5.1	0.8	0.0	0.0	3.4	1.2	0.2
132	4.5	74.8	2.5	1.7	2.3	1.5	0.0	0.0	7.4	0.8	1.6	2.5	0.4
213	2.8	3.0	74.3	5.6	2.7	1.6	0.8	4.7	2.9	1.2	0.2	0.1	0.2
231	2.1	2.1	4.4	77.2	2.0	4.6	0.0	0.1	0.9	1.6	3.9	0.5	0.4
312	2.7	1.9	2.8	2.6	75.9	5.5	1.9	1.7	0.3	4.5	0.0	0.1	0.1
321	1.9	3.2	4.2	3.0	3.9	75.2	1.3	2.5	0.0	0.4	0.4	3.7	0.4
124	5.3	0.4	0.5	0.1	2.9	1.9	71.4	10.8	0.5	0.4	0.4	0.8	4.6
214	1.0	0.0	5.7	0.6	1.6	4.3	8.0	71.8	0.4	1.0	0.4	0.4	4.9
134	0.1	5.3	3.1	1.8	0.6	0.1	0.4	0.4	71.8	9.9	1.0	0.3	5.2
314	0.1	0.2	1.8	4.3	5.8	0.3	0.2	0.7	7.4	73.3	0.3	0.7	4.7
234	3.8	1.3	0.4	5.7	0.1	0.6	0.4	0.4	0.8	0.3	71.7	9.4	5.0
324	1.0	2.8	0.2	1.2	0.5	6.1	0.9	0.4	0.6	0.4	8.7	71.4	5.8
444	0.8	0.1	0.4	0.8	0.6	0.5	3.7	4.4	4.7	4.9	4.1	4.6	70.5

Figure 6.31: Confusion Matrix for new Study YYY using a fully connected network trained on triples, double to triples, and false data from a 150MeV over 4096 epochs. The testing data used is from the MCDE model test1 150MeV 100K beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	75.2	6.2	1.9	1.7	1.4	2.2	6.0	0.7	0.0	0.0	2.6	1.9	0.0
132	3.4	77.2	1.9	1.4	1.7	1.7	0.0	0.0	5.5	0.2	0.5	6.0	0.5
213	2.7	2.9	76.4	4.1	1.7	1.2	0.0	7.5	2.7	1.0	0.0	0.0	0.0
231	2.2	2.7	4.6	71.8	4.8	4.3	0.0	0.0	1.0	2.2	4.8	1.2	0.5
312	2.7	3.4	1.2	1.2	77.1	4.8	1.9	1.4	0.7	4.8	0.0	0.0	0.7
321	1.9	2.7	3.6	2.2	2.2	79.3	1.2	3.4	0.0	0.0	0.5	3.1	0.0
124	5.5	0.3	0.5	0.2	2.4	1.9	71.9	10.0	0.5	0.2	0.4	0.8	5.6
214	1.0	0.0	5.2	0.6	1.6	3.7	7.3	73.9	0.3	0.9	0.3	0.5	4.7
134	0.4	5.4	3.0	1.7	0.9	0.0	0.5	0.3	73.8	8.7	1.2	0.6	3.5
314	0.1	0.2	1.0	3.7	6.3	0.4	0.3	1.0	6.8	74.1	0.4	0.4	5.3
234	3.1	1.3	0.6	5.4	0.2	1.0	0.2	0.2	0.6	0.2	72.0	9.5	5.7
324	1.9	3.2	0.1	0.6	0.3	6.0	1.2	0.2	0.4	0.6	9.1	70.6	5.7
444	0.4	0.6	0.4	0.6	0.3	0.4	4.6	4.0	4.6	5.0	5.7	4.6	68.8

Figure 6.32: Confusion Matrix for new Study YYY using a fully connected network trained on triples, double to triples,and false data from a 150MeV over 4096 epochs. The testing data used is from the MCDE model test1 150MeV 180K beam.

7 Ongoing Work and Preliminary Results

7.1 Normalization

After analyzing our promising studies, we ran a similar set of studies using different normalization methods compared to the initial 288 studies. In this new set of studies, validation rate and epochs were held constant at 0.2 and 256, respectively. The number of layers, neurons, batch size, dropout rate, and normalization methods were the variables changed The values for the changing variables are as follows:

- Number of layers: 128, 256
- Number of neurons: 64, 128
- Batch size: 2048, 4096, 8192
- Dropout rate: 0, 0.1

Based on the initial 288 studies we plan to create four different normalization methods. To do this we have decided to separate the spatial data and the energy data. This was done because the spatial data and the energy data have different ranges and units. All functions were derived from the sklearn library and are variations of the original normalization method. The first normalization method square roots all energy values and proceeds to use the MinMaxScaler function from the sklearn library on all energy values. It then uses the StandardScaler function from the sklearn library on the spatial data. The second normalization method uses the MinMaxScaler function on both the energy and spatial data separately. The third normalization method applies the PowerTransformer function from the sklearn library on both the energy and spatial data separately. Finally, the fourth normalization method utilizes a log function from the numpy library to transform the energy values and then uses the PowerTransformer function. It then applies the StandardScaler function on the spatial values.

Based on the above transformations and new parameters, we were able to generate 96 new studies. From these new studies we looked at the two with the highest accuracy. Each study is presented below with a hyperparameters table and a graph of it's validation and testing accuracy.

7.1.1 Normalization Results 1

The hyperparameter values for the Study are shown in the Table 7.1. Using this set of parameters, we found that the validation accuracy is still increasing, but it is still lower than our previous studies. The final validation accuracy is 0.7312, which is not significantly different than previous results.

Validation Rate	Dropout Rate	Neurons per Layer	Layers	Batch Size	Epochs	Normalizer
0.2	0	128	256	4096	256	Normalizer1

Table 7.1: Hyperparameters for study

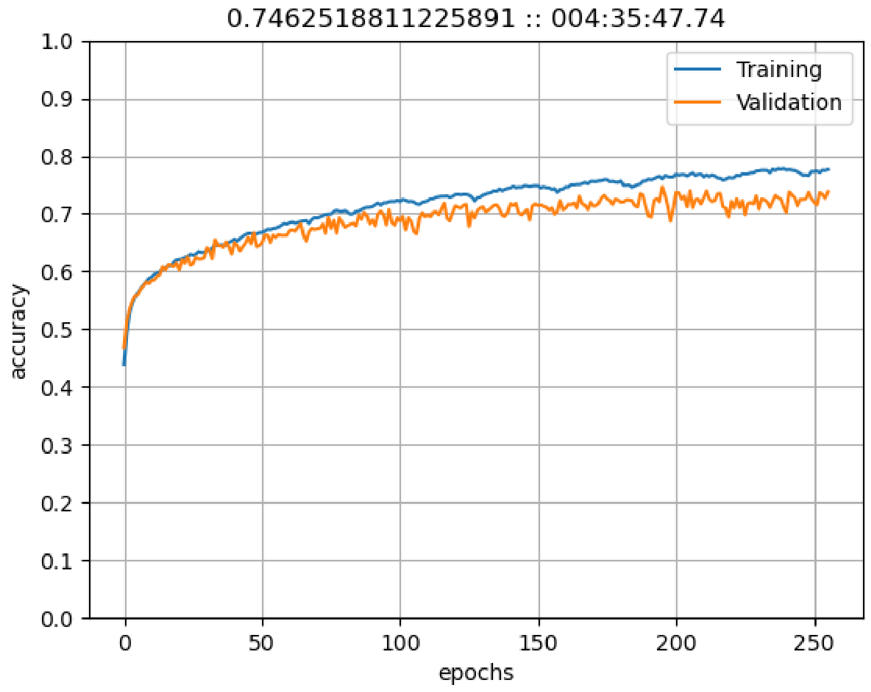

Figure 7.1: Training and validation plot for Study 1 using a fully connected network

7.1.2 Normalization Results 2

The hyperparameter values for the Study are shown in the Table 7.2. Using this set of parameters, we found that the validation accuracy is still increasing, but it is still lower than our previous studies. The final validation accuracy is 0.723, which is not significantly different than previous results.

Validation Rate	Dropout Rate	Neurons per Layer	Layers	Batch Size	Epochs	Normalizer
0.2	0	128	128	4096	256	Normalizer1

Table 7.2: Hyperparameters for study

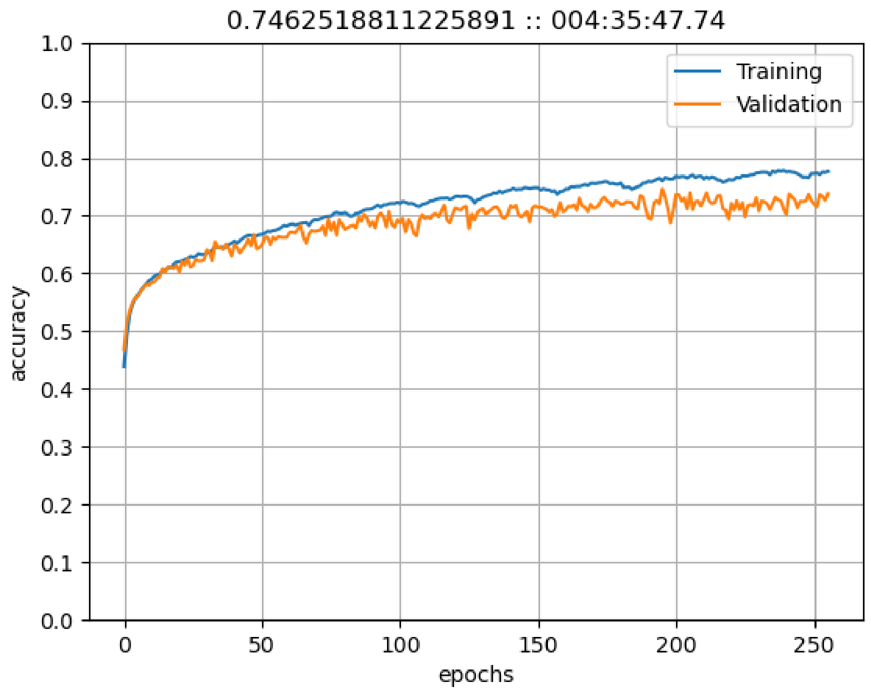

Figure 7.2: Training and validation plot for Study 1 using a fully connected network

7.2 Optimization

With the goal of improving the accuracy of our studies we proceed to change their optimization and momentum. Whereas the original studies use adam optimizer, which is the default optimizer in the tensorflow library, we changed it to either stochastic gradient descent (sgd) or nadam which are both from the tensorflow library. All other parameters of the studies remain the same as before.

- Number of layers: 128, 256
- Number of neurons: 64, 128
- Batch size: 2048, 4096, 8192
- Dropout rate: 0, 0.1
- Normalizers: 1,2,3,4,*Original*
- Optimizers: sgd,nadam

With the combination of all these new parameters we were able to generate 240 new studies. From those 240 studies we looked at the two with the highest accuracy. Both of the studies with the highest accuracy used nadam as the optimizer. Each study is presented with a hyperparameters table and a graph of it's validation and testing accuracy.

7.2.1 Optimization Results 1

The hyperparameter values for the Study are shown in the Table 7.3. Using this set of parameters, we found that the validation accuracy is still increasing, but it is still lower than our previous studies. The final validation accuracy is 0.7303, which is not significantly different than previous results.

Validation Rate	Dropout Rate	Neurons per Layer	Layers	Batch Size	Epochs	Normalizer
0.2	0	128	128	4096	256	Normalizer4

Table 7.3: Hyperparameters for study

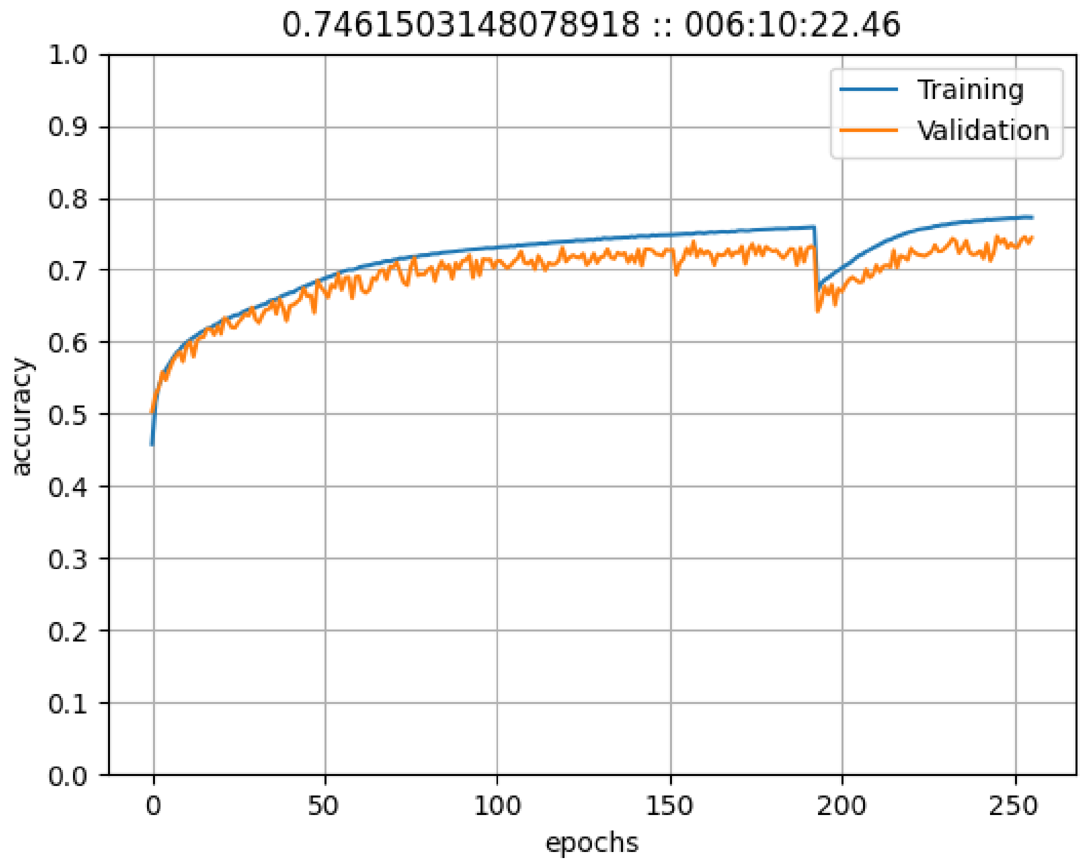

Figure 7.3: Training and validation plot for Study 1 using a fully connected network

7.2.2 Optimization Results 2

The hyperparameter values for the Study are shown in Table 7.4. Using this set of parameters, we found that the validation accuracy is still increasing, but it is higher than the training accuracy which could be caused by over fitting. The final validation accuracy is 0.7303, which is not significantly different than previous results.

Validation Rate	Dropout Rate	Neurons per Layer	Layers	Batch Size	Epochs	Normalizer
0.2	0	128	128	4096	256	Normalizer3

Table 7.4: Hyperparameters for study

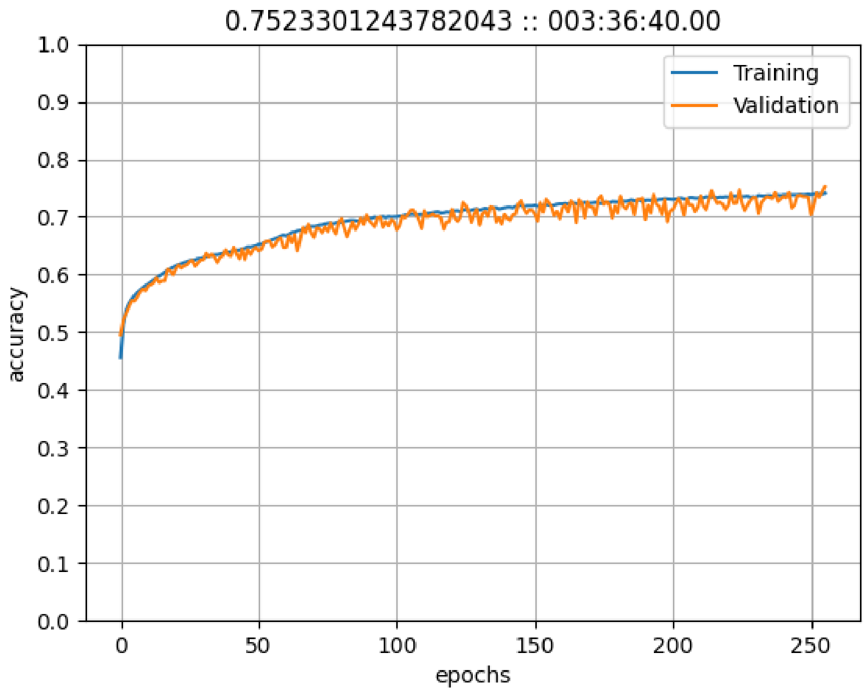

Figure 7.4: Training and validation plot for Study 1 using a fully connected network

7.3 Recurrent Neural Networks

In order to improve the accuracy of our estimates, we decided to create a new recurrent neural network (RNN). In these studies we plan to change the same parameters we had in the previous studies with the exception of the dropout rate. In addition to those parameters we will also be changing the learning rate and the layers. The range of each parameter was based on what we have learned and believe will give us the best results. These new parameters will generate a total of 960 new studies. The values for the new variables will be as follows.

- Number of layers: 1, 2, 4, 8
- Number of neurons: 64, 128,256
- Batch size: 2048, 4096
- Layer: GRU, LSTM
- Learning rate: 0.001, 0.0001
- Normalizers: 1, 2, 3, 4, *Original*
- Optimizers: adam, nadam

Out of 960 studies we looked at the two studies with the highest validation accuracy. Both of the studies with the highest accuracy used GRU layers, 128 neurons per layer, 2048 batch size, and Normalizer1. The validation rate was 0.2, dropout rate was 0, and we run for epochs 512 for all studies. Each study is presented with a hyperparameters table and a graph of it's validation and training accuracy.

7.3.1 RNN Results 1

The hyperparameter values for the Study are shown in the Table 7.5. Using this set of parameters, we found that the validation accuracy is still increasing and better compared to other RNN studies. The final validation accuracy is 0.7217, which is not significantly different than previous results.

Learning Rate	Neurons per Layer	Num of Layers	Layer	Batch Size	Optimizer	Normalizer
0.0001	128	4	GRU	2048	nadam	Normalizer1

Table 7.5: Hyperparameters for study

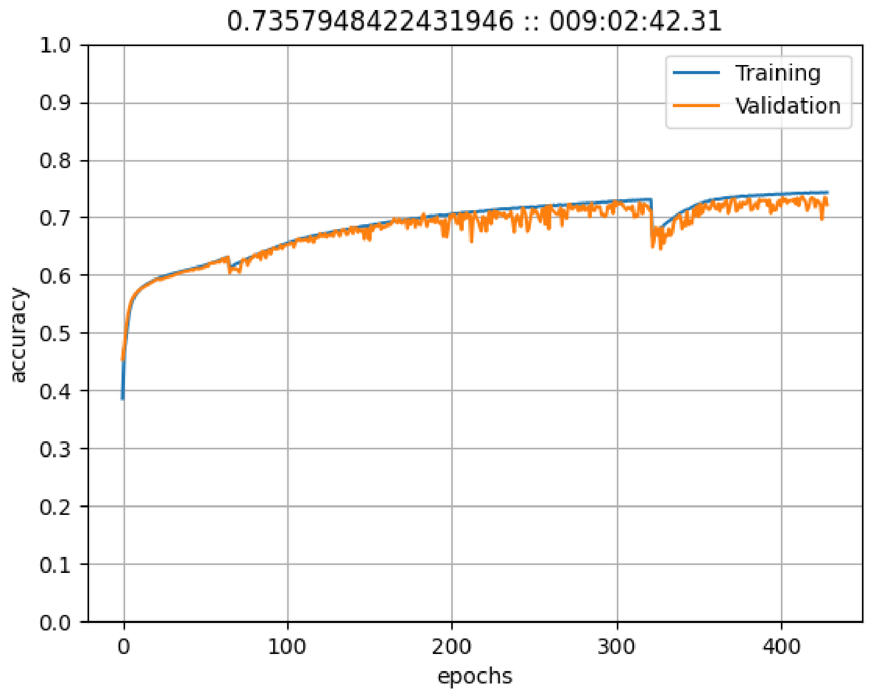

Figure 7.5: Training and validation plot for RNN Result 1 using a recurrent neural network

7.3.2 RNN Results 2

The hyperparameter values for the Study are shown in the Table 7.6. Using this set of parameters, we found that the validation accuracy is still increasing and better compared to other RNN studies. The final validation accuracy is 0.716, which is not significantly different than previous results.

Learning Rate	Neurons per Layer	Num of Layers	Layer	Batch Size	Optimizer	Normalizer
0.001	128	2	GRU	2048	adam	Normalizer1

Table 7.6: Hyperparameters for study

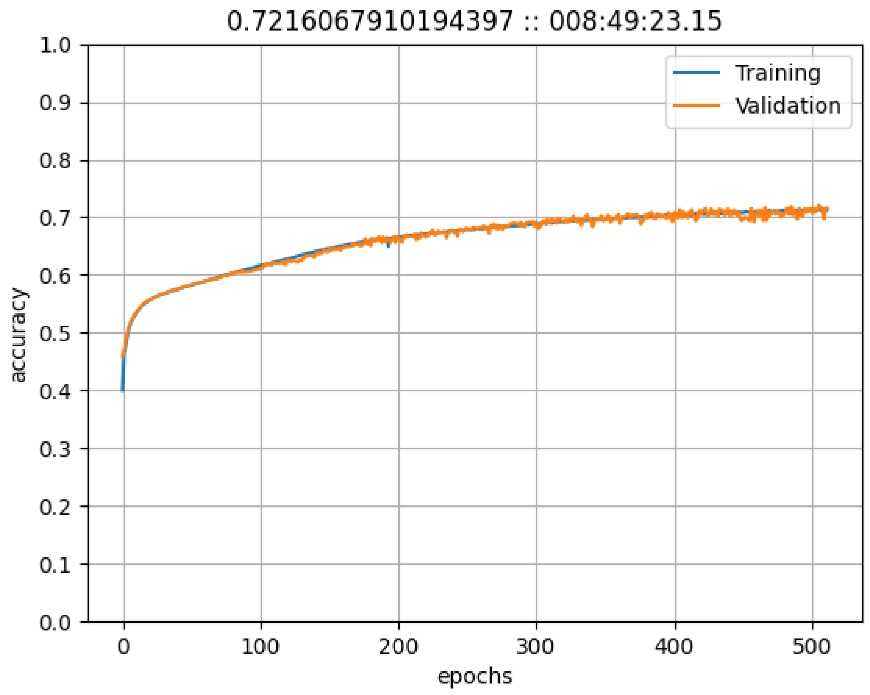

Figure 7.6: Training and validation plot for RNN Result 2 using a recurrent neural network

8 Conclusions and Future Work

Section 6 contains the highlights of our attempts to create a more compact neural network than is seen in [1]. We define a "compact network" as a network with similar or superior performance but contains less total parameters and less neurons the previous networks with similar function. The less neurons the network has, the more computationally cheap it is to use, and the faster the network can classify records. If a network is sufficiently cheap one may not even need a GPU for real-world usage! Under this same idea we refer to the networks in [1] as "complex networks" because the networks contain significantly more neurons and parameters than the ones we are attempting to create here. To create a more compact network we conduct hyperparameter studies using the grid search method.

In Section 6.1 we have seen that our networks' ability to classify events of all input classes is good enough to warrant further experimentation but there are still many problems that plague us. All of our promising studies that used a dropout rate of 0 had high peak validation accuracies but this performance did not carry over into the testing data. For example when we examine the peak validation in Figure 6.1 we have 77%. Yet, when we look at the confusion matrices we see that correct classification percentages fall into the mid 60's to low 70's. When we look at our worst results with the worst validation accuracy, not listed under our results in this work, we see that the common factor is a dropout rate greater than 0. This is not a huge surprise as dropout rate is known to cause lower initial accuracy in exchange for better network generalization when trained for more epochs. When using a dropout rate of 0 we cannot take the training and validation plots as the definitive proof of performance. Instead we would need to run hyperparameters studies which include dropout rates only greater than 0 while providing enough epochs for learning to potentially plateau. The first 288 studies yielded high performance during training and validation but substandard performance during testing. Even if the performance of these networks were, in general, not high enough to replace those in past works, we can still use their outcomes to guide our future hyperparameter decisions when trying to do more searches in the future.

In Section 6.2 we attempted to demonstrate this by taking sets of hyperparameters that had the best performance and running them for longer with the hope that we may see improved testing performance. We also chose to only consider studies whose dropout rate was greater than 0 with the hope that this would help fix the discrepancy between validation and testing accuracy seen in the previous studies. In our extension studies we see that our best performing studies plateaued early in their training process. We see that the usage of even a small amount of dropout has brought our confusion matrices' accuracies much closer to our validation accuracy seen in the training and validation plots. The triples now have comparable accuracy to a more complex network but the doubles-to-triples and false data are still 1% to 7% worse than the more complex network. This is an improvement over our previously trained networks which used no dropout rate. Our training time for these studies was eerily high and we were not able to locate bugs or problems within TensorFlow; the results are unaffected by this strange increase in training time. The more compact networks show great promise in terms of achieving comparable accuracy to the best performing networks seen in [1]. Currently our networks still do not have the classification performance for real-world use. If we can tackle the long training times with more hyperparameter tuning then we can most likely create a network that is easier to train and cheaper to use than previous networks.

Particular studies, if given considerably more training time, could yield competitive, if not superior, testing accuracy to existing architectures while maintaining a simpler structure. In Section 7, based on these observations, we conducted experiments with (i) different normalizers, (ii) different optimizers, and (iii) initial tests with recurrent neural networks, (RNN). These studies did not yield significantly better results than the original studies, at least so far. In particular, more improvements are still needed for clinical use and we are currently experimenting with recurrent neural networks to test the viability of this type of architecture for this application.

Acknowledgments

This work is supported by the grant "REU Site: Online Interdisciplinary Big Data Analytics in Science and Engineering" from the National Science Foundation (grant no. OAC–2050943). This work is supported by the National Institutes of Health National Cancer Institute under award number R01CA187416. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health. Co-author Carlos Barajas additionally acknowledges support as HPCF RA. The hardware used in the computational studies is part of the UMBC High Performance Computing Facility (HPCF). The facility is supported by the U.S. National Science Foundation through the MRI program (grant nos. CNS–0821258, CNS–1228778, and OAC–1726023) and the SCREMS program (grant no. DMS–0821311), with additional substantial support from the University of Maryland, Baltimore County (UMBC). See hpcf.umbc.edu for more information on HPCF and the projects using its resources.

References

[1] Carlos A. Barajas, Gerson C. Kroiz, Matthias K. Gobbert, and Jerimy C. Polf. Deep learning based classification methods of Compton camera based prompt gamma imaging for proton radiotherapy. Technical Report HPCF–2021–1, UMBC High Performance Computing Facility, University of Maryland, Baltimore County, 2021.

[2] Carlos A. Barajas, Gerson C. Kroiz, Matthias K. Gobbert, and Jerimy C. Polf. Using deep learning to enhance Compton camera based prompt gamma image reconstruction data for proton radiotherapy. *Proc. Appl. Math. Mech. (PAMM)*, in press (2021).

[3] Jonathan N. Basalyga, Carlos A. Barajas, Matthias K. Gobbert, Paul Maggi, and Jerimy Polf. Deep learning for classification of Compton camera data in the reconstruction of proton beams in cancer treatment. *Proc. Appl. Math. Mech. (PAMM)*, 20(1):e202000070, 2021.

[4] Jonathan N. Basalyga, Carlos A. Barajas, Gerson C. Kroiz, Matthias K. Gobbert, Paul Maggi, and Jerimy Polf. Improvements to the deep learning classification of Compton camera based prompt gamma imaging for proton radiotherapy. Technical Report HPCF–2020–29, UMBC High Performance Computing Facility, University of Maryland, Baltimore County, 2020.

[5] Jonathan N. Basalyga, Gerson C. Kroiz, Carlos A. Barajas, Matthias K. Gobbert, Paul Maggi, and Jerimy Polf. Use of deep learning to classify Compton camera based prompt gamma imaging for proton radiotherapy. Technical Report HPCF–2020–14, UMBC High Performance Computing Facility, University of Maryland, Baltimore County, 2020.

[6] François Chollet. *Deep Learning with Python*. Manning Publications Co., 2018.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[8] Gerson C. Kroiz, Carlos A. Barajas, Matthias K. Gobbert, and Jerimy C. Polf. Exploring deep learning to improve Compton camera based prompt gamma image reconstruction for proton

radiotherapy. In *The 17th International Conference on Data Science (ICDATA'21)*, accepted (2021).

[9] Paul Maggi, Steve Peterson, Rajesh Panthi, Dennis Mackin, Hao Yang, Zhong He, Sam Beddar, and Jerimy Polf. Computational model for detector timing effects in Compton-camera based prompt-gamma imaging for proton radiotherapy. *Phys. Med. Biol.*, 65(12):125004, 2020.

[10] Enrique Muñoz, Ana Ros, Marina Borja-Lloret, John Barrio, Peter Dendooven, Josep F. Oliver, Ikechi Ozoemelam, Jorge Roser, and Gabriela Llosá. Proton range verification with MACACO II Compton camera enhanced by a neural network for event selection. *Sci. Rep.*, 11(1):9325, 2021.

[11] Jerimy C. Polf, Carlos A. Barajas, Gerson C. Kroiz, Stephen W. Peterson, Paul Maggi, Dennis S. Mackin, Sam Beddar, and Matthias K. Gobbert. A study of the clinical viability of a prototype Compton camera for prompt gamma imaging based proton beam range verification. In *AAPM Virtual 63rd Annual Meeting*, submitted (2021).

[12] Jerimy C. Polf and Katia Parodi. Imaging particle beams for cancer treatment. *Phys. Today*, 68(10):28–33, 2015.

[13] Andreas Zoglauer and Steven E. Boggs. Application of neural networks to the identification of the Compton interaction sequence in Compton imagers. In *IEEE Nuclear Science Symposium Conference Record*, 2007.