Sequence-Based Models for the Classification of Compton Camera Prompt Gamma Imaging Data for Proton Radiotherapy on the GPU Clusters Taki and Ada

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

Joseph Clark[1], Anaise Gaillard[2], Justin Koe[3], Nithya Navarathna[4],
Daniel J. Kelly[5], Matthias K. Gobbert[5], Carlos A. Barajas[5], Jerimy C. Polf[6]

[1]Department of Mathematics and Computing, Lander University
[2]Department of Computational and Data Sciences, George Mason University
[3]Department of Electrical Engineering, The Cooper Union
[4]Department of Biological Sciences, University of Maryland, Baltimore County
[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–2022–12, hpcf.umbc.edu > Publications

Proton beam 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 the proton beam can make this form of therapy more precise and safer for the patient during delivery. 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. Unfortunately, some of the Compton camera data is flawed and the reconstruction algorithm yields noisy and insufficiently detailed images to evaluate the proton delivery for the patient. Previous work used a deep residual fully connected neural network. The use of recurrent neural networks (RNNs) has been proposed, since they use recurrence relationships to make potentially better predictions. In this work, RNN architectures using two different recurrent layers are tested, the LSTM and the GRU. Although the deep residual fully connected neural network achieves over 75% testing accuracy and our models achieve only over 73% testing accuracy, the simplicity of our RNN models containing only 6 hidden layers as opposed to 512 is a significant advantage. Importantly in a clinical setting, the time to load the model from disk is significantly faster, potentially enabling the use of Compton camera image reconstruction in real-time during patient treatment.

1 Introduction

Because to its many advantages, proton beam therapy has gained popularity as a form of cancer treatment. Most types of radiation therapies work with the objective to damage the cellular DNA of target cancer cells that reside in the nucleus of every cell. X-ray therapy is able to deliver dosage at the tumor site, but its 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 beams have a finite range that can be controlled and they deposit the majority of their energy just before they stop. This sharp energy increase of the proton beam right before stopping is known as the Bragg peak. Since almost no radiation is delivered beyond the Bragg peak, healthy tissue can be spared from unnecessary radiation [9]. In order to take full advantage of these properties of proton therapy, we must have an efficient technique to image the prompt gamma rays produced by the beam in real-time as they travel through the patient's body. A Compton camera can be used to detect the prompt gamma rays emitted when the proton beam travels through the body, and an algorithm is available to reconstruct the beam's

image from the prompt gamma data, which then provides an indirect image of the proton beam. Unfortunately, a lot of the raw data of the Compton camera is flawed and the reconstruction algorithm yields noisy and insufficiently detailed images to evaluate the proton delivery for the patient [7, 8].

Machine learning can be used to clean the raw Compton camera data by identifying and removing false data before image reconstruction [7,8]. Research efforts to clean the Compton camera data have led to the use of neural networks. Shallow networks like the one in [7] use 1 to 2 hidden layers to perform simple classifications of simulated prompt gamma data under ideal conditions that do not represent the irradiation conditions encountered during clinical proton beam radiotherapy. This *shallow* network in [7] is a binary classification network that simply determines which event data are true events and should be used for reconstruction and which are false events that should not be used for reconstruction. This is improved upon in [8] using the *deep* residual fully connected neural network described in [3] for triple event classification. This neural network consists of 64 residual blocks with 8 fully connected layers per block yielding a total of 512 hidden layers. Each layer had 256 neurons per layer, a 45% dropout rate, and used leaky ReLU activation. More detailed results and discussions about the impact of neural network processing on the use and viability of Compton camera based imaging in clinical proton radiotherapy are the focus of [8], while providing details on the network and its training are the focus of [3]. The full capabilities of the described neural network are specified in [2], where preprocessing the data, all classification capabilities, and postprocessing output data are described in detail. Other recent work [1, 11] focused on hyperparameter studies on the deep residual fully connected neural network from [3], varying batch sizes, neurons, and layers. The use of recurrent neural networks (RNNs) is proposed in [1], since they use recurrence relationships in sequence data sets to make potentially better predictions. The potential for RNNs to be an improvement over feedforward neural networks (FNNs) is shown in [6].

In this work, we test RNN architectures using two different recurrent layers because of their potential for classifying sequence data, the Long Short-term Memory (LSTM) (discussed in Section 3.1.1) and the Gated Recurrent Unit (GRU) (discussed in Section 3.1.2). The LSTM uses memory cells with gates and a carry track to encode and learn from sequence data. The GRU uses two gating units to encode and learn from sequence data. We also examine transformer blocks (discussed in Section 3.2) which use multi-headed self-attention mechanisms and not recurrence on sequence data to help make predictions. The goal in this change in type of network architecture is to examine data as a sequence of interactions rather than one single event, but preliminary results do not show any benefit. We use models with 4 GRU layers and with 4 LSTM layers and achieve similar testing accuracy as the deep residual fully connected model from [3]. The model with 4 GRU layers outperforms the deep residual fully connected model in 3 classes but has a larger gap (within 10%) in accuracy in the other 10 classes. The model with 4 LSTM layer outperforms the previous deep residual fully connected model in only 2 classes but has a smaller gap (within 6%) in accuracy in the other 11 classes. Although the deep residual fully connected model achieves a slightly higher accuracy in nearly every class, the simplicity of our RNN models containing only 6 hidden layers (4 recurrent and 2 fully connected) as opposed to 512 is an advantage. And importantly in a clinical setting, the time to load the model from disk is significantly faster, potentially enabling the use of Compton camera image reconstruction in real-time during patient treatment.

The remainder of this report is organized as follows: Section 2 provides background on proton beam therapy to treat cancer and the use of a Compton camera to image promp gammas. Section 3 details the basics of machine learning and recurrent neural networks, while also providing details on the LSTM, GRU, and transformer networks. Section 4 outlines the hardware and software we use to carry out this research project. Sections 5 and 6 contain the results of our work. Section 7 contains our conclusions and future work.

2 Application Background

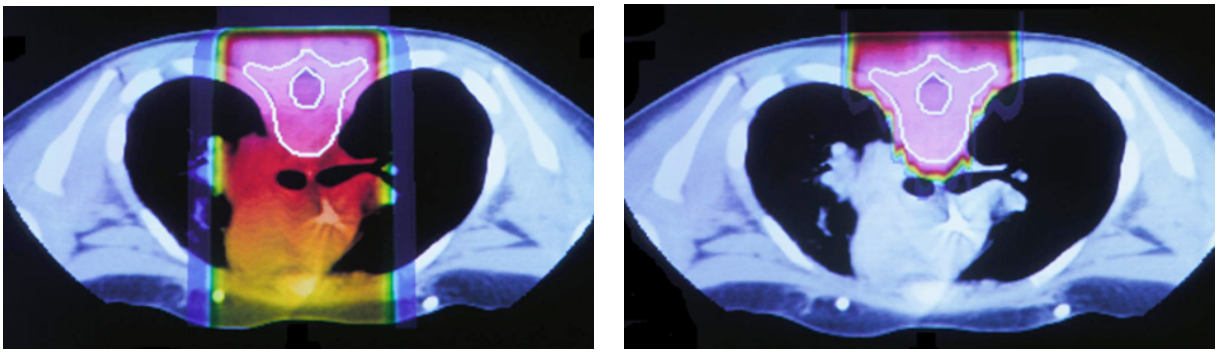
2.1 Proton Beam Therapy

Radiation therapy is a form of cancer treatment that uses high doses of radiation to kill cancer cells. X-ray therapy, a form of radiation therapy, is a common technique used for cancer treatment, where 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, which is another form of radiation therapy, 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) shows 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 problems. 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.

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 slightly 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



(a) (b)

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

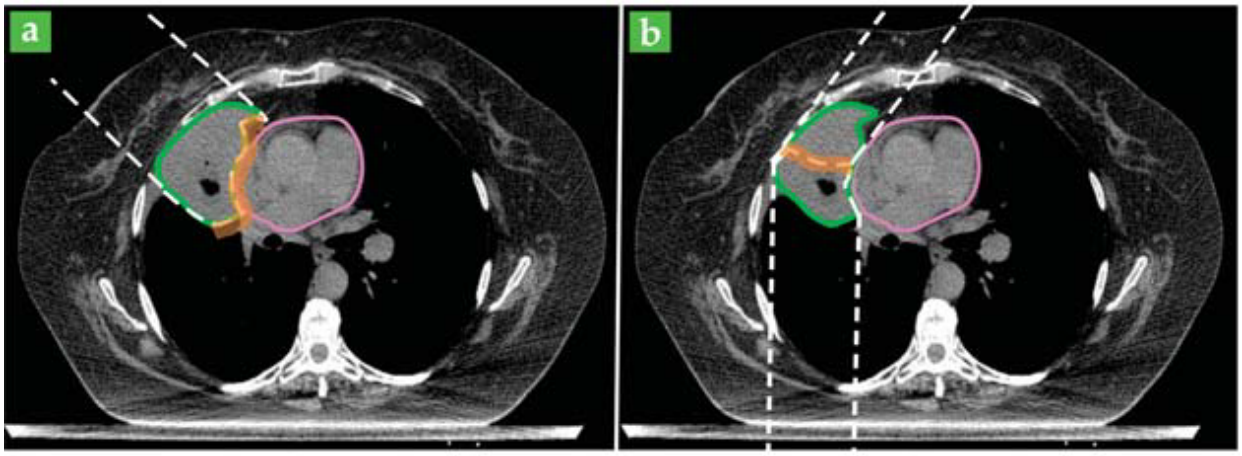


Figure 2.2: (a) Optimal proton beam trajectory. (b) Suboptimal trajectory necessary to protect the heart.

heart will more than likely receive a small dosage of radiation. To stay clear of radiating the heart, Figure 2.2 (b) suggests a sub-optimal 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.

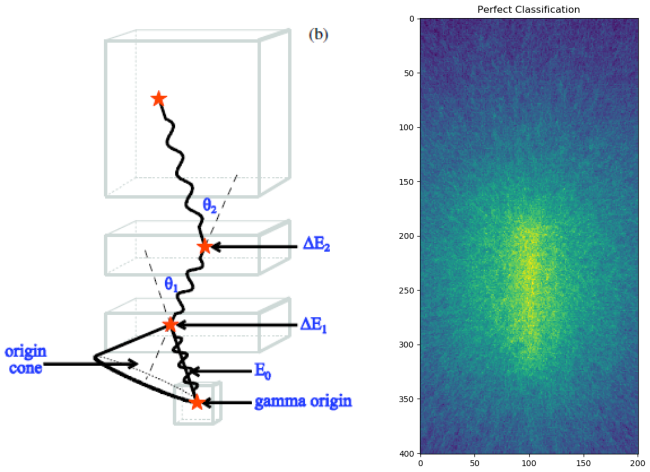
2.2 Compton Camera

The Compton camera is a multi-stage detector as shown in Figure 2.3 that produces data used to generate images of proton beams used in proton beam therapy [3]. As protons from the beam enter the body, they interact with cells in the body causing the emission of prompt gamma rays. Some of these gamma rays will collide with the Compton camera. An interaction is when a prompt gamma collides with a stage of the Compton camera. For each interaction, 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 $i = 1, 2, 3$ for the three stages of the Compton camera, 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 scan 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 [9]. 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.

Prompt gammas are emitted at speeds close to the speed of light consequently the camera is unable to detect the true ordering of interactions in an event. The false events cause noise in the image created impacting the usefulness of the image [3]. Next we describe the different type of Compton camera scatters.



(Phys. Med. Biol., 2010.)

(a) (b)

Figure 2.3: Depiction of Compton camera's multiple stages (a) and image reconstruction (b).

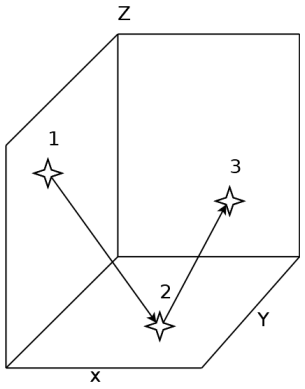
2.3 Scatter Types

There are three different types of scatter events: True Triples, Double-to-Triples (DtoT), and False triples [3].

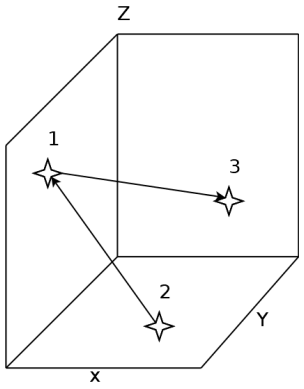
True Triples In the True Triples event, the Compton camera will detect the path of the prompt gamma as shown in Figure 2.4 (a). However, it is possible that the true path is the one seen in Figure 2.4 (b). There are a total of 6 total combinations of True Triple scatters: 123, 132, 213, 231, 312, 321 and, as the data stands, only the 123 ordering is usable.

Double-to-Triples (DtoT) In the DtoT event, the Compton camera will detect the path of a single prompt gamma as a true triple seen in Figure 2.4 (a). However, in reality, there were two prompt gammas who had varying paths as seen in Figure 2.4 (c). One prompt gamma could have detected as the first and third interaction and the second prompt gamma could have been mistaken as the second interaction. Similar to true triples, there are a total of 6 misdetection orderings: 124, 134, 214, 234, 324, 314. The second prompt gamma interaction is classified as "4" in the misdetection orderings. In this case, without processing the data, all 6 orderings are unusable.

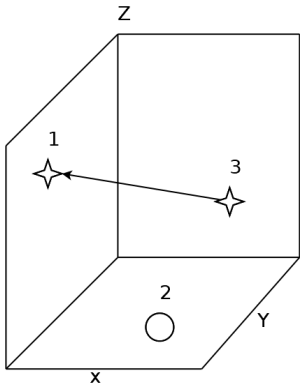
False Triples In a false triples event, the Compton camera will detect a true triple (Figure 2.4 (a)) whereas in reality, there were actually three different prompt gammas whose possible locations can be seen in Figure 2.4 (d). As a result, this entire event provides no insight into the path of a single prompt gamma and must be discarded.



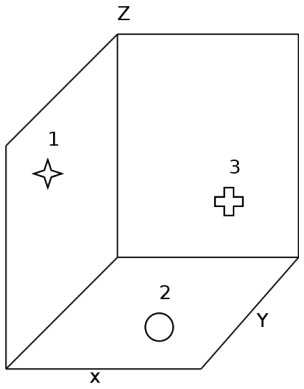
(a) True Triple scatter path of prompt gamma detected by Compton camera.



(b) Possible True Triple scatter path detected by Compton camera.



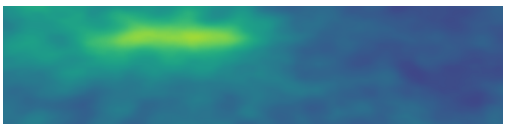
(c) Possible Double-to-Triple path of prompt gamma detected by Compton camera.



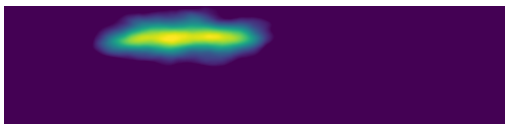
(d) False triple path of prompt gamma detected by Compton camera.

Figure 2.4: Scatter events.

The Need for Machine Learning In order to make proton beam therapy more effective, real-time imaging is needed to verify location and effectiveness of the proton beam, in particular the location of the Bragg peak. Machine learning is capable of classifying which type of scatter event occurred based upon data provided by the Compton camera. These classifications lead to removal of unusable data which will clean the resulting image. A clearer image allows for treatment verification. A sufficiently accurate machine learning model could produce an image that is clear enough to be used in proton beam therapy as a form of treatment verification. Figure 2.5 shows a sample comparison of an uncleaned image from raw data and a cleaned image after machine learning from [3]. This indicates how effective machine learning can be at cleaning image data. A machine learning algorithm will need approximately 90% testing accuracy to be useful for clinicians.



(a) Uncleaned image.



(b) Cleaned image.

Figure 2.5: Images from Compton camera data with a dose of 100kMU before and after machine learning.

3 Machine Learning

Machine learning is a type of artificial intelligence where a machine is trained to identify specific trends and patterns to make predictions from data. In the case of Compton camera data, the machine learning algorithm will try to predict the appropriate class for a scatter event. The main benefit of machine learning is its efficiency in producing results that would take humans alone much longer. There are four different ways that a machine can be taught: supervised, unsupervised, semi-supervised, and reinforcement. Supervised learning is a form of learning where the machine is provided a labeled data set that has regular input data as well as the desired output data. This allows the machine to produce a model that has been fitted appropriately. Unsupervised learning is used when one wants to identify hidden patterns within an unlabeled data set. This allows the machine to identify any trends it finds in the data without special instruction. Semi-supervised learning is a mixture of supervised and unsupervised where the model is provided some labeled data and a large amount of unlabeled data. Reinforcement learning is similar to the way humans learn where the machine will interact with the data and there will be either a positive or negative reward depending on whether the machine does something the programmer wants or not. The method used in this study is supervised learning because the data set contains both the data from the scatter event and the corresponding label of which event scatter took place.

3.1 Recurrent Neural Networks

Recurrent neural networks (RNNs) are an efficient neural network used for time series tasks. They work similar to a coupling process in biology. They rely on information from the previous system or "loop" to move forward with the next. In this type of neural networks, the sequence or order of the network is very important. The system can be read and executed differently if the elements of both series are in different orders. In the case of RNNs, elements include an input layer, hidden layers, and an output layer.

RNNs also have the same parameters throughout the various layers of the network and don't vary as in feed forward networks. The weight of each parameter remains the same with RNNs. In order to train an RNN, a loss function is defined to measure and decrease the error between the true label and the output.

RNNs use back-propagation through time to illustrate gradients. The difference between RNN

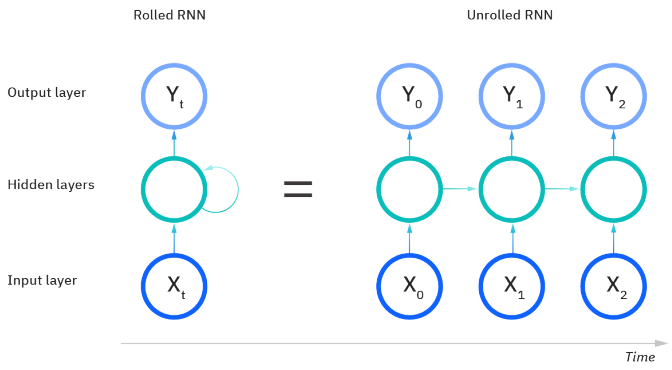


Figure 3.1: Recurrent neural network vs. feedforward neural network, IBM Cloud Education, https://www.ibm.com/cloud/learn/recurrent-neural-networks.

back propagation and other methods such as in a feed forward network is that sum errors are necessary at each time step because of the shared parameters throughout the network. There are several types of RNNs that are distinguished by the pathways between inputs and outputs. RNNs may also contain activation functions that allow a neuron to translate the input into a specific output. Finally, there are a few RNN structures that vary depending on the desired use. There are bidirectional recurrent neural networks, long short-term memory, and gated recurrent units. Bidirectional recurrent networks rely on future data to generate predictions.

RNNs are a viable option for Compton camera data because of their ability to encode information about previous events. Shaping an event in the Compton camera as a sequence of three interactions each with five features, we have transformed the data produced by the Compton camera to a sequence. Using the sequence of interactions the RNN will be able to predict the ordering of interactions, i.e., the appropriate scatter.

3.1.1 Long Short-Term Memory

A Long Short-Term Memory (LSTM) neural network is a type of RNN that is traditionally used for natural language processing and time series forecasting. The unique aspect of LSTM is that it contains a memory cell. This memory cell is used to store certain pieces of information that may be needed later in the training process, called a state. The memory cell has three gates to determined the state: forget gate, input gate, and output gate. The forget gate controls what stored information can be forgotten. The input gate controls what information should be used to change the state of the memory cell, and the output gate controls which part of that information is needed at a given time. As stated previously, RNNs use the output of one step and carry it over into the next step in addition to the new data input. This is depicted in Figure 3.2 as the "Carry track". The memory cell is depicted as "Compute New Carry" in Figure 3.2 where new information as well as the output of the previous step enter, the different gates classify the needed and unneeded information, and the new state is outputted for the next step. The memory cell was added to combat the main issue with RNNs which is long-term dependency where as more and more information is fed into the RNN, it becomes less effective in learning because the network cannot remember everything.

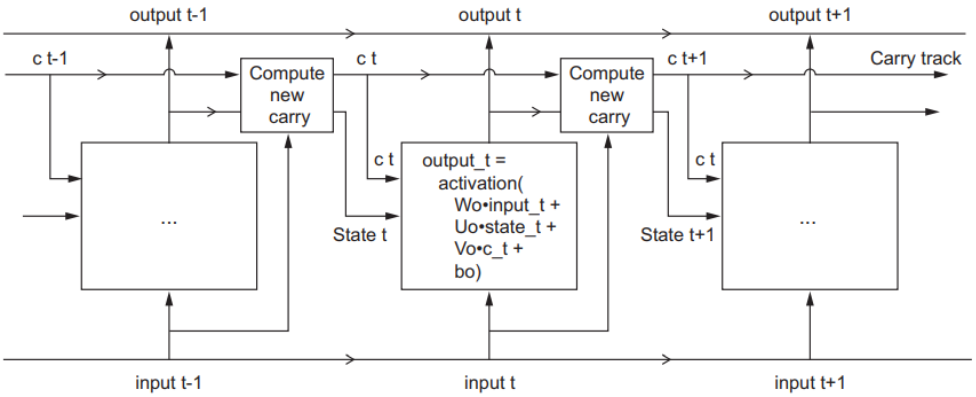


Figure 3.2: LSTM diagram [4]

The LSTM that was used in this initial study had 2 layers. Although the data being used is not time-series data, this test was done to determine whether a RNN could be used to classify the different scatter events. In addition to a new network, a hyper-parameter study was conducted

that manipulated batch size, neurons, and learning rates.

3.1.2 Gated Recurrent Unit

A Gated Recurrent Unit is essentially a streamlined version of the LSTM in Section 3.1.1. The GRU has gating units that modulate the flow of information inside of the unit. The GRU factors in the previous short-term dependency with a reset gate by using a linear interpolation between the previous activation function value and the current one. The GRU also factors in previous long-term dependencies with an update gate by taking a linear sum between existing state and the newly computed state. Unlike the LSTM the GRU does not have separate memory cells. In Figure 3.3 we see a diagram of the GRU [5].

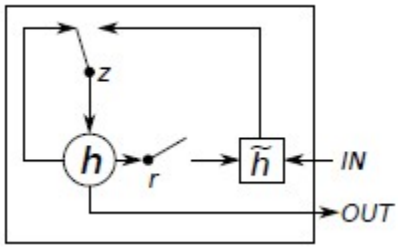


Figure 3.3: GRU diagram where r and z are the reset and update gates, and h and $\tilde{h}$ and the activation and the candidate activation [5].

3.2 Transformer Networks

Transformer networks are a more recent state of the art sequence modeling technique. Transformer models use attention mechanisms to capture sequence dependencies between input and output. Self-attention mechanisms that relates different positions of the same sequence in order to compute a representation of the sequence. Similar to other neural networks, attention mechanisms are based off of biological facts. A non-volitional cue is when the brain focuses on an object involuntarily such as a colored object in a row of black and white objects. A volitional cue is when the brain focuses on an object by choice. In an attention mechanism a query is a volitional cue that is a chosen object for the network to pay attention. For each query, attention mechanisms bias selection via sensory inputs called values. Every value is paired with a key which is a non-volitional cue for a given sensory input [12]. In self-attention a single source provides the query, key, and values. The transformer model uses multi-headed self-attention mechanisms. The multi-headed self-attention mechanism linearly projects the query, key, and value h times and concatenates the results allowing the model to jointly attend to information from different representation sub-spaces simultaneously [10].

The transformer network uses stack self-attention layers and point-wise, fully connected layers to create an encoder and decoder structure used in sequence modeling. The transformer model has been shown to learn long range dependencies easier than other RNNs which could be helpful for our model [10]. We are exploring appending transformer blocks to recurrent layers to determine if the transformer architecture could help classify Compton camera data. The hyper-parameters that we can tune are the number of heads of the transformer block which is the number of linear projection of the query,key, and value. The size of each of those heads which will be the dimension

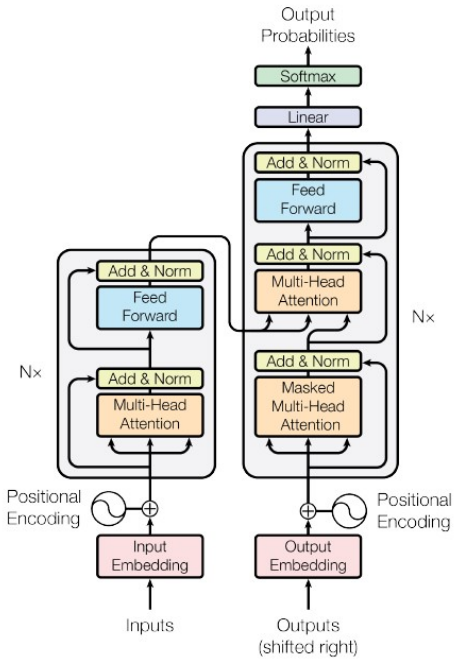


Figure 3.4: The transformer architecture [10].

of the key matrix, and the number of transformer blocks. Figure 3.4 shows the architecture of the transformer block.

4 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. The taki system has two GPU partitions 2013 and 2018. For 2018, 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). The 2013 GPU node contains 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).

Networks built on Taki were built using Tensorflow v2.4.0 (www.tensorflow.org) with the bundled Keras module. We also used scikit-learn v0.23.dev0 (https://scikit-learn.org/stable/) to preprocess and normalize the data. Moreover pandas v1.1.0 (https://pandas.pydata.org/) and numpy v1.18.1 (www.numpy.org) were also used to help preprocess the data. Finaly we used the matplotlib v3.5.1 (www.matplotlib.org) library to graph our results.

We also used the GPU cluster ada in the UMBC High Performance Computing Facility. The ada system has 3 distinct node types. Four nodes each with 8 Nvidia RTX 2080 Ti GPUs each with 11GB GPU memory. Seven nodes with 8 Nvidia Quadro RTX 6000 GPUs each with 24GB of GPU memory. Two nodes each with 8x Nvidia Quadro RTX 8000 GPUs each with 48GB memory. Each node has 384 GB of CPU memory (12×32 GB DDR4 at 2933 MT/s) except the two RTX

8000 nodes which have 768GB of CPU memory(12 $\times$ 64GB DDR4 at 2933 MT/s).

Networks built on ada were built with the software package Anaconda3 and Tensorflow v2.6.0 with the bundled Keras module.

5 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 1,443,993 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. Both the training and testing datasets were reshaped to be sequentially read. Therefore each record of 15 features was reshaped to 3 interactions of 5 features each: three spatial coordinates, Euclidean distance, and energy deposition. Each record is fed into the neural network as a sequence of 3 interactions. The testing data contains 37,151 testing data points for 20kMU/min, 17,425 for 100kMU/min, and 12,254 for 180kMU/min from MCDE model test 1 150MeV.

5.1 Performance tests on taki and ada Partitions

The tests below were performed to compare the performances of the 2013 taki partition with the 2018 taki partition, along with the three GPUs of ada- RTX 2080 Ti, RTX 6000, and RTX 8000. All studies in the performance comparison study were run using a deep fully connected neural network whose architecture is similar to the model in [3] with residual blocks and some hyperparameter changes. The hyperparamters used for these tests include 128 layers with 256 neurons, a batch size of 8192, 1024 epochs and a learning rate of 1e-3. Table 5.1 records the performance times of each partition. The taki 2018 partition performs the fastest with completing the job in 4 hours, 13 minutes, and 42 seconds. The slowest performance is that of the taki 2013 partition, which takes 15 hours, 49 minutes, and 31 seconds. All three ada partitions perform similarly, and are slightly slower than the taki 2018 partition. The taki 2018 partition is at least three times faster when compared to the taki 2013 partition, and is the most efficient partition to use for future studies.

Table 5.1: Table of taki and ada performances.

Cluster	*Partition*	*Total Time (s)*	*Total Time (HH:MM:SS)*
taki	2013	56,971.1	15:49:31
taki	2018	15,222.6	04:13:42
ada	RTX 2080 Ti	18,547.5	05:09:07
ada	RTX 6000	19,125.4	05:18:45
ada	RTX 8000	19,541	05:25:41

5.2 Hyperparameter study on Recurrent Neural Networks

Previous research explored fully connected networks in depth. We explore recurrent neural networks using the LSTM and GRU layers. We begin by examining the number of epochs, the batch size, and the learning rate. We then explore the number of layers and number of neurons in order to determine a promising configuration for a recurrent neural network. RNNs with both GRU and LSTM models are examined. These studies lead us to the use of 4 recurrent layers of 128 neurons with a batch size of 2048 and learning rate of 10^{-3}. Table 5.2 shows the constant parameters for all RNN studies.

Hyperparameter	Value
Recurrent Layer Activation	Tanh
Final activation	Softmax
Output Layer	13 Neurons
Optimizer	Nadam
Loss Function	Categorical Crossentropy
Train,Validation Split	0.8/0.2

Table 5.2: Constant RNN parameters

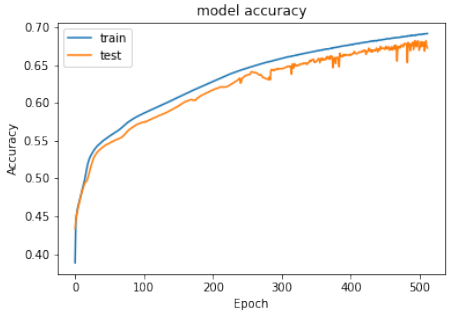
5.2.1 First RNN Hyperparameter study

In the first RNN hyperparameter study, we tested the models using two batch sizes, two number of epochs, two number of neurons, and two learning rates. In each training and validation plot accuracy is plotted with the accuracy as a decimal percent against the number of epochs. The variable parameters are as follows:

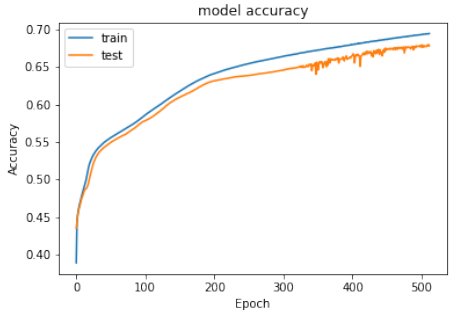
- Batch size: 2048, 4096
- Epochs: 256, 512
- Neurons: 64, 128
- Learning rate: 10^{-3}, 10^{-4}

In this test we determine that 512 epochs, a batch size of 2048, and 128 neurons is the most promising configuration. The learning rates did not make much of a difference on accuracy on this study. The 2 LSTM layer model shown in Figure 5.1 (a) and Figure 5.1 (b) show the model's training and validation accuracy plotted against the number of epochs. The model has a validation accuracy of 68%.

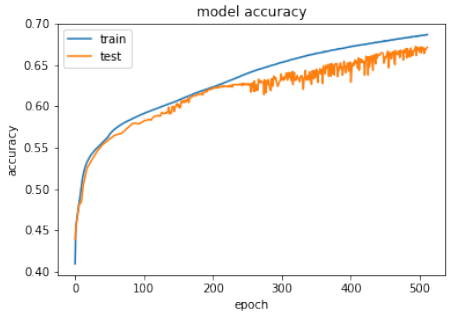
Similarly, the 2 GRU layer model is tested using the hyperparameter configuration. The GRU layer model achieves very similar results to the LSTM layer model. The GRU layer model shown in Figure 5.1 (c) and Figure 5.1 (d) have a validation accuracy of 67%. We also notice in Figure 5.1 that validation accuracy begins at around 40% much higher than initial accuracy in [3] which is around 7%. Further it appears as if the model is still learning suggesting the number of epochs can be increased.



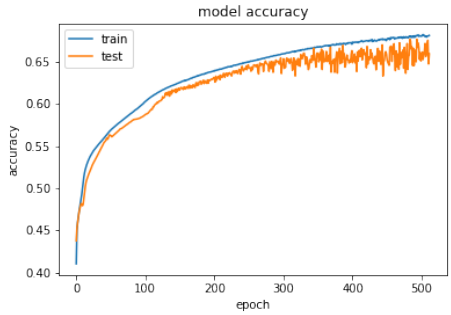
(a) 2 LSTM layer model accuracy using batch size of 2048, 128 neurons, and 10^{-3} learning rate.



(b) 2 LSTM layer model accuracy using batch size of 2048, 128 neurons, and 10^{-4} learning rate.



(c) 2 GRU layer model accuracy using batch size of 2048, 128 neurons, and 10^{-3} learning rate.



(d) 2 GRU layer model accuracy using batch size of 2048, 128 neurons, and 10^{-4} learning rate.

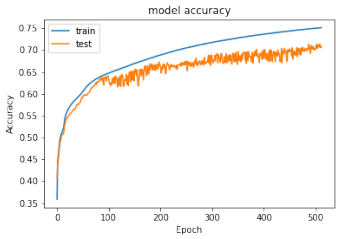
Figure 5.1: Model accuracy of LSTM and GRU of 2048 batch size, 128 neurons and 10^{-3} and 10^{-4} learning rates.

5.2.2 Second RNN Hyperparameter Study

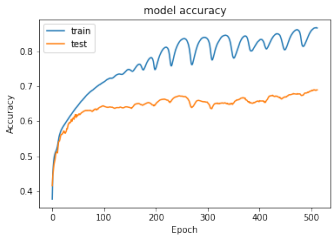
From the previous hyperparameter study, it was found that layers and neurons had the largest impact on model accuracy. Therefore, the following studies manipulate the number of layers and neurons to find a better configuration. We have fixed the batch size to 2048, the number of epochs to 512, and the learning rate to 10^{-3}. We test four different number of layers, and two number of neurons. The models with 4 recurrent layers performs better than those with 2 recurrent layers. Models that have over 128 neurons are overfit that is training accuracies continues to rise while validation or testing accuracy does not. Models that have over 4 recurrent layers are either overfit or do not learn (training and validation accuracy plateau). Figure 5.2 shows the promise of the 4 recurrent layer models achieving validation accuracies above 70% while remaining close to the training accuracy. Future studies will use 4 layers and 128 neurons with the same batch size. The variable parameters are as follows:

- Layers: 4, 8, 16, 32
- Neurons: 128, 256, 512

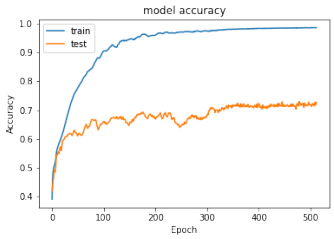
In Figure 5.2 (a) we see the 4 LSTM layer model with 128 neurons showing a steady logarithmic rise in training and validation accuracy while the two curves are beginning to split they are relatively close within 5%. However with 5.2 (b-c) we see a large difference in training and validation accuracies. The model's training accuracies continue to rise while the validation accuracies do not.. Especially in 5.2 (c) we see the model have almost 100% training accuracy with a validation accuracy around 70%. The model has memorized the data at this point. Figures 5.2 (d-f) show the exact same trends for the GRU layer models. The two layers are performing almost identically in the study under these parameters.



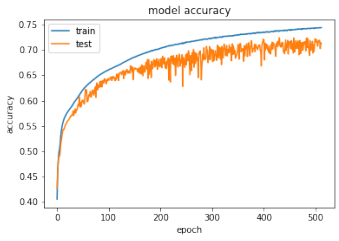
(a) 128 Neurons.



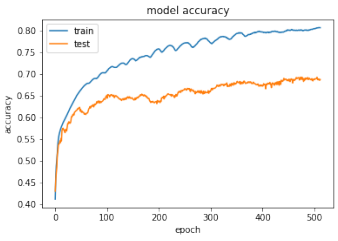
(b) 256 Neurons.



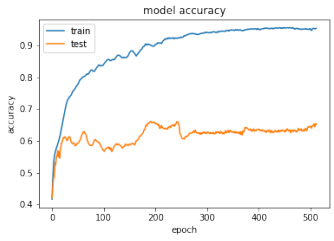
(c) 512 Neurons.



(d) 128 Neurons.



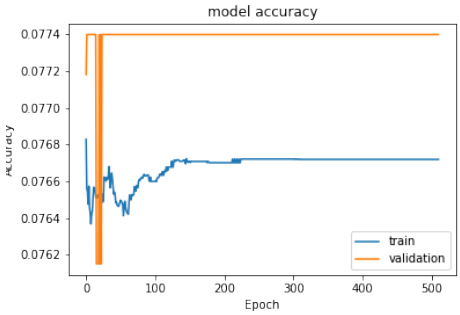
(e) 256 Neurons.



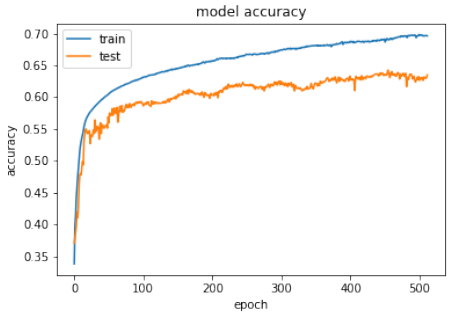
(f) 512 Neurons.

Figure 5.2: LSTM (a-c) and GRU (d-f) runs with 4 layers and 128, 256, and 512 neurons.

In Figure 5.3 we see the training and validation plots for the 16 recurrent layers each of 128 neurons models. In Figure 5.3 (a) the model had accuracies of 7% which is equivalent to the model making a random guess. In Figure 5.3 (b) we see the 16 GRU layer model performing much better than the LSTM layer model. This is the first meaningful difference in the two models. The 16 GRU layer model's training and validation accuracies are not converging which is the reason for our belief that 16 GRU layers is too many and for us to continue with the 4 recurrent layer models. It is possible that an 8 or 16 GRU layer model could be adjusted and regularized. The 32 recurrent layer models never achieve testing or validation accuracy above a random guess.



(a) 16 LSTM Layers.



(b) 16 GRU Layers.

Figure 5.3: LSTM (a) and GRU (b) runs with 16 Layers.

5.3 Learning Rate Scheduler and Long Jobs

Studies with many more epochs were run based on the results of section 5.2.2. Our goal is to discover how long these models could be trained before plateauing. Testing models with more epochs than 512 showed an increase in validation accuracy. However, after 1024 epochs the model learns very slowly. A 4 GRU layer model with 1024 epochs has a validation accuracy of 71% and the same model with 8192 epochs has a validation accuracy of 77%.

A learning rate scheduler is used to change the learning rate during model training. One possible learning rate schedule is a step schedule which changes the learning rate at certain epochs. This can be done using a Keras callback which will adjust the learning rate during training. A piece-wise function such as the one in Equation (5.1) can represent a step learning schedule. Equation (5.1) is the same schedule as used in [2]. We use i to represent the current epoch and p to represent the total number of epochs, then L is a function of epochs and determines our learning rate at the i^{th} epoch. Our initial studies using a step learning rate schedule showed that a learning rate schedule could make approximately a 6 to 7% increase in accuracy while preventing overfitting.

$$L(i) = \begin{cases} 10^{-3} & i \leq \frac{p}{3} \\ 10^{-4} & \frac{p}{3} < i \leq \frac{2p}{3} \\ 10^{-5} & \frac{p}{3} < i \leq \frac{5p}{6} \\ 10^{-6} & \frac{5p}{6} < i \end{cases} \tag{5.1}$$

Figure 5.4 shows a 4 LSTM layer model with and without a learning rate schedule. We are able to notice in the training validation plots that the model has a sharp rise in accuracy when the learning rate changes at first and that the lower learning rates at the end prevent the overfitting.

The impact of the learning rate schedule on accuracy and generalization lead us to study a 32,000 epoch model with the learning rate schedule defined by Equation (5.1) that also has 2 dense layers of 128 and 64 neurons respectively. We test these parameters on both the models with LSTM and the models with GRU layers. The final models will then be tested using a new data set and will have confusion matrices made to verify their accuracy. A confusion matrix contains all 13 misdetection orderings as well as a percentage that is determined by how frequently the model classified each event correctly or incorrectly. The main diagonal of a given matrix shows the percentage of correct classifications of the network. All other entries in the matrix are percentages where the network incorrectly classified an event.

(a) 4 LSTM layers without learning rate scheduler.

(b) 4 LSTM layers with learning rate scheduler.

Figure 5.4: Impact of the learning rate scheduler.

5.3.1 32,000 Epoch Network with Learning Rate Schedule

The 4 LSTM layer model has a very high training and validation accuracy. The dense layers for the LSTM model have the ReLU activation function for both layers. This causes us to believe the model could be a possible improvement over the previous model. We notice the rise in accuracy at the epochs where the learning rate is lowered. We also notice how training and validation accuracy converge at the end with the lowering learning rates. The model has a final validation accuracy of 89% which is a significant improvement over all previous studies. The model however is overfit. There is a significant difference in the validation accuracy shown in Figure 5.5 and the testing accuracy shown in Table 5.3 through Table 5.5.

The dense layers for the GRU model have the leaky ReLU activation function for both layers the parameters are the same for the GRU. The GRU layers produce a model with a slightly lower final validation accuracy of 86%. However the model performs different on the test data. Figure 5.6 shows the training validation plot for this model. Tables 5.6 through 5.8 show the confusion matrices. Ultimately, this model is still overfit with the large difference in validation and testing accuracy. The overfitting of these models tells us that we should try and apply some regularization to them such as dropout layers in order to make the model more general. Regularization techniques will help bring the testing accuracy closer to the validation accuracy.

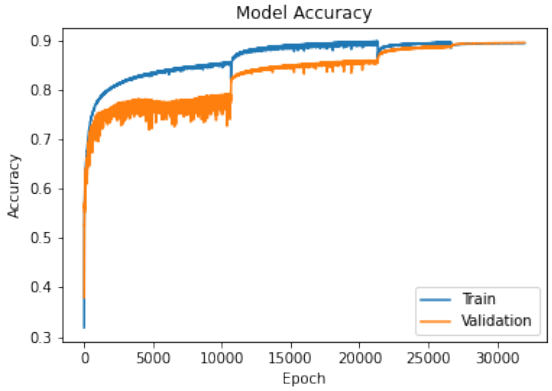


Figure 5.5: Accuracy of 4 layer LSTM model with 2 Dense layers run on 32000 epochs.

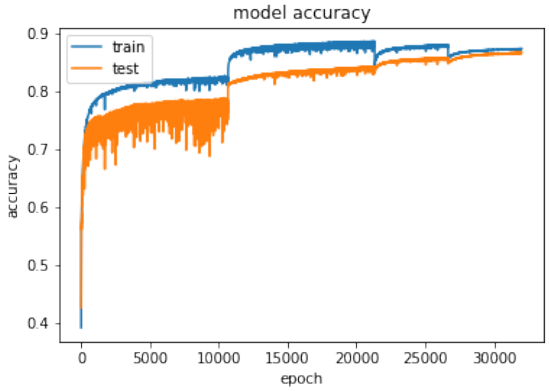


Figure 5.6: Accuracy of 4 layer GRU model with 2 Dense layers run on 32000 epochs.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	60.6	7.3	4.1	4.0	4.5	3.7	7.1	1.1	0.3	0.1	4.1	2.4	0.7
132	7.8	58.8	3.6	3.7	5.3	4.2	0.5	0.2	7.0	1.7	2.7	4.0	0.6
213	4.7	3.8	60.9	6.8	3.4	3.4	1.2	7.7	4.9	2.1	0.4	0.1	0.5
231	4.3	3.5	7.6	57.9	3.9	5.9	0.1	0.4	2.3	4.3	7.7	1.7	0.5
312	3.8	5.2	3.6	4.1	60.3	6.1	4.3	2.3	1.7	7.3	0.2	0.5	0.6
321	4.3	3.6	3.8	6.1	6.3	58.5	2.1	4.2	0.2	0.7	1.6	7.9	0.6
124	6.0	0.4	1.3	0.1	4.0	2.3	64.4	11.1	1.1	0.8	0.6	2.2	5.7
214	1.1	0.2	7.3	0.7	1.9	3.7	12.8	63.2	0.6	1.8	1.1	0.7	4.9
134	0.8	7.4	4.9	2.5	1.8	0.3	0.9	0.6	59.6	13.5	1.9	0.8	5.0
314	0.1	2.1	2.4	5.2	6.9	0.7	0.8	1.3	13.0	59.7	0.7	1.0	6.1
234	4.8	2.5	0.0	6.2	0.1	1.8	0.7	1.0	1.6	1.3	58.3	15.1	6.5
324	2.8	3.9	0.2	1.0	0.3	7.3	1.8	0.5	1.1	1.7	14.1	59.8	5.5
444	1.3	1.6	0.9	1.6	0.3	0.6	6.6	4.1	6.3	5.0	7.8	8.5	55.5

Table 5.3: Confusion matrix for 4 LSTM layer model with learning rate schedule Equation (5.1) trained on triples, double to triples, and false data from a 150MeV over 32000 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	60.3	6.5	4.9	3.9	2.9	3.9	7.4	1.0	0.4	0.1	5.3	3.0	0.5
132	7.0	56.4	4.5	3.5	5.9	4.5	0.2	0.0	8.6	2.1	2.7	3.7	0.9
213	4.9	4.0	57.7	8.1	5.0	3.6	1.2	6.8	4.7	3.1	0.4	0.2	0.3
231	3.0	3.8	8.3	60.3	3.7	5.5	0.2	0.4	2.8	4.4	6.0	1.4	0.3
312	3.7	4.3	2.9	3.8	62.0	7.3	3.9	2.1	0.9	7.8	0.1	0.7	0.6
321	3.5	4.4	4.4	4.7	7.8	56.6	2.3	5.6	0.3	0.4	1.5	8.2	0.3
124	7.8	0.4	1.5	0.1	4.7	1.7	64.6	11.0	0.8	0.7	0.7	1.2	4.7
214	1.3	0.1	7.0	0.4	1.7	4.6	10.6	64.6	0.7	1.7	0.7	0.8	5.8
134	0.4	6.7	4.0	1.9	1.0	0.1	0.7	0.5	60.6	14.9	1.5	0.8	6.9
314	0.1	1.3	2.7	5.2	6.8	0.4	0.5	1.0	13.9	59.7	0.8	1.1	6.5
234	5.4	2.5	0.5	7.3	0.2	1.3	0.5	0.8	1.8	0.5	56.8	15.4	7.0
324	2.2	4.6	0.4	1.6	0.5	7.1	1.3	0.6	0.7	0.8	15.2	57.4	7.7
444	1.0	0.4	0.8	0.8	0.9	0.7	6.5	6.0	6.8	6.0	6.0	6.0	57.9

Table 5.4: Confusion matrix for 4 LSTM layer model with learning rate schedule Equation (5.1) trained on triples, double to triples, and false data from a 150MeV over 32000 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	62.3	8.7	4.6	3.6	3.1	3.4	7.2	0.2	0.7	0.2	3.8	2.2	0.0
132	6.0	58.2	3.6	3.6	6.2	3.6	1.2	0.0	6.0	1.0	3.8	5.8	1.0
213	4.1	4.6	55.2	8.4	4.8	2.7	1.7	9.9	5.1	2.4	1.0	0.2	0.0
231	3.4	3.6	8.2	55.9	6.3	7.0	0.2	0.7	1.7	3.9	6.7	1.2	1.2
312	4.3	6.0	2.9	2.9	60.5	6.3	4.6	2.2	1.7	6.3	0.0	1.0	1.4
321	4.1	3.1	4.6	3.4	6.3	59.5	2.9	5.3	0.0	0.2	2.7	7.0	1.0
124	6.5	0.6	1.4	0.2	3.5	2.1	65.1	11.2	0.9	0.5	0.6	1.5	6.0
214	1.0	0.0	6.3	0.5	1.7	4.2	9.2	68.0	0.6	1.2	0.8	0.7	5.8
134	0.8	7.1	4.5	2.9	1.6	0.1	0.6	0.6	61.8	12.2	1.5	0.5	5.8
314	0.1	1.3	2.1	4.8	6.7	0.5	0.4	1.7	12.8	60.3	0.5	1.1	7.7
234	4.3	2.9	0.7	6.7	0.2	1.8	0.2	0.5	1.7	0.6	59.8	14.1	6.5
324	2.5	4.3	0.1	1.4	0.6	6.4	1.6	0.6	1.1	1.1	14.8	57.9	7.5
444	1.0	1.2	0.5	0.8	0.8	0.3	5.8	5.9	6.2	6.5	7.2	7.4	56.5

Table 5.5: Confusion matrix for 4 LSTM layer model with learning rate schedule Equation (5.1) trained on triples, double to triples, and false data from a 150MeV over 32000 epochs. The testing data used is from the MCDE model test1 150MeV 180K beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	66.8	5.4	2.7	4.0	3.4	2.3	6.9	0.7	0.3	0.1	4.6	2.1	0.6
132	5.5	65.0	3.6	2.7	3.2	3.8	0.4	0.1	7.2	1.1	1.9	4.7	0.8
213	2.7	3.5	68.5	5.6	2.3	3.0	0.8	7.0	4.2	1.8	0.2	0.1	0.4
231	3.3	2.9	5.7	64.8	3.4	3.2	0.1	0.2	1.7	4.6	8.5	1.0	0.5
312	3.5	3.6	2.3	3.4	67.3	4.4	4.4	1.9	1.3	6.9	0.1	0.4	0.4
321	2.6	3.4	3.6	2.5	4.9	67.6	1.6	3.7	0.1	0.4	0.9	7.8	0.7
124	6.1	0.5	1.0	0.1	3.9	2.1	69.3	8.3	0.7	0.6	0.4	1.6	5.3
214	0.7	0.3	6.7	0.3	2.1	4.3	8.5	69.0	0.6	1.2	0.4	0.8	5.1
134	0.5	5.9	4.4	2.6	1.2	0.2	0.9	0.3	63.4	12.9	1.8	0.9	5.0
314	0.1	1.3	3.0	5.0	6.1	0.4	0.8	1.1	12.6	62.9	0.5	1.2	5.2
234	3.8	2.6	0.3	8.4	0.1	0.9	0.4	1.0	2.3	1.0	62.7	10.7	5.7
324	1.5	4.5	0.1	0.9	0.3	8.1	1.1	0.5	0.6	1.2	9.7	66.1	5.5
444	1.3	1.6	0.3	0.6	0.0	0.0	6.6	5.6	5.6	5.6	6.6	5.3	60.8

Table 5.6: Confusion matrix for 4 GRU layer model with learning rate schedule Equation (5.1) trained on triples, double to triples, and false data from a 150MeV over 32000 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	68.5	5.5	2.8	3.3	4.1	1.8	6.5	0.8	0.0	0.1	3.9	2.2	0.4
132	4.3	63.6	3.6	2.8	4.1	3.6	0.0	0.0	8.3	1.7	2.1	5.3	0.6
213	4.3	3.7	66.2	5.8	3.0	2.6	0.9	5.8	4.7	1.9	0.6	0.0	0.5
231	3.5	2.1	6.5	67.6	3.1	2.6	0.2	0.5	2.1	3.2	7.7	0.6	0.3
312	2.5	3.1	3.2	2.7	70.9	4.9	3.2	1.4	0.6	6.7	0.0	0.5	0.4
321	1.9	4.0	4.1	2.8	6.2	65.4	1.3	4.5	0.1	0.3	1.1	7.8	0.5
124	6.7	0.7	0.8	0.0	3.8	1.6	70.4	8.3	1.1	0.5	0.7	1.6	3.9
214	1.1	0.2	7.3	0.3	1.6	4.5	7.9	68.7	0.5	1.3	1.0	0.5	5.0
134	0.4	6.4	3.6	2.7	1.4	0.1	0.8	0.4	63.3	13.5	1.3	0.7	5.5
314	0.1	1.2	2.6	3.8	5.6	0.2	0.3	0.8	13.1	64.1	0.7	1.1	6.4
234	5.0	2.2	0.5	7.1	0.2	0.9	0.5	0.7	1.3	0.8	64.2	10.4	6.1
324	1.3	4.9	0.2	0.9	0.5	6.9	1.0	0.4	0.8	0.7	10.1	64.6	7.7
444	0.9	0.8	0.7	1.0	0.7	0.5	5.2	4.4	5.5	5.1	5.1	6.4	63.7

Table 5.7: Confusion matrix for 4 GRU layer model with learning rate schedule Equation (5.1) trained on triples, double to triples, and false data from a 150MeV over 32000 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	66.8	4.8	3.1	2.9	3.4	2.9	7.0	0.7	0.7	0.0	5.8	1.9	0.0
132	4.8	65.1	4.3	1.9	4.1	2.9	0.2	0.2	7.2	0.5	2.2	5.3	1.2
213	3.9	2.4	67.0	5.5	2.9	3.9	0.7	8.0	4.1	1.4	0.2	0.0	0.0
231	3.4	1.9	6.7	62.7	5.1	3.6	0.0	1.0	1.7	4.8	7.7	0.2	1.2
312	2.7	5.1	1.7	2.9	70.8	3.6	1.9	2.2	0.7	7.0	0.0	0.5	1.0
321	1.9	1.7	3.9	2.2	5.3	68.9	1.7	4.8	0.7	0.2	0.5	7.5	0.7
124	7.4	0.8	0.7	0.2	3.6	1.8	69.0	8.3	1.0	0.2	0.5	1.0	5.6
214	0.5	0.1	5.2	0.6	2.1	4.1	7.1	72.2	0.6	0.8	0.5	0.7	5.7
134	0.6	7.4	4.6	1.9	1.3	0.2	0.4	0.6	63.6	12.2	1.5	0.9	4.8
314	0.1	0.6	1.4	3.7	6.4	0.5	0.6	1.6	13.4	62.8	1.1	0.7	7.1
234	5.2	2.2	0.4	7.4	0.3	1.2	0.2	0.2	1.3	0.4	64.3	9.7	7.1
324	2.0	4.7	0.2	1.4	0.5	7.4	1.7	0.2	0.8	0.6	11.0	63.0	6.6
444	1.0	0.8	0.4	0.8	0.3	0.5	5.3	5.0	5.8	5.7	6.8	6.5	61.2

Table 5.8: Confusion matrix for 4 GRU layer model with learning rate schedule Equation (5.1) trained on triples, double to triples, and false data from a 150MeV over 32000 epochs. The testing data used is from the MCDE model test1 150MeV 180K beam.

5.3.2 Regularization

We adjusted the number of epochs to be 16384 and added dropout layers between every hidden layer with a dropout rate of 20%. The model used Equation (5.2) as the learning rate schedule.

$$L(i) = \begin{cases} 10^{-3} & i \leq \frac{p}{8} \\ 10^{-4} & \frac{p}{8} < i \leq \frac{p}{4} \\ 10^{-5} & \frac{p}{4} < i \leq \frac{p}{2} \\ 10^{-6} & \frac{p}{2} < i \end{cases} \tag{5.2}$$

This helps regularize the model. While drastically reducing the model's validation accuracy to below 80%; the validation and testing accuracy are much closer. This model performs almost as well as the deep residual fully connected model for both the LSTM and the GRU. While the model is almost as accurate as the deep residual fully connected model. Its load time is 10s for the GRU model and 7s for the LSTM model which is an advantage. The fully connected model loads in 47s. Also having still only 4 recurrent layers and 2 dense layers is an advantage because there is a great deal more than can be done with such a simple model. Table 5.10 through Table 5.15 show the confusion matrices for the regularized GRU and LSTM models.

Class	GRU	DRFCN	GRU - DRFCN
123	76.4	79.1	-2.7
132	79.4	76.0	3.4
213	73.5	76.4	-2.9
231	79.1	80.7	-1.6
312	83.1	82.4	0.7
321	76.2	76.5	-0.3
124	71.9	76.0	-4.1
214	74.0	75.0	-1
134	72.0	75.4	-3.4
314	78.3	75.4	2.9
234	63.9	73.6	-9.7
324	73.9	75.3	-1.4
444	63.5	72.6	-9.1

Table 5.9: Comparison of GRU model with deep fully connected network.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	73.8	5.4	1.7	3.5	4.2	2.0	5.6	0.5	0.2	0.1	1.6	1.1	0.2
132	2.5	79.4	1.9	1.6	3.3	2.6	0.1	0.0	5.0	0.8	0.4	2.2	0.2
213	1.5	3.4	75.7	3.5	2.5	2.9	0.6	4.7	3.1	1.8	0.1	0.0	0.2
231	2.0	2.1	3.4	77.3	4.0	2.2	0.0	0.1	0.7	3.5	3.7	0.7	0.2
312	1.6	2.0	1.2	1.7	81.1	2.7	2.1	0.9	0.5	6.0	0.0	0.1	0.0
321	1.2	2.7	2.1	2.3	4.3	78.7	0.7	2.1	0.0	0.4	0.4	4.9	0.2
124	4.2	0.4	0.8	0.1	5.2	2.3	70.6	9.0	0.9	1.0	0.3	0.9	4.2
214	0.4	0.3	5.1	0.3	2.3	4.4	5.7	74.6	0.3	2.1	0.3	0.4	3.7
134	0.6	5.8	3.0	1.7	0.8	0.1	0.4	0.3	72.7	10.5	0.5	0.8	2.7
314	0.0	1.5	1.7	4.1	6.0	0.4	0.5	0.9	6.8	74.4	0.1	0.9	2.6
234	3.6	2.8	0.1	8.2	0.4	1.0	0.1	1.1	1.8	1.0	65.7	9.4	4.8
324	1.3	5.7	0.1	0.4	0.4	7.3	0.9	0.3	0.6	1.2	4.3	74.4	3.2
444	0.9	2.2	0.3	0.9	0.0	0.0	5.6	6.3	5.6	7.5	4.4	7.2	58.9

Table 5.10: Confusion matrix for 4 GRU layer model with learning rate schedule Equation (5.2) trained on triples, double to triples, and false data from a 150MeV over 16384 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	76.4	4.4	1.8	3.1	3.5	1.7	4.5	0.8	0.2	0.0	2.0	1.4	0.1
132	2.0	79.4	1.5	1.2	3.5	2.0	0.0	0.0	6.8	0.6	0.6	1.9	0.3
213	2.0	3.7	73.5	3.7	3.2	1.9	0.5	4.9	3.6	2.7	0.2	0.0	0.1
231	1.9	2.2	3.1	79.1	3.1	1.9	0.0	0.0	1.2	3.4	3.4	0.6	0.1
312	1.1	1.7	1.5	2.1	83.1	2.7	1.7	0.6	0.3	4.7	0.0	0.2	0.1
321	0.8	3.3	2.7	2.8	5.6	76.2	0.6	3.2	0.0	0.6	0.3	3.8	0.2
124	5.9	0.5	0.5	0.1	5.3	1.6	71.9	7.7	0.8	0.7	0.2	1.3	3.4
214	0.7	0.2	5.8	0.4	2.5	4.2	5.6	74.0	0.5	2.1	0.2	0.5	3.4
134	0.3	6.5	2.8	1.7	0.7	0.0	0.4	0.5	72.0	11.0	0.6	0.4	3.2
314	0.1	0.5	1.6	2.9	5.8	0.2	0.1	0.7	5.9	78.3	0.1	1.1	2.8
234	4.7	3.1	0.5	8.0	0.2	0.9	0.4	0.5	1.1	0.8	63.9	10.3	5.4
324	1.2	4.7	0.2	0.9	0.4	6.1	0.7	0.3	0.8	1.3	5.4	73.9	4.1
444	1.0	0.9	0.4	0.6	0.9	1.0	4.2	4.6	6.4	8.3	3.0	5.2	63.5

Table 5.11: Confusion matrix for 4 GRU layer model with learning rate schedule Equation (5.2) trained on triples, double to triples, and false data from a 150MeV over 16384 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	72.1	7.2	2.2	2.4	4.3	2.6	6.2	0.5	0.0	0.0	1.7	0.7	0.0
132	1.7	81.0	1.2	1.2	1.2	2.2	0.2	0.0	5.5	1.0	0.7	3.6	0.5
213	1.0	3.1	75.4	4.1	2.4	2.9	0.2	6.5	3.4	1.0	0.0	0.0	0.0
231	1.4	1.7	4.3	75.4	5.3	1.9	0.2	0.2	0.7	4.1	3.6	0.7	0.2
312	1.0	3.4	0.7	1.7	80.7	2.9	1.9	0.7	0.5	5.5	0.0	0.2	0.7
321	1.7	1.7	2.4	2.7	4.8	78.1	1.4	3.4	0.0	0.2	0.2	3.4	0.0
124	6.0	0.8	0.6	0.0	4.7	1.9	70.8	8.2	0.5	0.4	0.2	1.1	4.9
214	0.8	0.0	5.1	0.6	2.9	4.4	5.2	74.0	0.5	1.5	0.2	0.6	4.1
134	0.6	6.4	3.3	1.7	1.4	0.0	0.3	0.2	71.6	10.7	0.9	0.5	2.3
314	0.1	0.2	0.6	3.7	6.3	0.3	0.1	1.0	6.4	77.4	0.1	0.6	3.3
234	3.7	2.5	0.6	7.6	0.3	1.1	0.3	0.5	1.6	0.6	67.0	8.7	5.4
324	1.4	4.8	0.1	0.8	0.6	6.6	1.4	0.2	0.9	1.0	6.0	72.1	4.2
444	0.8	1.2	0.3	0.7	0.6	0.4	4.9	4.4	6.5	7.7	4.0	5.9	62.6

Table 5.12: Confusion matrix for 4 GRU layer model with learning rate schedule Equation (5.2) trained on triples, double to triples, and false data from a 150MeV over 16384 epochs. The testing data used is from the MCDE model test1 150MeV 180K beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	78.2	3.6	1.3	2.4	2.2	1.8	7.2	0.4	0.1	0.0	1.8	0.8	0.2
132	4.3	74.9	2.3	1.8	2.7	2.9	0.4	0.0	6.6	0.7	0.7	2.3	0.3
213	2.2	3.1	75.0	3.0	1.9	2.5	0.8	6.3	3.6	1.3	0.1	0.0	0.3
231	3.6	2.3	3.8	73.3	3.4	2.3	0.1	0.2	1.3	3.6	5.3	0.6	0.3
312	3.4	2.2	1.5	2.3	74.2	3.2	4.0	1.2	0.7	6.9	0.0	0.1	0.2
321	2.2	2.5	2.5	1.9	3.6	76.5	1.6	3.1	0.0	0.2	0.5	5.0	0.4
124	4.4	0.4	0.6	0.2	2.8	1.5	75.9	7.3	0.7	0.7	0.2	0.9	4.4
214	0.8	0.3	4.7	0.1	1.1	3.3	8.9	74.7	0.5	1.2	0.3	0.3	4.0
134	0.6	4.7	2.8	1.6	0.5	0.3	0.8	0.5	75.3	8.4	0.9	0.4	3.2
314	0.0	1.0	2.1	3.3	5.0	0.4	0.7	1.1	8.3	72.8	0.2	1.0	4.0
234	4.9	2.1	0.2	6.4	0.1	0.6	0.5	1.2	1.8	0.8	67.8	8.0	5.7
324	1.8	5.4	0.1	0.4	0.3	6.7	1.8	0.4	0.6	1.0	7.1	70.6	3.9
444	1.3	1.9	0.3	0.9	0.0	0.0	7.2	6.0	4.1	5.3	5.0	6.0	62.1

Table 5.13: Confusion matrix for 4 LSTM layer model with learning rate schedule Equation (5.2) trained on triples, double to triples, and false data from a 150MeV over 16384 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	80.0	3.8	1.6	2.5	1.3	1.3	5.9	0.6	0.1	0.0	1.9	0.8	0.1
132	3.4	75.8	2.2	1.2	3.1	2.5	0.0	0.0	8.1	0.7	0.6	2.1	0.3
213	2.3	2.8	72.5	3.9	3.0	1.8	1.0	6.2	4.1	1.7	0.2	0.1	0.3
231	3.1	1.5	4.4	75.9	2.7	2.6	0.1	0.2	1.7	3.0	4.2	0.5	0.2
312	2.6	2.0	2.2	2.5	76.1	3.9	2.7	1.0	0.6	6.1	0.0	0.1	0.2
321	1.5	3.1	3.3	2.8	4.7	73.5	1.9	3.9	0.0	0.3	0.2	4.5	0.3
124	5.8	0.4	0.3	0.1	2.8	1.0	77.1	6.7	0.5	0.7	0.3	0.8	3.4
214	1.0	0.1	5.2	0.4	1.7	3.3	8.2	74.2	0.5	1.0	0.1	0.4	3.9
134	0.4	4.9	2.7	1.4	0.5	0.1	0.7	0.7	74.2	9.1	0.8	0.2	4.3
314	0.1	0.6	1.8	2.9	4.5	0.2	0.5	1.1	8.9	73.7	0.3	0.8	4.6
234	5.6	2.2	0.4	6.2	0.2	0.7	0.5	0.5	1.2	0.5	68.5	7.7	5.6
324	1.8	4.2	0.1	0.8	0.4	6.7	1.2	0.5	0.7	0.8	7.3	69.6	6.0
444	1.1	0.6	0.3	0.6	0.5	0.4	6.0	4.9	5.6	5.0	3.0	3.7	68.3

Table 5.14: Confusion matrix for 4 LSTM layer model with learning rate schedule Equation (5.2) trained on triples, double to triples, and false data from a 150MeV over 16384 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	77.6	6.2	1.9	1.2	1.9	1.7	6.0	0.5	0.0	0.0	2.2	0.7	0.0
132	3.1	78.8	1.7	1.2	1.4	1.7	0.5	0.0	5.8	1.2	0.7	3.4	0.5
213	1.2	2.2	75.4	4.8	1.2	3.1	1.2	6.0	4.1	0.5	0.2	0.0	0.0
231	3.4	1.7	5.3	71.6	4.1	2.4	0.0	0.0	1.2	4.3	4.8	0.5	0.7
312	3.9	2.4	1.0	1.7	73.7	3.1	4.8	1.0	0.5	6.7	0.0	0.2	1.0
321	1.7	2.2	2.7	1.9	3.1	76.9	3.1	3.6	0.0	0.2	0.7	3.4	0.5
124	5.4	0.8	0.6	0.1	3.0	1.1	76.6	6.6	0.5	0.3	0.3	0.9	3.8
214	1.0	0.0	4.1	0.5	1.8	3.3	8.8	74.2	0.4	0.7	0.3	0.2	4.7
134	0.5	5.0	3.5	1.3	0.8	0.1	0.6	0.4	73.7	8.8	1.0	0.8	3.6
314	0.0	0.2	0.9	3.0	5.4	0.3	0.5	1.3	8.3	74.3	0.2	0.8	4.8
234	4.4	1.7	0.6	5.8	0.2	0.9	0.3	0.6	1.0	0.6	71.2	6.4	6.1
324	2.5	4.4	0.0	0.6	0.5	6.7	2.0	0.4	0.5	0.6	6.5	69.9	5.4
444	0.6	1.0	0.3	0.7	0.3	0.4	6.8	4.9	5.8	5.1	4.7	4.2	65.3

Table 5.15: Confusion matrix for 4 LSTM layer model with learning rate schedule Equation (5.2) trained on triples, double to triples, and false data from a 150MeV over 16384 epochs. The testing data used is from the MCDE model test1 150MeV 180K beam.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	76.3	4.0	1.6	2.8	2.7	1.9	6.4	0.5	0.2	0.0	2.5	1.0	0.3
132	3.4	76.7	2.5	2.0	2.3	2.8	0.3	0.0	5.5	0.5	1.1	2.6	0.3
213	1.5	3.0	77.7	3.2	1.7	2.6	0.8	5.4	2.8	1.1	0.1	0.0	0.2
231	2.7	2.0	3.7	75.8	3.2	2.0	0.0	0.1	0.9	2.6	6.1	0.7	0.2
312	2.9	2.3	1.8	2.3	76.4	3.2	3.4	1.3	0.7	5.4	0.0	0.2	0.2
321	1.5	2.6	2.8	2.1	3.5	76.4	1.1	3.0	0.0	0.2	0.5	6.1	0.3
124	4.4	0.5	0.9	0.1	3.3	1.3	72.9	9.3	0.4	0.6	0.3	1.0	4.9
214	0.6	0.3	5.2	0.1	1.8	3.8	6.6	75.1	0.2	1.0	0.4	0.5	4.4
134	0.7	5.2	4.2	2.1	1.1	0.2	0.6	0.5	72.6	7.4	1.5	0.6	3.5
314	0.0	1.3	2.4	4.6	6.4	0.4	0.5	1.2	7.4	69.8	0.3	0.9	4.9
234	3.7	1.7	0.3	6.0	0.1	0.9	0.3	1.1	1.3	0.6	72.1	7.1	4.9
324	1.6	4.7	0.1	0.6	0.3	6.2	1.1	0.4	0.6	0.7	6.7	73.6	3.5
444	0.6	1.6	0.0	0.9	0.0	0.0	6.3	6.3	4.1	4.1	8.2	6.0	62.1

Table 5.16: Confusion matrix for 4 LSTM layer model with dense layers of 256 and 128 neurons respectively, with the learning rate schedule Equation (5.2) trained on triples, double to triples, and false data from a 150MeV over 16384 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	77.8	4.0	1.6	3.1	2.7	1.3	5.0	0.8	0.1	0.0	2.2	1.2	0.1
132	3.4	76.7	1.8	1.8	2.2	2.2	0.2	0.0	7.4	0.7	0.8	2.3	0.3
213	1.8	3.3	75.8	4.2	2.1	1.5	0.5	5.8	3.1	1.2	0.2	0.1	0.3
231	3.0	2.1	3.3	78.4	2.7	1.5	0.1	0.3	1.4	2.1	4.5	0.3	0.2
312	2.1	2.0	2.1	2.1	77.8	4.0	3.0	0.5	0.6	5.0	0.2	0.3	0.1
321	1.2	3.2	3.3	2.8	4.5	73.7	1.1	3.4	0.0	0.5	0.4	5.5	0.4
124	6.0	0.3	0.5	0.1	3.3	1.2	74.5	8.2	0.6	0.5	0.5	0.9	3.4
214	0.8	0.1	5.8	0.3	1.9	3.7	7.4	74.0	0.5	0.7	0.2	0.4	4.3
134	0.2	6.2	3.3	1.9	0.8	0.0	0.5	0.5	71.5	8.9	0.7	0.5	4.9
314	0.1	0.7	2.0	3.5	5.8	0.3	0.2	1.1	8.5	71.7	0.6	0.9	4.7
234	4.8	2.0	0.5	5.5	0.2	0.4	0.5	0.5	0.7	0.3	71.8	7.8	4.9
324	1.4	2.9	0.2	0.7	0.5	5.5	1.0	0.5	0.8	0.5	7.1	74.2	4.7
444	0.9	0.6	0.5	0.6	0.9	0.4	4.6	5.0	4.8	4.4	4.3	4.8	68.2

Table 5.17: Confusion matrix for 4 LSTM layer model with dense layers of 256 and 128 neurons respectively, with the learning rate schedule Equation (5.2) trained on triples, double to triples, and false data from a 150MeV over 16384 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	5.5	1.9	2.6	2.9	1.9	5.5	0.7	0.0	0.0	2.6	1.0	0.0
132	2.4	80.3	2.2	0.7	1.0	2.6	0.5	0.0	4.6	0.2	1.0	4.1	0.5
213	1.0	2.4	77.6	4.8	1.9	2.4	0.2	6.3	2.9	0.5	0.0	0.0	0.0
231	2.9	1.9	5.8	73.5	4.1	1.4	0.0	0.2	0.5	3.1	5.5	0.5	0.5
312	1.9	3.4	0.5	1.7	77.6	3.4	3.1	1.0	0.5	6.3	0.0	0.0	0.7
321	2.2	1.4	3.4	1.7	2.2	77.8	1.7	3.6	0.0	0.0	0.2	5.3	0.5
124	5.4	0.6	0.6	0.2	2.9	1.3	74.5	7.8	0.7	0.1	0.2	0.9	4.9
214	1.2	0.0	4.9	0.6	1.8	3.2	7.2	74.1	0.4	0.6	0.2	0.3	5.4
134	0.3	6.6	4.4	2.0	1.2	0.0	0.7	0.5	70.8	8.0	1.3	0.7	3.6
314	0.1	0.3	1.3	4.0	6.0	0.4	0.4	1.3	8.2	71.7	0.4	0.6	5.3
234	3.4	1.5	0.4	5.0	0.3	0.6	0.1	0.3	0.8	0.2	74.1	7.7	5.5
324	2.1	4.2	0.1	0.6	0.6	5.6	1.3	0.3	0.5	0.5	7.1	71.9	5.3
444	0.5	1.0	0.5	1.0	0.4	0.4	5.3	5.2	5.1	4.8	5.9	5.3	64.7

Table 5.18: Confusion matrix for 4 LSTM layer model with dense layers of 256 and 128 neurons respectively, with the learning rate schedule Equation (5.2) trained on triples, double to triples, and false data from a 150MeV over 16384 epochs. The testing data used is from the MCDE model test1 150MeV 180K beam.

Comparing the GRU and LSTM models with the deep residual fully connected (DRFCN) model from [2] in each classification at the dosage rate of 100kMU/min, we see in Table 5.9 that the GRU model outperforms the deep fully connected model in three categories and is within 10% in all categories. Similarly, the LSTM only outperforms in two categories but is within 6% as shown in Table 5.19. Finally in Table 7.1 we see a comparison of overall accuracy and the load times.

Class	LSTM	DRFCN	LSTM - DRFCN
123	80.0	79.1	0.9
132	75.8	76.0	-0.2
213	72.5	76.4	-3.9
231	75.9	80.7	-4.8
312	76.1	82.4	-6.3
321	73.5	76.5	-3
124	77.1	76.0	1.1
214	74.2	75.0	-0.8
134	74.2	75.4	-1.2
314	73.7	75.4	-1.7
234	68.5	73.6	-5.1
324	69.6	75.3	-5.7
444	68.3	72.6	-4.3

Table 5.19: Comparison of LSTM model with deep fully connected network.

Another test was run on the 4 layer LSTM with 2 dense layers model except the number of neurons per dense layer was increased from 128 and 64 to 256 and 128. The dropout rate remained at 0.2. Table 5.20 shows the comparison results. This model is within 5% within every classification. Tables 5.16 through 5.18 show the confusion matrices for this model.

Class	LSTM	DRFCN	LSTM - DRFCN
123	77.8	79.1	-1.3
132	76.7	76.0	0.7
213	75.8	76.4	-0.6
231	78.4	80.7	-2.3
312	77.8	82.4	-4.6
321	73.7	76.5	-2.8
124	74.5	76.0	-1.5
214	74.0	75.0	-1
134	71.5	75.4	-3.9
314	71.7	75.4	-3.7
234	71.8	73.6	-1.8
324	74.2	75.3	-1.1
444	68.2	72.6	-4.4

Table 5.20: Comparison of the 4 layer LSTM with dense layers of 256 and 128 neurons with deep fully connected network at 100kMU dose rate.

5.4 LSTM with Residual blocks

Seeing that the LSTM layers alone were leading to overfitting, the deep residual fully connected model (DRFCN) from [2] was combined with varying numbers of LSTM layers. The models tested used 1, 2, or 4 LSTM layers in front of 64, 128, 256, or 512 fully connected layers (FCL) shaped into residual blocks with 8 FCL in each block. The models were trained in 4096 epochs, 2048 batch size, 128 neurons per LSTM layer, 256 neurons per fully connected layer, and the learning rate scheduler seen in Equation (5.2). Overall, the models are not overfitting as the validation accuracies and testing accuracies are not drastically different.

5.4.1 1 LSTM layer with 256 Fully Connected Layers

One of the better performing models from the above trials was 1 LSTM layer with 256 FCL. Although it achieved a lower validation accuracy of 75%, it can be seen from the confusion matrix that its testing accuracy is significantly higher than the model discussed in subsection 5.3.1. The testing accuracy of this model if 74.6%. Table 5.21 shows accuracy comparison between this model and the deep fully connected model. Although this model does not outpeform the DRFCN, the accuracy remains close across all classes.

Class	LSTM	DRFCN	LSTM - DRFCN
123	78.7	79.1	-0.4
132	74.7	76.0	-1.3
213	76.4	76.4	0.0
231	79.2	80.7	-1.5
312	79.5	82.4	-2.9
321	74.4	76.5	-2.1
124	72.6	76.0	-3.4
214	74.2	75.0	-0.8
134	71.8	75.4	-3.6
314	72.3	75.4	-3.1
234	71.4	73.6	-2.2
324	71.2	75.3	-4.1
444	71.6	72.6	-1.0

Table 5.21: Comparison of the 1 layer LSTM with 256 FCL with DRFCN at the 100kMU dose rate.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	76.4	3.8	2.0	3.0	2.9	1.5	5.9	0.4	0.1	0.0	2.6	1.0	0.4
132	3.9	75.3	2.5	2.3	3.1	2.8	0.2	0.0	5.3	0.7	0.9	2.7	0.3
213	1.8	2.6	77.4	3.8	1.9	2.3	0.6	5.5	2.7	1.0	0.1	0.0	0.3
231	2.1	1.6	3.5	77.5	2.9	2.6	0.0	0.2	0.9	2.5	5.4	0.7	0.2
312	2.9	2.7	1.7	2.8	76.0	3.2	3.3	1.2	0.5	5.4	0.0	0.2	0.1
321	1.8	2.0	2.7	3.2	3.4	76.9	0.8	2.6	0.0	0.2	0.8	5.3	0.2
124	4.6	0.4	0.9	0.2	3.5	1.7	71.4	9.0	0.4	0.5	0.5	1.2	5.6
214	0.3	0.3	4.7	0.3	1.6	3.3	6.8	75.1	0.2	1.2	0.5	0.6	5.0
134	0.5	5.6	3.3	2.2	0.6	0.2	0.6	0.3	71.7	8.5	1.4	0.6	4.5
314	0.0	1.3	2.1	4.5	5.7	0.4	0.6	1.2	7.7	70.9	0.1	0.9	4.7
234	3.3	1.4	0.1	6.3	0.1	1.1	0.4	0.9	1.2	0.6	70.2	8.6	5.8
324	1.5	3.7	0.1	0.6	0.3	5.7	1.0	0.4	0.6	0.8	8.9	72.3	4.1
444	0.3	0.9	0.3	1.3	0.0	0.0	5.0	6.0	2.2	4.4	5.6	5.3	68.7

Table 5.22: Confusion matrix for 1 LSTM layer model with 256 fully connected layers with learning rate schedule Equation (5.2) 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	78.7	2.8	2.0	2.9	1.8	1.5	4.4	0.9	0.2	0.0	3.4	1.3	0.1
132	4.2	74.7	1.6	2.2	3.5	2.5	0.0	0.0	6.5	0.5	1.1	3.1	0.1
213	2.6	2.5	76.4	4.7	1.9	1.5	0.4	5.1	3.4	0.8	0.3	0.0	0.4
231	2.2	1.6	3.7	79.2	2.0	2.9	0.1	0.2	1.3	2.0	4.0	0.6	0.1
312	2.1	2.2	1.9	2.6	79.5	3.5	2.5	0.7	0.2	4.4	0.0	0.3	0.1
321	1.4	2.3	3.4	3.4	4.9	74.4	0.8	3.8	0.0	0.3	0.4	4.3	0.6
124	6.2	0.5	0.6	0.0	3.4	1.6	72.6	8.6	0.7	0.5	0.5	0.8	4.1
214	0.8	0.1	5.6	0.6	1.7	3.7	6.2	74.2	0.4	1.0	0.5	0.4	4.7
134	0.2	5.8	3.5	1.7	0.6	0.1	0.5	0.7	71.8	9.0	0.8	0.2	5.0
314	0.1	0.5	2.2	3.7	5.5	0.3	0.2	0.8	7.7	72.3	0.4	0.8	5.6
234	3.4	1.8	0.4	6.2	0.2	0.5	0.5	0.4	1.0	0.4	71.4	8.8	5.0
324	1.6	3.2	0.1	1.4	0.2	5.0	0.9	0.4	0.5	0.7	8.6	71.2	6.1
444	0.8	0.3	0.3	0.3	0.5	0.4	3.9	4.4	4.7	4.5	3.9	4.4	71.6

Table 5.23: Confusion matrix for 1 LSTM layer model with 256 fully connected layers with learning rate schedule Equation (5.2) 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	5.3	2.9	2.2	2.4	1.7	5.3	1.0	0.0	0.0	3.1	0.7	0.0
132	2.2	77.9	2.6	1.7	2.9	2.2	0.0	0.0	4.6	0.5	1.0	4.1	0.5
213	1.0	2.9	79.5	3.6	1.4	1.0	0.5	6.3	3.1	0.7	0.0	0.0	0.0
231	1.9	1.4	4.6	76.1	3.4	2.7	0.0	0.2	0.5	2.7	5.3	0.7	0.5
312	1.9	3.4	0.5	1.9	78.6	3.1	2.9	1.2	0.5	4.8	0.2	0.2	0.7
321	0.7	1.7	3.1	2.9	3.1	78.6	1.7	3.9	0.0	0.0	1.0	3.4	0.0
124	5.8	0.5	0.6	0.2	2.5	1.4	72.9	8.6	0.5	0.2	0.5	1.0	5.5
214	1.1	0.0	4.5	0.5	1.7	3.2	5.2	76.2	0.4	0.8	0.2	0.5	5.7
134	0.4	5.9	4.1	1.3	0.9	0.0	0.3	0.4	72.4	8.8	1.3	0.6	3.7
314	0.1	0.2	0.9	3.7	6.1	0.5	0.3	1.0	7.9	72.5	0.3	0.6	5.9
234	3.8	1.1	0.6	5.2	0.1	0.6	0.1	0.4	0.4	0.4	74.0	7.1	6.1
324	2.0	3.3	0.1	0.7	0.3	5.2	1.3	0.3	0.4	0.5	8.3	72.2	5.5
444	0.8	0.7	0.3	0.7	0.2	0.5	4.6	4.3	4.0	4.8	5.3	4.8	69.2

Table 5.24: Confusion matrix for 1 LSTM layer model with 256 fully connected layers with learning rate schedule Equation (5.2) 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.

5.4.2 2 LSTM layers with 128 Fully Connected Layers

Another promising model was one with 2 LSTM layers followed by 128 FCL divided into residual blocks of 8 layers. This model achieved a validation accuracy of 76%, however, similar to the previous model (section 5.4.1), the testing accuracy is much higher than the 4 layer LSTM model alone. The testing accuracy of this model if 74.2% which is seen in Tables 5.26 to 5.28. Table 5.25 shows accuracy comparison between this model and the deep fully connected model where it can be seen that it outperforms the DRFCN in 2 classes while staying close in accuracy for the other classes.

Class	LSTM	DRFCN	LSTM - DRFCN
123	79.9	79.1	0.8
132	76.7	76.0	0.7
213	74.8	76.4	-1.6
231	76.0	80.7	-4.7
312	77.8	82.4	-4.6
321	74.3	76.5	-2.2
124	74.4	76.0	-1.6
214	73.3	75.0	-1.7
134	71.8	75.4	-3.6
314	70.7	75.4	-4.7
234	71.4	73.6	-2.2
324	73.4	75.3	-1.9
444	71.0	72.6	-1.6

Table 5.25: Comparison of the 2 layer LSTM with 128 FCL with DRFCN at the 100kMU dose rate.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	78.3	4.0	1.7	2.1	2.7	1.4	5.7	0.5	0.2	0.0	2.2	1.0	0.3
132	3.6	76.9	2.4	1.8	2.6	2.5	0.2	0.0	5.6	0.5	1.0	2.7	0.3
213	2.2	2.9	77.2	3.1	1.7	2.3	0.7	5.2	3.2	1.1	0.1	0.0	0.2
231	2.9	2.2	4.5	73.9	3.0	2.0	0.1	0.2	1.3	3.0	6.2	0.6	0.2
312	2.6	3.2	1.5	2.1	76.0	3.0	3.4	1.2	0.7	5.8	0.0	0.2	0.2
321	1.9	2.5	2.8	1.7	3.3	76.5	1.1	2.9	0.0	0.3	0.7	6.0	0.4
124	4.8	0.3	0.9	0.1	3.5	1.8	72.4	7.9	0.6	0.4	0.3	1.5	5.5
214	0.7	0.3	4.9	0.1	1.5	3.5	7.8	73.6	0.1	1.3	0.6	0.5	5.2
134	0.7	5.1	3.3	1.6	0.8	0.1	0.6	0.4	72.9	9.0	1.1	0.6	3.8
314	0.0	1.4	2.2	3.8	5.4	0.4	0.7	1.1	9.2	69.3	0.2	0.9	5.4
234	3.7	2.1	0.2	5.4	0.1	0.9	0.7	0.9	1.3	0.8	70.9	7.6	5.4
324	1.1	4.5	0.1	0.4	0.3	5.9	1.2	0.3	0.7	1.0	7.4	72.5	4.6
444	0.6	1.9	0.0	1.3	0.0	0.0	5.0	5.6	3.1	3.8	4.1	7.5	67.1

Table 5.26: Confusion matrix for 2 LSTM layer model with 128 fully connected layers with learning rate schedule Equation (5.2) 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	79.9	3.8	1.8	1.9	2.3	1.1	4.4	0.6	0.2	0.0	2.7	1.1	0.1
132	3.0	76.7	1.8	1.6	3.4	2.3	0.1	0.0	7.2	0.5	0.6	2.3	0.4
213	2.3	3.2	74.8	3.9	2.8	1.5	0.6	5.5	3.8	1.0	0.2	0.1	0.3
231	3.0	2.1	4.5	76.0	2.8	2.3	0.1	0.1	1.2	2.6	4.8	0.2	0.3
312	1.9	2.9	2.3	1.8	77.8	3.5	3.0	0.8	0.5	5.1	0.0	0.2	0.1
321	1.5	3.0	3.7	1.3	4.4	74.3	1.4	3.6	0.0	0.4	0.6	5.3	0.5
124	6.4	0.3	0.4	0.0	3.0	1.3	74.4	7.7	0.5	0.5	0.4	1.1	4.0
214	0.7	0.0	6.1	0.3	1.9	3.2	7.3	73.3	0.2	1.0	0.5	0.8	4.7
134	0.2	5.7	3.3	1.4	0.5	0.1	0.7	0.5	71.8	9.5	0.8	0.3	5.2
314	0.1	0.7	1.9	3.4	5.0	0.2	0.4	1.0	9.9	70.7	0.4	1.1	5.2
234	4.6	2.2	0.4	5.3	0.2	0.4	0.5	0.5	0.9	0.5	71.4	7.9	5.4
324	1.3	3.7	0.2	0.8	0.5	4.9	1.1	0.5	0.4	0.4	7.0	73.4	5.8
444	0.8	0.3	0.3	0.3	0.6	0.5	4.0	4.4	5.0	4.6	3.7	4.4	71.0

Table 5.27: Confusion matrix for 2 LSTM layer model with 128 fully connected layers with learning rate schedule Equation (5.2) 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	1.4	1.7	3.1	2.2	6.2	0.5	0.0	0.0	3.1	0.2	0.0
132	1.9	79.3	2.4	1.4	1.9	1.9	0.0	0.0	4.8	0.0	1.2	4.6	0.5
213	1.4	2.9	76.6	3.6	1.7	1.7	0.5	7.7	2.9	1.0	0.0	0.0	0.0
231	2.9	2.2	5.3	72.8	4.1	1.9	0.0	0.2	1.2	3.6	5.3	0.2	0.2
312	3.9	3.6	1.4	1.4	76.4	2.9	3.6	0.7	0.7	5.1	0.0	0.0	0.2
321	1.2	2.7	2.7	1.4	3.6	75.7	1.7	4.3	0.0	0.0	0.7	5.3	0.7
124	6.0	0.6	0.7	0.2	2.4	1.4	73.6	7.9	0.8	0.1	0.3	0.9	5.2
214	1.0	0.0	4.9	0.5	2.0	2.9	6.9	74.2	0.3	0.7	0.2	0.6	5.8
134	0.3	5.6	3.6	1.5	1.0	0.0	0.5	0.6	71.6	9.8	1.2	1.0	3.3
314	0.1	0.2	1.0	3.0	6.8	0.3	0.2	1.3	9.6	70.6	0.4	0.6	5.8
234	3.3	2.1	0.5	4.7	0.2	0.6	0.2	0.3	1.0	0.2	72.4	8.2	6.4
324	2.1	3.7	0.1	0.4	0.5	5.5	1.4	0.2	0.7	0.6	8.0	71.1	5.6
444	0.7	0.9	0.4	0.5	0.2	0.5	4.4	3.8	4.8	4.8	5.3	5.3	68.5

Table 5.28: Confusion matrix for 2 LSTM layer model with 128 fully connected layers with learning rate schedule Equation (5.2) 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.

5.4.3 4 LSTM layers with 64 Fully Connected Layers

The 4 LSTM layers with 64 FCL divided into residual blocks of 8 layers, also performed relatively well by achieving a validation accuracy of 79% and a testing accuracy of 70% as seen from Tables 5.30 to 5.32. Table 5.29 shows the accuracy comparison between this model and the deep fully connected model and it can be seen that this model performs worse in every class.

Class	LSTM	DRFCN	LSTM - DRFCN
123	70.9	79.1	-8.2
132	69.4	76.0	-6.6
213	68.1	76.4	-8.3
231	70.1	80.7	-10.6
312	70.1	82.4	-12.3
321	68.6	76.5	-7.9
124	73.4	76.0	-2.6
214	71.6	75.0	-3.4
134	68.4	75.4	-7.0
314	68.8	75.4	-6.6
234	68.0	73.6	-5.6
324	71.3	75.3	-4.0
444	68.0	72.6	-4.6

Table 5.29: Comparison of the 4 layer LSTM with 64 FCL with DRFCN at the 100kMU dose rate.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	69.6	5.8	2.2	3.2	3.4	2.3	7.0	0.7	0.3	0.1	3.4	1.6	0.5
132	4.2	70.0	2.6	2.4	5.2	3.1	0.2	0.1	6.4	0.7	1.3	3.5	0.4
213	2.4	2.8	70.1	5.7	2.7	2.8	0.8	6.6	3.8	1.6	0.2	0.1	0.5
231	3.2	2.5	6.2	69.2	3.4	2.9	0.1	0.2	1.4	3.3	6.4	0.9	0.3
312	3.2	5.4	2.6	2.3	68.8	4.3	3.7	1.6	0.9	6.5	0.1	0.2	0.4
321	2.1	2.7	3.2	2.5	4.1	72.4	1.6	3.0	0.0	0.4	0.9	6.7	0.4
124	4.9	0.4	0.9	0.1	3.5	2.0	70.3	9.3	0.6	0.6	0.3	1.6	5.4
214	0.6	0.3	5.2	0.3	1.5	4.0	7.1	73.6	0.3	1.2	0.6	0.7	4.6
134	0.4	5.6	3.6	2.1	0.8	0.1	0.9	0.4	68.7	10.5	1.4	0.7	4.7
314	0.1	1.5	2.1	4.0	5.1	0.4	0.5	1.1	12.0	67.3	0.2	1.1	4.7
234	3.5	1.6	0.1	6.3	0.1	0.9	0.4	1.2	1.5	0.6	66.8	10.9	5.9
324	1.3	4.3	0.1	0.4	0.4	6.1	1.2	0.3	0.5	0.9	7.8	72.1	4.5
444	0.6	1.6	0.3	0.6	0.0	0.6	7.2	5.3	4.1	6.3	6.3	6.9	60.2

Table 5.30: Confusion matrix for 4 LSTM layer model with 64 fully connected layers with learning rate schedule Equation (5.2) 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	70.9	5.1	2.0	3.6	2.6	2.2	6.2	0.6	0.2	0.0	4.2	1.9	0.4
132	3.2	69.4	2.3	2.0	6.4	2.5	0.3	0.0	7.5	1.3	1.2	3.4	0.5
213	2.6	3.2	68.1	6.8	3.8	2.6	0.6	5.6	4.4	1.8	0.1	0.1	0.3
231	2.7	1.5	7.5	70.1	2.8	3.0	0.0	0.1	2.0	2.9	6.4	0.7	0.4
312	2.1	5.2	2.8	2.8	70.1	5.0	3.4	0.9	1.2	5.6	0.1	0.5	0.3
321	2.8	3.0	3.7	2.7	6.1	68.6	1.4	3.8	0.0	0.6	0.8	5.8	0.7
124	5.5	0.2	0.7	0.2	3.5	1.8	73.4	7.9	0.5	0.5	0.5	1.3	4.0
214	0.8	0.1	5.9	0.3	1.9	4.2	7.6	71.6	0.4	0.7	0.4	0.8	5.1
134	0.4	5.6	3.6	2.2	1.5	0.0	0.6	0.5	68.4	10.6	1.0	0.4	5.3
314	0.1	1.1	1.9	3.3	4.4	0.3	0.4	0.8	11.2	68.8	0.7	1.0	5.8
234	3.7	2.5	0.4	5.9	0.1	0.8	0.8	0.7	1.0	0.5	68.0	9.7	5.8
324	1.7	3.4	0.1	0.8	0.5	5.8	1.1	0.6	0.5	0.5	8.2	71.3	5.4
444	0.6	0.6	0.6	0.3	0.6	0.6	4.4	4.6	5.0	5.8	3.6	5.3	68.0

Table 5.31: Confusion matrix for 4 LSTM layer model with 64 fully connected layers with learning rate schedule Equation (5.2) 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	68.8	7.0	1.7	2.4	4.1	2.4	6.7	1.2	0.0	0.0	3.4	1.9	0.5
132	2.6	72.6	2.2	2.2	3.6	2.4	0.2	0.2	5.8	1.2	1.2	5.3	0.5
213	2.4	2.4	68.7	7.0	3.1	3.4	1.0	7.0	3.6	1.0	0.2	0.0	0.2
231	2.7	1.2	7.5	67.2	2.9	3.9	0.2	0.2	1.2	4.6	7.2	1.0	0.2
312	2.2	6.3	1.9	2.2	68.9	4.3	4.6	1.7	0.7	6.0	0.0	0.2	1.0
321	1.9	2.7	2.9	1.4	2.7	73.3	2.4	5.1	0.0	0.0	0.7	6.5	0.5
124	5.8	0.3	0.6	0.2	2.9	2.1	72.5	7.6	0.6	0.1	0.5	1.0	5.8
214	0.9	0.1	4.7	0.2	1.7	3.5	7.4	73.7	0.5	0.7	0.6	0.4	5.7
134	0.4	5.8	3.2	1.9	1.3	0.0	0.5	0.5	68.6	11.7	1.1	0.9	4.2
314	0.2	1.0	1.7	3.8	5.7	0.3	0.6	1.1	11.8	66.7	0.7	0.9	5.6
234	3.6	1.7	0.5	4.4	0.2	1.0	0.3	0.6	0.7	0.7	70.8	9.0	6.5
324	1.7	4.5	0.1	1.4	0.5	5.6	1.7	0.2	0.6	0.5	8.6	69.0	5.8
444	0.6	0.7	0.4	0.7	0.3	0.5	5.3	4.9	5.2	5.0	6.3	5.6	64.6

Table 5.32: Confusion matrix for 4 LSTM layer model with 64 fully connected layers with learning rate schedule Equation (5.2) 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 Further Studies

Transformer Blocks Initial studies using transformer blocks after recurrent layers were tested to see if the transformer embedding would increase model accuracy. The transformer hyperparameters examined are the number of blocks, the number of heads, and the head size (key dimension). Two values are tested for each hyperparameter. The hyperparameter space for this architecture is still quite large. We can not conclude whether the transformer blocks are a viable option or not only that our initial examination did not yield promising results. Further explorations could further explore the hyperparameter space or different architectures using the transformer blocks. Table 6.1 shows the initial results of the transformer study the same model without a transformer block has 77% accuracy.

Blocks	*Heads*	*Head Size*	*Epochs*	*Val Accuracy*
1	2	3	7850	78%
1	2	5	8192	79%
1	4	3	7448	76%
1	4	5	8192	78%
2	2	3	8192	78%
2	2	5	8192	78%
2	4	3	8192	78%
2	4	5	8192	78%

Table 6.1: Table of Initial Transformer Hyper-parameter Study Results.

Bidirectional LSTM The Bidirectional LSTM (BiLSTM) uses sequence from the past as well as the future to learn. A BiLSTM layer is made up of 2 connected unidirectional LSTM layers where past information flows forward through one and future information flows backwards through the second. The studies done in section 5.3.2 were extended with 4 BiLSTM layers rather than 4 unidirectional LSTM layers. The use of a BiLSTM depends on the data but it has shown promise in certain classification problems. Tables 6.2 and 6.3 show the confusion matrices for the BiLSTM models at the 100K dose rate where it can be seen that the models do not perform better than the unidirectional LSTM. However, the BiLSTM model did perform better in certain classes such as 213 and 234 where the unidirectional LSTM model achieved 72.5% and 68.5% respectively whereas the BiLSTM model achieved 75.0% and 72.8% respectively.

	123	132	213	231	312	321	124	214	134	314	234	324	444
123	78.1	3.6	2.0	2.2	2.5	1.3	4.7	0.9	0.3	0.1	3.0	1.1	0.1
132	4.6	70.4	3.1	1.8	3.1	1.8	0.2	0.2	8.9	0.7	1.5	3.5	0.2
213	2.2	2.1	75.0	3.7	2.9	1.5	0.5	5.6	4.2	1.8	0.1	0.0	0.3
231	3.4	1.8	4.3	75.1	2.2	1.7	0.2	0.2	1.5	3.4	5.6	0.3	0.2
312	2.6	1.9	2.5	2.8	75.9	3.1	3.1	1.6	0.8	4.9	0.0	0.6	0.3
321	2.5	2.5	4.1	2.6	5.4	69.9	1.4	5.0	0.0	0.5	0.6	5.1	0.5
124	6.2	0.4	0.7	0.0	3.9	1.1	71.4	9.7	0.7	0.5	0.3	0.8	4.2
214	0.9	0.2	5.5	0.3	1.4	3.1	7.5	73.5	0.5	1.3	0.4	0.4	5.1
134	0.4	4.2	3.8	1.7	0.7	0.1	0.6	0.5	72.6	9.6	1.0	0.4	4.6
314	0.0	0.6	1.7	3.4	5.0	0.4	0.7	0.9	10.1	70.4	0.5	0.9	5.5
234	4.8	1.6	0.4	4.7	0.2	0.5	0.4	0.7	0.8	0.7	72.8	6.8	5.8
324	2.2	3.0	0.1	0.5	0.5	5.6	1.0	0.6	0.8	0.4	10.1	68.5	6.7
444	1.0	0.3	0.6	0.5	0.6	0.3	4.6	5.1	6.0	5.1	4.7	3.7	67.6

Table 6.2: Confusion matrix for 4 BiLSTM layer model with dense layers of 128 and 64 neurons and 0.2 dropout rate, with the learning rate schedule Equation (5.2) for 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	77.6	3.5	1.5	2.7	2.1	1.2	5.2	0.9	0.2	0.0	3.0	1.6	0.4
132	3.0	75.1	2.1	1.7	2.9	2.5	0.2	0.0	6.6	0.8	1.2	3.3	0.6
213	2.2	2.3	74.4	4.2	3.3	1.9	0.5	5.6	3.2	1.5	0.3	0.1	0.4
231	2.9	1.0	3.7	78.0	2.7	2.1	0.0	0.2	1.1	2.9	4.6	0.5	0.3
312	2.1	2.7	1.9	2.6	75.8	3.8	3.2	1.2	0.7	5.3	0.1	0.3	0.3
321	0.7	2.6	2.8	2.1	4.7	74.9	0.9	3.9	0.0	0.1	0.8	5.7	0.8
124	5.5	0.4	0.6	0.1	3.4	1.7	72.9	8.0	0.3	0.7	0.4	1.3	4.8
214	0.9	0.1	5.2	0.3	1.7	4.0	8.0	71.8	0.4	1.0	0.7	0.6	5.3
134	0.2	5.9	3.2	1.4	0.4	0.1	0.7	0.8	70.7	10.3	0.8	0.3	5.2
314	0.1	0.4	1.6	3.8	5.2	0.3	0.4	1.1	9.6	70.9	0.6	0.8	5.3
234	3.5	1.7	0.3	5.5	0.1	0.7	0.5	0.6	1.2	0.7	71.2	8.8	5.1
324	1.3	2.9	0.1	0.8	0.7	5.8	0.9	0.4	0.7	0.5	8.5	72.1	5.2
444	0.6	0.4	0.3	0.6	0.8	0.4	5.0	4.8	4.8	4.9	5.2	5.3	66.9

Table 6.3: Confusion matrix for 4 BiLSTM layer model with dense layers of 256 and 128 neurons and 0.2 dropout rate, with the learning rate schedule Equation (5.2) for 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.

7 Conclusions and Future Work

In Section 5.1, the taki 2018 GPU cluster has the fastest GPU node. However, the performance of the GPU nodes on ada are very similar to those on taki 2018 GPU, but ada has many more available GPUs. There are 56 RTX 6000 GPUs available and only 4 GPUs available on taki 2018. The taki 2013 GPUs are too slow for the studies in this research. The number of high performance GPUs on ada are a huge advantage for performing numerous simulations simultaneously.

In Section 5.2 we conduct a hyperparameter study to determine the configurations for batch size, number of neurons, and number of recurrent layers. From the studies in Section 5.2, a batch size of 2048 is used with 4 recurrent layers each with 128 neurons in the future studies. These parameters are held constant through the remainder of Section 5.

Results from the RNN (recurrent neural network) hyperparameter study in Section 5.3 demonstrated that a learning rate scheduler benefits the model by increasing accuracy and efficiency.

Figure 5.4 demonstrates the effect a learning rate schedule has on model accuracy. The learning rate schedule improves validation accuracy between 6% to 7%. Test results showed that at a higher number of epochs and with a smaller learning rate, the accuracy of the network increases. Due to the success of the learning rate scheduler, the LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) models were trained using the scheduler. For these studies, a piece-wise function was created to illustrate the change in learning rate. From Section 5.3.1, the maximum training accuracy for the model reached was 89% with 32,000 epochs.

In Section 5.3.2, we use dropout layers in between each recurrent layer to randomly zero out 20% of the neurons in each layer. A model with 4 GRU layers of 128 neurons and 2 dense layers of 128 and 64 neurons, respectively, has a testing accuracy of 73.4%. The model is able to load from its saved state to an active state, i.e., load from disk to GPU memory in 10s. A model with 4 LSTM layers of 128 neurons and 2 dense layers of 128 and 64 neurons, respectively, has a testing accuracy of 73.2%. The model is able to load from disk in 7s. The major advantage of this model are that it contains only 6 hidden layers which leaves a tremendous amount of space for further research and growth while already having a testing accuracy of 73%. Further, in real-time imaging, loading to disk is a potentially significant advantage when treating patients.

Section 5.4 contains models that combine LSTM layers with the deep residual fully connected network that contained residual blocks of 8 fully connected layers each discussed in [2]. The number of LSTM layers varied from 1, 2, or 4 and the number of deep full connected layers varied from 64, 128, 256, and 512. The 1 LSTM layer with 256 fully connected layers achieved the highest testing accuracy of 74.6% and loads from disk in 24 seconds. The 2 LSTM layer with 128 fully connected layers and 4 LSTM layer with 64 fully connected layers achieved testing accuracies of 74.2% and 70.0%, respectively. These models show that adding LSTM layers to the deep residual fully connected network did not improve accuracy.

The key results of this work are summarized in Table 7.1. The Model column refers to the architecture of the model. The first row shows the results of the deep residual fully connected network (DRFCN) in [2]; this model has 512 fully connected layers (FCL). The row 1 LSTM, 256 FCL represents the model in Section 5.4.1. 4 LSTM w/ more neurons represents the 4 LSTM layer model with two dense layers of 256 and 128 neurons. 2 LSTM, 128 FCL represents the model in Section 5.4.2. 4 LSTM, 64 FCL represents the model in Section 5.4.3. 4 GRU represents the model with 4 GRU layers and 2 dense layers of 128 and 64 neurons. 4 LSTM represents the model with 4 LSTM layers and 2 dense layers of 128 and 64 neurons. The Accuracy column represents the overall testing accuracy of the model at the dosage rate of 100kMU/min. The Load Time column represents the observed wall clock time in seconds to load the model from its saved state to an active state, i.e., from disk to GPU memory. These measurements report observations on a reference computer, a basic laptop with an 11th Gen Intel Core i7–1165G7 CPU at 2.80 GHz with 16 GB of memory. The laptop has Intel Optane Memory H10 with 512 GB Intel QLC 3D NAND solid state drive connected by PCIe 3.0 x4 with NVMe interface. The GPU on the laptop is an Intel Iris Xe Graphics card. On a large cluster like taki or ada, described in Section 4, these times would in fact be *slower*, since the central rotating disk storage is much larger and connected only via network cables to the compute nodes. Even with high-performance fiber-optic cables, this is slower than direct connection from solid state storage inside a laptop. However, such direct connection and use of solid state storage is more realistic for the type of computer used in a clinical setting in a treatment room.

The DRFCN model has the highest accuracy of 75.8% with the load time of 47s. The models in the last two rows of the table have accuracies of 73.4% and 73.2% respectively while loading in 10s and 7s. These 4 GRU and 4 LSTM models are much simpler with only 6 hidden layers instead of 512. In particular, they have a factor 85 fewer layers while being only 2% less accurate. These

Model	Accuracy	Load Time
DRFCN (512 FCL)	75.8%	47s
1 LSTM, 256 FCL	74.6%	24s
4 LSTM w/ more neurons	74.4%	15s
2 LSTM, 128 FCL	74.2%	13s
4 LSTM, 64 FCL	70.0%	11s
4 GRU	73.4%	10s
4 LSTM	73.2%	7s

Table 7.1: Comparison of top performing models with the deep residual fully connected network (DRFCN) from [3].

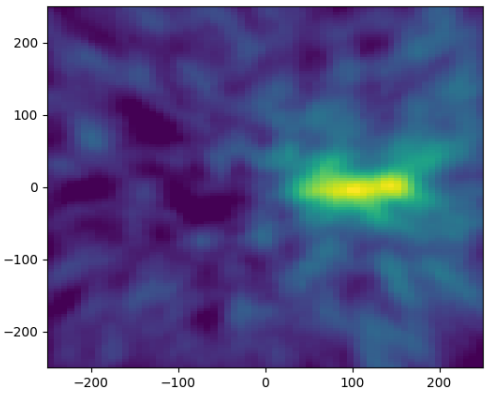
two recurrent models are also 4 times faster to disk an advantage when treating the patient.

To illustrate the effect that network event classification can have on the PG images produced from the camera data, reconstructed PG images are shown in Figure 7.1 for the GRU model and Figure 7.2 for the LSTM model. In Figures 7.1 and 7.2, there are three rows of PG image reconstructions for each dose rate corresponding the the MCDE model test1 150MeV. The images in the left column are the respective PG images reconstructed with raw data prior to NN classification, called the "uncleaned" data. The images in the right column are the respective PG images reconstructed with data after it has been corrected based on the NN classifications, called the "cleaned" data. Since each PG image is from data collected during delivery of the same 150MeV proton beam they will have the same position and range even though they are reconstructed from data collected at different dose rates. We observed an improved visual appearance of the beam in which the start point and end point are now easily distinguishable at all three dose rates. The method used to reconstruct these images is described in [3].

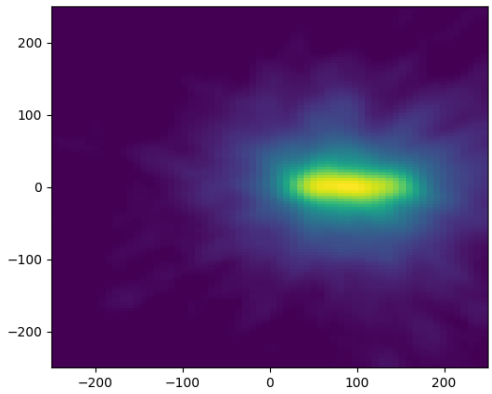
The 6 hidden layer model has a large space for improvement due to its simplicity. Transformer networks were briefly explored in Section 6 during this study and its initial results did not increase accuracy as expected. However, the hyperparameter space is very large and there is still potential in finding the optimal combination and architecture. The bidirectional LSTM is also tested in Section 6. More complex architectures than 4 recurrent layers and 2 dense layers should be explored in addition to more techniques of regularization. There is also room in the RNN and DRFCN merged models where, rather than stacking the RNN layers in front of the DRFCN, the RNN layers could be dispersed between the FCLs, placed inside the residual blocks, or placed behind the DRFCN. From the results of this work, it is still possible that the optimal configuration of hyperparameters has still not been achieved for the more complex recurrent architectures (RNNs with residual blocks and transformers). Therefore, hyperparameter searches and exploring different optimization techniques could increase the accuracy of those models.

Acknowledgments

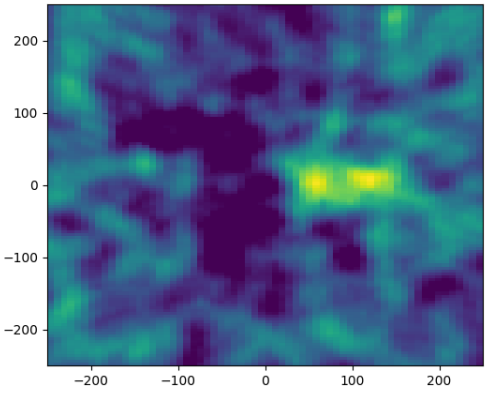
This work is supported by the grant "REU Site: Online Interdisciplinary Big Data Analytics in Science and Engineering" from the National Science Foundation (grant no. OAC–2050943). Co-author Kelly additionally acknowledges support as HPCF RA. Co-author Polf acknowledges support from the NIH. The hardware 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, OAC–1726023, and CNS—1920079) and the SCREMS program (grant no. DMS–0821311), with additional substantial



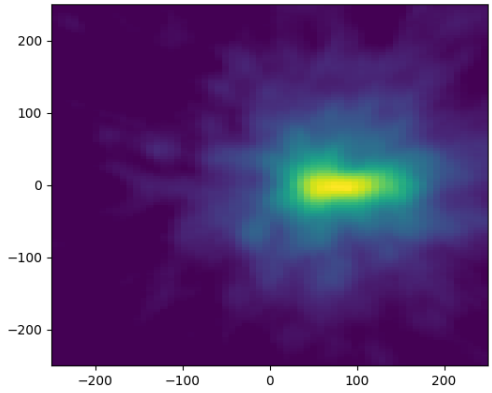
(a) 20kMU/min Uncleaned



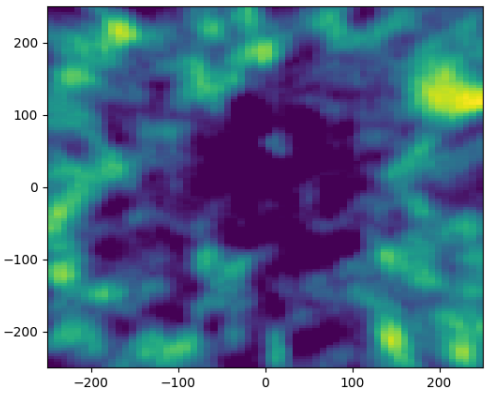
(b) 20kMU/min Cleaned



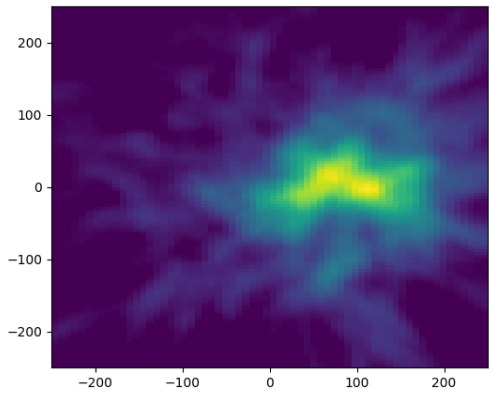
(c) 100kMU/min Uncleaned



(d) 100kMU/min Cleaned

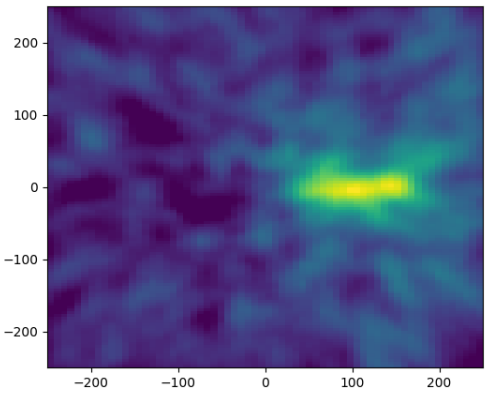


(e) 180kMU/min Uncleaned

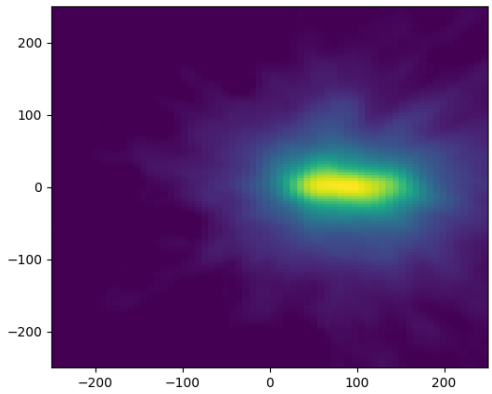


(f) 180kMU/min Cleaned

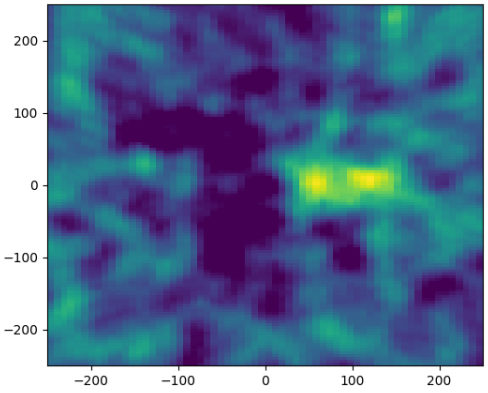
Figure 7.1: The left column (a), (c), (e) uses testing data without the NN classification for data correction, called the "uncleaned" data. The right column (b), (d), (f) uses testing data with NN classification for data correction, called the "cleaned" data with the 4 layer GRU model described in Section 5.3.2. Testing data used comes from MCDE model test1 150MeV.



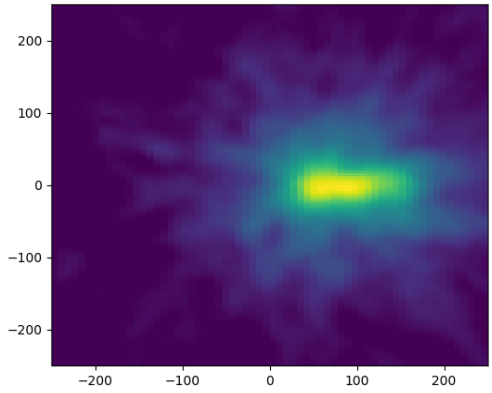
(a) 20kMU/min Uncleaned



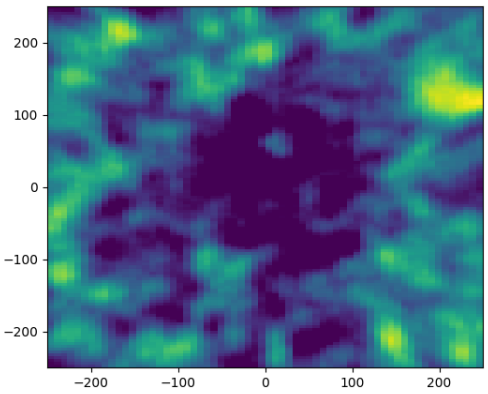
(b) 20kMU/min Cleaned



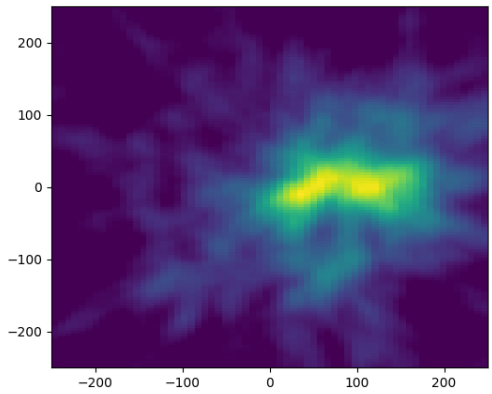
(c) 100kMU/min Uncleaned



(d) 100kMU/min Cleaned



(e) 180kMU/min Uncleaned



(f) 180kMU/min Cleaned

Figure 7.2: The left column (a), (c), (e) uses testing data without the NN classification for data correction, called the "uncleaned" data. The right column (b), (d), (f) uses testing data with NN classification for data correction, called the "cleaned" data with the 4 layer LSTM model described in Section 5.3.2. Testing data used comes from MCDE model test1 150MeV.

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] Alina M. Ali, David Lashbrooke, Rodrigo Yepez-Lopez, Sokhna A. York, Carlos A. Barajas, Matthias K. Gobbert, and Jerimy C. Polf. Towards optimal configurations for deep fully connected neural networks to improve image reconstruction in proton radiotherapy. Technical Report HPCF–2021–12, UMBC High Performance Computing Facility, University of Maryland, Baltimore County, 2021.

[2] Carlos A. Barajas. *Neural Networks for the Sanitization of Compton Camera Based Prompt Gamma Imaging Data for Proton Radiotherapy.* Ph.D. Thesis, Department of Mathematics and Statistics, University of Maryland, Baltimore County, 2022.

[3] Carlos A. Barajas, Matthias K. Gobbert, and Jerimy C. Polf. Deep residual fully connected neural network classification of Compton camera based prompt gamma imaging for proton radiotherapy, submitted (2022).

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

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555, 2014.

[6] Fengdi Guo. Comparison of feedforward and recurrent neural networks for predicting pavement roughness, 2021.

[7] 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.

[8] Jerimy C. Polf, Carlos A. Barajas, Stephen W. Peterson, Dennis S. Mackin, Sam Beddar, Lei Ren, and Matthias K. Gobbert. Applications of machine learning to improve the clinical viability of Compton camera based in vivo range verification in proton radiotherapy. *Front. Phys.*, 10:838273, 2022.

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

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

[11] Sokhna A. York, Alina M. Ali, David Lashbrooke, Rodrigo Yepez-Lopez, Carlos A. Barajas, Matthias K. Gobbert, and Jerimy C. Polf. Promising hyperparameter configurations for deep fully connected neural networks to improve image reconstruction in proton radiotherapy. In *2021 National Symposium for NSF REU Research in Data Science, Systems, and Security (REU 2021 Symposium)*, in press (2021).

[12] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into Deep Learning. Interactive deep learning book with code, math, and discussions, 2021.