Improving Gamma Imaging in Proton Therapy by Sanitizing Compton Camera Simulated Patient Data using Neural Networks through the BRIDE Pipeline

Michael O. Chen
Departments of Mathematics
Dartmouth College, USA

Julian Hodge
Dept. of Mathematics and Statistics
U. of Maryland, Baltimore County, USA

Peter L. Jin
James M. Bennett High School
Salisbury, MD, USA

Ella Protz
Department of Mathematics and Sciences
Florida Atlantic University, USA

Elizabeth Wong
Department of Mathematics
Brookdale Community College, USA

Ruth Obe
Dept. of Computer Science and of Software Engineering
U. of Houston—Clear Lake, USA

Ehsan Shakeri
Dept. of Mathematics and Statistics
U. of Maryland, Baltimore County, USA

Mostafa Cham
Dept. of Information Systems
U. of Maryland, Baltimore County, USA

Matthias K. Gobbert
Dept. of Mathematics and Statistics
U. of Maryland, Baltimore County, USA

Carlos A. Barajas
Dept. of Mathematics and Statistics
U. of Maryland, Baltimore County, USA

Vijay R. Sharma
Dept. of Radiation Oncology
U. of Maryland School of Medicine, USA

Sina Mossahebi
Department of Radiation Oncology
U. of Maryland School of Medicine, USA

Lei Ren
Department of Radiation Oncology
U. of Maryland School of Medicine, USA

Stephen W. Peterson
Dept. of Physics
U. of Cape Town, South Africa

Jerimy C. Polf
M3D, Inc.
USA

***Abstract*—Precision medicine in cancer treatment increasingly relies on advanced radiotherapies, such as proton beam radiotherapy, to enhance efficacy of the treatment. When the proton beam in this treatment interacts with patient matter, the excited nuclei may emit prompt gamma ray interactions that can be captured by a Compton camera. The image reconstruction from this captured data faces the issue of mischaracterizing the sequences of incoming scattering events, leading to excessive background noise. To address this problem, several machine learning models such as Feedfoward Neural Networks (FNN) and Recurrent Neural Networks (RNN) were developed in PyTorch to properly characterize the scattering sequences on simulated datasets, including newly-created patient medium data, which were generated by using a pipeline comprised of the GEANT4 and Monte-Carlo Detector Effects (MCDE) softwares. These models were implemented using the novel 'Big-data REU Integrated Development and Experimentation' (BRIDE) platform, a modular pipeline that streamlines preprocessing, feature engineering, and model development and evaluation on parallelized GPU processors. Hyperparameter studies were done on the novel patient data as well as on water phantom datasets used during previous research. Patient data was more difficult than water phantom data to classify for both FNN and RNN models. FNN models had higher accuracy on patient medium data but lower accuracy on water phantom data when compared to RNN models. Previous results on several different datasets were reproduced on BRIDE and multiple new models achieved greater performance than in previous research.**

***Index Terms*—Proton beam therapy, Compton camera, Classification, Recurrent neural network, PyTorch**

I. Introduction

Radiotherapy, a common cancer treatment, involves delivering a clinically determined dose of X-ray, electron, or proton radiation to a tumor for destruction. Unlike X-ray radiotherapy, which delivers a high initial dose to the tumor and often exposes healthy tissue to radiation, proton beam therapy provides a more precise and targeted approach, significantly reducing unnecessary radiation exposure [7], [10]. This advantage stems from the fact that proton beams release most of their energy in a concentrated area known as the Bragg peak. Accurately determining the Bragg peak location is essential to ensure effective treatment and the protection of healthy tissue in medical settings.

A Compton camera is a gamma-ray imaging device that detects and visualizes gamma rays through the Compton scattering process. It can detect prompt gamma rays generated by the proton beams interacting with patient tissues, enabling real-time determination of the Bragg peak location. However, uncertainties arise due to the non-zero time resolution of the Compton camera, which can lead to simultaneous detection of interactions [2]. Using machine learning, the background noise in the reconstructed images can be mitigated. Deep learning techniques, such as fully connected neural networks (FCNs) or recurrent neural networks (RNNs), can differentiate scattering interactions, filter out false data, and improve image reconstruction accuracy, thus enhancing the precise localization of the proton beam.

Previous studies use the same dataset derived from Monte Carlo simulations of a water phantom, a constant density water medium, in the training and testing of models [5], [9], [15]. Recent improvements have allowed novel datasets based off simulated *patient* medium to provide comparable data to clinical cases. It is unclear whether these results can immediately transfer to this new data. Hence, this work aims to test the extensibility of these previous results onto the simulated patient data. We first improve the PyTorch implementation of multi-processor training in [9]. Next, we refine the best models of [1] and [6] on the new data in our improved environments. Finally, we perform a hyperparameter study on the new data for both FCN and RNN models for comparison with water phantom data results.

Finally, please note that all code mentioned in this paper, including the implementation of BRIDE, can be found in [3] in the folder `2024-projects/team-2`. The full project code base exists within the hpcf cluster at UMBC.

The remainder of this paper is organized as follows: Section II covers the background information on proton beam therapy. Section III begins with technical background on machine learning and the physical and software resources utilized. It reviews the data generation process. Section IV covers the results and comparisons of our tests on both water phantom and patient data. Section V summarizes additional studies done in this research. Section VI concludes our findings and summarizes potential clinical applications.

II. Application Background

A. Proton Beam Radiotherapy

Radiotherapy is an advanced cancer treatment that delivers a high dosage of radiation to destroy cancer cells. One of the most common treatments, X-ray therapy, delivers the full dosage of radiation upon entering the body en route to the tumor, leading to unnecessary radiation exposure.

In the relatively novel proton beam therapy, proton beams deposit the vast majority of the radiation at the tumor site. Unlike in X-ray therapy, the proton beam does not travel further than the tumor site, minimizing damage to healthy tissue. Hence, this treatment is widely considered to be more effective for some types of cancer. Importantly, using the Bragg peak of proton therapy allows for treatment plans that deliver radiation precisely to the tumor and avoid surrounding healthy tissue. However, the small distance between healthy and tumor tissue requires the Bragg peak to be accurately located [12]. Clinicians need real-time information to determine its location for safe treatment. Fig. 1 demonstrates the safety margin for treating a lung tumor with minimal damage to the heart.

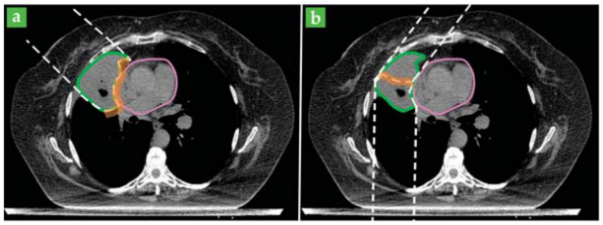


Fig. 1: **a:** Optimal proton treatment beam (dashed) targeting a tumor (green) with safety margin (orange) that overlaps with the healthy heart tissue (magenta). **b**: Suboptimal treatment plan of two beams that do not overlap with heart [12].

B. Compton Camera and Image Reconstruction

When high energy proton beams collide with patient matter, prompt gamma rays are emitted and captured by the Compton camera [12]. The Compton camera produces real-time images to visualize the prompt gamma radiation and scattering events. Prompt gammas are emitted at a specific angle of displacement determined from the energy levels of the proton collision with the nucleus. Prompt gamma rays interact with the camera: for each interaction, the camera calculates (x_i, y_i, z_i) coordinates and the energy level e_i of the scatter. The Compton cone of emission is used to project the potential trajectories that this collision could occur. Using this cone of emission, the origin of the gamma is determined mathematically [11].

After the scattering events, image reconstruction algorithms recover a visualization of the path of the proton beam. However, the Compton camera has a major problem of a finite time resolution. It does not explicitly record the sequential order of the prompt gamma ray interactions. This causes noise in the reconstructed images, rendering them partially unusable in a medical setting [11].

1) Scatter Types: Due to prompt gamma radiation emission at approximately the speed of light, the sequence of the scattering events can become distorted. To help identify false events that create image noise, scattering events are organized as scatter types. There are 13 types of scatterings, which are separated into three groups.

1) **True Triples:** True triples are three sequential interactions with the Compton camera. The ordering of the interactions can be one of six combinations: 123, 132, 213, 231, 312, and 321. Out of these, only the 123 combination is currently usable for image reconstruction purposes.
2) **Doubles to Triples (DtoT):** DtoT events are double and single interactions that occur independently of each other but are detected as one event by the Compton camera. There are six possible combinations of this

event: 124, 134, 214, 234, 324, and 314, where the ”4” refers to the second prompt gamma interaction in the misdetection events.

3) **False Triples:** False triples are events that are detected as a true triple, but in reality are comprised of three independent events. These false events may result in images with noise and must be discarded [2], [8], [11].

III. Methodology

A. Machine Learning

In order to make proton therapy more effective, real-time imaging is required for treatment to verify proton beam dosage and the Bragg peak location. Machine learning can be used to sanitize Compton camera data by removing classified false events; it uses algorithms to identify and generalize specific trends within data. The main form of machine learning employed in this work are neural networks. Using complex techniques, neural networks were designed to emulate the ability of the human brain to discover relationships between each observation and its class. The two main types applied in this work are Feed Forward Neural Networks and Recurrent Neural Networks, as they have shown some success in past work [14].

1) Feed Forward Neural Networks: The simplest deep learning model is the Feed Forward Neural Network (FNN). FNNs involve unidirectional flow of information, channeling the input through the hidden activation layers to become the output [13]. Each neuron in one layer is connected to every neuron in the next layer, forming a fully-connected layer; in other words, the output of a neuron in a layer serves as the inputs for all of the neurons on the subsequent layer. These models are widely used for many machine learning applications, including search engines, image classification and economic forecasting.

2) Recurrent Neural Networks: Recurrent Neural Networks (RNN) are multi-directional networks with recurrent units and equal weights in each layer. Each recurrent unit has forward activation units with a memory state needed to store information about the network at particular epoch and a backward propagation for training the network. While remembering the last input, the memory state is updated continuously with new information [14]. The drawbacks of RNNs include sensitivity to hyperparameter changes and exploding gradient issues.

A type of RNN is the Long Short Term Memory (LSTM) neural network, which features long-term dependencies. In addition to the defaults in RNNs, information of an LSTM can be sent to the input gate for memory, discarded in the forget gate, or produced as output from memory [14]. These three gates in the LSTM feature a unique aspect, the memory cell, which stores information for later use during model training. Though LSTMs are typically used for natural language processing and time series forecasting, the model was projected to perform well on our datasets due to its robust learning capabilities.

3) Custom Pairwise Loss Function: The data used only lists 13 classes; it does not explicitly have the three event groups (true triple, DtoT, false triple) as classes. Hence, machine learning models wouldn’t explicitly recognize three separate groupings. To address this, we implemented a novel custom loss function designed to boost accuracy by penalizing incorrect predictions outside the event group more than those inside the correct event group. We defined the custom pairwise loss term L_P to be

$$L_P = (1 + \text{avg}((D \cdot t) \cdot p))^h \tag{1}$$

where h is the penalty (a hyperparameter), D is a 13×13 matrix [1], and t and p are the one-hot encodings of the target and prediction vectors of shape $13\times$[batch size]. This custom loss function was used for both FCNs and LSTMs.

B. Related Works

In recent years, significant progress has been made in the application of machine learning to image reconstruction from prompt gamma radiation. It is important to note that prior research was all performed on water phantom data as opposed to simulated patient data. The research in [1] focuses on optimizing deep FCNs for image reconstruction; they achieved a best accuracy of 75%. The research in [6] had a primary contribution in terms of RNNs. The RNNs demonstrated comparable performance to the FCNs in [1], with a key advantage: the RNN models had a simpler architecture, leading to faster loading times and enhanced usability. Finally, [9] built upon the prior two works and used distributed learning via PyTorch versus the Tensorflow setup in the past. Their peak accuracy was lower, at 69%.

C. Dataset Generation

Due to the measurement limitations encountered during clinical proton beam delivery, the GEANT4 toolkit is used to perform high fidelity simulations of the interactions of a proton beam with simulated matter, producing prompt gamma rays that interact with a Compton camera. The resulting data is then fed into the Monte-Carlo Detector Effects (MCDE) modeling package to determine scattering types and orderings for class labels.

Novel patient data is incorporated into the current preprocessing pipeline early on with CT images made from patient tissue measurements. Tissue measurements are non-uniform, corresponding to different densities as opposed to a uniform density water phantom. We generated three major datasets with 499,000 rows (simulated patient), 1.8 million (water phantom), and 3.8 million rows, which is a hybrid collection of both the water phantom and simulated patient datasets. Class balancing the raw MCDE data to create training data results in less of the patient simulated data being usable when compared to class balancing the combined dataset of both the water phantom and simulated patient data. Consequently, the hybrid collection contains more data than simply adding the total rows of the datasets after they are put through preprocessing.

[1] See [3] for matrix.

Each row of the dataset corresponds to the data of the triple scatter interaction and its scatter type class label. Training data is preprocessed to consist of 15 features total. The first 12 features consist of three groups of the energy level and coordinates (e_i, x_i, y_i, z_i), each group corresponding to a prompt gamma detection in the triples scatter interaction. The remaining 3 features correspond to the euclidean distance between each scatter, as introduced in [2]. All features are normalized, scaled, and have outliers removed.

D. The BRIDE Platform

In this research, we developed the Big-Data REU Integrated Development and Experimentation (BRIDE) coding platform to ensure reproducibility, expandability, rigorous testing, and flexible experimentation. BRIDE is a workflow built using PyTorch Lightning and contains the processes of dataset formation, model implementation, evaluation, and selection, as displayed in Fig. 2.

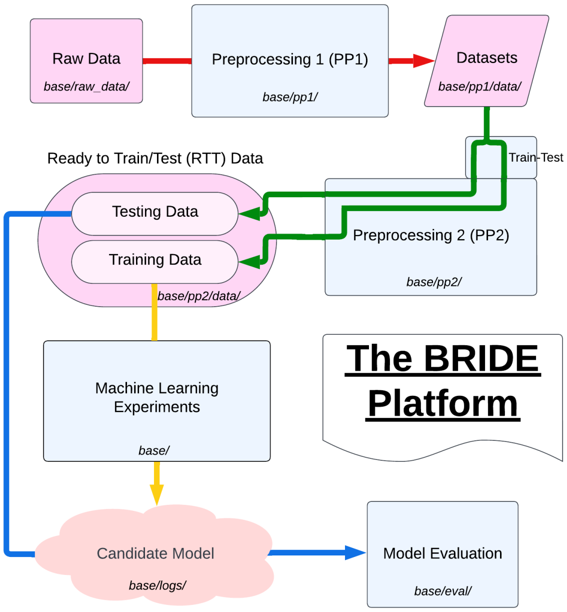


Fig. 2: The structure of the BRIDE platform.

BRIDE is modular because each process is completely separate from every other process. In particular, PP1 contains the first step of preprocessing which is converting raw data to an array, class balancing, and cleaning data, while PP2 contains feature engineering such as scaling. Rigorous testing is enforced with no possibility of data leaks. In BRIDE, each run has a distinct `run_id` for functionality and record-keeping purposes. The platform allows for the use of Tensorboard for visually tracking runs; it has csv logging for results and checkpointing callbacks for accessing models, along with automatic training curve and confusion matrix creation. Further details on BRIDE can be found in [4].

E. Hardware and Software

For this research, we utilized the Graphics Processing Units (GPU) in the ada cluster maintained by the UMBC High Performance Computing Facility (hpcf.umbc.edu). The ada GPU cluster consists of four nodes with eight 2018 Ti GPUs each, seven nodes with eight RTX 6000 GPUs each, and two nodes with eight RTX 6000 GPUs and an additional 384 GB of memory each.

Deep learning models were built and implemented using PyTorch v2.3.1 (https://PyTorch.org). For data preprocessing and manipulation, we used scikit-learn v1.3.0 (https://scikit-learn.org/stable/), pandas v2.2.2 (https://pandas.pydata.org/), and numpy v1.26.4 (https://numpy.org/). To visually display our results, we used matplotlib v3.8.4 (https://matplotlib.org/) and seaborn v0.13.2 (https://seaborn.pydata.org/). Our models were built inside of the python environment Anaconda3 (https://www.anaconda.com/).

1) Parallelization: Parallelization is widely used in various computing aspects, where multiple nodes work on distinct aspects of a problem simultaneously. It can be highly beneficial in terms of speed and efficiency, with more processors working on a single task.

We utilized PyTorch's Distributed Data Parallel (DDP) for the training of machine learning models among several GPUs. The DDP process involves distributing the input data in unique subsets among the various GPUs; each processor then uses its given set of data to construct a model. With forward and backward passes, gradients among the GPUs are synchronized and averaged, and weights are updated on the entire model.

IV. Results

A. Patient Data

In order to test the extensibility of machine learning models from water phantom to patient data and to gain insights on patient data, we conduct 2 initial hyperparameter studies (one study each for FCNs and LSTMs) on a 499,000 row simulated patient dataset. These tuning studies can be divided into three stages:

- Stage 1: A grid of tests was run to determine a set of constant starting parameters and to identify 3 candidate hyperparameters that tuning could benefit.
- Stage 2: Hyperparameter Importance. For each of the 3 candidate hyperparameters, 2 values are chosen, and (2)(2)(2) = 8 tests are done to identify the 2 most influential parameters. The hyperparameter that is less influential is fixed at an optimal value.
- Stage 3: Final Tuning. For each of the 2 influential hyperparameters, 3 values are chosen, and (3)(3) = 9 tests are done to identify the optimal configuration of hyperparameters.

1) FCN Hyperparameter Study: After testing a large grid of hyperparameters for Stage 1, we identified the three candidate hyperparameters listed in Table I. An intuitive 'binary' labeling scheme was used to name tests: the two values of each of the candidate hyperparameters are assigned '0' and '1', and each run is a string of these binary digits in the order *batch size* | *dropout* | *neuron configuration*.

Hyperparameter	0	1
Batch Size	1024	4096
Neuron Configuration	18 layers, 582 ANL[2]	12 layers, 343 ANL[3]
Dropout	0.05	0.15

TABLE I: FCN Stage 1 Candidate Hyperparameters.

For Stage 2, to identify the 2 most influential hyperparameters from the candidate hyperparameters, we implemented runs adjusting just the candidate hyperparameter in consideration; all others were held constant. The results of the 8 tests are shown in Table II. Table III contains the constant hyperparameters. Each test was run with periodic checkpointing and early stopping so that the job would end after 500 epochs without improvement in validation loss. If a run displayed erratic behavior or very low validation accuracy, it was not tested.

Test	Epoch	Max Train Acc.	Max Val. Acc.	Test Acc.
000	604	0.7005	0.52	0.517
001	574	0.36	0.36	-
010	551	0.669	0.53	0.523
100	658	0.7566	0.53	0.515
011	645	0.5607	0.53	0.527
110	576	0.7316	0.52	0.525
101	709	0.43	0.44	-
111	616	0.5975	0.53	0.532

TABLE II: Stage 2 Hyperparameter Tests.

Hyperparameter	Value
Hardware	2 RTX6000 GPUs
Validation Split	0.1
Loss Function	Cross Entropy + Custom Pairwise Loss (p=1)
Optimizer	AdamW
Activation Function	ReLU
Learning Rate	0.001
Learning Rate Change	0.95
Learning Rate Step	500
L2	0.01

TABLE III: FCN Stage 2 Constant Hyperparameters.

From Stage 2 results, we can conclude that batch size has little effect on validation and test accuracy. A larger batch size also tends to lead to overfitting. Implementing dropout had a small positive effect on validation and test accuracy if training did not diverge; as expected, it decreased training accuracy significantly since it is used to prevent overfitting. The second neuron configuration led to significantly better predictions with a slight decrease in training accuracy, compared to the first configuration. All models were overfitting. Overall, dropout and neuron configuration were the 2 influential hyperparameters to be studied in Stage 3. In addition, Stage 3 constant parameters were adjusted according to these results to a batch size of 512 and a learning rate of 0.0008.

For Stage 3 hyperparameter tests, the values for dropout and neuron configuration are shown in Table IV. A similar naming scheme is used with the ternary digits 1, 2, and 3 are used. Results of these tests are shown in Table V. Note the lack of peaks in the descriptions of the validation curves in comparison to Stage 2; this may be due to the measures designed to reduce overfitting causing plateaus.

Hyperparameter	1	2	3
Neuron Configuration	12 layers 315 ANL	8 layers 272 ANL	8 layers 68 ANL[4]
Dropout	0.05	0.2	0.4

TABLE IV: FCN Stage 3 Hyperparameters.

Test	Epoch	Max Train Acc.	Max Val. Acc.	Test Acc.
11	578	0.6068	0.53	0.528
12	1531	0.51	0.52	0.514
13	581	0.39	0.41	-
21	564	0.6232	0.56	0.550
22	3649	0.5387	0.55	0.540
23	1999	0.475	0.50	0.495
31	2061	0.5274	0.54	0.534
32	1999	0.4698	0.48	0.483
33	1393	0.3306	0.35	-

TABLE V: FCN Stage 3 tests; highest accuracy result is highlighted in green.

In conclusion, the model with greatest accuracy, highlighted in green in Table V achieved an accuracy of 55%. The accuracy learning curves and confusion matrices for this model are included in Fig. 3 and Fig. 4. The adjusted hyperparameters for this model were: Neuron Configuration: 8 layers, 272 ANL; Dropout: 0.05; Batch Size: 512; L2: 0.01; Loss Function: Cross Entropy + Custom Pairwise Loss; Optimizer: AdamW; Activation: ReLU. It is clear that the model is still overfitting, but does much better at classifying false triples.

2) LSTM Hyperparameter Study: In light of the previous success of a 4-layer LSTM model in [6], we implemented versions of this model on simulated patient data. As part of Stage 1, we found three candidate hyperparameters to tune as displayed in VI. A binary naming scheme was utilized.

Hyperparameter	0	1
Batch Size	2048	4096
Hidden Layers	[128, 64]	[128,128,128,128]
Dropout	0.15	0.45

TABLE VI: LSTM Stage 1 Candidate Hyperparameters.

For Stage 2, to identify the two influential hyperparameters from among the candidates, we adjusted each candidate hyperparameter singularly, as before. The hidden layer values are the neuron dimensions of the hidden layers after the 4 LSTM layers. Table VII contains the constant parameters. This stage's results are shown in Table VIII.

[2] Average neurons per layer; full configuration is [4096, 2048, 1024, 512, 256, 256, 256, 256, 256, 256, 256, 256, 256, 256, 128, 64, 32, 16].

[3] Average nuerons per layer; full configurations is [512, 1024, 512, 256, 256, 256, 256, 256, 256, 256, 256, 128].

[4] Full configurations: [1024, 512, 256, 256, 256,256,256, 256, 256, 256, 128, 64], [512, 256, 256, 256, 256, 256, 256, 128], and [128, 64, 64, 64, 64, 64, 64, 32]

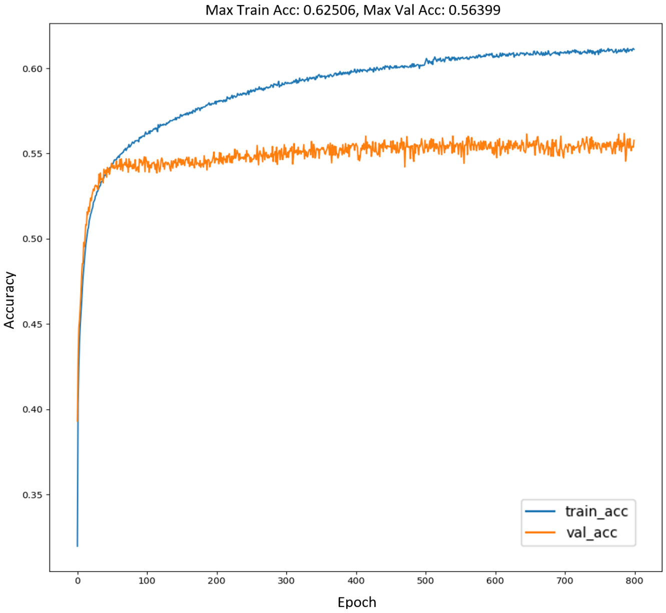
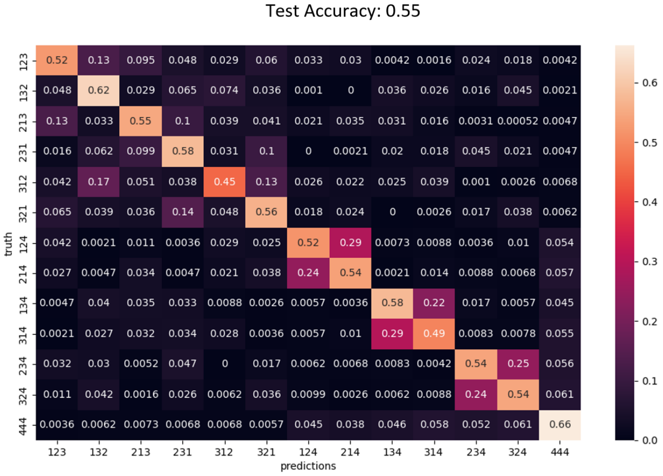


Fig. 3: Accuracy curves for Test 21.



	123	132	213	231	312	321	124	214	134	314	234	324	444
123	0.52	0.13	0.095	0.048	0.029	0.06	0.033	0.03	0.0042	0.0016	0.024	0.018	0.0042
132	0.048	0.62	0.029	0.065	0.074	0.036	0.001	0	0.036	0.026	0.016	0.045	0.0021
213	0.13	0.033	0.55	0.1	0.039	0.041	0.021	0.035	0.031	0.016	0.0031	0.00052	0.0047
231	0.016	0.062	0.099	0.58	0.031	0.1	0	0.0021	0.02	0.018	0.045	0.021	0.0047
312	0.042	0.17	0.051	0.038	0.45	0.13	0.026	0.022	0.025	0.039	0.001	0.0026	0.0068
321	0.065	0.039	0.036	0.14	0.048	0.56	0.018	0.024	0	0.0026	0.017	0.038	0.0062
124	0.042	0.0021	0.011	0.0036	0.029	0.025	0.52	0.29	0.0073	0.0088	0.0036	0.01	0.054
214	0.027	0.0047	0.034	0.0047	0.021	0.038	0.24	0.54	0.0021	0.014	0.0088	0.0068	0.057
134	0.0047	0.04	0.035	0.033	0.0088	0.0026	0.0057	0.0036	0.58	0.22	0.017	0.0057	0.045
314	0.0021	0.027	0.032	0.034	0.028	0.0036	0.0057	0.01	0.29	0.49	0.0083	0.0078	0.055
234	0.032	0.03	0.0052	0.047	0	0.017	0.0062	0.0068	0.0083	0.0042	0.54	0.25	0.056
324	0.011	0.042	0.0016	0.026	0.0062	0.036	0.0099	0.0026	0.0062	0.0088	0.24	0.54	0.061
444	0.0036	0.0062	0.0073	0.0068	0.0068	0.0057	0.045	0.038	0.046	0.058	0.052	0.061	0.66

Fig. 4: Test set confusion matrix for Test 21.

Hyperparameter	Value
Hardware	2 RTX6000 GPUs
Validation Split	0.1
Loss Function	CrossEntropy + Custom Pairwise Loss (p=1)
Optimizer	AdamW
Activation Function	Leaky ReLU
Learning Rate	0.001
Learning Rate Change	0.95
Learning Rate Step	100
L2	0.01

TABLE VII: LSTM Constant Parameters.

Test	Epoch	Max Train Acc.	Max Val. Acc.	Val. Loss Minus Train Loss
000	671	64.2%	55.3%	0.89
001	683	61.8%	55.1%	1.02
010	712	62.5%	55.0%	0.67
011	699	61.0%	54.7%	0.83
100	967	60.5%	55.6%	0.429
101	705	62.5%	55.3%	0.77
110	902	60.1%	55.2%	0.35
111	765	60.5%	54.7%	0.74

TABLE VIII: LSTM Stage 2 Tests; best result is highlighted in green.

As displayed in Table VIII, most results attain very close final training and validation accuracies of 60-61% and 54-55%, respectively. Among all runs, training ended when validation accuracy remained stagnant for 500 epochs. We can conclude that dropout, as expected, decreased overfitting in terms of accuracy. The [128,64] hidden layer neuron architecture performed slightly better than the more complex [128,128,128,128], suggesting that simpler models may have better prediction on patient data. A greater batch size resulted in a negligible increase in validation accuracy and seemed to reduce overfitting. In conclusion, hidden layer neuron configuration and batch size seemed to have the largest effect on accuracy. However, unlike the FCN hyperparameter study, no Stage 2 tests led to a reasonable increase in model performance. Hence, Stage 3 of this LSTM study was not implemented. Our best LSTM model has a validation accuracy of 55.6%, as shown in Fig. 5. The adjusted hyperparameters are: Batch Size: 4096; Hidden Layer Configuration: [128, 64]; Dropout: 0.15.

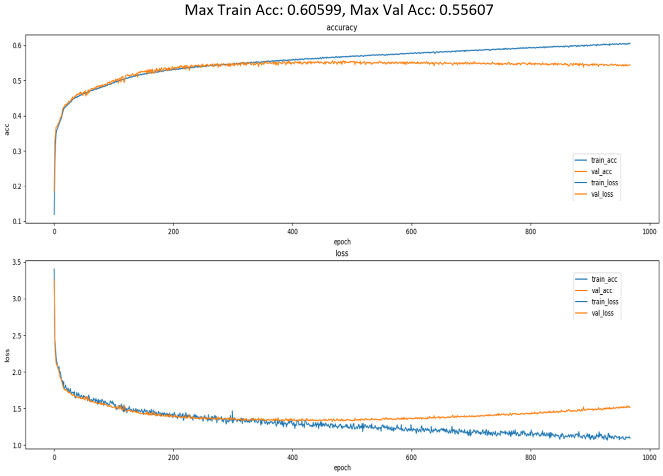


Fig. 5: Accuracy and loss learning curve for Test 100.

B. Comparisons between Water Phantom and Real Patient Data

We note that the results of [5] were reproduced with similar accuracy results on BRIDE using networks developed in PyTorch, verifying the validity of BRIDE experiments as well as presenting a significant improvement for the Pytorch models in [9] which were previously unable to achieve similar results those of Tensorflow in [5]: we achieved 72.5% testing accuracy for a LSTM model, as shown by the training curve of Fig. 6, and 65% testing accuracy for a FCN model, both on 1.8 million row water phantom data.

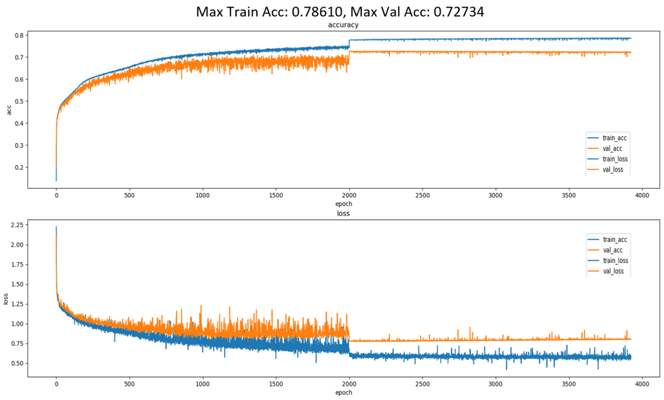


Fig. 6: LSTM Model BRIDE reproduction results. **Top:** Training (blue) and validation (orange) accuracies. **Bottom:** Training (blue) and validation (orange) loss curves.

The key results from our hyperparameter studies on simulated patient data can be summarized: FCN models had a greatest accuracy of 55.0%and [512, 256, 256, 256, 256, 256, 256, 128] architecture, and LSTM models had a greatest accuracy of 55.6% and 4 LSTM + [128,64] Fully Connected Layer (FCL) architecture. We can conclude that model predictive performance is significantly lower when trained on simulated patient data compared to both current reproduced water phantom results and past work [5], [9], [15]. Second, LSTM models do better relative to the FCN on water phantom data, when compared to patient data, suggesting that FCN architecture may be specifically advantageous for patient data or LSTM models could be better on water phantom data. However, these conclusions have the caveat that the simulated patient dataset was smaller than the water phantom, which may have led to the poorer generalization performance of the models on patient data.

V. Additional Studies

We conducted runs on a novel hybrid water phantom and simulated patient dataset of 3.8 million observations with the intention of producing models that could perform well on and gain insights from both types of data. As shown by Fig. 7, we implemented a model containing 4 LSTM and 4 FCN layers resulting in a test accuracy of 76.1%. Our hyperparameters, listed in Table IX, were an adaptation of a somewhat similar model in [5].

Hyperparameter	Value
Hardware	4 RTX6000 GPUs
Validation split	0.1
Loss Function	Cross Entropy
Optimizer	Adam
Activation Function	ReLU
Batch Size	4096
Neurons	128
Learning Rate	0.001
Learning Rate Change	0.1
Learning Rate Step	2000
Dropout	0.0

TABLE IX: LSTM hyperparameters on the hybrid dataset.

Because of previous results plateauing after a few hundred epochs, this model implemented a distinct learning rate scheduler. The learning rate was multiplied by 0.1 every 2000 epochs; the models were also run for several thousand, as opposed to several hundred, epochs. As displayed by Fig. 7, there was a large increase in accuracy at the 2000th epoch learning rate change. This 76.1% accuracy achieved is greater than any comparable results in previous work [5], [9], [15]. This may be due to a a larger training dataset (which also seemed to limit overfitting). Hence, this model may be a more robust application in a medical situation.

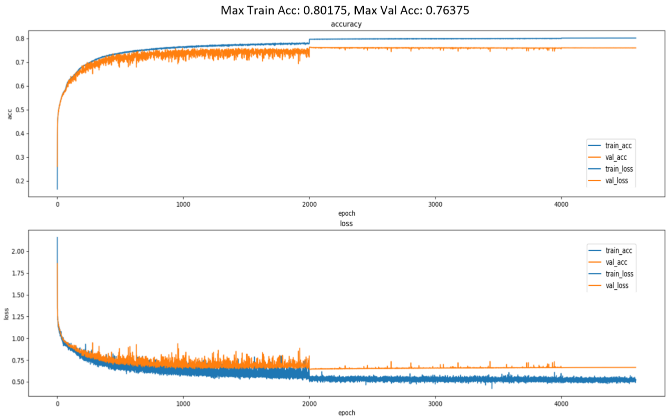


Fig. 7: Accuracy and loss learning curve for LSTM model on hybrid data.

VI. Conclusions and Future Work

We found that the best FCN model on patient data had architecture [512, 256, 256, 256, 256, 256, 256, 128] and achieved a 55% testing accuracy while the best LSTM had architecture 4 LSTM + [128,64] FCL and achieved an accuracy of 55.6%. Given that these accuracies are substantially lower than our best accuracies on the water phantom data, these results may indicate that patient data is more difficult to learn and predict on compared to water phantom. LSTM models performed better on water phantom data, while FCN models had a slightly higher accuracy for patient data, suggesting that FCL layer models could be more suited for patient data in this setting. To consider the implications of these results on applied machine learning research in general, they sugest that deep neural networks are very sensitive. Even small changes in data generation can have large impacts on model performance.

This work introduced BRIDE, a machine learning code implementation platform to ensure rigorous testing, experimentation, expandability, and reproducibility in a parallelized context. The best models of [1] and [6] were reproduced on

BRIDE with 72.5% accuracy for a LSTM model and 65% accuracy for a FCN model. We developed a custom pairwise loss function for both FCNs and LSTMs to improve model performance. The utility for the efficiency of our research of having an effective integrated development and experimentation platform cannot be overstated.

Finally, this research explored several avenues in increasing the predictive performance and effectiveness of various machine learning models. Hybrid LSTM + FCL neural networks trained on a larger hybrid water phantom and real patient dataset achieved a testing accuracy of 76%. This may indicate that the relatively small size of the simulated patient data may have contributed to the poor accuracies. The poorer testing accuracy on patient data could have also been due to the relative complexity of patient data compared to water phantom data and a smaller dataset.

These results may be scaled to practical use through further work to address overfitting in simulated patient data. Our models can be considered more scalable and understandable due to less complex architectures. This, in turn, could reduce prediction speed for real-time proton beam range verification in a medical setting, compared to more complex machine learning models.

In the future, to improve these results for less noisy reconstructed images, more training data could be generated, which may lead to better model predictive performance. Another option would be to explore changing the constant hyperparameters in both the FCN and LSTM hyperparameter studies, such as L2 regularization, learning rate, and optimizer algorithm. Additional neural network model types could also be explored for this classification purpose. De-noised reconstructed images through further models could be compared between water phantom and simulated patient data.

Acknowledgment

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–2348755). Undergraduate assistant co-author Obe acknowledges support from an REU Supplement. Co-authors Sharma and Ren acknowledge 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 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, Matthias K. Gobbert, and Jerimy C. Polf. Deep residual fully connected neural network classification of Compton camera based prompt gamma imaging for proton radiotherapy. *Front. Phys.*, 11:903929, 2023.

[3] Michael Chen, Julian Hodge, Peter Jin, Ella Protz, and Elizabeth Wong. Github repository. https://github.com/big-data-lab-umbc/big-data-reu, 2024.

[4] Michael O. Chen, Julian Hodge, Peter L. Jin, Ella Protz, Elizabeth Wong, Ruth Obe, Ehsan Shakeri, Mostafa Cham, Matthias K. Gobbert, Carlos A. Barajas, Zhuoran Jiang, Vijay R. Sharma, Lei Ren, Sina Mossahebi, Stephen W. Peterson, and Jerimy C. Polf. Using neural networks to sanitize Compton camera simulated data through the BRIDE pipeline for improving gamma imaging in proton therapy on the ada cluster. Technical Report HPCF–2024–5, UMBC High Performance Computing Facility, University of Maryland, Baltimore County, 2024.

[5] Joseph Clark, Anaise Gaillard, Justin Koe, Nithya Navarathna, Daniel J. Kelly, Matthias K. Gobbert, Carlos A. Barajas, and Jerimy C. Polf. Multi-layer recurrent neural networks for the classification of Compton camera based imaging data for proton beam cancer treatment. In *9th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT 2022)*, pages 213–222, 2022.

[6] Joseph Clark, Anaise Gaillard, Justin Koe, Nithya Navarathna, Daniel J. Kelly, Matthias K. Gobbert, Carlos A. Barajas, and Jerimy C. Polf. Sequence-based models for the classification of Compton camera prompt gamma imaging data for proton radiotherapy on the GPU clusters taki and ada. Technical Report HPCF–2022–12, UMBC High Performance Computing Facility, University of Maryland, Baltimore County, 2022.

[7] Jonathan R. Hughes and Jason L. Parsons. FLASH radiotherapy: Current knowledge and future insights using proton-beam therapy. *Int. J. Mol. Sci.*, 21(18):6492, 2020.

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

[9] Ruth Obe, Brandt Kaufmann, Kaelen Baird, Sam Kadel, Yasmin Soltani, Mostafa Cham, Matthias K. Gobbert, Carlos A. Barajas, Zhuoran Jiang, Vijay R. Sharma, Lei Ren, Stephen W. Peterson, and Jerimy C. Polf. Accelerating real-time imaging for radiotherapy: Leveraging multi-GPU training with PyTorch. In *2023 International Conference on Machine Learning and Applications (ICMLA 2023)*, pages 1735–1742, 2023.

[10] Costanza M. V. Panaino, Ranald I. Mackay, Karen J. Kirkby, and Michael J. Taylor. A new method to reconstruct in 3D the emission position of the prompt gamma rays following proton beam irradiation. *Sci. Rep.*, 9(1):18820, 2019.

[11] 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.

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

[13] Michael J. Smith and James E. Geach. Astronomia ex machina: a history, primer and outlook on neural networks in astronomy. *R. Soc. Open Sci.*, 10:221454, 2023.

[14] Martin Sundermeyer, Hermann Ney, and Ralf Schlüter. From feed-forward to recurrent LSTM neural networks for language modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):517–529, 2015.

[15] Sokhna A. York, Alina M. Ali, David C. Lashbrooke Jr, 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 IEEE International Conference on Big Data (Big Data 2021)*, pages 5648–5657, 2021.