Enhancing Real-Time Imaging for Radiotherapy: Leveraging Hyperparameter Tuning with PyTorch

Kaelen Baird[1], Sam Kadel[2], Brandt Kaufmann[3], Ruth Obe[4], Yasmin Soltani[5], Mostafa Cham[6], Matthias K. Gobbert[7], Carlos A. Barajas[7], Zhuoran Jiang[8], Vijay R. Sharma[9], Lei Ren[9], Stephen W. Peterson[10], Jerimy C. Polf[11]

[1]Dept. of Computer Science and of Mathematics, Skidmore College
[2]Dept. of Computer Science and of Psychology, Mount Holyoke College
[3]Dept. of Mathematics and Statistics, University of San Francisco
[4]Dept. of Computer Science, University of Houston—Clear Lake
[5]Dept. of Biomedical Engineering, University of Houston
[6]Dept. of Information Systems, Univ. of Maryland, Baltimore County
[7]Dept. of Mathematics and Statistics, Univ. of Maryland, Baltimore County
[8]Medical Physics Graduate Program, Duke University
[9]Dept. of Radiation Oncology, University of Maryland School of Medicine
[10]Dept. of Physics, University of Cape Town, South Africa
[11]H3D, Inc.

Acknowledgments: NSF (Big Data REU, MRI), NIH, UMBC, CIRC

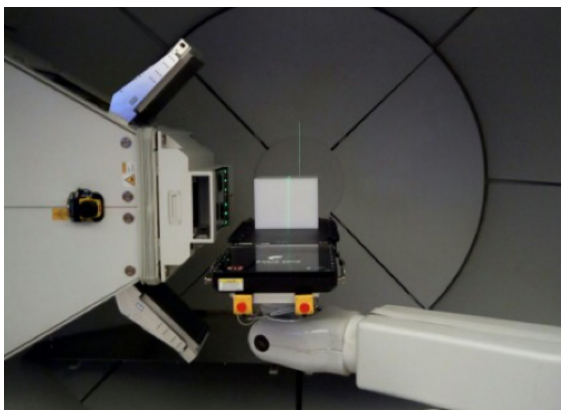
Maryland Proton Treatment Center

Maryland Proton Treatment Center

This work was done in collaborations with the Maryland Proton Treatment Center located in Baltimore, Maryland. Opened in 2016, the center was the first in the Maryland/DC region to offer proton therapy for cancer treatment. It has trained more than 200 health care professionals in proton therapy and, with its state of the art facilities and four treatment rooms, has been able to treat over 3,000 patients (`www.mdproton.com`).

Proton Beam Therapy

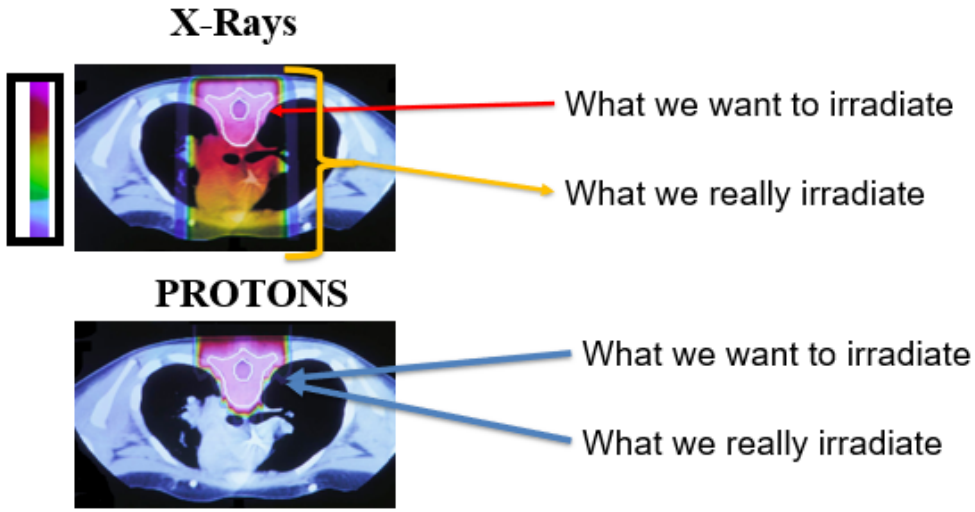
- Cancer is the second highest cause of death in the US (`www.cdc.gov` 2020).
- Proton Therapy is a form of radiotherapy that delivers a high dose of radiation to a local tumor site using proton beams.
- The patient lies on the black tray, where the white box (phantom) is placed. Proton beams come out of the machine on the left.



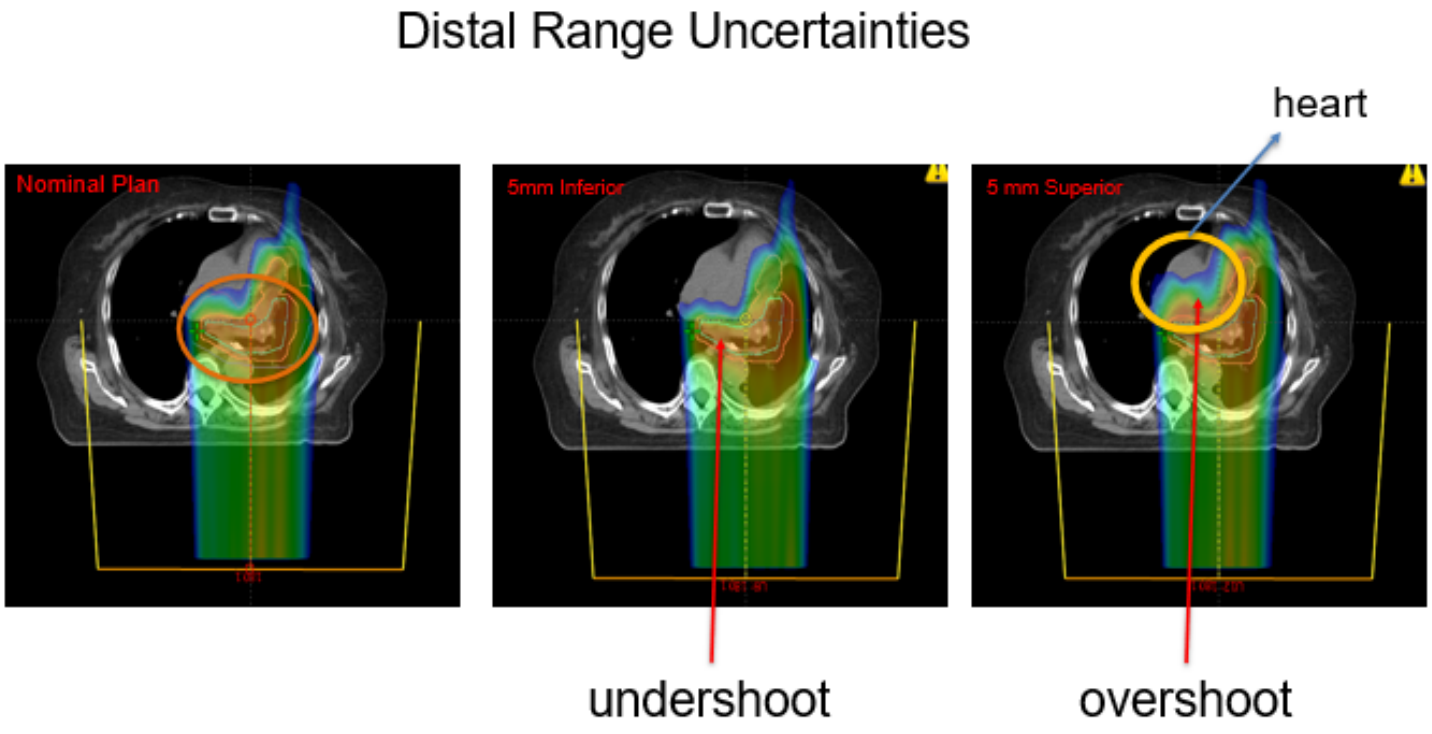
Treatment table in MD Proton Treatment Center. (Maggi 2019)

Proton Beam Therapy

- Proton beams deposit the majority of their energy/dose just before they stop.
  - This sharp energy increase of the proton right before stopping is known as a **Bragg peak**.
- Since almost no radiation is delivered beyond the Bragg peak, healthy tissue is spared from unnecessary radiation.



The Need for Real-Time Imaging



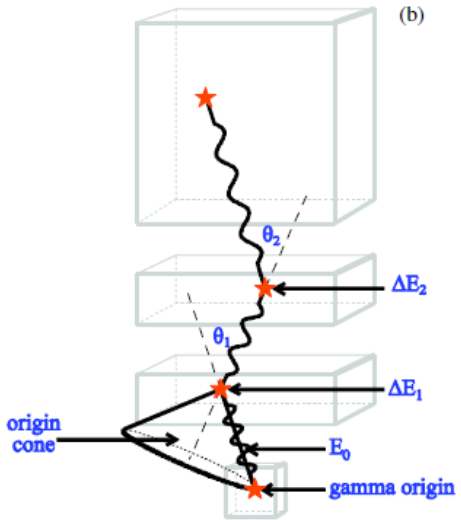
- Uncertainties in the beam's position limit proton beam therapy's advantages.
- Imaging the beam in near real time would reduce uncertainties and allow the advantages of the Bragg peak to be fully exploited.

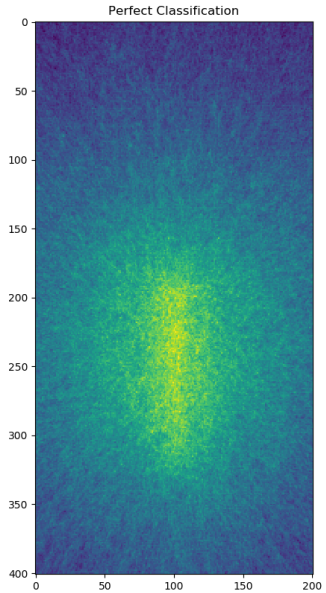
Compton Camera to Visualize Beam Trajectory

- Proton beam radiation can be detected and recorded via several different emissions, including thermoacoustic waves, ionoacoustic, and gamma radiation.
- Prompt Gamma imaging allows one to detect specific shifts in the location of the Bragg peak.
- Prompt gamma rays also allow for spectroscopic analysis of tissue that has been exposed to radiation.

Polf and Parodi, *Physics Today*, 2015

Image Reconstruction Using Compton Camera





(Phys. Med. Biol., 2010.)

(a) (b)

- (a) Nuclear reactions between beam and tissue produce prompt gamma rays. A Compton camera records the position and energies of each interaction.
- (b) By analysing how prompt gammas scatter through the Compton camera, we can reconstruct their origin and image the beam.

Events in the Compton Camera

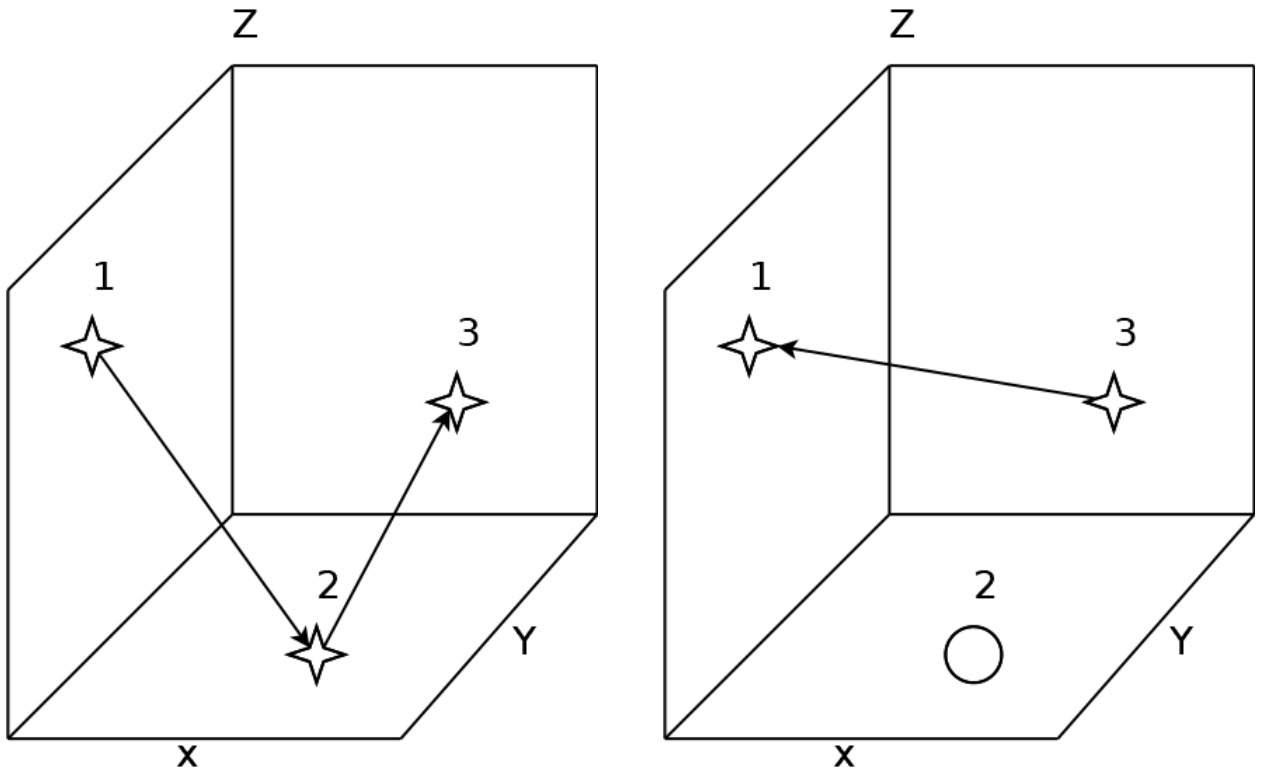
- Prompt gamma rays emitted from nuclear interactions of protons and tissue travel at approximately the speed of light.
- This causes the Compton camera to detect interactions as occurring simultaneously.
- We define a **interaction** as a prompt gamma colliding with a stage of the Compton camera. For each interaction the output data from the camera is (e_i, x_i, y_i, z_i) where $i = 1, 2, 3$; x_i, y_i, z_i are coordinate locations in 3D space, and e_i is the energy level.
- An **event** is the readout of interactions in a single period.

Barajas, Gobbert, Polf, *Frontiers in Physics*, 11:903929, 2023.

Limitations of the Compton Camera

- The Compton camera simply records events as a **single**, **double**, or **triple** scatters.
- The Compton camera cannot determine the correct orderings of camera events.
- The Compton camera cannot determine if a **double** or **triple** scatter event was triggered by prompt gammas originating from different physics events that just happened to enter the camera at the same time.
  1. This lack of distinction means that **single** events can end up paired together as a **double** or **triple** event.
  2. Coupled **single**s are referred to as **false** events.
  3. A double to triple or, **dtot**, is when a true **double** is incorrectly paired with a **single** which appears as a **triple**.

Double-to-Triples (DtoTs)



(a) Detected Path (b) True Path

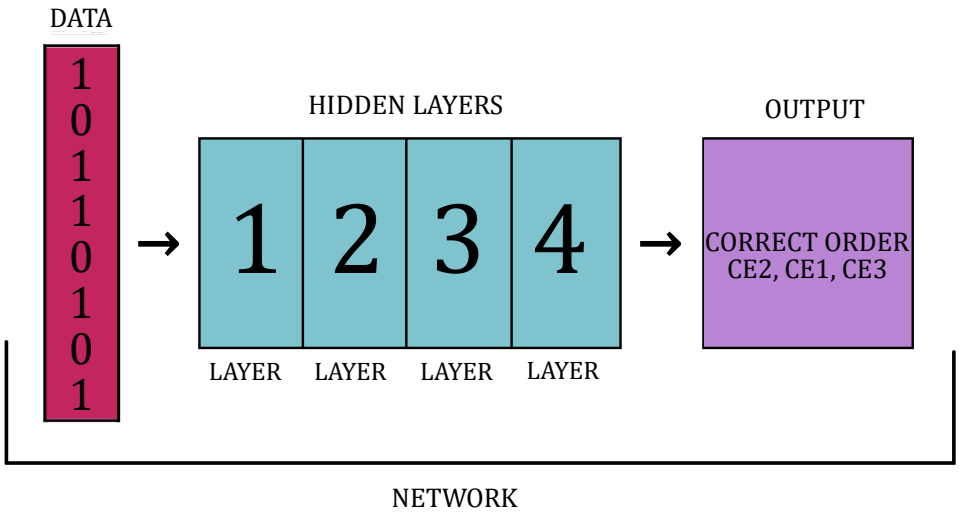
- The Compton Camera detected a single prompt gamma whose path is (a) but really there are two prompt gammas with varying paths as seen in(b).
- This causes a triple to be computed when it is actually a double.
- There are six different misdetection orderings each of which is equally likely: 124, 134, 214, 234, 324, 314.

All The Data and Classes

Class	Interaction 1				Interaction 2				Interaction 3				Event Type
123	e_1	x_1	y_1	z_1	e_2	x_2	y_2	z_2	e_3	x_3	y_3	z_3	Ordered Triple
132	e_1	x_1	y_1	z_1	e_3	x_3	y_3	z_3	e_2	x_2	y_2	z_2	Misordered Triple
213	e_2	x_2	y_2	z_2	e_1	x_1	y_1	z_1	e_3	x_3	y_3	z_3	Misordered Triple
231	e_2	x_2	y_2	z_2	e_3	x_3	y_3	z_3	e_1	x_1	y_1	z_1	Misordered Triple
312	e_3	x_3	y_3	z_3	e_1	x_1	y_1	z_1	e_2	x_2	y_2	z_2	Misordered Triple
321	e_3	x_3	y_3	z_3	e_2	x_2	y_2	z_2	e_1	x_1	y_1	z_1	Misordered Triple
124	e_1	x_1	y_1	z_1	e_2	x_2	y_2	z_2	single				Ordered DtoT
214	e_2	x_2	y_2	z_2	e_1	x_1	y_1	z_1	single				Misordered DtoT
134	e_1	x_1	y_1	z_1	single				e_2	x_2	y_2	z_2	Ordered DtoT
314	e_2	x_2	y_2	z_2	single				e_1	x_1	y_1	z_1	Misordered DtoT
234	single				e_1	x_1	y_1	z_1	e_2	x_2	y_2	z_2	Ordered DtoT
324	single				e_2	x_2	y_2	z_2	e_1	x_1	y_1	z_1	Misordered DtoT
444	single				single				single				False Event

- All 13 classes are shown in the left column, the top row indicates the output of the Compton Camera, and the rest show the unusable combinations of interactions.
- The true triple classes are 123, 132, 213, 231, 312, 321.
- The double-to-triple classes are 124, 214, 134, 234, 324, 313.
- The false triples are the 444 class.
- In the data set we have 1,443,992 data points are used for training and validation.

Design of Neural Networks



- A neural network is made up of three sections:
  1. The input layer. The data is converted into a shape and type that is conducive to learning.
     For example, an image might be converted to a 3 layer tensor.
  2. The hidden layers. These layers perform repeated mathematical operations on the data as it passes through each layer storing bits of information associated with the data aka "learning from the data".
  3. The output layer. The final choice, decision, or feedback that the network produces based on the hidden layers digestion of the input data.

Chollet, *Deep Learning with Python*, 2018

Translation to PyTorch

- Translated the code base from TensorFlow to PyTorch for easy setup of distributed and paralleled training and GPU optimization
- Used torch.distributed packages for proper functionalities of distributed training
- Defined model architecture using PyTorch's nn.Module class for forward pass in the forward method of the model
- No direct equivalent to the CSVLogger like in Keras
- Created a CSVLogger class to log desired information such epoch time, loss, accuracy, validation loss, and validation accuracy
- PyTorch creates an instance of the Linear while Keras creates a KerasTensor object
- Tested DistributedDataParallelism on both single and multiple GPUs using UMBC HPCF CUDA device on ADA nodes with NVIDIA RTX 2080, 6000, and 8000

Distributed Data Parallel (DDP)

- Implemented distributed data parallelism in PyTorch for training across multiple GPUs or machines.
- Training data divided into smaller subsets
- Accelerated research iterations with reduced training time, allowing for quicker validation and refinement of models
- Reduces the memory footprint by sharing parameters, and ensures consistency across the model during training.
- Provides fault tolerance mechanisms to handle failures during distributed training
  `https://pytorch.org/tutorials/intermediate/ddp_tutorial.html`

Hyperparameter Study

- Tuning of the model parameters: number of layers, learning rate scheduler, train-validation split, clip gradient, and dropout rate.

Hyperparameter	*Value*
Indim	15
Outdim	13
Optimizer	Adam
Loss Function	Categorical Crossentropy

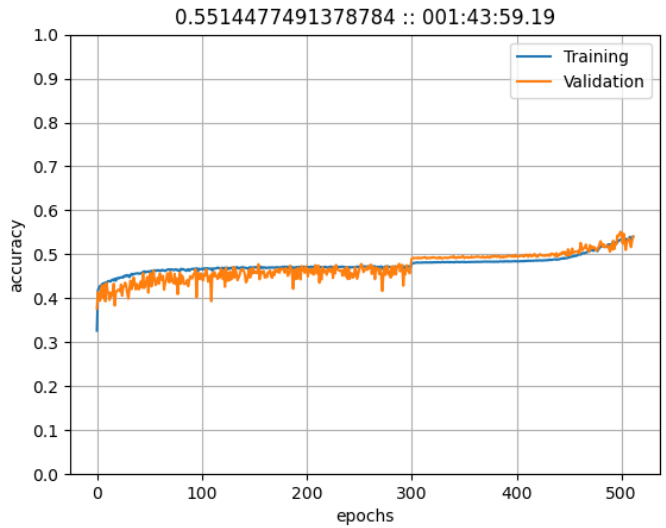
Table: Constant Model Parameters

Hyperparameter Variables

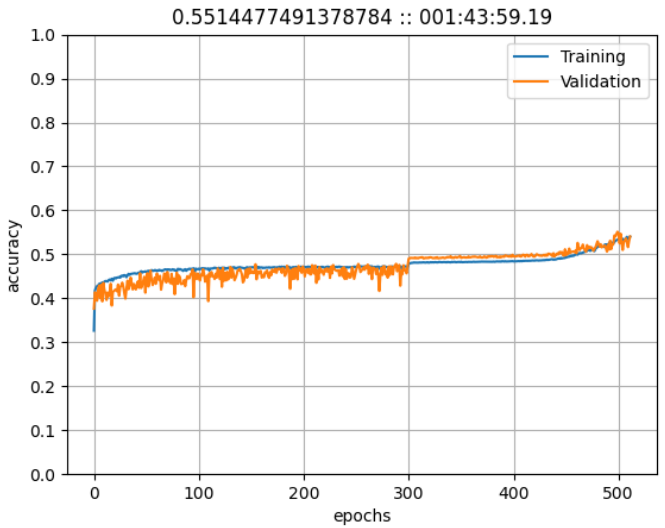
Hyperparameter	*Value*
Layers	256
Neurons	256
Batch Size	8192
Learning Rate	1.0e-3
Train/Validation	0.8/0.2
Dropout	0.45
Inter-activation	leakyrelu
Clip Gradient	0
Layer Type	Dense

Table: Variable Model Parameters

Number of Layers



(a) Single Layer



(b) 11 Layers

- We ran models with layer sizes 1-16, 32, 64, 128, and 256.
- Concluded that a lower number of levels increased our accuracy of the model but if the number of layers were too low then it would cause the model to under fit.

Hyperparameter	*Value*
Learning Rate	1.0e-3
Learning Rate Change	1e-1
Learning Rate Step	300
Train/Validation	0.8/0.2
Batch Size	8192
Neurons	256
Dropout	0.45
Inter-activation	leakyrelu
Layer Type	Dense
Epochs	512

Table: Constant hyperparameters in number of layers models

1-16 Dense Layer Investigation

No. Dense Layers	Accuracy	Val. Accuracy
1	58.37	60.71
2	48.14	49.18
3	60.02	57.69
4	48.74	49.18
5	58.35	56.68
6	52.24	56.02
7	56.97	56.85
8	50.02	26.99
9	48.37	49.61
10	48.57	49.70
12	48.72	50.00
13	48.45	49.67
14	48.32	49.33
15	48.28	49.20
16	48.26	49.46

Table: Dense layer tests on 2048 epochs

Learning Rate Scheduler

- We trained 2, 4, 6, 11 layer dense models with learning changes of 1e-1, 3e-1, and 5e-1 with variable learning step sizes of 682 epochs and 341 epochs.
- Learning change had an insignificant effect on the model accuracy with all of the runs being withing 1-2% of each other
- Halving the learning step improved the accuracy of all of the models.

Learning rate Investigation

No. Dense Layers	Learning Rate change	Epochs per step	Accuracy	Validation Accuracy
2	1.0e-1	682	62.29	64.44
	3e-1	682	62.44	64.54
		348	62.07	64.17
	5e-1	682	62.22	62.98
		348	62.42	64.11
4	1.0e-1	682	62.34	64.37
	3e-1	682	62.31	64.34
		348	62.23	64.28
	5e-1	682	61.47	62.72
		348	62.37	64.30
6	1.0e-1	682	62.39	64.47
	3e-1	682	62.06	64.23
		348	62.15	64.17
	5e-1	682	56.81	57.21
		348	62.36	63.74
11	3e-1	682	61.60	63.40
		348	61.78	64.29
	5e-1	682	57.71	56.49
		348	62.00	64.06

Table: Learning rate tests over 2048 epochs

Train-Validation Split

Validation split	Accuracy	Val. Accuracy
10%	62.25	64.29
15%	62.32	64.54
20%	61.45	64.08
25%	61.80	63.99
30%	61.67	64.02

- We trained six 11-layer feed-forward models using train-validation splits of 0.9/0.1, 0.85/0.15, 0.8/0.2, 0.75/0.25, 0.7/0.3.
- Train-validation split had an insignificant effect on our model as all of the tested models were within 1% accuracy of one another at 64%.

Clip Gradient

Clip Gradient	Accuracy	Val. Accuracy
5.0e-3	61.44	63.98
5.0e-4	61.41	64.10
1.0e-4	61.77	63.98

Table: Clip gradient on accuracy on 11-layer model

Clip Gradient

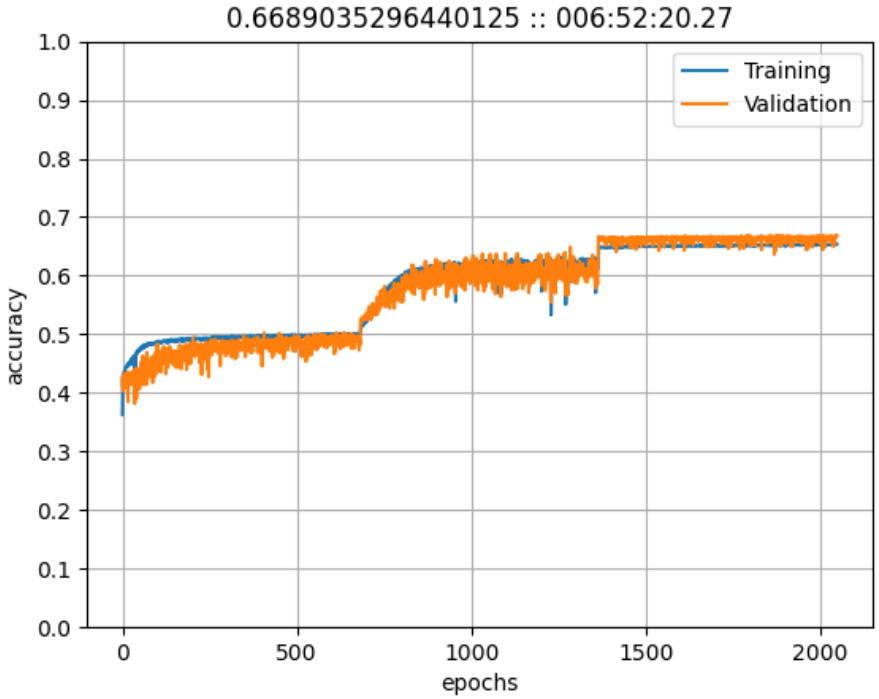


Figure: 1.0e-4 Clip Gradient

- Clip Gradient has little to no change on peak accuracy, producing results that are less than .0005 different then our preset clip gradient.
- 1.0e-4 seems to be the best value in this range, but this is not a very powerful variable in changing peak accuracy.

Batch Size

Batch	Accuracy	Val. Accuracy
8192	61.63	63.96
16384	61.73	63.89
32768	49.87	52.74
65536	48.75	50.32

Table: Batch size on 11-layer dense model accuracy

- Sometimes increasing batch sizes improves models, but it can cost memory performance
- Keeping the batch size at 8192 maximized accuracy in our model

Dropout

- We trained six different models with variable dropout rates of 0.15, 0.20, 0.30, 0.40, 0.45, and 0.60 dropout values.
- For dense layers, lowering the dropout percentage to 0.15 increased the accuracy to 66%.
- In the GRU layers, a dropout rate of 10% led to an accuracy of 43%. A dropout rate of 60%, on the other hand, generated an accuracy of 7%

Dense Dropout

Dropout	Accuracy	Val. Accuracy
60%	59.27	62.76
55%	60.02	63.17
40%	62.04	64.31
30%	63.29	65.17
20%	64.81	66.37
15%	65.28	66.85

Table: Dropout rate on accuracy in 11-layer dense model

GRU Tests

No. GRU layers	No. Dense layers	Accuracy	Val. Accuracy
2	10	74.20	64.45
	16	73.63	66.04
4	16	87.59	55.13
	64	8.629	8.418
16	16	7.675	7.691
	64	7.678	7.600

Table: GRU tests (Dropout = 35%, lr = 1e-3, step = 450, lr multiplier = 0.95)

LSTM Tests

No. of LSTM layers	No. of Dense layers	L2 Regularization	Accuracy	Validation Accuracy
2	10	0	74.17	63.33
		0	73.78	63.70
	16	1e-2	89.62	54.05
		1e-4	88.16	56.56
4	16	0	89.36	56.36
		1e-2	74.11	63.67
		1e-3	73.83	66.52
	64	0	76.90	76.03
16	16	0	7.66	7.64
	64	0	7.69	7.73
128	0	0	7.677	7.604

Table: LSTM tests on LSTM layers, dense layers, and L2 regularization

Best Model

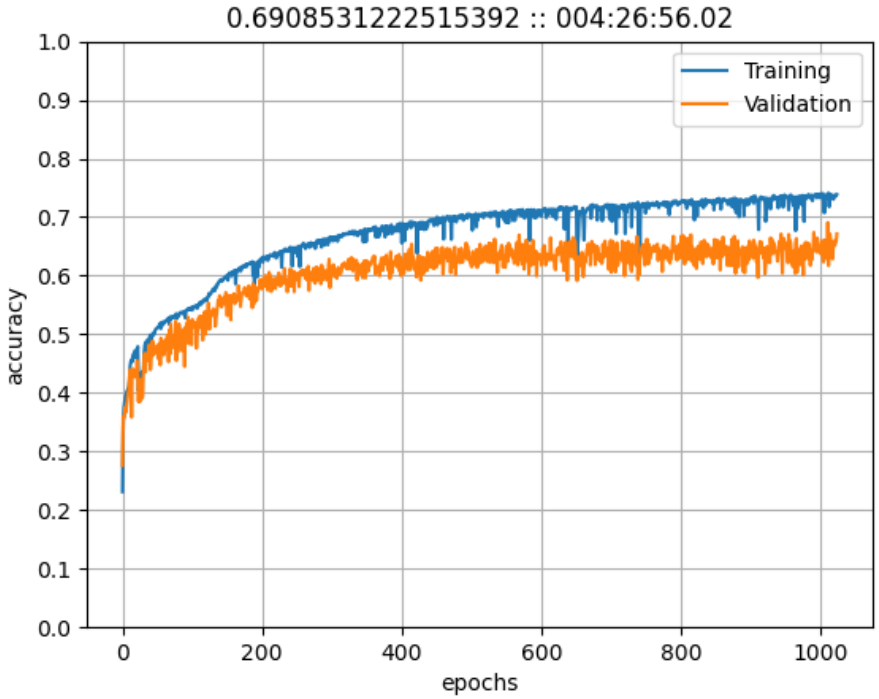


Figure: Accuracy graph of our 2 GRU + 10 FCL model

- Combination of recurrent and fully connected layers reached a peak validation accuracy of 69%.
- Number of layers were the most important, with increases in accuracy from dropout and learning rate scheduler parameters.

Future Work

- Create a Transformer model in the current pytorch model
- Test different criterion (cross entropy loss vs. negative log likelihood)
- Test different optimizers and inter-activation layers (leakyrelu, prelu, relu, and sigmoid)
- Test smoother learning rate curves
- Improve speed of multi-GPU runs

Conclusions

- Successfully translated base code from Keras to Pytorch for potential faster training, increased user support, better readability, rich package environment, and more standard code across the industry.
- Our most accurate model uses a combination of 2 GRU layers and 10 fully connected layers, and was able to reach an accuracy of 69% in 1024 epochs.
- Number of layers, dropout rate, and learning rate scheduler were the most impactful hyperparameters.

For complete information: Baird, Kadel, Kaufmann, Obe, Soltani, et al., Tech. Report HPCF–2023–12, 2023. `hpcf.umbc.edu`