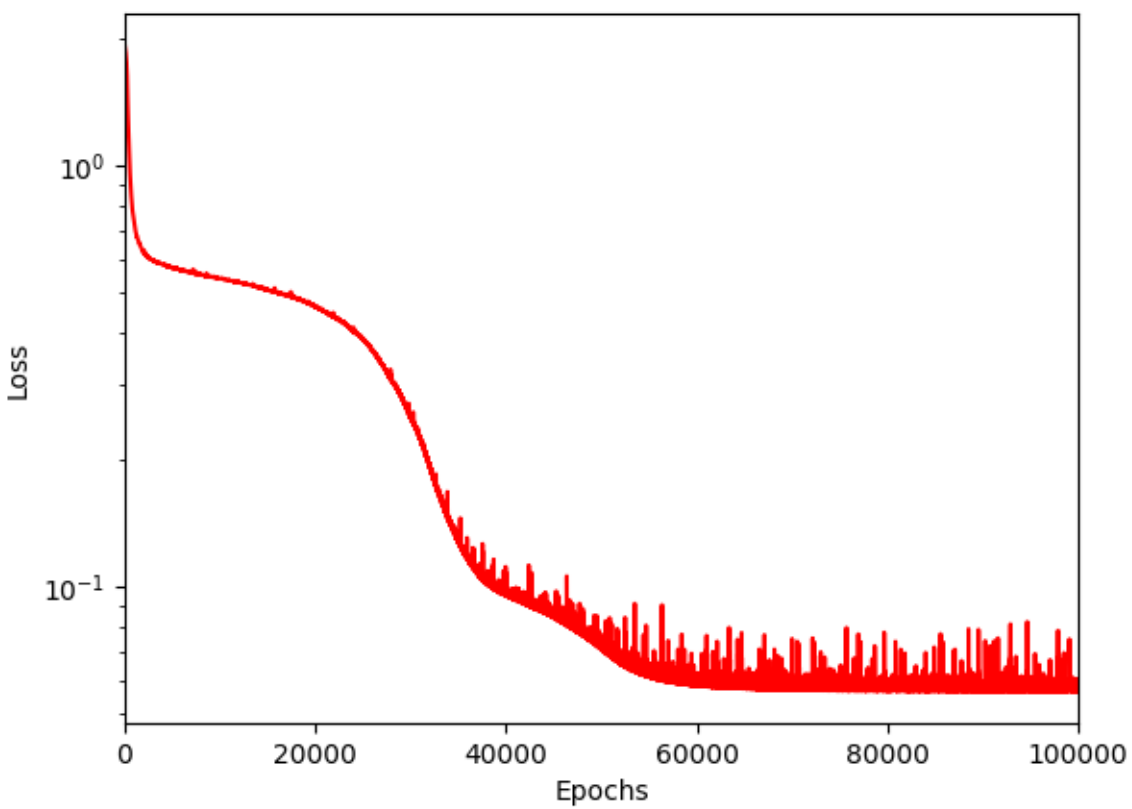
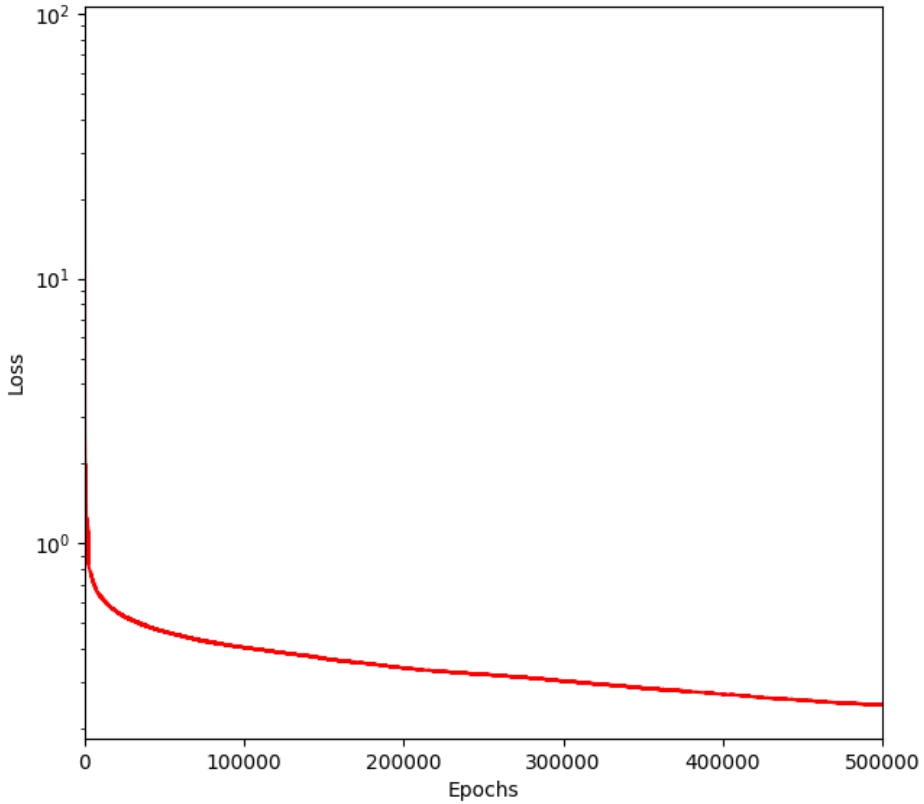
Current Status

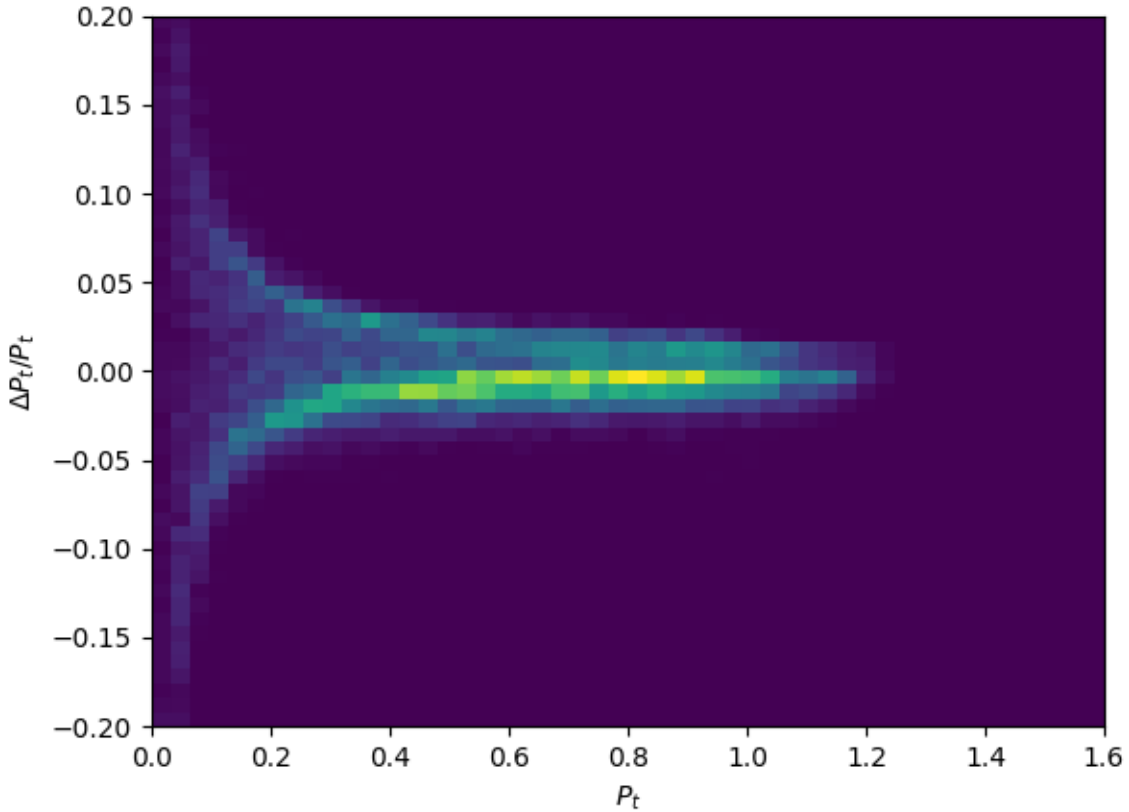
- The previous plot I sent was somewhat optimistic, as I accidentally indexed poorly and contaminated the test sample with one of the files the network was trained on.
- But, by employing a learning rate scheduler set to decrease the learning rate by 0.5 on a significant enough plateau, have been able to create a steady increase in precision with more training time



Custom Loss Function / Poor Combination of Parameters

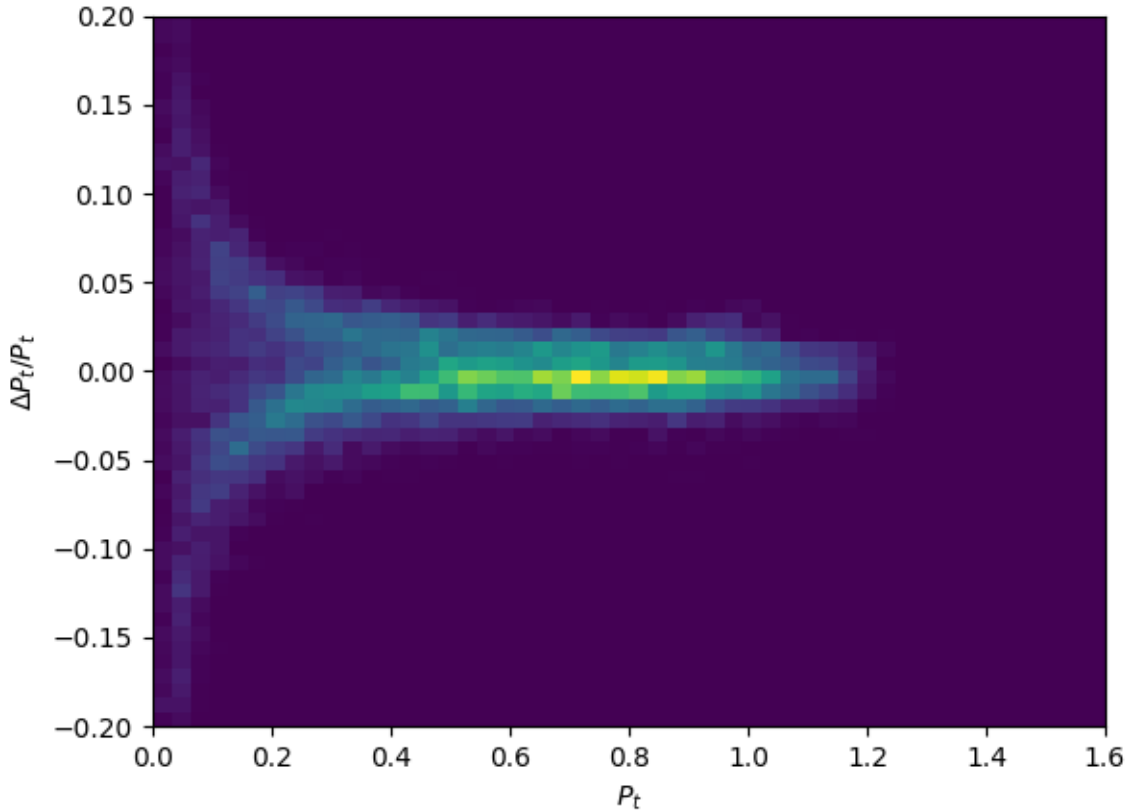


Huber Loss / Ideal-Ish Combination of Parameters



Custom Loss Function

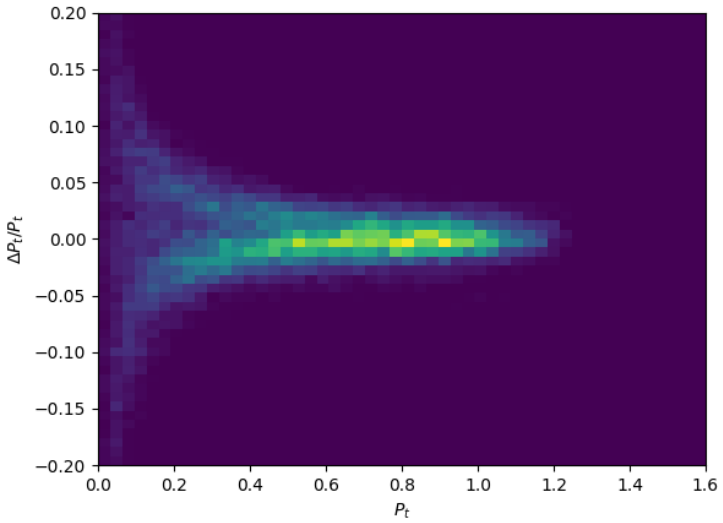
No further obvious room for improvement



Huber Loss Function

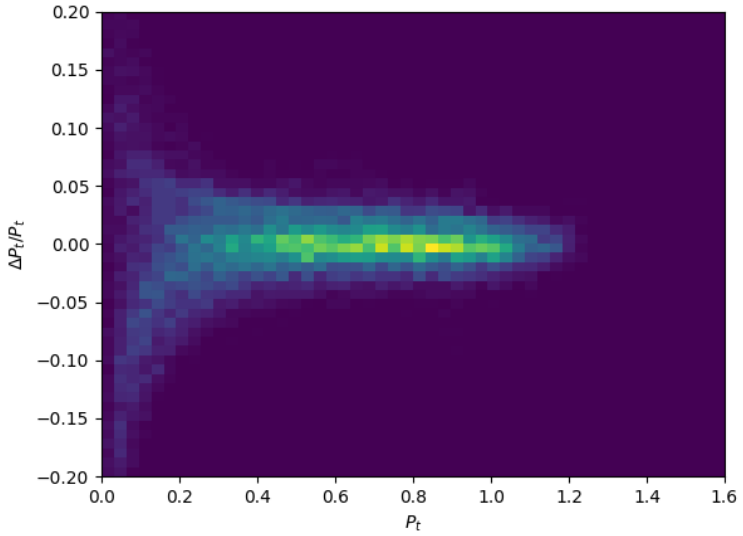
Seems it can do better with more epochs

Stdev: 2.51%



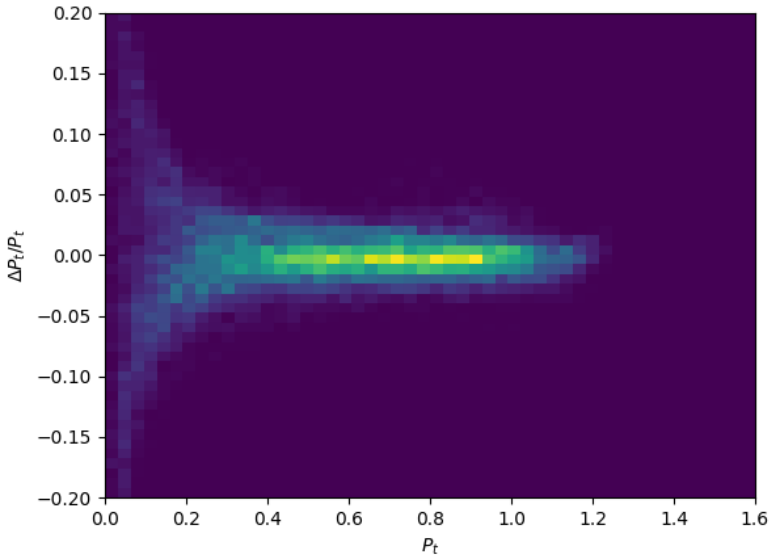
5 hrs

Stdev: 2.31%



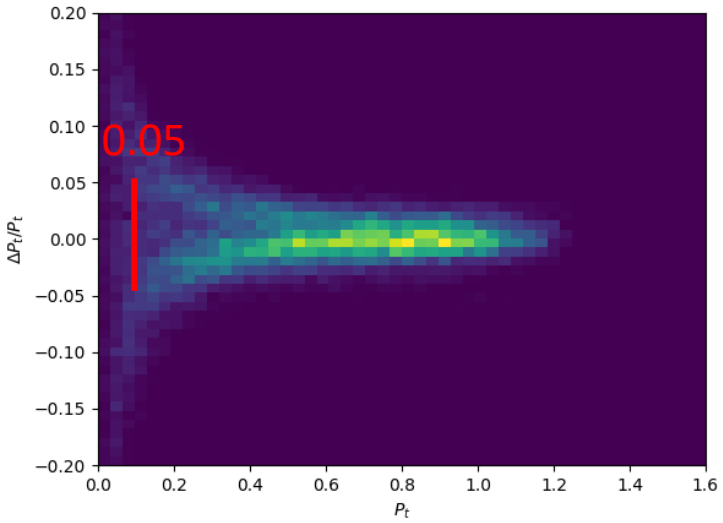
16 hrs

Stdev: 1.95%



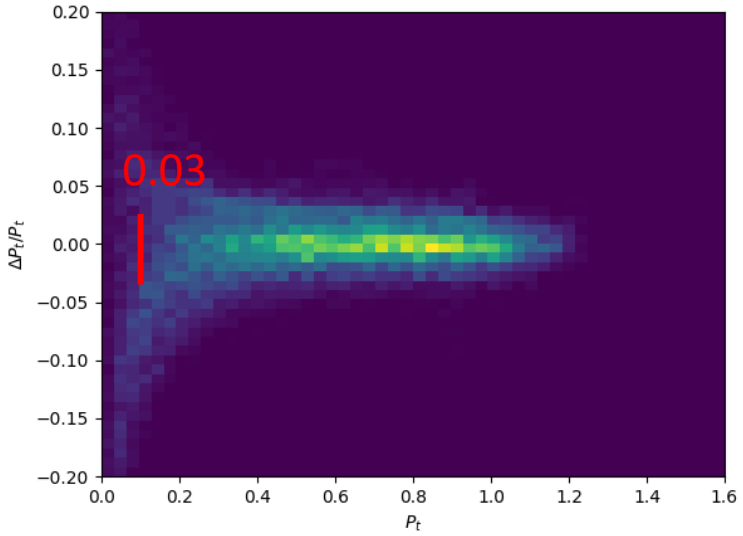
25 hrs

Stdev: 2.51%



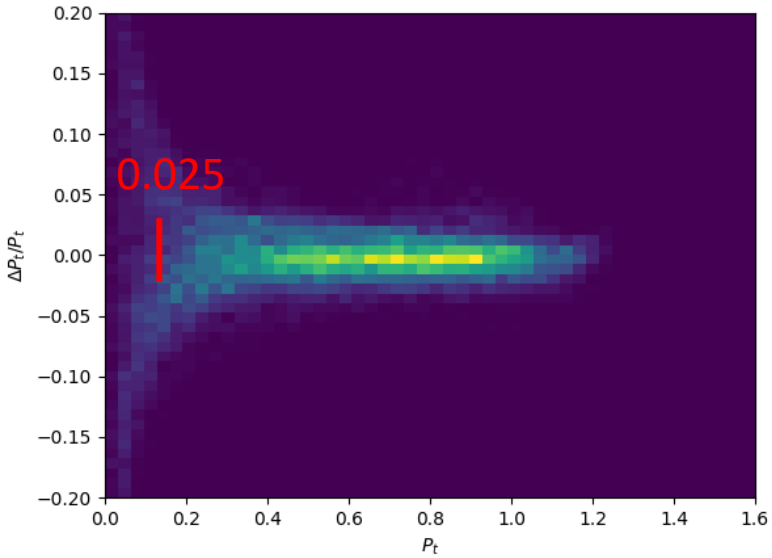
5 hrs

Stdev: 2.31%



16 hrs

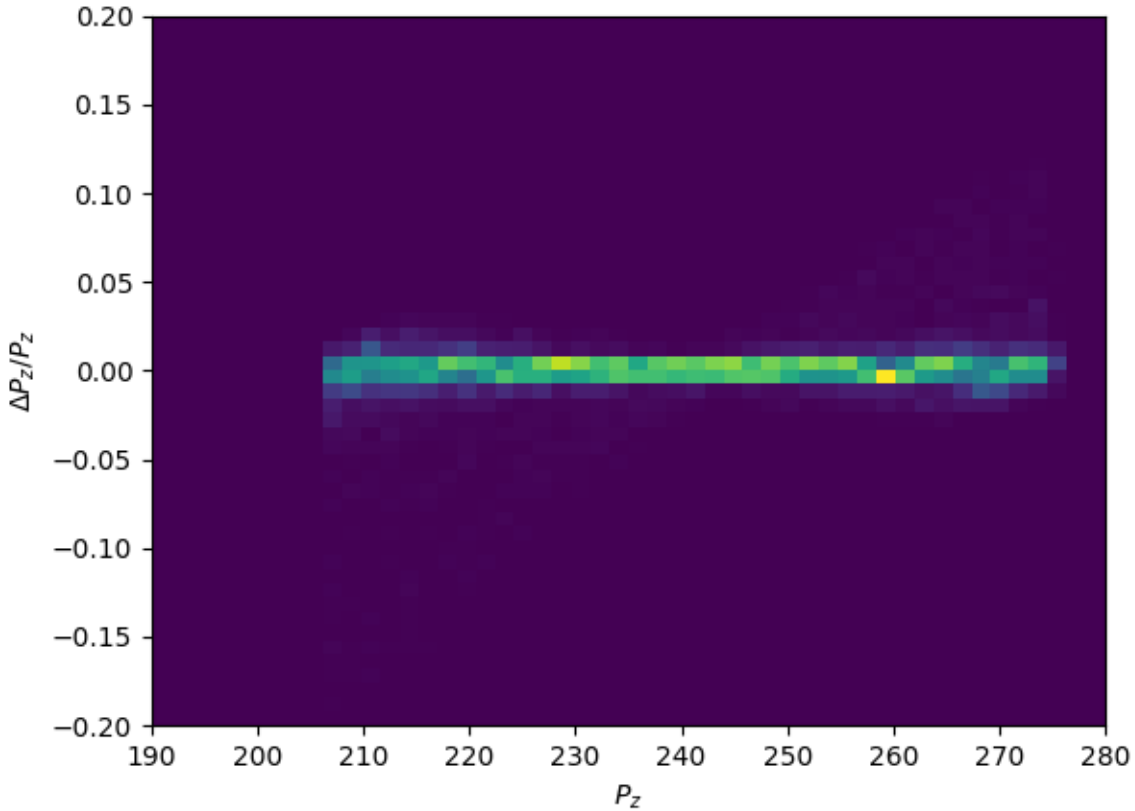
Stdev: 1.95%

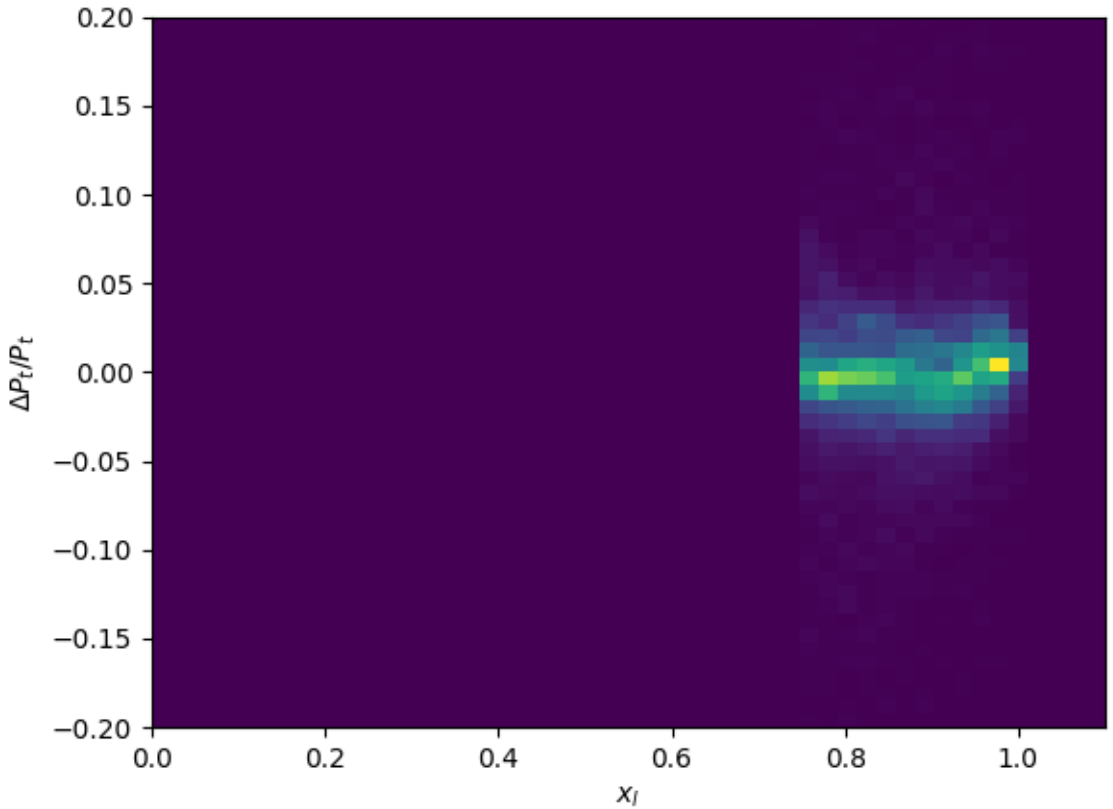


25 hrs

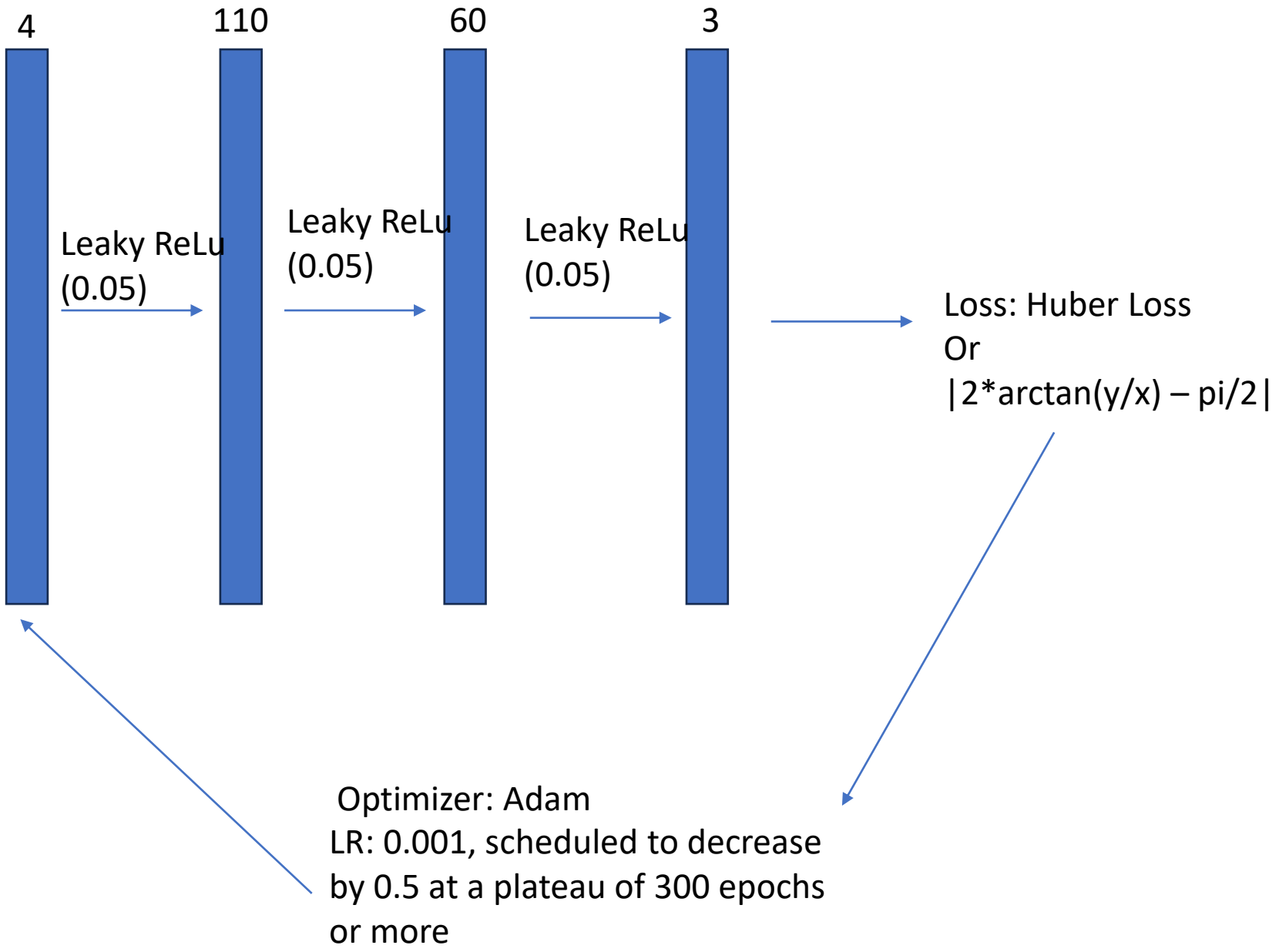
Next Steps

- Loss shows precision can still improve, but need to improve efficiency of network or use a cluster that can run the training for a very long time to get better results
- Continue to work on tuning efficiency, fine-tune the learning rate scheduler, and use UNH/Jlab/Brookhaven farm to run the training for several days.


0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
$\Delta P_z/P_z$
190
200
210
220
230
240
250
260
270
280
P_z


0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
$\Delta P_t/P_t$
0.0
0.2
0.4
0.6
0.8
1.0
x_l

Current Best Format:
Dense Network



Steps to Reproduce

- Use PyTorch to structure a sequential model, with Linear and Leaky ReLU layers alternating
- Convert input and output files to Torch tensors
- Loop through some number of Epochs
- For each epoch, use the model on the input data set
- Compare the output to the known training data and calculate the loss using some function
- Pass the loss to the backward function of the optimizer, which tunes the model parameters
- Repeat and ideally the loss will decrease as you run over more epochs
- Save the model, load it with the same parameters to test

EICRecon Integration

- Training file can be completely separate, run as its own command to update a parameter file
- EICRecon Factory can then pull from the parameter file with a matching model structure
- Work so far is in Python – rewrite with c++ PyTorch frontend?