Classification Baseline

Matt Zhang | June 08, 2018

Classification Bug

The classification architecture we used in NIPS had 4 hidden layers. It turns out that all 4 layers were constrained to have the same weight. This has been fixed and resulted in slightly better accuracy using NIPS architecture.

Before, we showed that scanning over n_hidden_layers had little effect. Due to the discovery of this bug, this result is obviously called into question.

```
class Classifier_Net(nn.Module):
    def __init__(self, hiddenLayerNeurons, nHiddenLayers, dropoutProb, windowSize):
        super().__init__()
        self.windowSize = windowSize
        self.nHiddenLayers = nHiddenLayers
        self.input = nn.Linear(windowSize * windowSize * 25, hiddenLayerNeurons)
        self.hidden = nn.Linear(hiddenLayerNeurons, hiddenLayerNeurons)
        self.hidden.cuda()
        self.dropout = nn.Dropout(p = dropoutProb)
        self.dropout.cuda()
        self.output = nn.Linear(hiddenLayerNeurons, 2)
    def forward(self, x, _):
        lowerBound = 26 - int(math.ceil(self.windowSize/2))
        upperBound = lowerBound + self.windowSize
        x = x[:, lowerBound:upperBound, lowerBound:upperBound]
        x = x.contiguous().view(-1, self.windowSize * self.windowSize * 25)
        x = self.input(x)
        for _ in range(self.nHiddenLayers-1):
            x = F.relu(self.hidden(x))
            x = self.dropout(x)
        x = F.softmax(self.output(x), dim=1)
        return x
```

NIPS architecture (when windowSize == 25)

```
class Classifier_Net(nn.Module):
    def __init__(self, hiddenLayerNeurons, nHiddenLayers, dropoutProb, windowSize):
        super().__init__()
        self.windowSize = windowSize
        self.nHiddenLayers = nHiddenLayers
        self.input = nn.Linear(windowSize * windowSize * 25, hiddenLayerNeurons)
        self.hidden = [None] * self.nHiddenLayers
        self.dropout = [None] * self.nHiddenLayers
        for i in range(self.nHiddenLayers):
            self.hidden[i] = nn.Linear(hiddenLayerNeurons, hiddenLayerNeurons)
            self.hidden[i].cuda()
            self.dropout[i] = nn.Dropout(p = dropoutProb)
            self.dropout[i].cuda()
        self.output = nn.Linear(hiddenLayerNeurons, 2)
    def forward(self, x, _):
        lowerBound = 25 - int(math.ceil(self.windowSize/2))
        upperBound = lowerBound + self.windowSize
        x = x[:, lowerBound:upperBound, lowerBound:upperBound]
        x = x.contiguous().view(-1, self.windowSize * self.windowSize * 25)
        x = self.input(x)
        for i in range(self.nHiddenLayers-1):
            x = F.relu(self.hidden[i](x))
            x = self.dropout[i](x)
        x = F.softmax(self.output(x), dim=1)
        return x
```

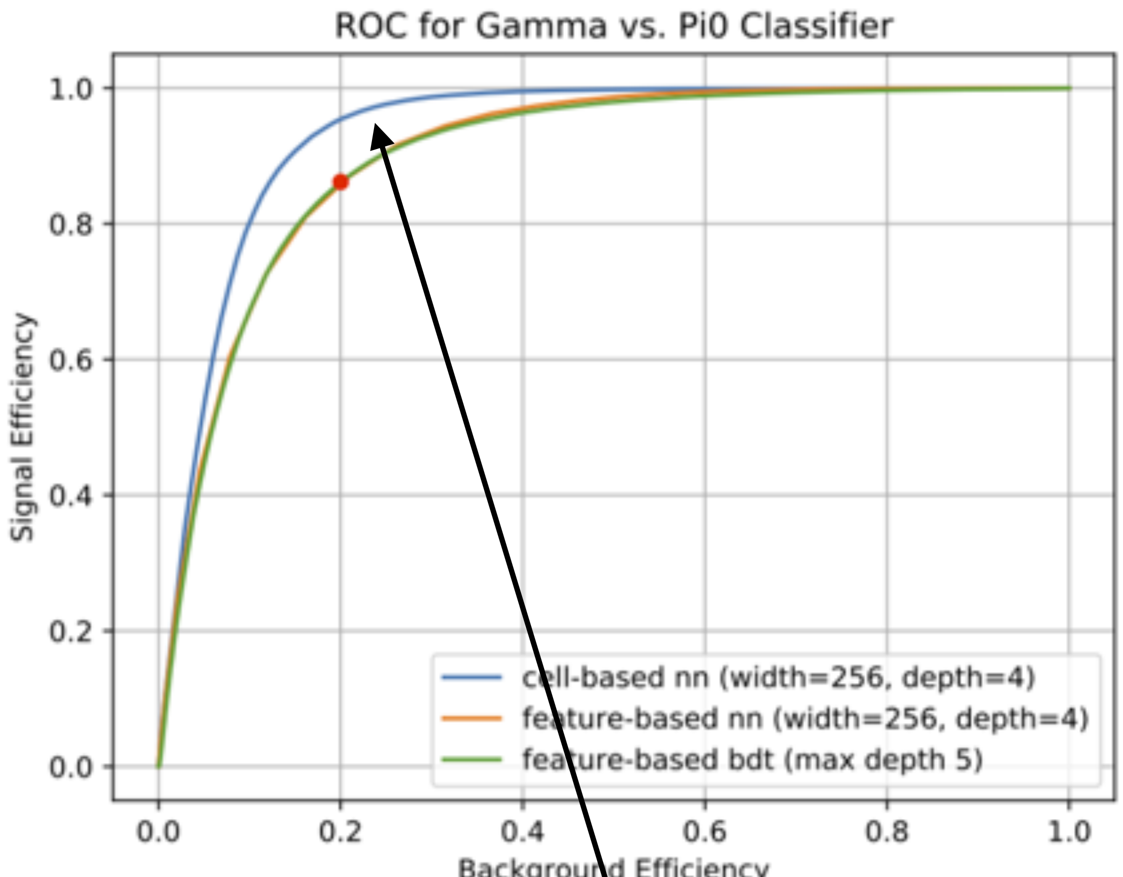
fixed architecture

Baseline Classification

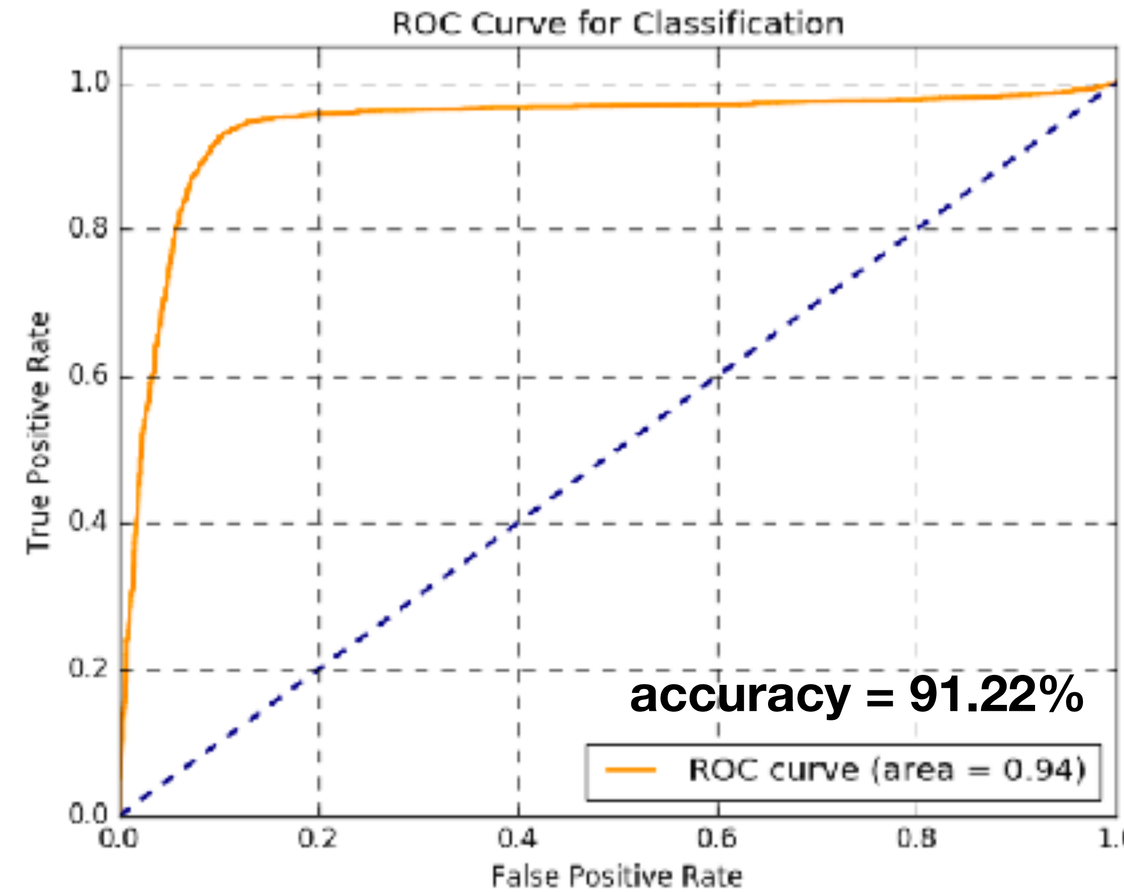
Using a window size of 25x25 on new fixed-angle samples with energy between 50-70 GeV, we get an analogue of the data used in the NIPS paper.

When using the original buggy architecture, we got very slightly worse results (sorry, overwrote the plot). With the fixed architecture, our accuracy is better by about 4 percent.

NIPS paper

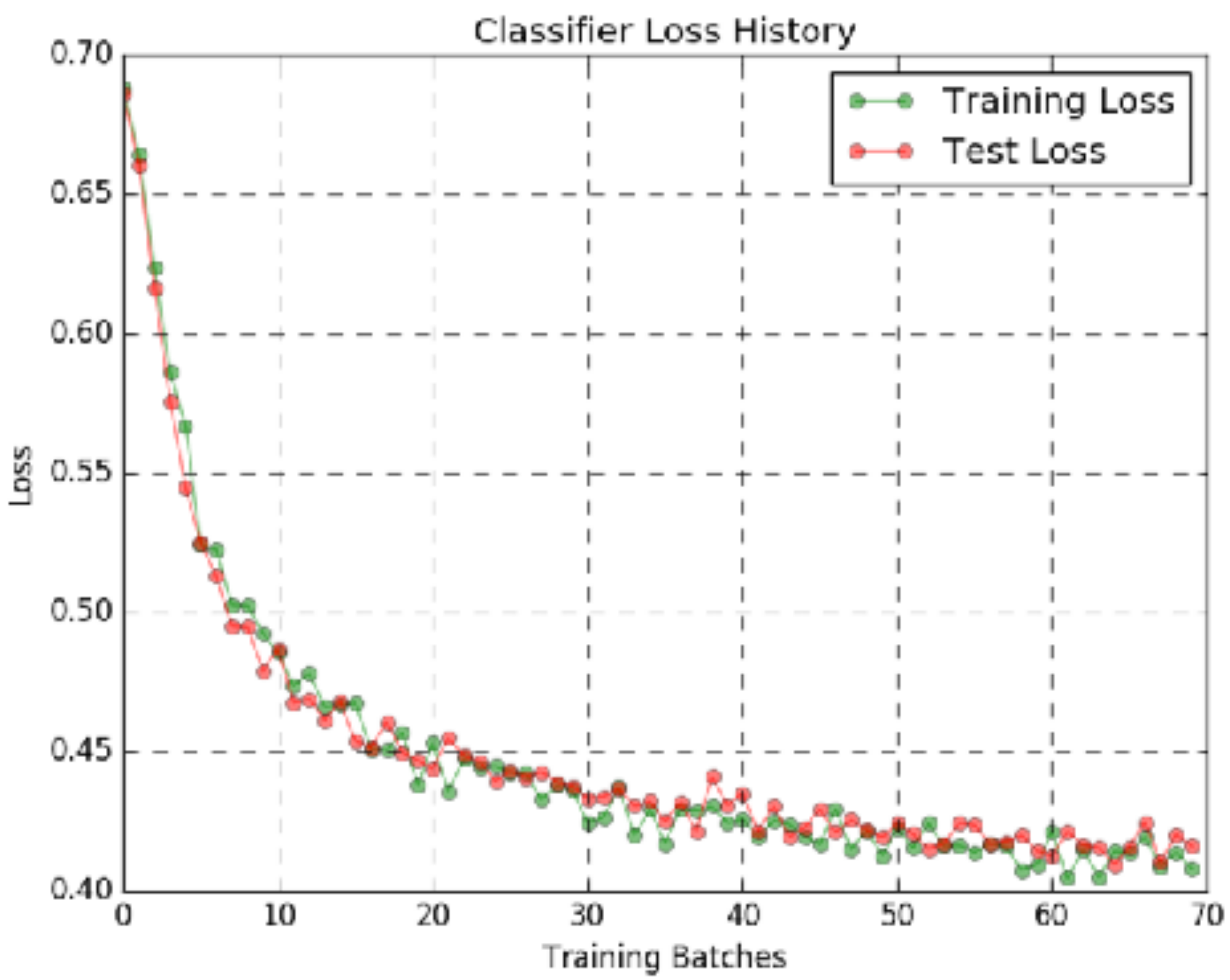


new samples

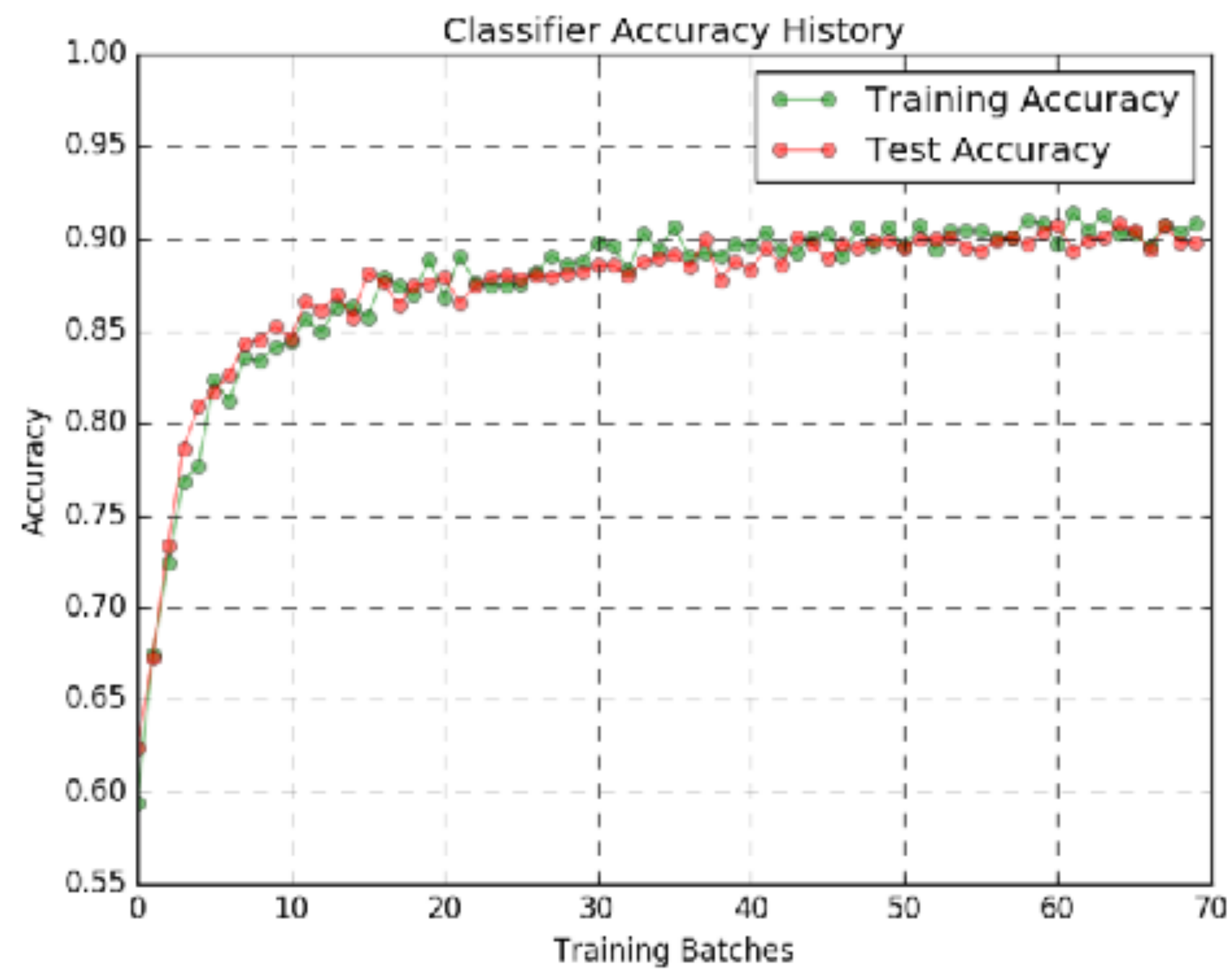


	γ vs. π^0				e vs. π			
Model	**acc.**	**AUC**	$\Delta\epsilon_{\mathrm{sig}}$	ΔR_{bkg}	**acc.**	**AUC**	$\Delta\epsilon_{\mathrm{sig}}$	ΔR_{bkg}
BDT	83.1%	89.8%	-	-	93.8%	98.0%	-	-
DNN (features)	82.8%	90.2%	0.9%	0.95	93.6%	98.0%	-0.1%	0.95
DNN (cells)	87.2%	93.5%	9.4%	1.63	99.4%	99.9%	4.9%	151

Table 1: Performance parameters for BDT and DNN classifiers.



Classifier Loss History
Training Loss
Test Loss
Loss
Training Batches
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
10
20
30
40
50
60
70

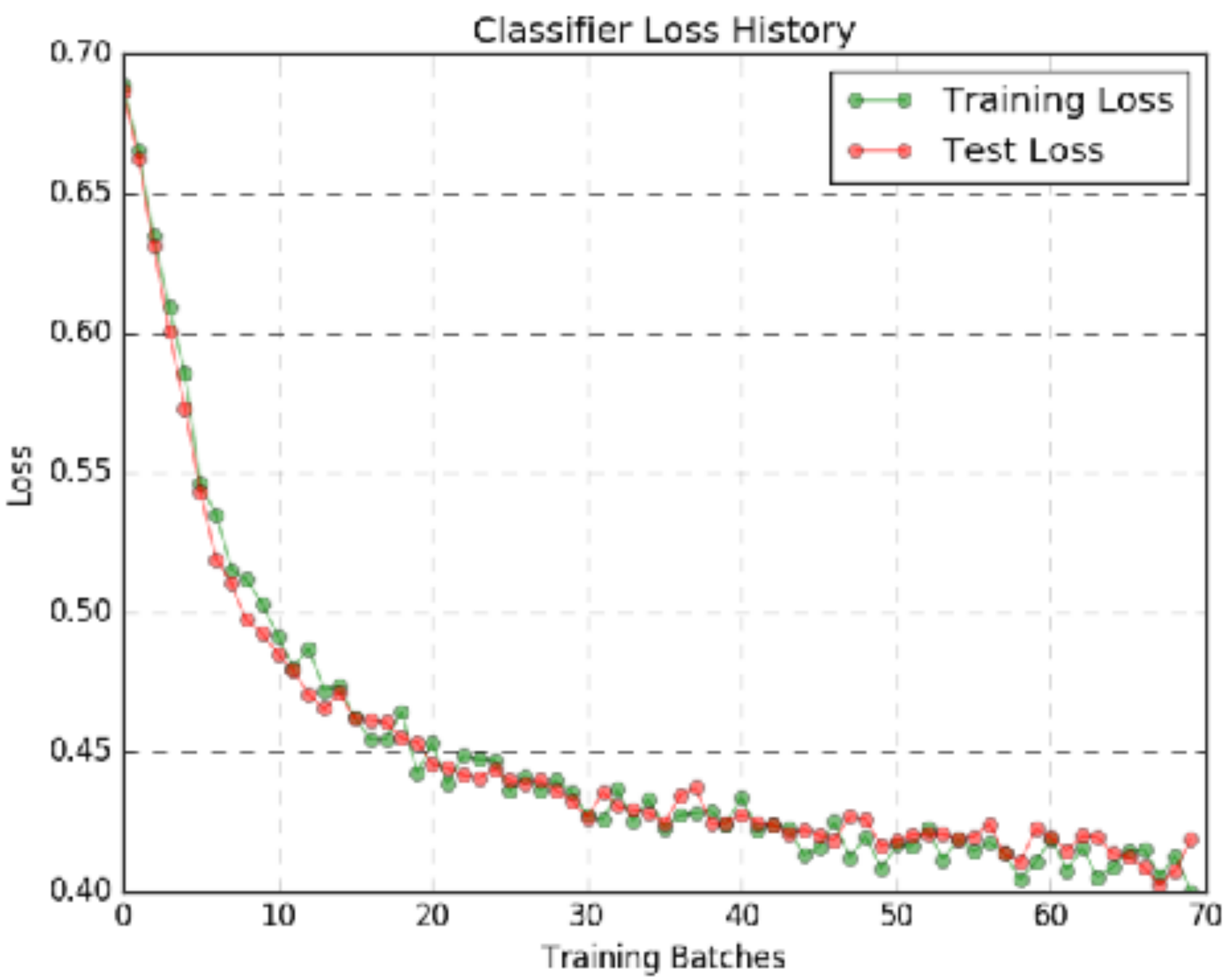


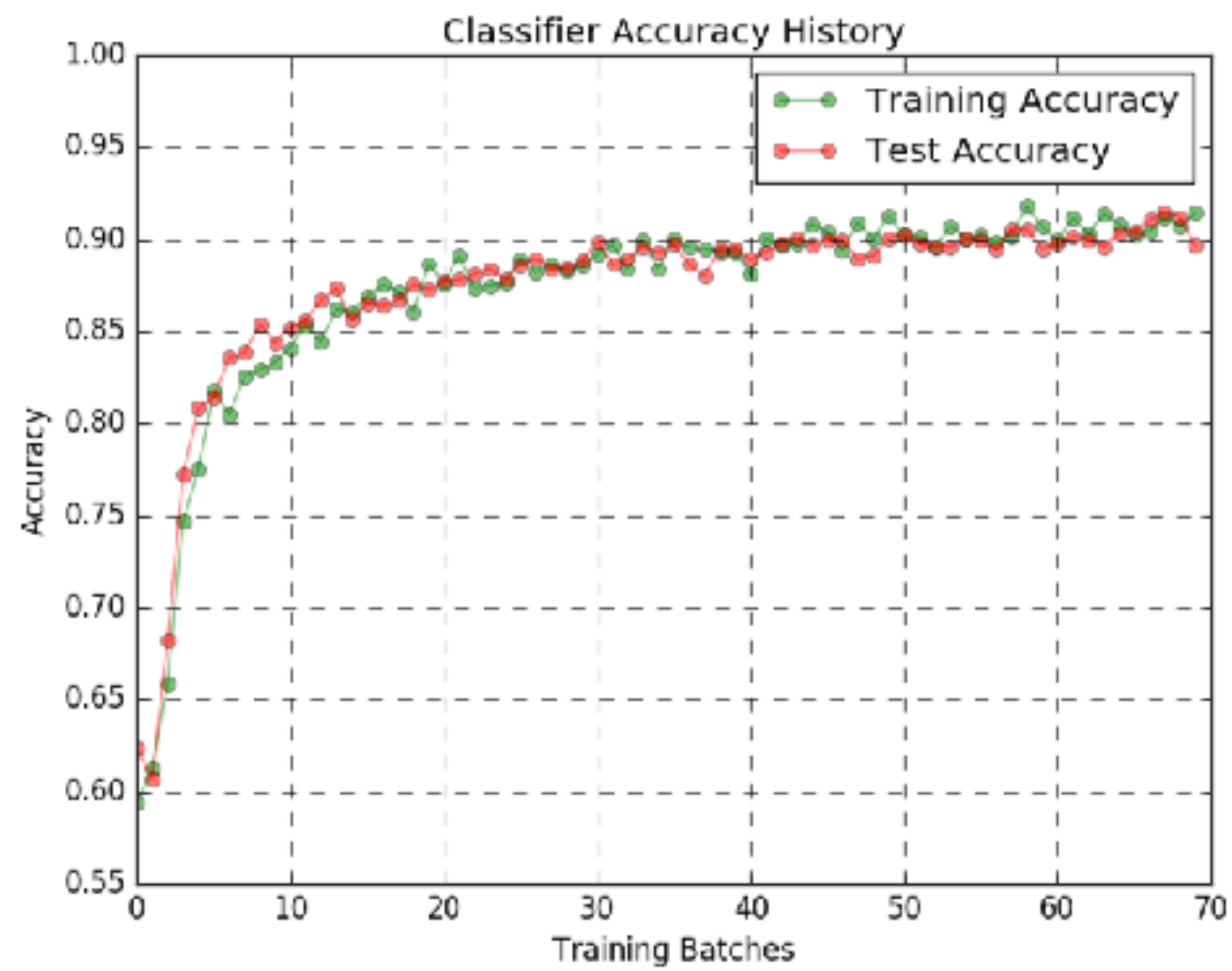
Classifier Accuracy History
Training Accuracy
Test Accuracy
Accuracy
Training Batches
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0
10
20
30
40
50
60
70

Baseline w/ Expanded Window

Using a window size of 51x51 on new fixed-angle samples with energy between 50-70 GeV, the same as before except with the larger window.

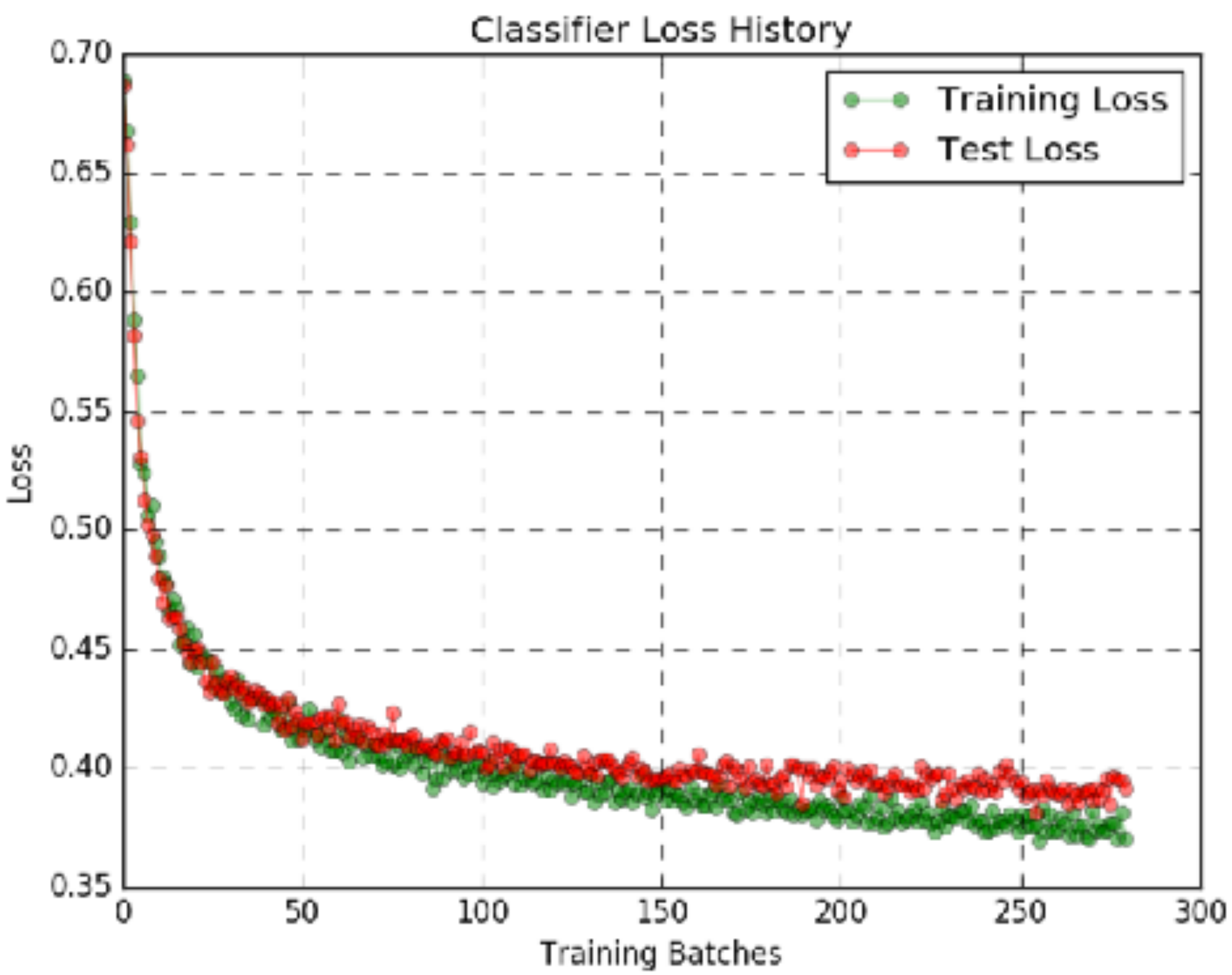
The results are maybe slightly better than the 25x25 window. Doesn't look like the larger window is either hurting or helping much. It's probably best to keep the larger window since it does contain some info.


Classifier Loss History
Training Loss
Test Loss
Loss
Training Batches
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
10
20
30
40
50
60
70


Classifier Accuracy History
Training Accuracy
Test Accuracy
Accuracy
Training Batches
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0
10
20
30
40
50
60
70

Expanded Window, Longer Train

Now training for 20 epochs instead of 5.



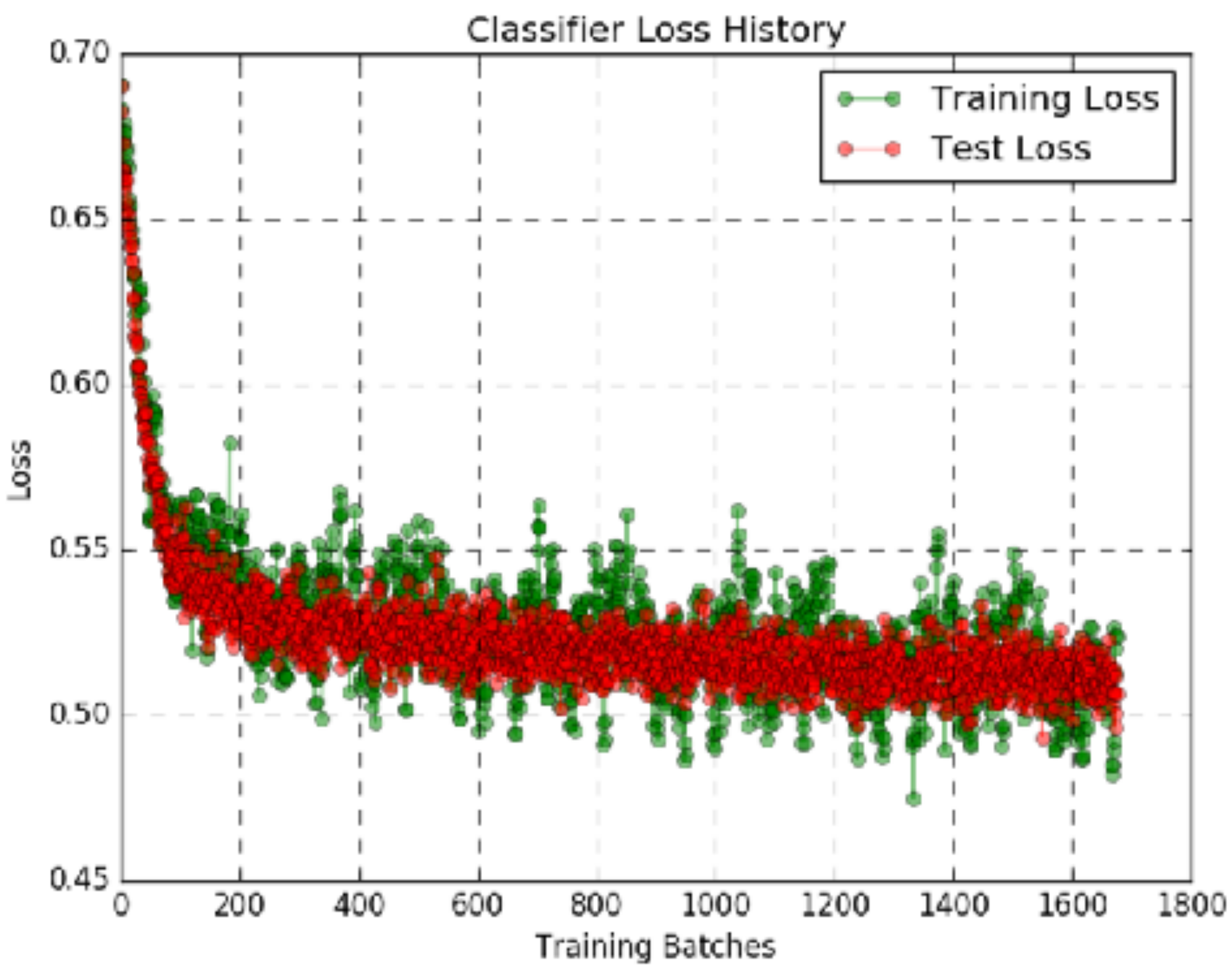
Classifier Loss History
Training Loss
Test Loss
Loss
Training Batches
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
50
100
150
200
250
300

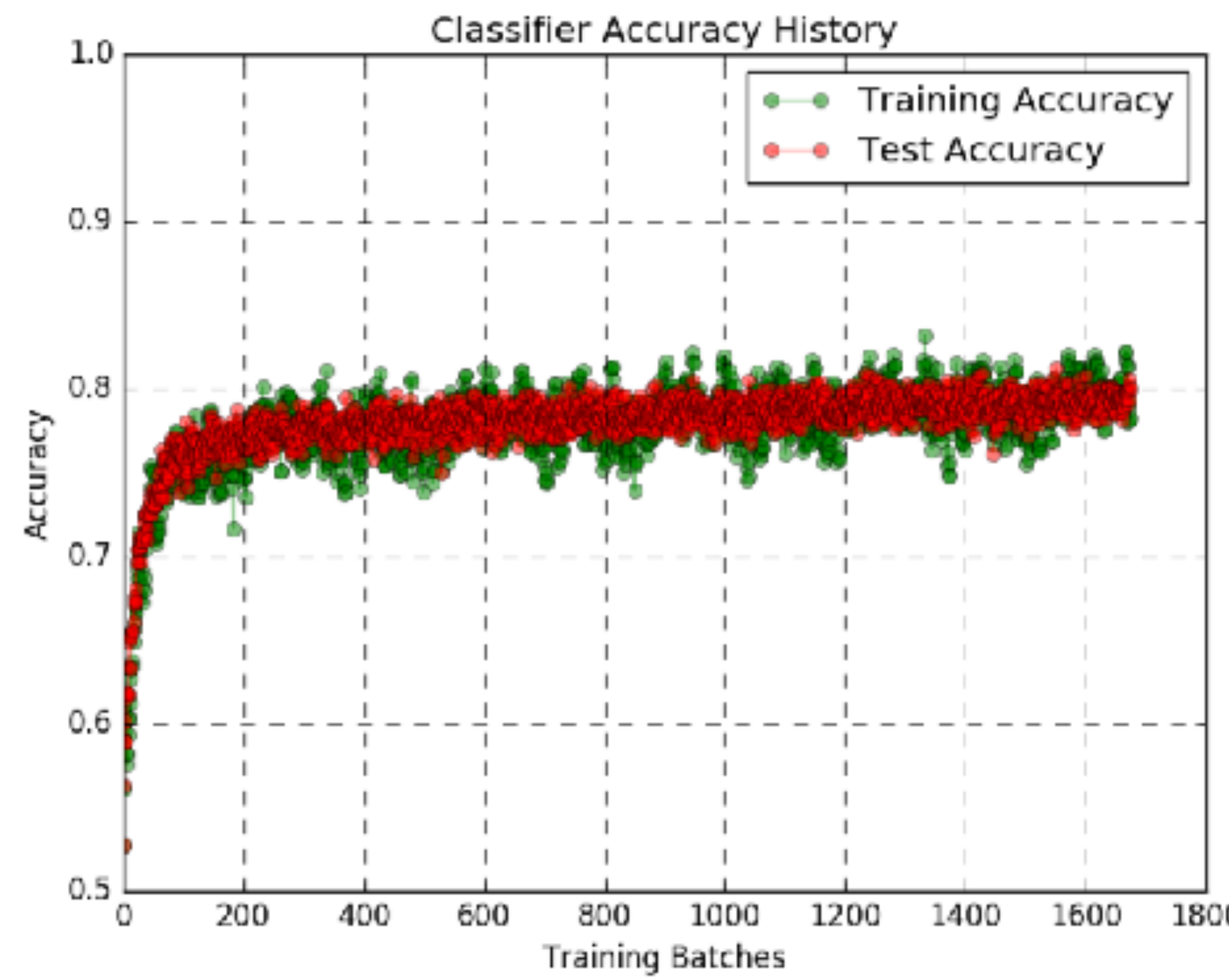
Classifier Accuracy History
Training Accuracy
Test Accuracy
Accuracy
Training Batches
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0
50
100
150
200
250
300

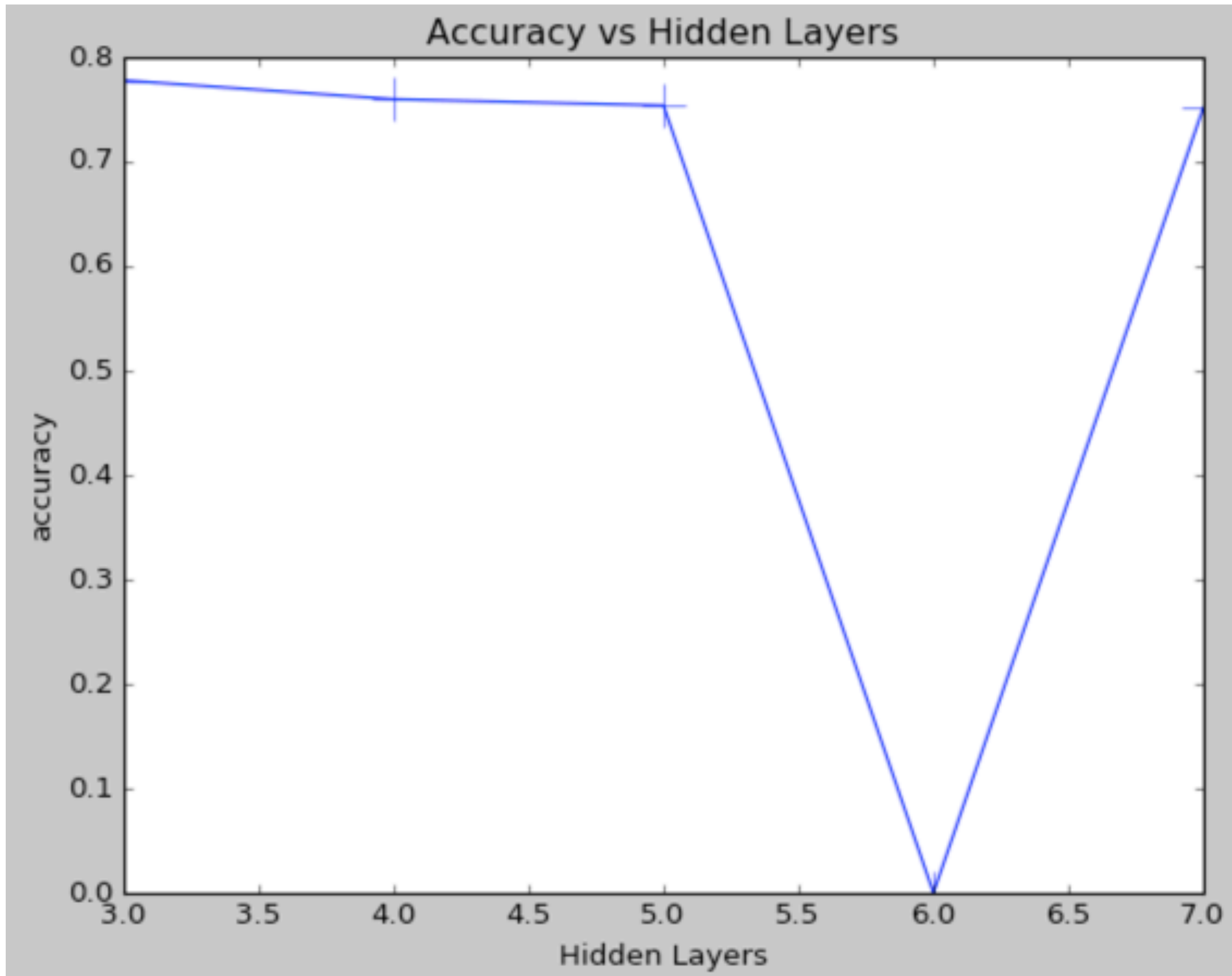
Energy-Range Classification

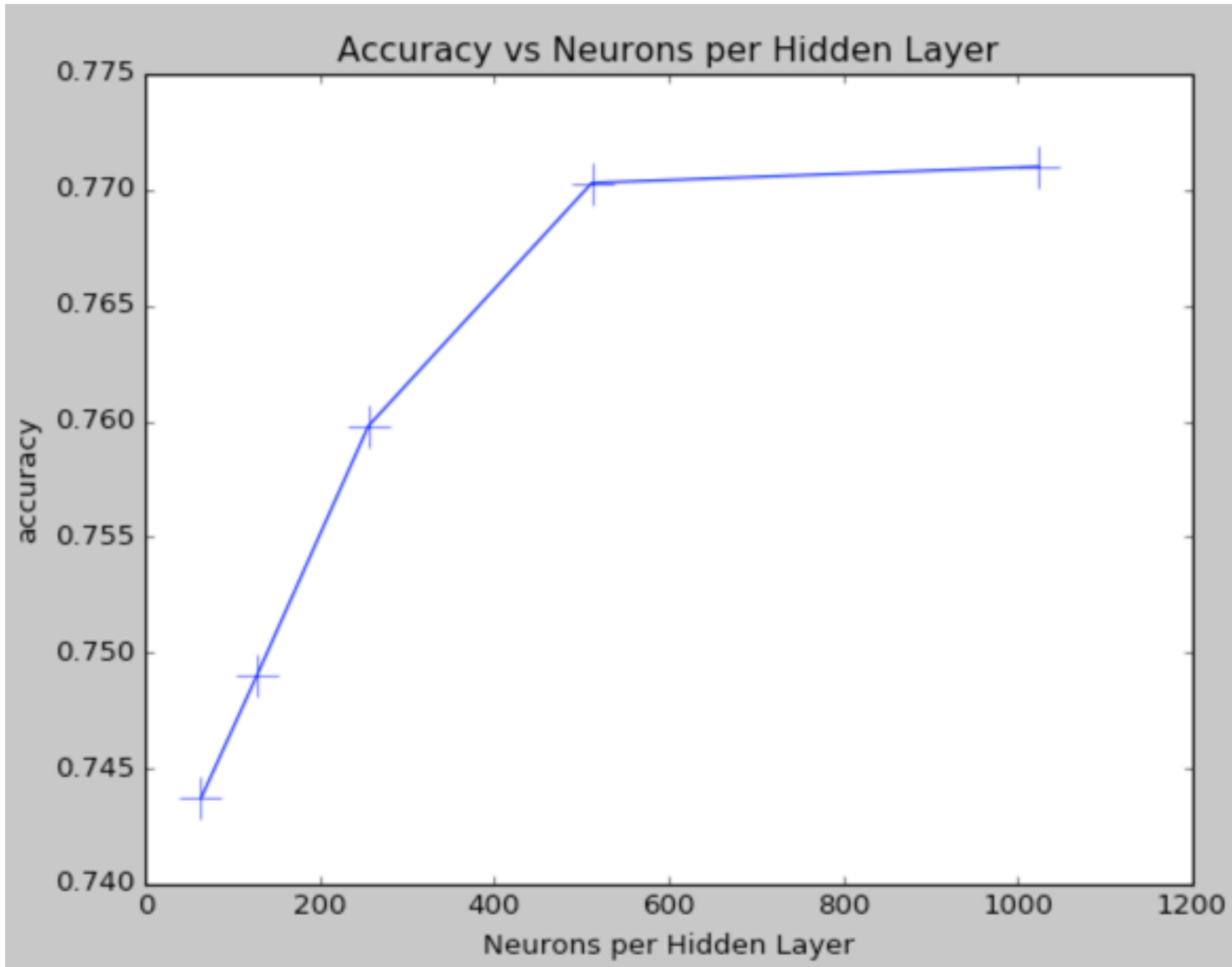
Still using the NIPS architecture with a window size of 25x25, we now use all of the new fixed-angle samples (instead of just those with energy between 50-70 GeV). Note there are more samples, so it takes longer to get through 5 epochs of training.

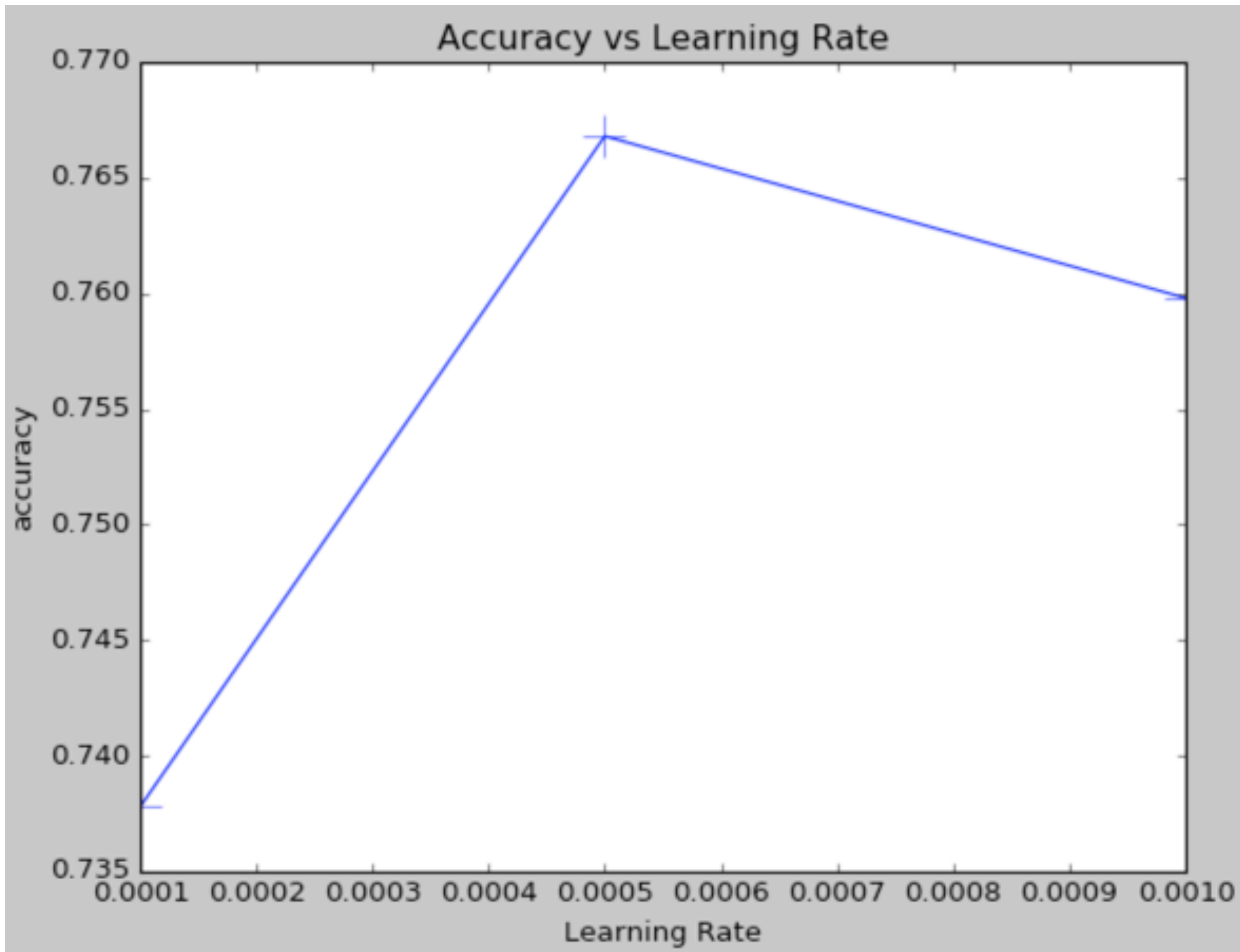
We also show the results of various hyperparameter scans.

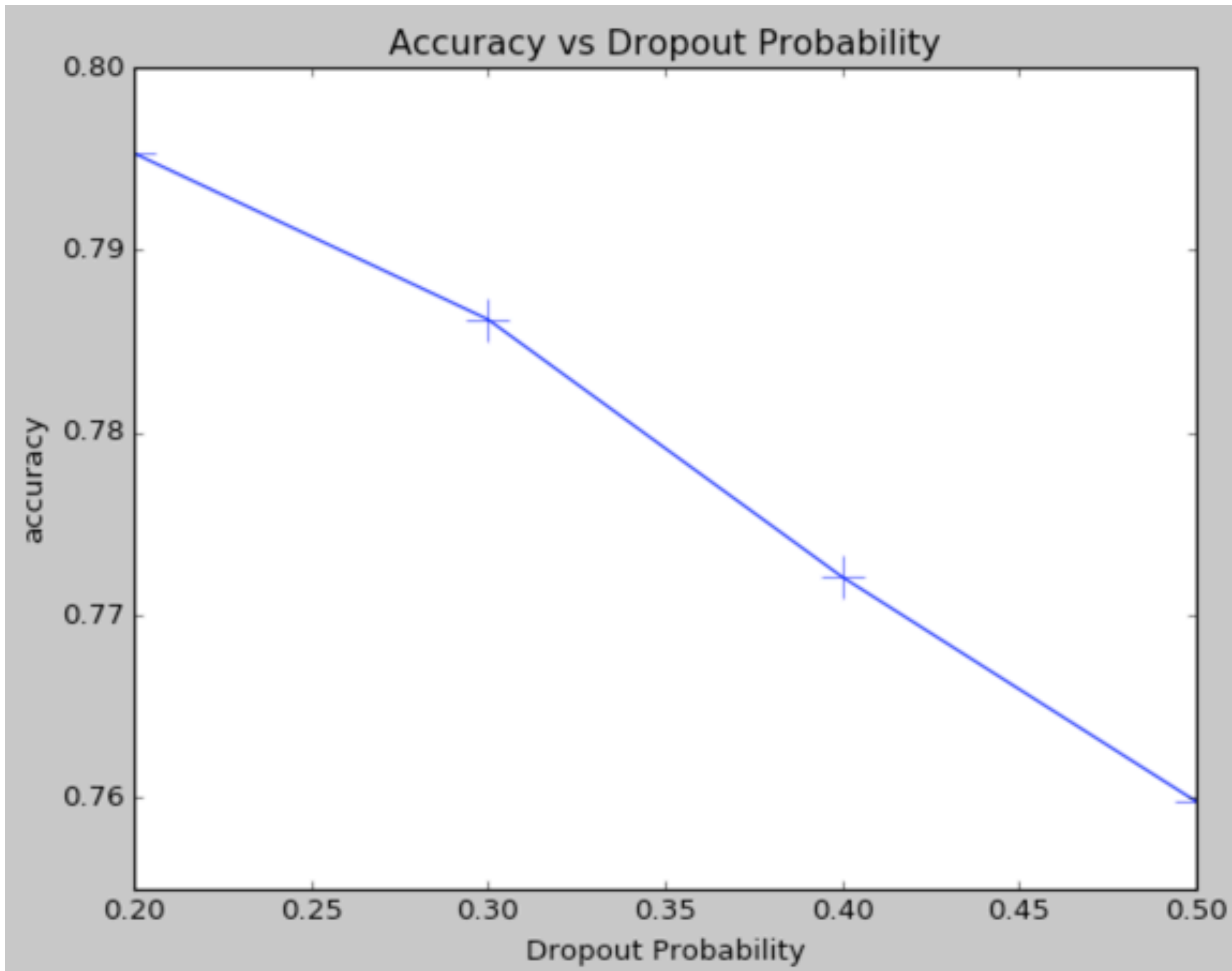

Classifier Loss History
Training Loss
Test Loss
Loss
Training Batches
0.70
0.65
0.60
0.55
0.50
0.45
0
200
400
600
800
1000
1200
1400
1600
1800

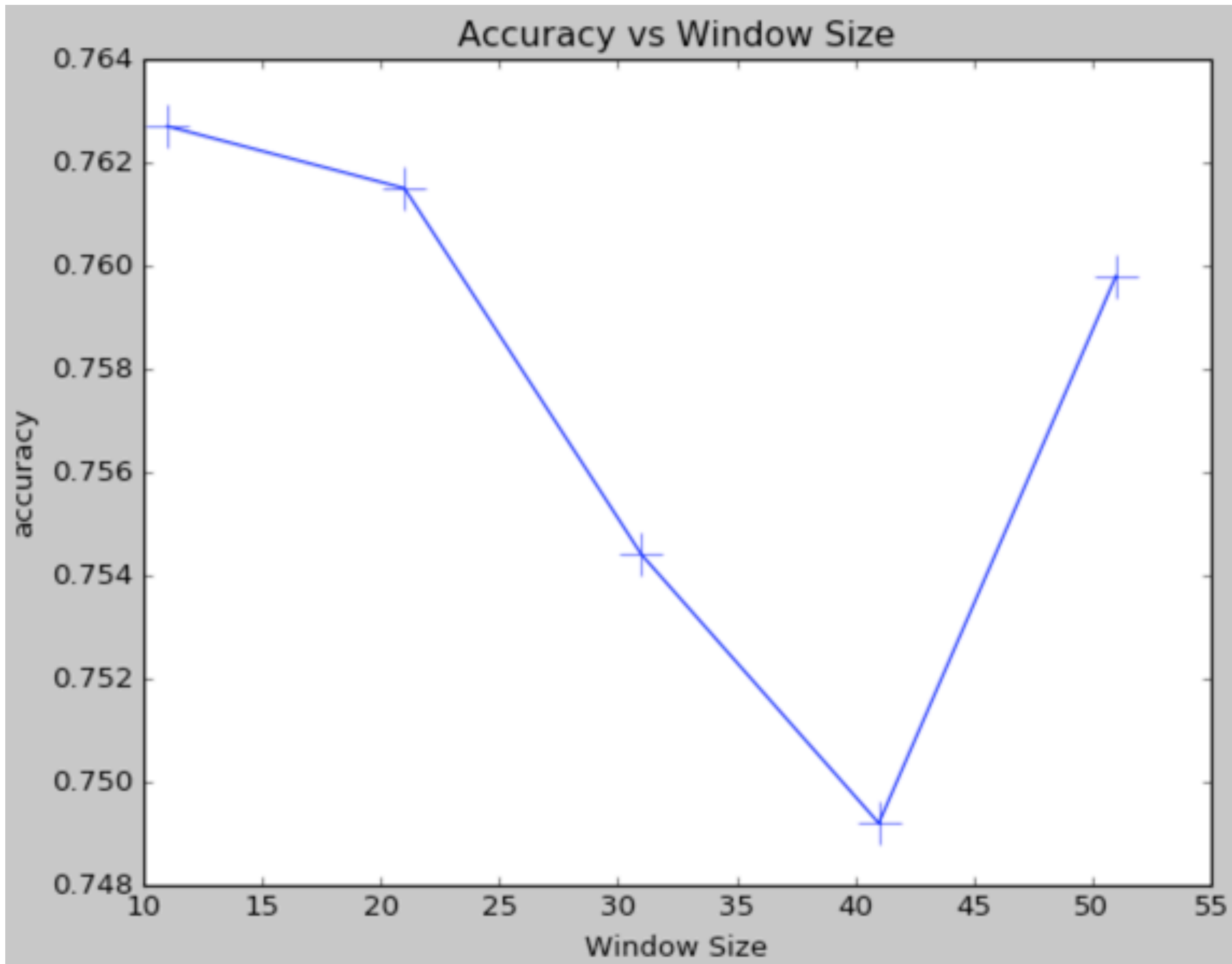

Classifier Accuracy History
Training Accuracy
Test Accuracy
Accuracy
Training Batches
1.0
0.9
0.8
0.7
0.6
0.5
0
200
400
600
800
1000
1200
1400
1600
1800

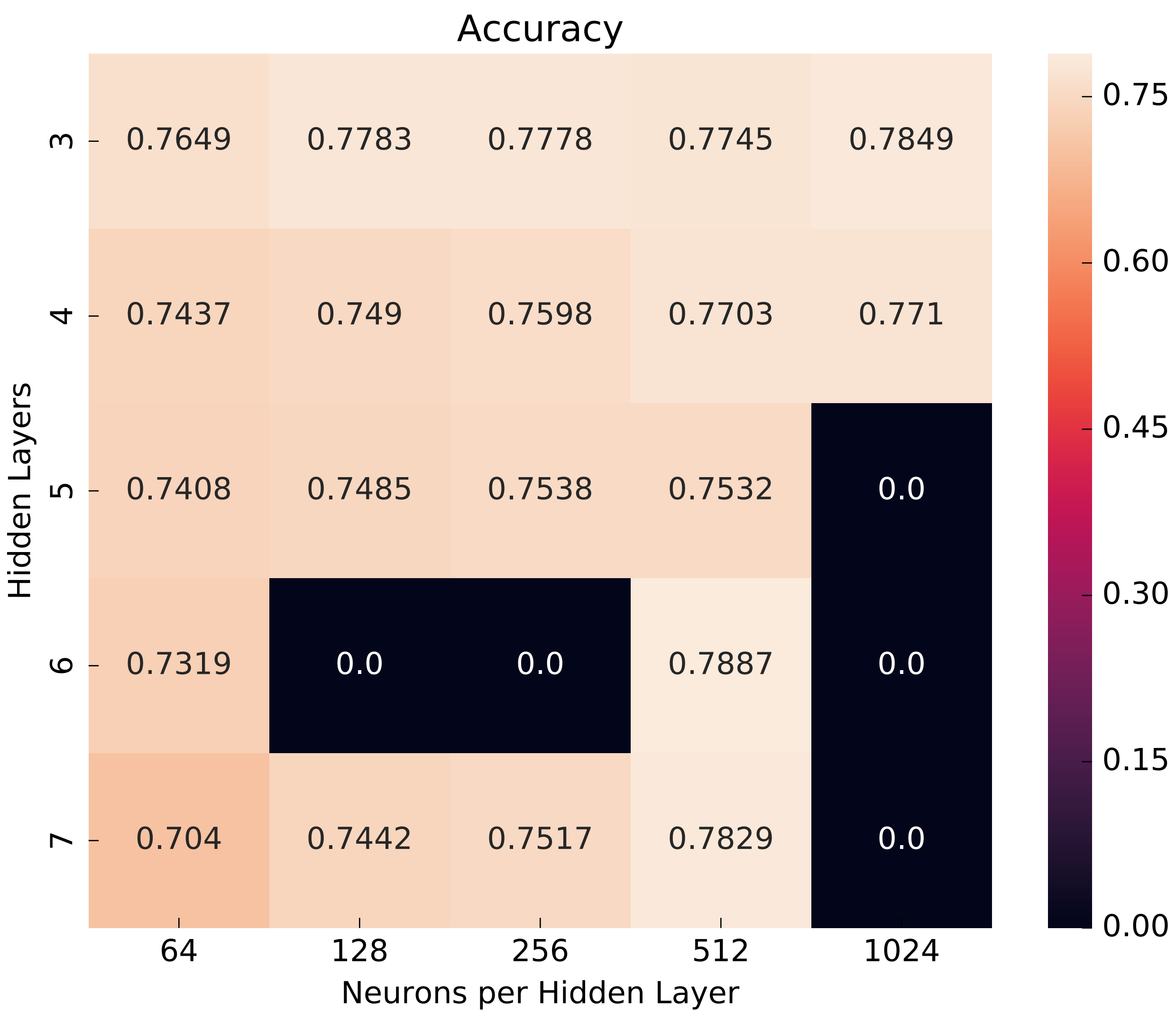

Accuracy vs Hidden Layers
accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
Hidden Layers

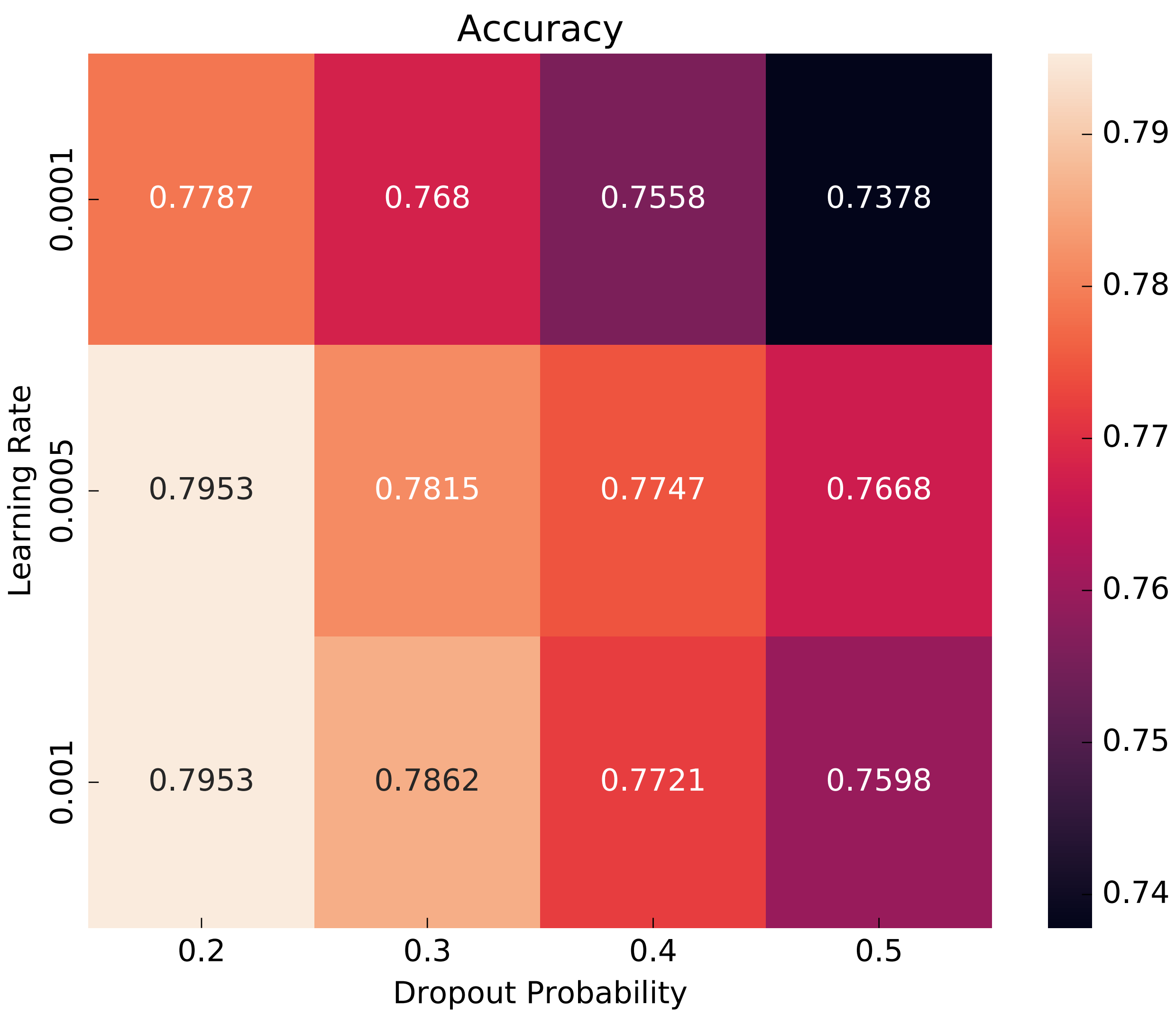

Accuracy vs Neurons per Hidden Layer
accuracy
0.775
0.770
0.765
0.760
0.755
0.750
0.745
0.740
0
200
400
600
800
1000
1200
Neurons per Hidden Layer


Accuracy vs Learning Rate
accuracy
0.770
0.765
0.760
0.755
0.750
0.745
0.740
0.735
0.0001
0.0002
0.0003
0.0004
0.0005
0.0006
0.0007
0.0008
0.0009
0.0010
Learning Rate

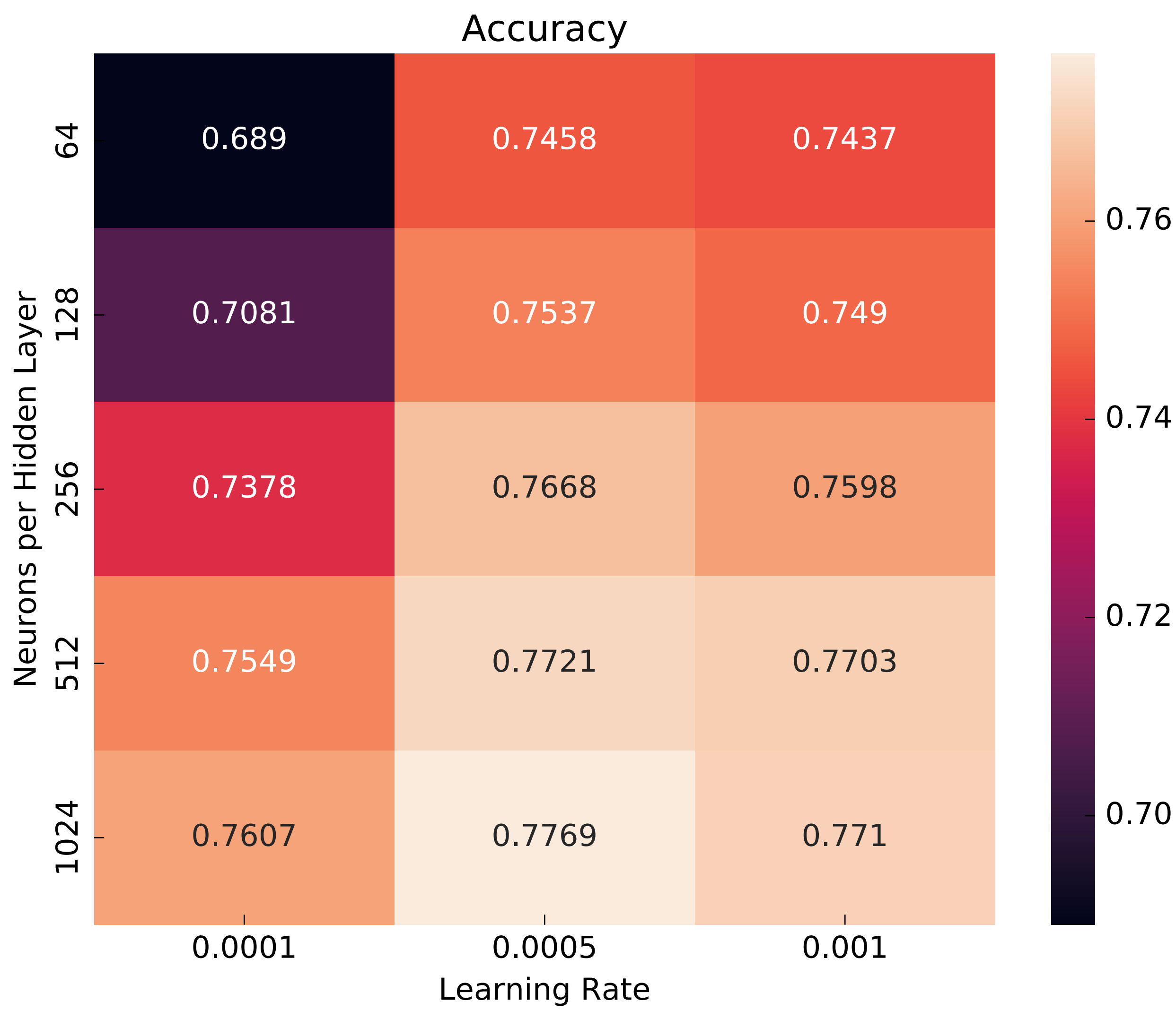

Accuracy vs Dropout Probability
0.80
0.79
0.78
0.77
0.76
accuracy
0.20
0.25
0.30
0.35
0.40
0.45
0.50
Dropout Probability

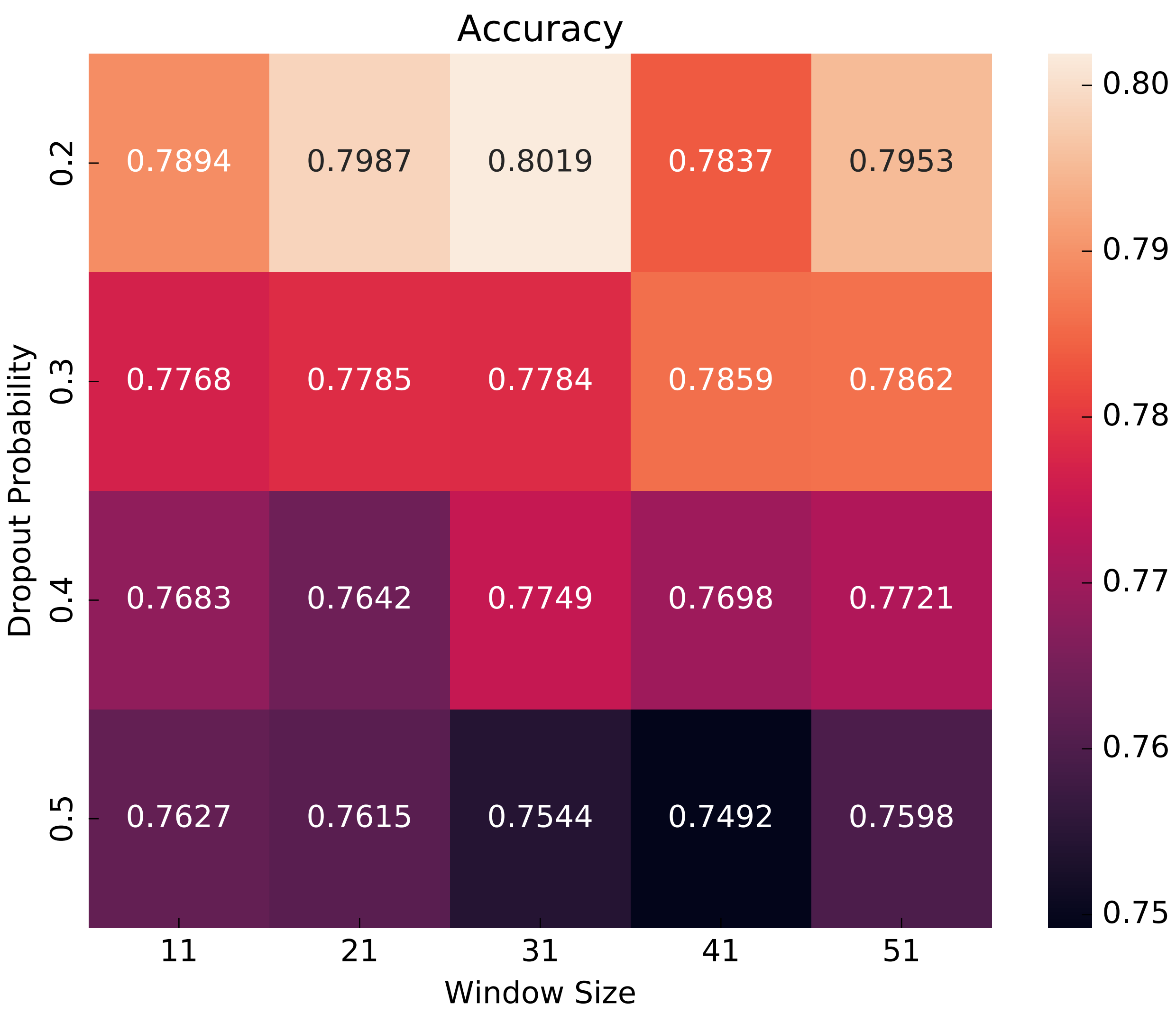

Accuracy vs Window Size
accuracy
0.764
0.762
0.760
0.758
0.756
0.754
0.752
0.750
0.748
10
15
20
25
30
35
40
45
50
55
Window Size


Accuracy
Hidden Layers
3
4
5
6
7
0.7649
0.7783
0.7778
0.7745
0.7849
0.7437
0.749
0.7598
0.7703
0.771
0.7408
0.7485
0.7538
0.7532
0.0
0.7319
0.0
0.0
0.7887
0.0
0.704
0.7442
0.7517
0.7829
0.0
64
128
256
512
1024
Neurons per Hidden Layer
0.75
0.60
0.45
0.30
0.15
0.00



Accuracy
0.7787
0.768
0.7558
0.7378
0.7953
0.7815
0.7747
0.7668
0.7953
0.7862
0.7721
0.7598
0.0001
0.0005
0.001
Learning Rate
0.2
0.3
0.4
0.5
Dropout Probability
0.79
0.78
0.77
0.76
0.75
0.74


Accuracy
Neurons per Hidden Layer
64
128
256
512
1024
Learning Rate
0.0001
0.0005
0.001
0.689
0.7458
0.7437
0.7081
0.7537
0.749
0.7378
0.7668
0.7598
0.7549
0.7721
0.7703
0.7607
0.7769
0.771
0.76
0.74
0.72
0.70


Accuracy
0.7894
0.7987
0.8019
0.7837
0.7953
0.7768
0.7785
0.7784
0.7859
0.7862
0.7683
0.7642
0.7749
0.7698
0.7721
0.7627
0.7615
0.7544
0.7492
0.7598
0.2
0.3
0.4
0.5
Dropout Probability
11
21
31
41
51
Window Size
0.80
0.79
0.78
0.77
0.76
0.75