Using ANN to Predict Credit Card Defaults Next Month

Juliusz Gasior

Problem Definition

Can a Taiwanese bank predict if a customer will default on their next month's credit card bill based on customer characteristics and their payment history from the last 6 months?

This is a binary classification problem where the target is:

- 1 (yes, the customer will default next month)
- 0 (no, the customer will pay next month)

Relevance

- Can be used to identify risky customers to the bank for further investigation on how to minimize these risks
  - Payment plans, reduction in available credit, removing as a customer
- This methodology can be used in other aspects
  - EX: a car manufacturer’s accounts receivable department may be able to identify high risk suppliers based on past payment obligations and the type of supplier

Executive Summary

- A multilayer ANN model that had gone through 13 experiments to maximize its accuracy was able to achieve a testing accuracy of about 81.7%

- The ANN model used was able to perform as well, from an accuracy point of view, when compared against a multilayer perceptron model used in a research paper using similar data. (Begüm & Ünal, 2019)

Methodology

- Model Selection
  - ANN model chosen due to nature of the data
- Architecture
  - Architecture of the model was chosen at random to start
  - By experiments #8 through to #11, the model architecture was finalized
- Techniques
  - Data exploration and model building done using Python libraries and in Jupyter Notebooks
  - Numpy used to create data arrays for parameter experimentation and holding performance data like losses and accuracies

Starting Architecture

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                [-1, 1, 64]           1,536
              ReLU-2                [-1, 1, 64]               0
            Linear-3                [-1, 1, 16]           1,040
              ReLU-4                [-1, 1, 16]               0
            Linear-5                 [-1, 1, 1]              17
           Sigmoid-6                 [-1, 1, 1]               0
================================================================
Total params: 2,593
Trainable params: 2,593
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.01
Estimated Total Size (MB): 0.01
----------------------------------------------------------------
```

Optimizer = SGD
Loss function = BCE

Ending Architecture

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                [-1, 1, 64]           1,536
              Tanh-2                [-1, 1, 64]               0
            Linear-3                [-1, 1, 64]           4,160
              Tanh-4                [-1, 1, 64]               0
            Linear-5                [-1, 1, 64]           4,160
              Tanh-6                [-1, 1, 64]               0
            Linear-7                [-1, 1, 64]           4,160
              Tanh-8                [-1, 1, 64]               0
            Linear-9                 [-1, 1, 1]              65
          Sigmoid-10                 [-1, 1, 1]               0
================================================================
Total params: 14,081
Trainable params: 14,081
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.05
Estimated Total Size (MB): 0.06
----------------------------------------------------------------
```

Optimizer = SGD
Loss function = BCE

Data Source

Variable	Description	Contents
X1	Amount of credit given	Integer
X2	Gender	1 = male; 2 = female
X3	Education	1 = graduate school; 2 = university; 3 = high school; 4 = others
X4	Marital Status	1 = married; 2 = single; 3 = others
X5	Age (year)	Integer
X6-X11	History of payment in last 6 months	-1 = duly paid; 1 = payment delay of 1 month; 2 = payment delay of 2 months; … 9 = payment delay of 9 months and so on
X12-X17	Amount of bill statement in last 6 months	Integer
X18-X23	Amount of previous payment in last 6 months	
Y	Default Payment	1 = Yes; 0 = No

Default of Credit Card Clients - UCI Machine Learning Repository

Hyperparameter Experimentation

	Hyperparameter	Thresholds Tested	Objective
1	Learning Rate	0.001 – 0.1	Improved accuracy and loss reduction
2	Number of Epochs	10,000 – 20,000	Loss convergence
3	Data Normalization	None, 0-1 Max, Z-Score	Loss reduction
4	Train Split Size	75%, 90%	Improved accuracy and loss reduction
5	Batch Sizes	16 - 8192	Loss reduction & train/test accuracy tracking
6	Epoch Resizing		Training speed

Experimentation Cont'd

7	Regularization	L1, L2, Dropout	Train and test accuracy
8	Breadth and Depth	Units: 8, 32, 64, 128 Layers: 1, 2, 3	Improved accuracy and loss reduction
9	Batch Normalization	With and Without	Improved accuracy and loss reduction
10	Output Activation Function	Sigmoid, ReLU	Improved accuracy and loss reduction
11	Hidden Layer Activation Functions	ReLU, ReLU6, Tanh	Improved accuracy and loss reduction

Experimentation Cont’d

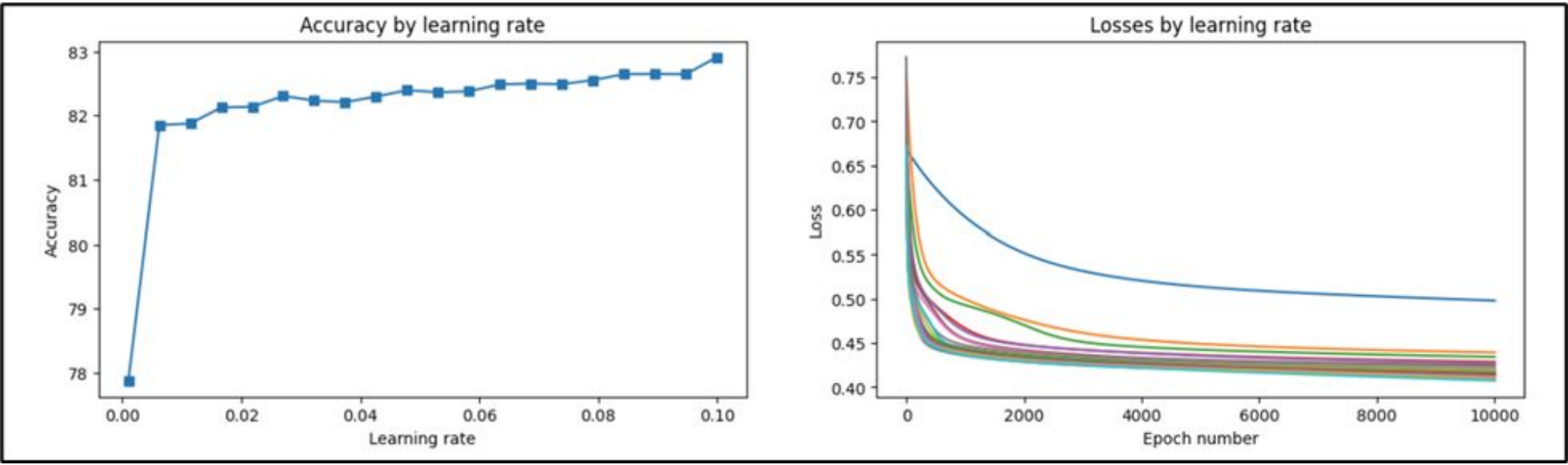
12	Optimizer	SGD, RMSprop, Adam	Improved accuracy and loss reduction
12b	Learning Rate	0.1 - 1	Improved accuracy and loss reduction
13	Momentum	0.000 – 0.999	Improved accuracy and loss reduction

Analysis

For most experiments, the models were evaluated using loss, training accuracy and testing accuracy

Main objective was to choose hyperparameters that would reduce loss as much as possible to some convergence value, as well as have the testing accuracy match the training accuracy as much as possible

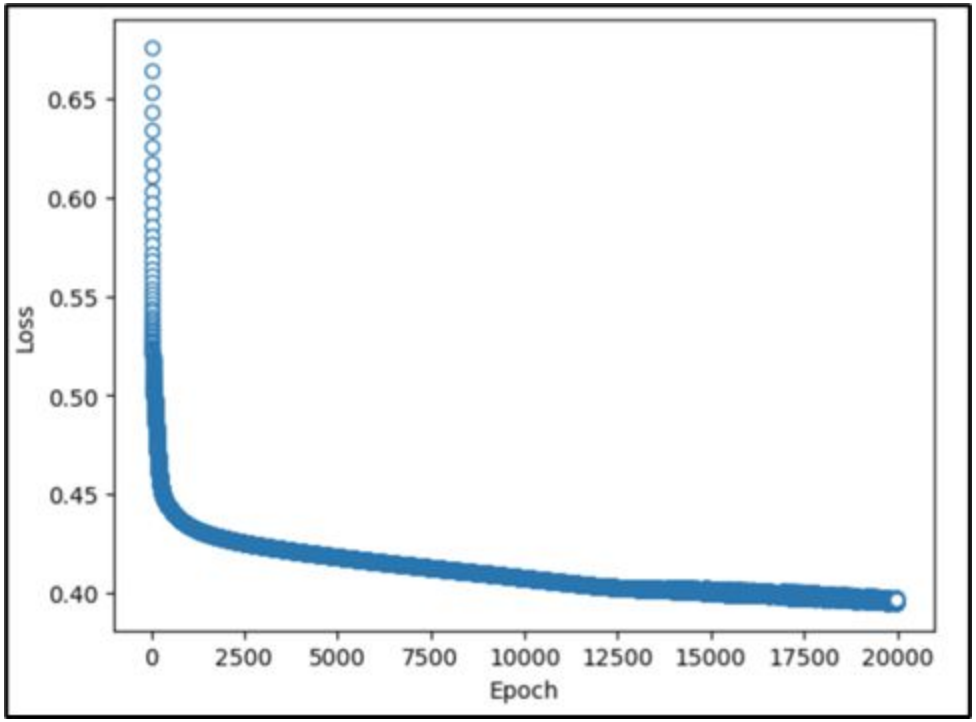
Experiment 1 - Learning Rate



Learning rate of 0.1 was chosen as default rate

However, there is evidence that a larger learning rate may provide a more accurate model

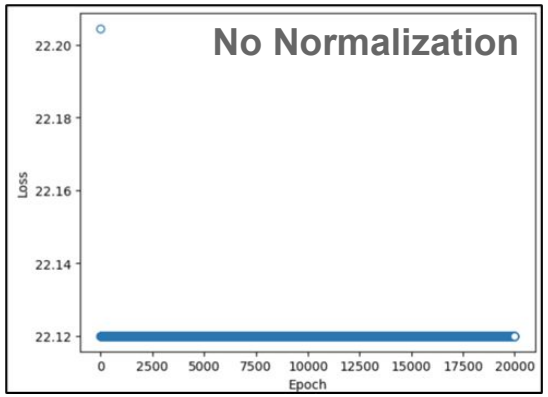
Experiment 2 - Epochs

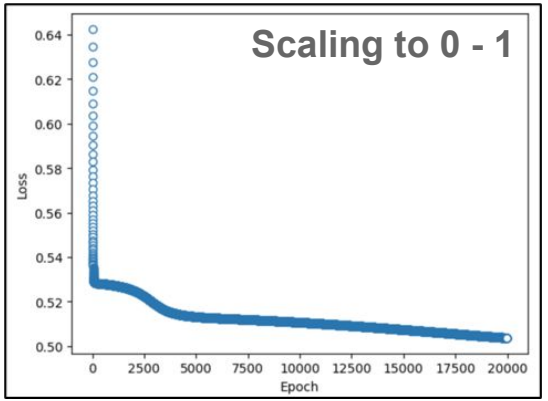


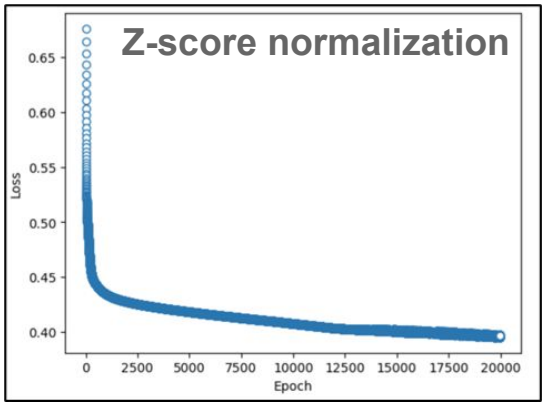
Slow linear decrease where convergence is not well seen

Keep number of epochs at 20,000 to keep training speed reasonable

Experiment 3 - Data Normalization

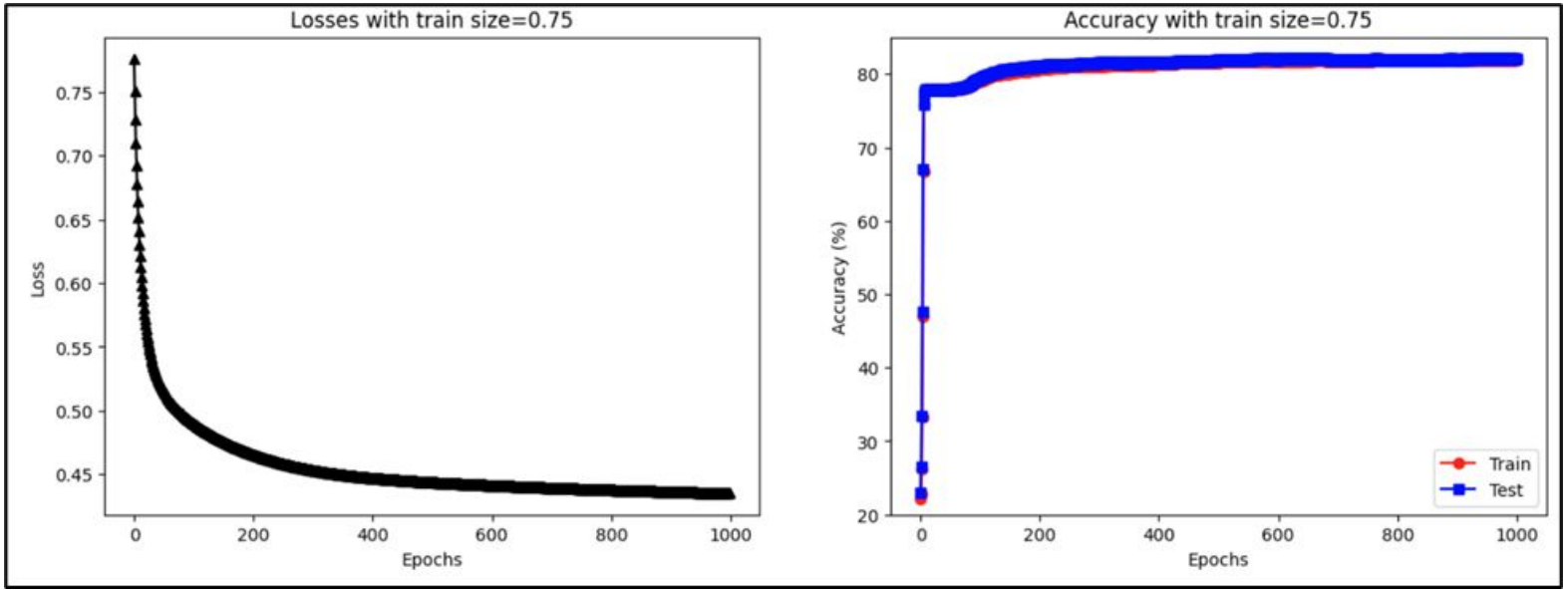






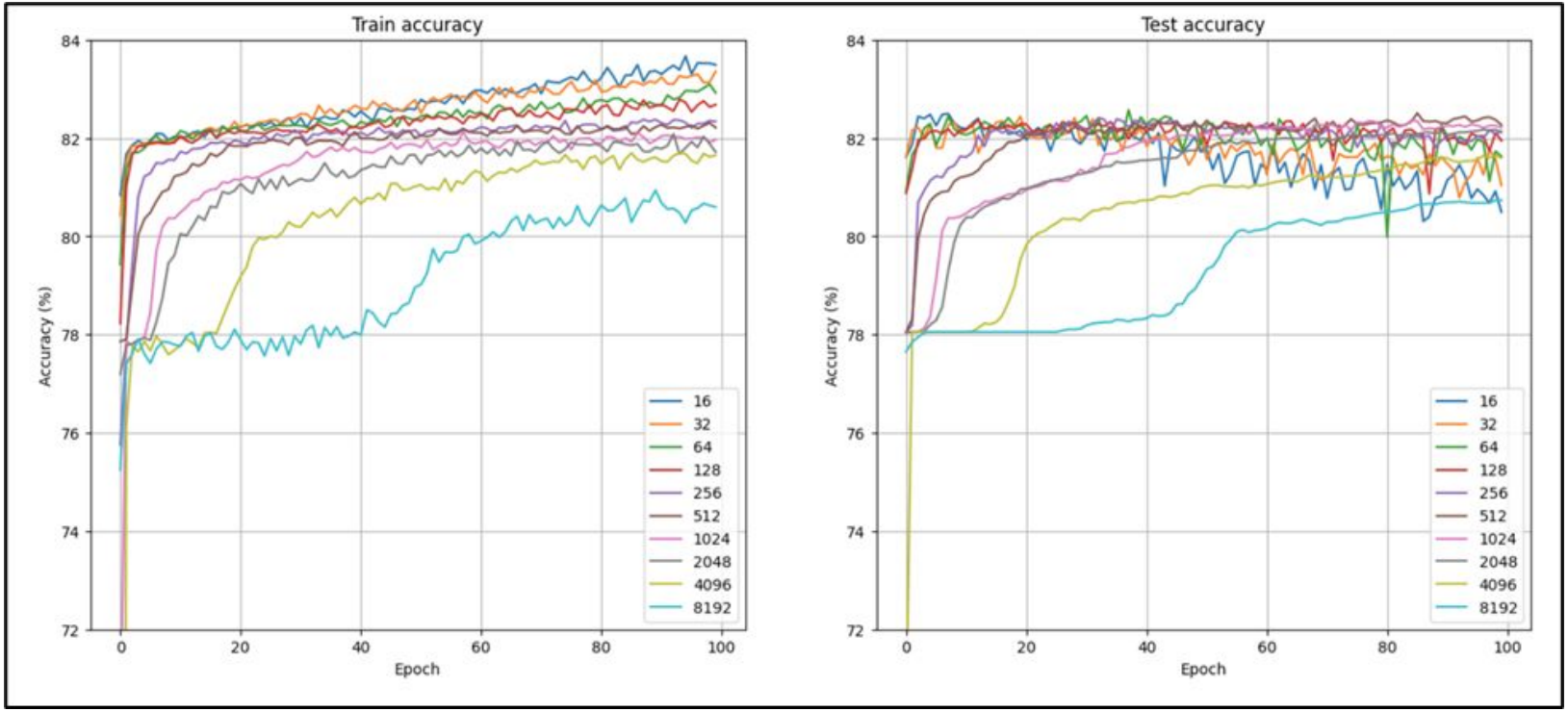
Loss curve for Z-score looks best and converges to 0.4
This is the better option

Experiment 4 - Train Split Size

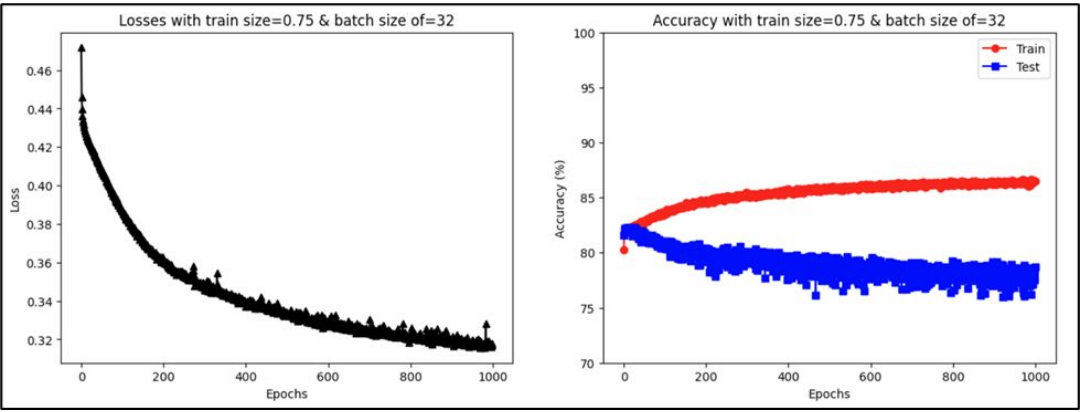


Tested train split sizes of 75% and 90% (both seem adequate)
Ultimately used 75% for subsequent experiments for more robust testing

Experiment 5 - Batch Sizes



Batch Sizes Cont'd

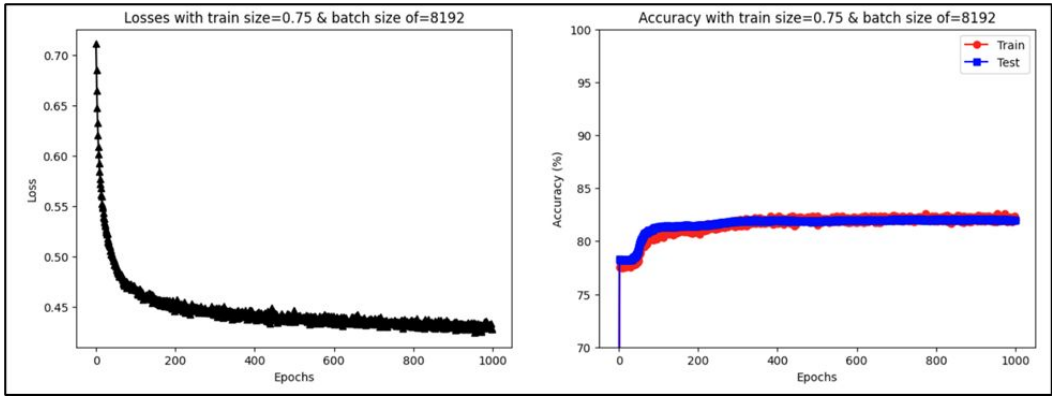


Small batch sizes provided lower losses but test accuracy deviated too much from training accuracy

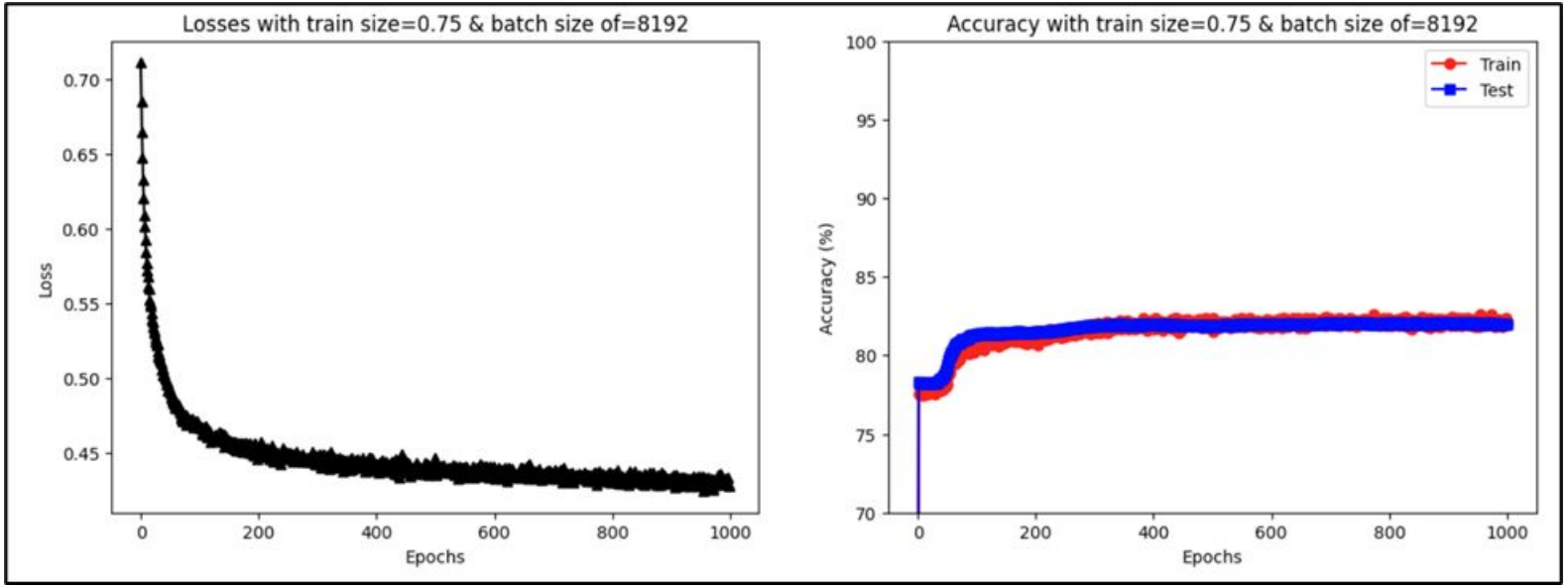
Losses also did not converge quickly

Larger batch sizes provided better loss convergence, albeit at a higher rate, and better test accuracy tracking to training accuracy

Chose a batch size of 8192 for experiments



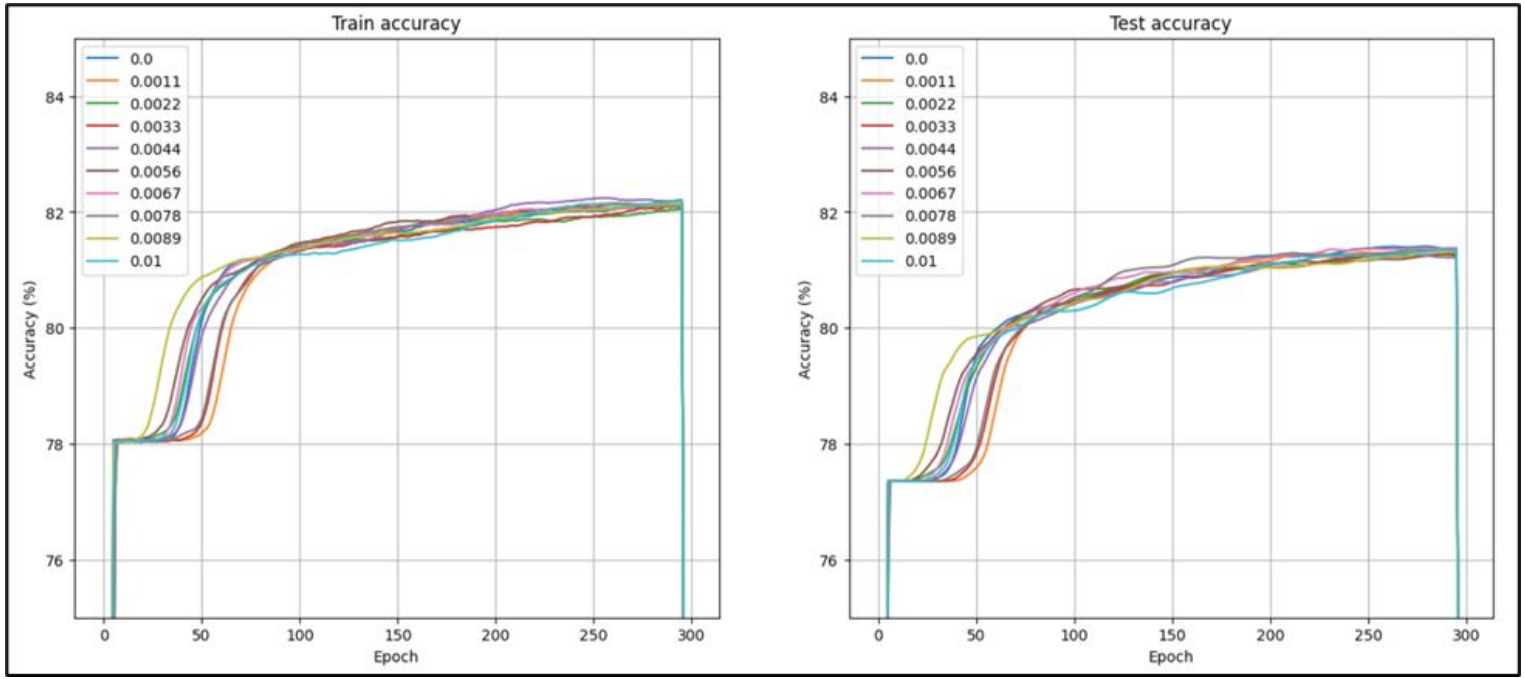
Experiment 6 - Epoch Resizing



Losses start to converge at 300 epochs and model accuracies stabilize around 300 epochs

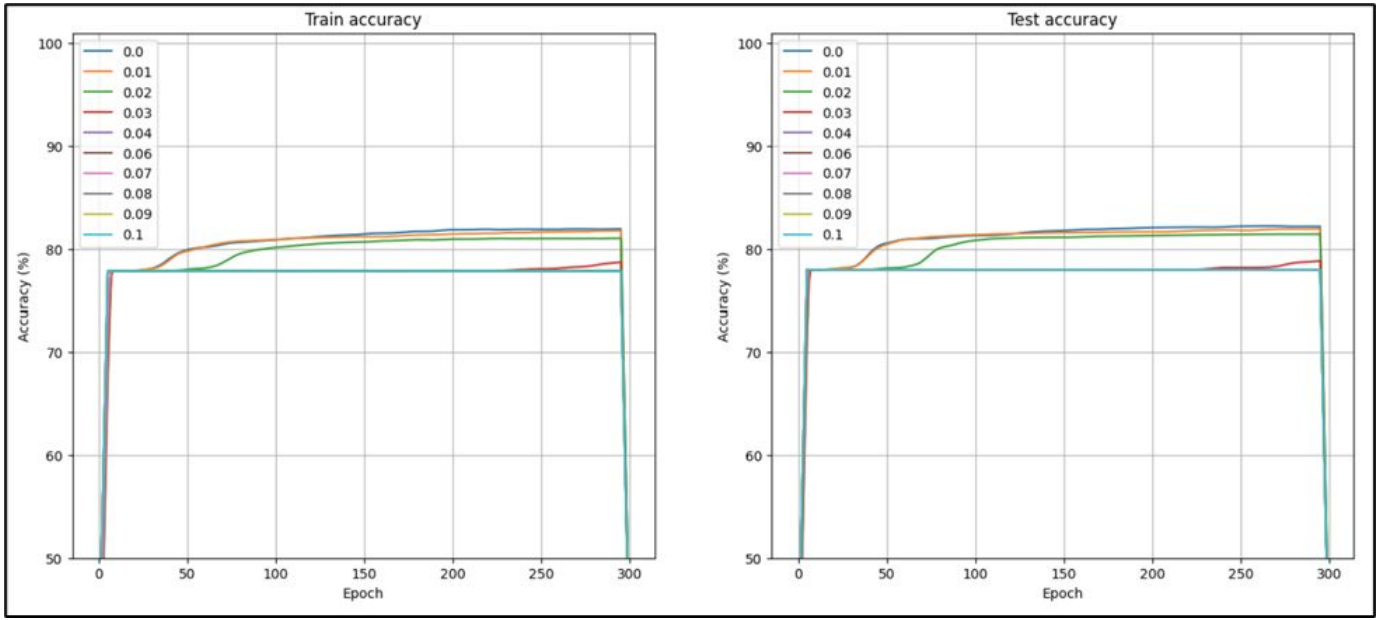
Subsequent experiments will use 300 epochs instead of 1000

Experiment 7 - L1 Regularization



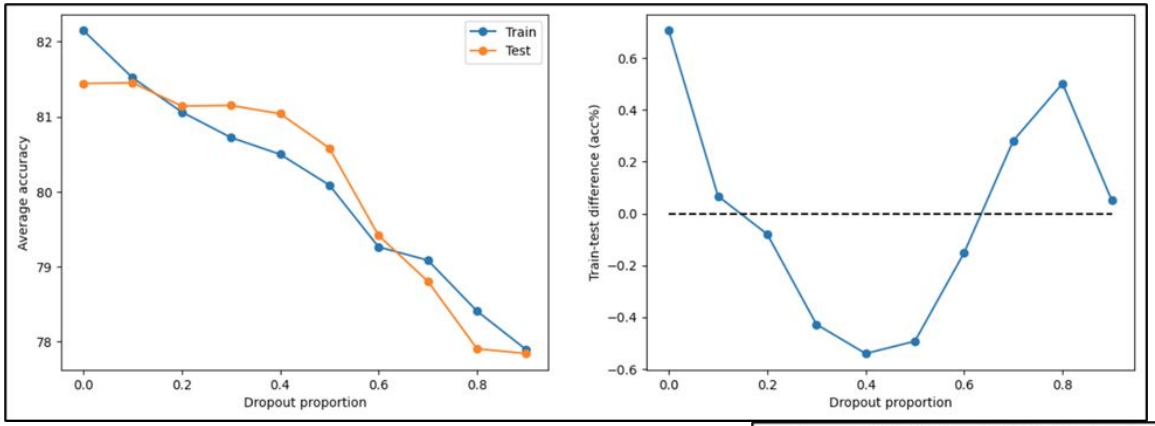
No meaningful change to accuracy so will not utilize

Experiment 7 - L2 Regularization



No added benefit to accuracy so will not utilize in model

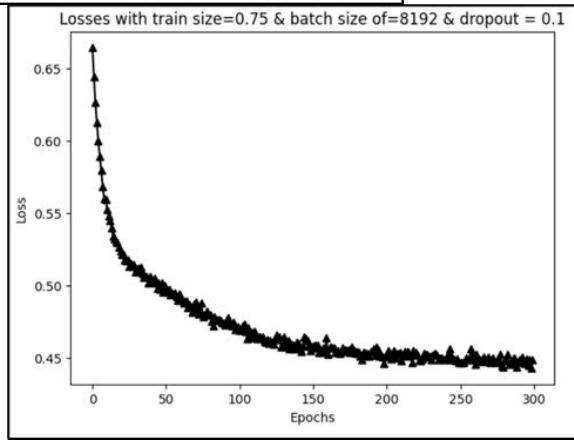
Experiment 7 - Dropout Regularization

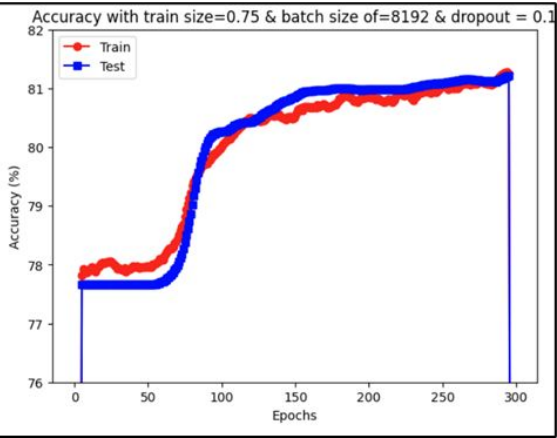


Any amount of dropout decreases accuracy

Any amount of dropout decreases accuracy. A dropout rate of 0.1 seems to create a slightly overfitted model

Ultimately, did not use any dropout in subsequent experiments





Experiment 8 - Breadth and Depth

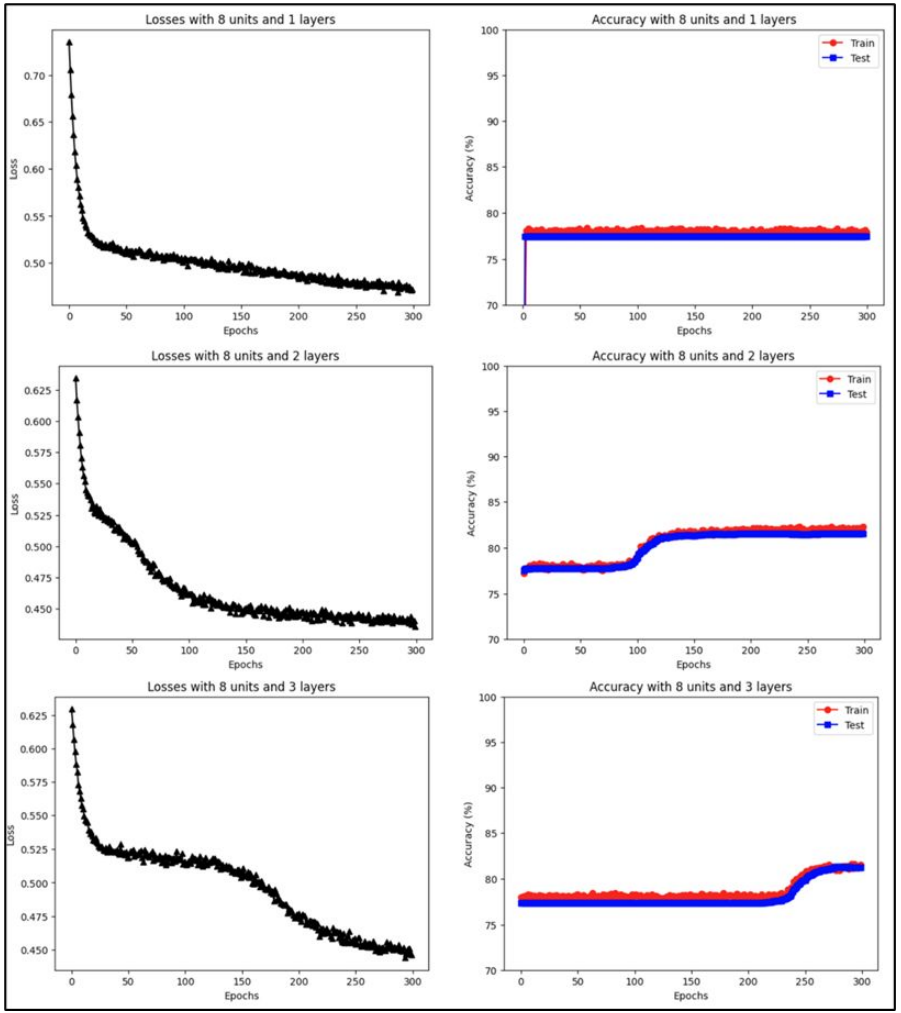
- 15 experiments ran
- Number of units in each layer were set to values of 8, 32, 64, 128
- Number of hidden layers (between input and output layers) was set to 1, 2, and 3

Starting Architecture in Experiment

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                [-1, 1, 8]             192
              ReLU-2                [-1, 1, 8]               0
            Linear-3                [-1, 1, 8]              72
              ReLU-4                [-1, 1, 8]               0
            Linear-5                [-1, 1, 1]               9
           Sigmoid-6                [-1, 1, 1]               0
================================================================
Total params: 273
Trainable params: 273
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.00
Estimated Total Size (MB): 0.00
----------------------------------------------------------------
```

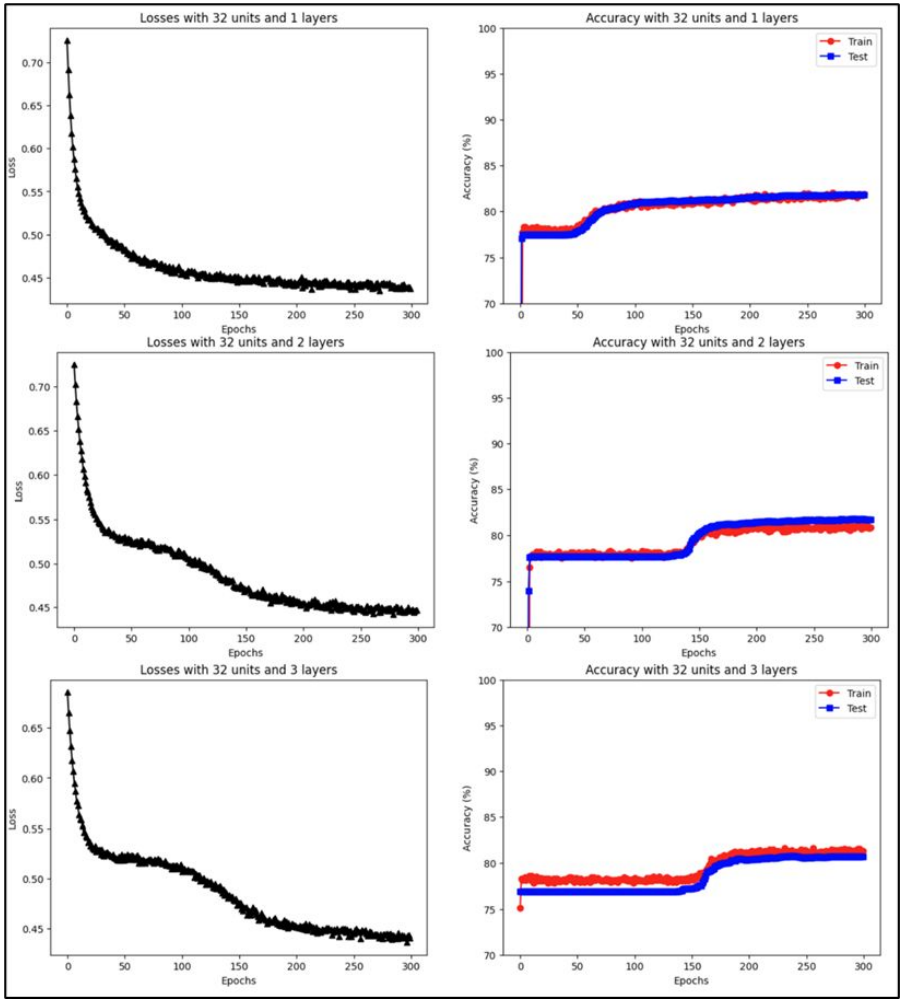
Ending Architecture in Experiment

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1               [-1, 1, 128]           3,072
              ReLU-2               [-1, 1, 128]               0
            Linear-3               [-1, 1, 128]          16,512
              ReLU-4               [-1, 1, 128]               0
            Linear-5               [-1, 1, 128]          16,512
              ReLU-6               [-1, 1, 128]               0
            Linear-7               [-1, 1, 128]          16,512
              ReLU-8               [-1, 1, 128]               0
            Linear-9                 [-1, 1, 1]             129
          Sigmoid-10                 [-1, 1, 1]               0
================================================================
Total params: 52,737
Trainable params: 52,737
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.01
Params size (MB): 0.20
Estimated Total Size (MB): 0.21
----------------------------------------------------------------
```



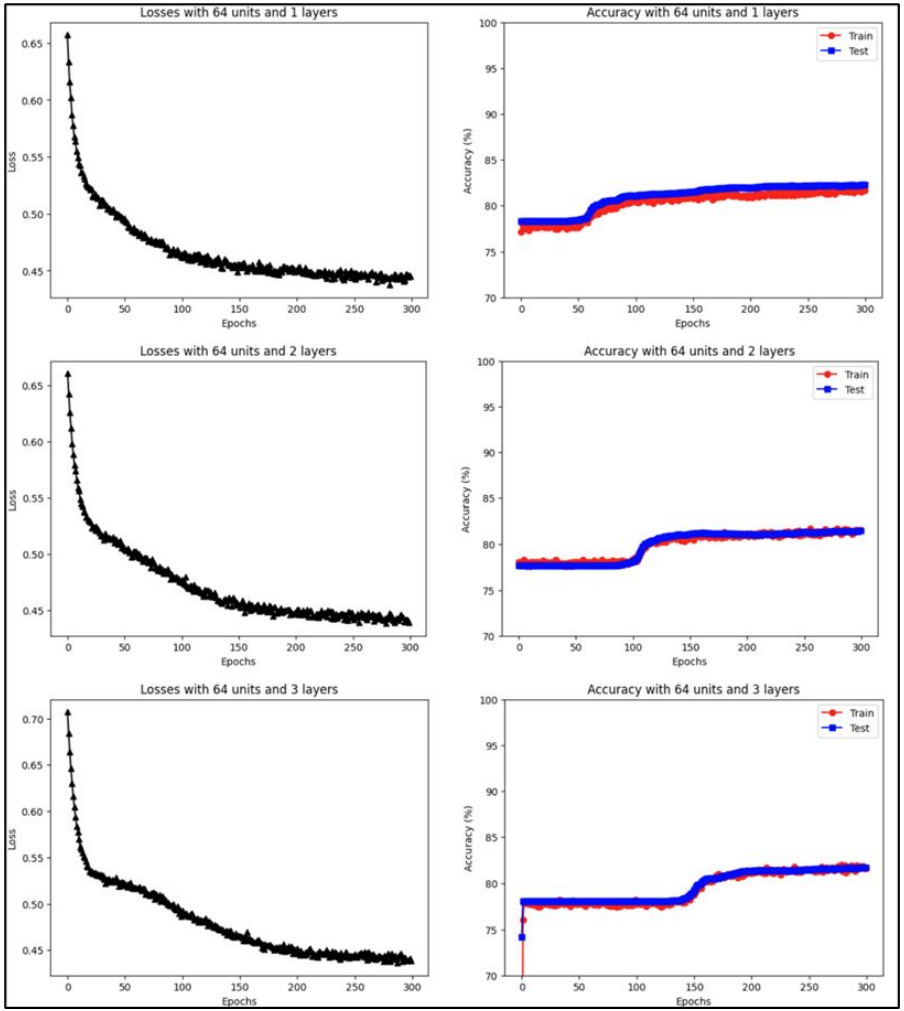
Breadth and Depth Cont'd

- 8 units and 1 layer showed fit issues since testing accuracy plateaued
  - May not be complex enough
- 8 units and 2 layers showed promise with good loss curve and tracking accuracies
- 8 units and 3 layers had testing accuracy trail



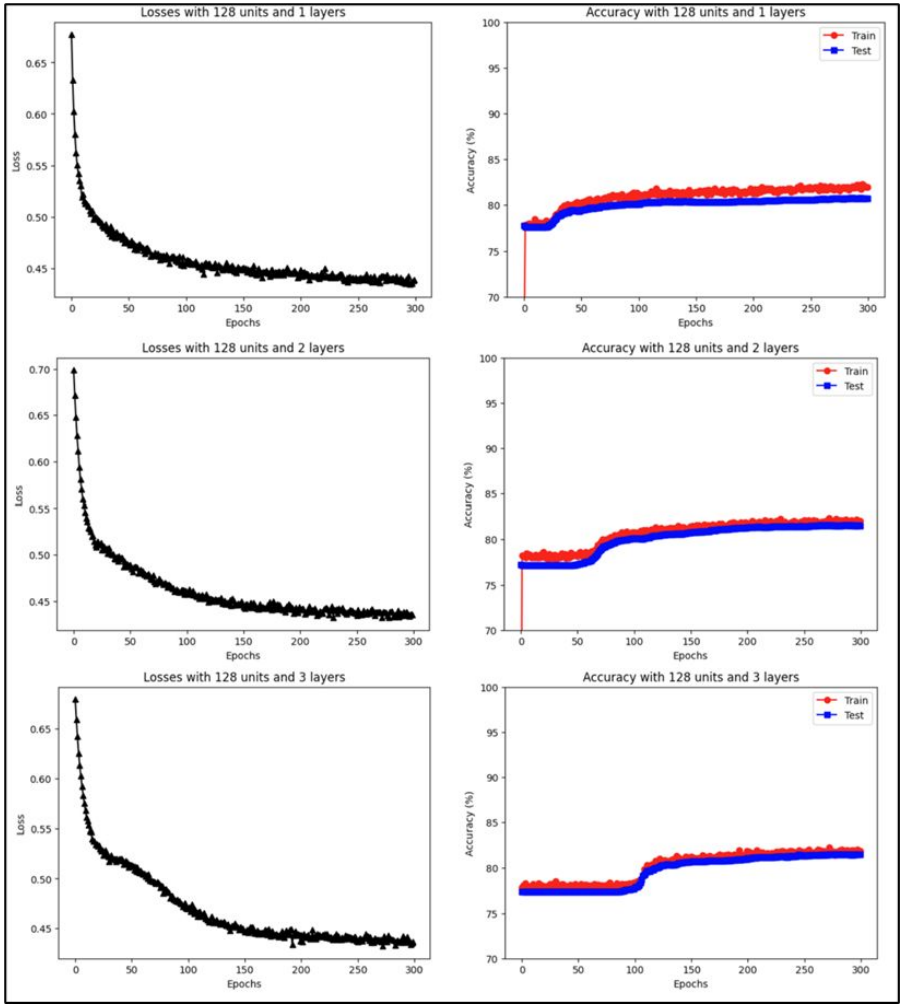
Breadth and Depth Cont'd

- 32 units and 1 layer showed promise as has good loss curve and tracking accuracies
- 32 units and 2 layers showed some overfitting
- 32 units and 3 layers showed poor generalization



Breadth and Depth Cont'd

- 64 units and 1 layer showed slight overfitting
- 64 units and 2 layers showed promise with good loss curve and some accuracy tracking
- 64 units and 3 layers showed good loss curve with excellent accuracy tracking
  - **Ultimately chose this as architecture for rest of experiments**



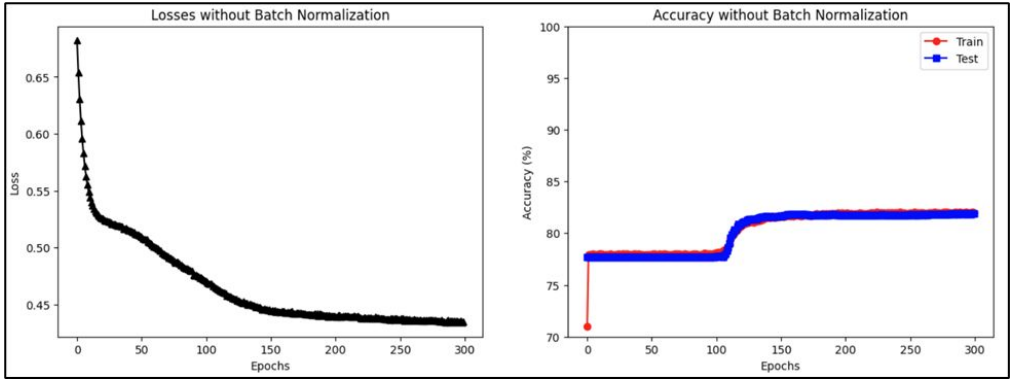
Breadth and Depth Cont'd

- All models showed slightly poorer generalization than experiments using 64 units

Quick Recap:
Architecture Being Used Now

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                [-1, 1, 64]           1,536
              ReLU-2                [-1, 1, 64]               0
            Linear-3                [-1, 1, 64]           4,160
              ReLU-4                [-1, 1, 64]               0
            Linear-5                [-1, 1, 64]           4,160
              ReLU-6                [-1, 1, 64]               0
            Linear-7                [-1, 1, 64]           4,160
              ReLU-8                [-1, 1, 64]               0
            Linear-9                 [-1, 1, 1]              65
          Sigmoid-10                 [-1, 1, 1]               0
================================================================
Total params: 14,081
Trainable params: 14,081
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.05
Estimated Total Size (MB): 0.06
----------------------------------------------------------------
```

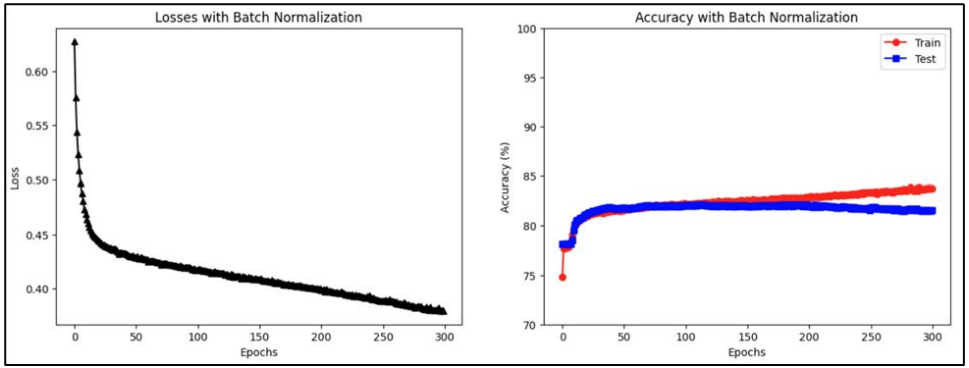
Experiment 9 - Batch Normalization



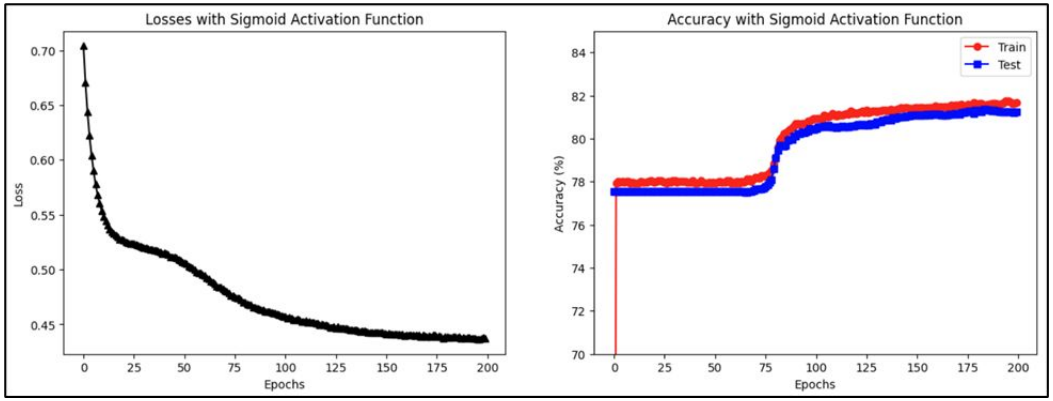
Models without batch normalization have good generalization in this specific case.

Ultimately, chose not to use batch normalization and to reduce epochs to 200

Models with batch normalization show poor generalization at epoch 100 onward



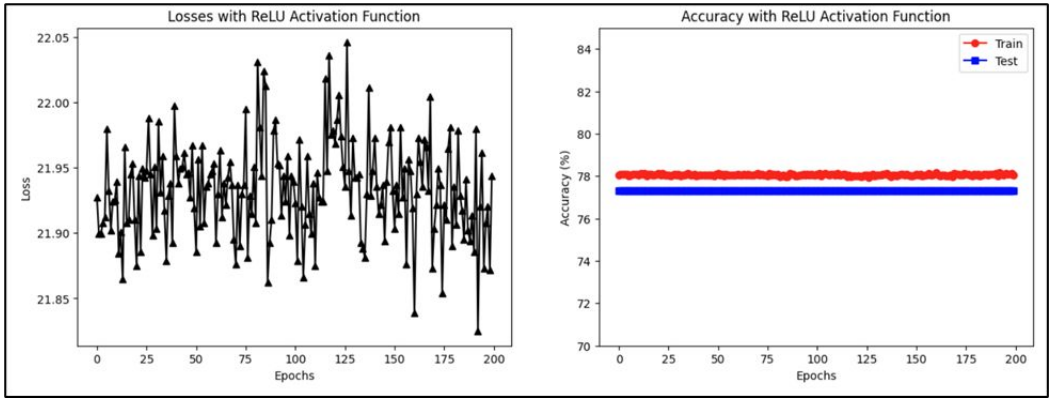
Experiment 10 - Output Layer Activation Functions



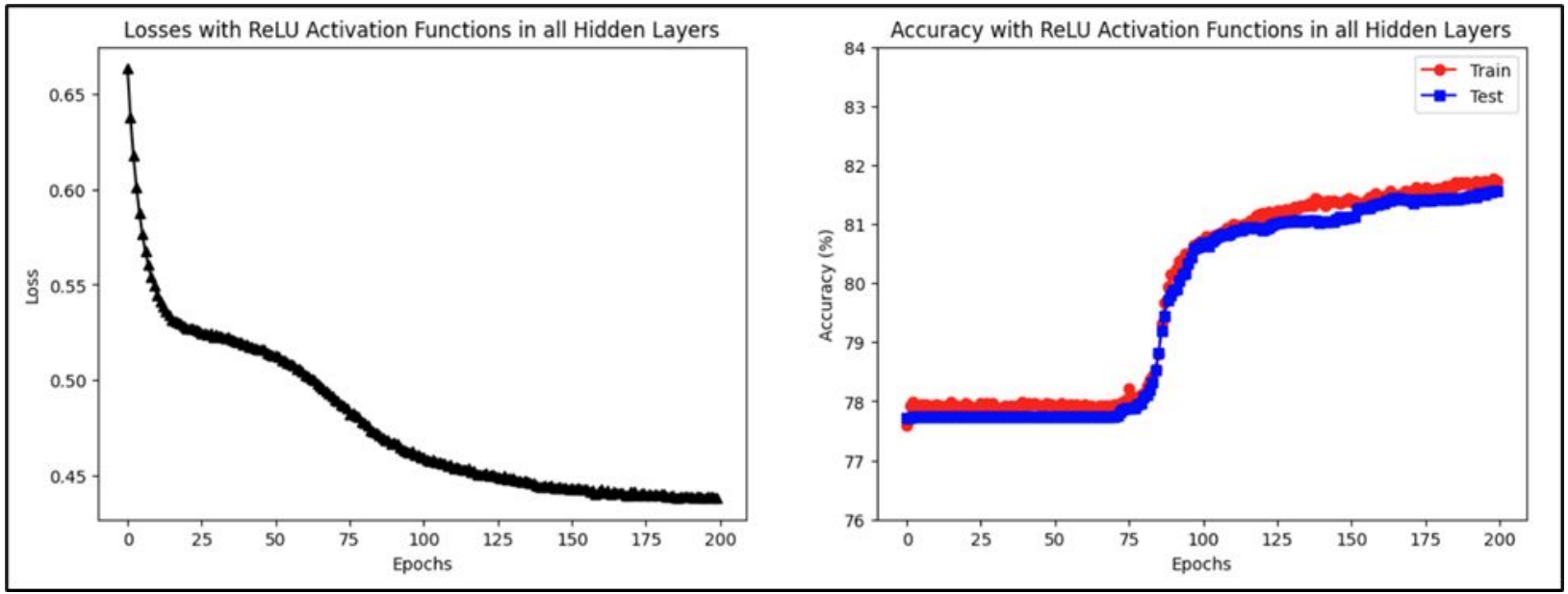
Sigmoid function has been the default activation function and shows good generalization

Ultimately, chose to continue to use Sigmoid

Using ReLU as an activation function in the output layer lead to unstable behaviour judging by the loss curve
The training and test accuracies showed no signs of improvement over epochs



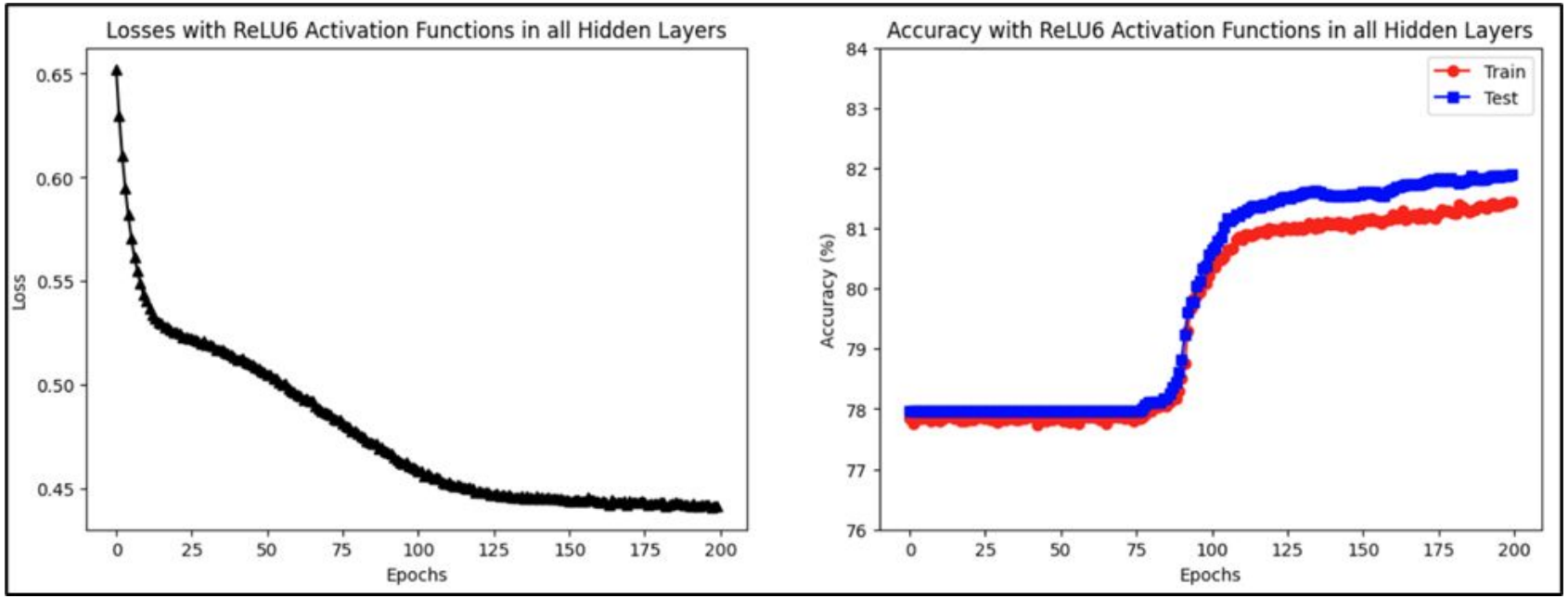
Experiment 11 - Hidden Layer Activation Functions



ReLU has been the default activation function until now

Does a good job of converging loss error as well as generalizing the model

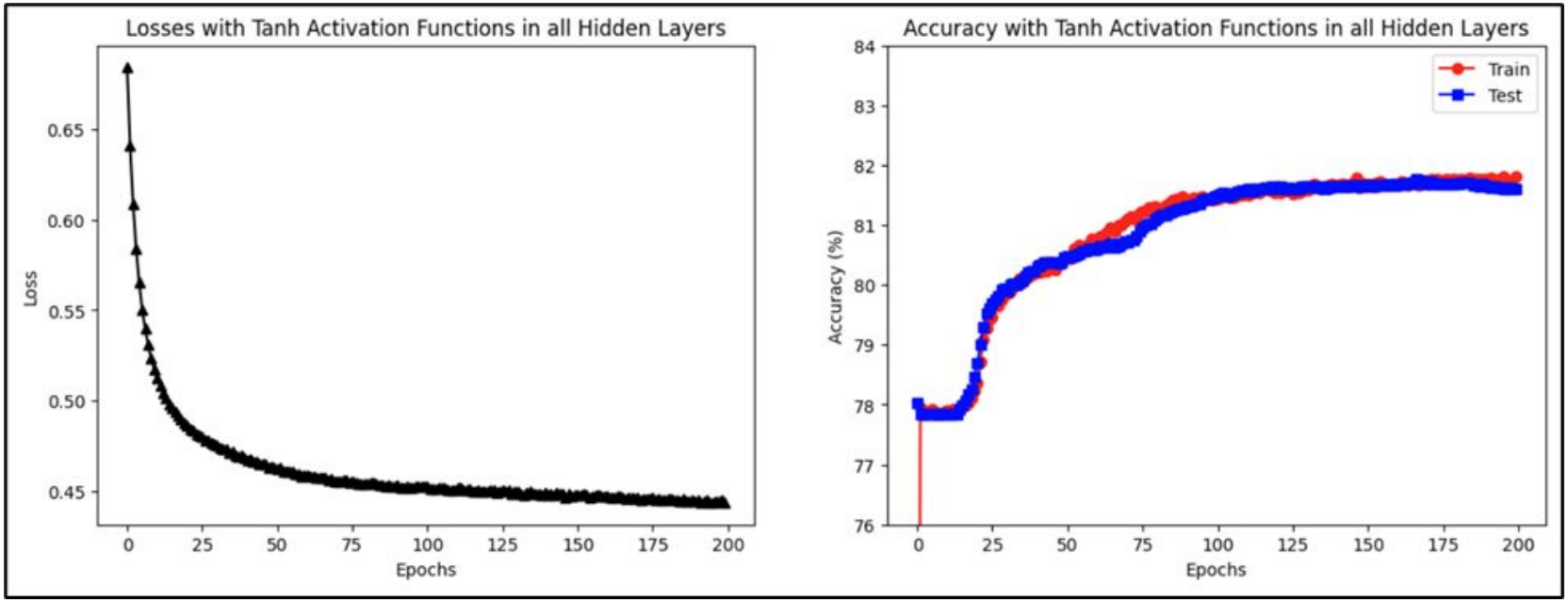
Experiment 11 Cont'd



ReLU6 showed a tendency to create an overfitted model

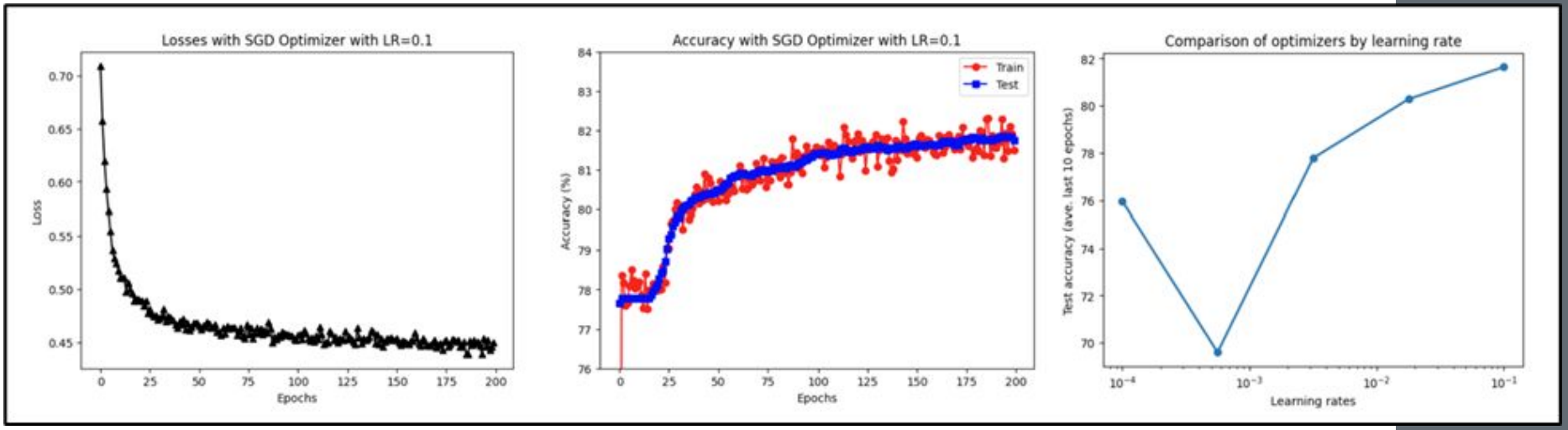
This is seen by the testing accuracy being higher than the training accuracy after epoch 100

Experiment 11 Cont'd



Tanh shows an excellent loss curve and a well fit test to train accuracy
This suggest good generalization and thus Tanh was chosen as the activation for subsequent experiments
However, ReLU would have still been an excellent choice

Experiment 12 - Optimizers

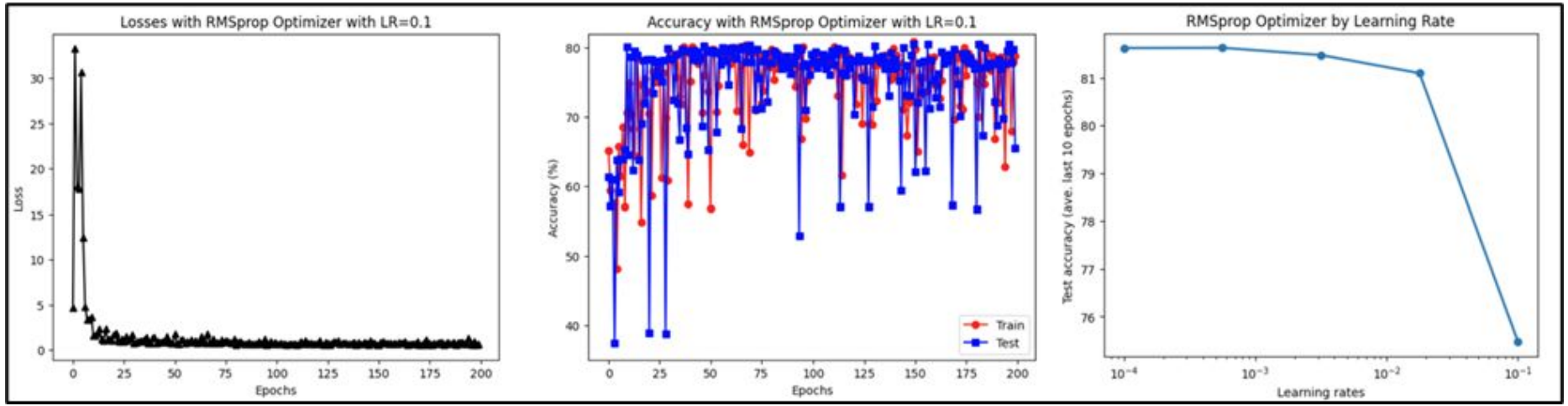


SGD shows good generalization and loss curve

Judging by the test accuracy vs learning rate graph, there is more opportunity to optimize the learning rate

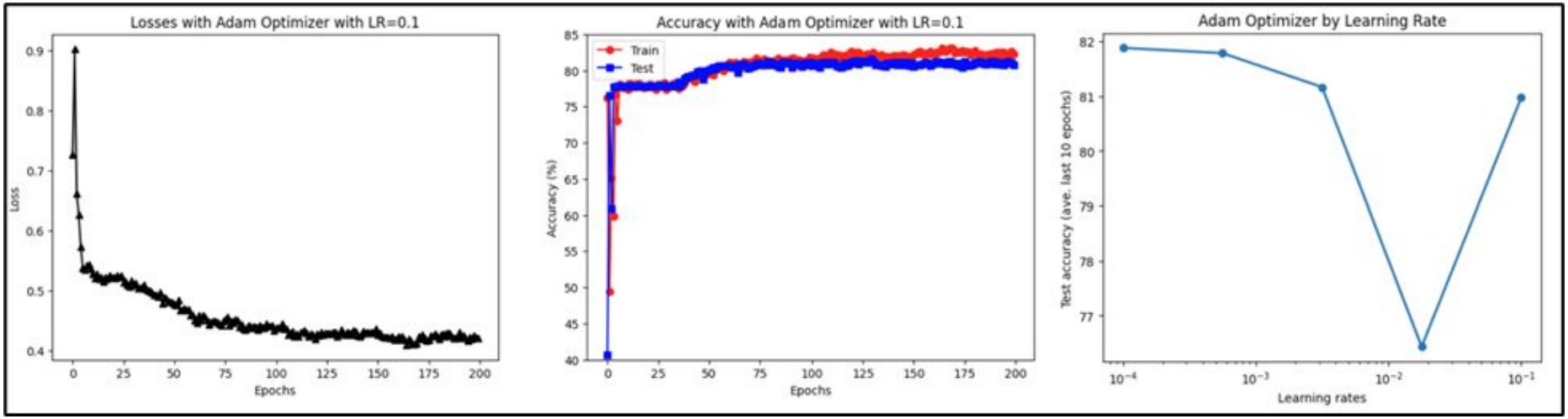
Chose to continue to use SGD over Adam and RMSprop because of the train and testing accuracy tracking

Experiment 12 - Optimizers Cont'd



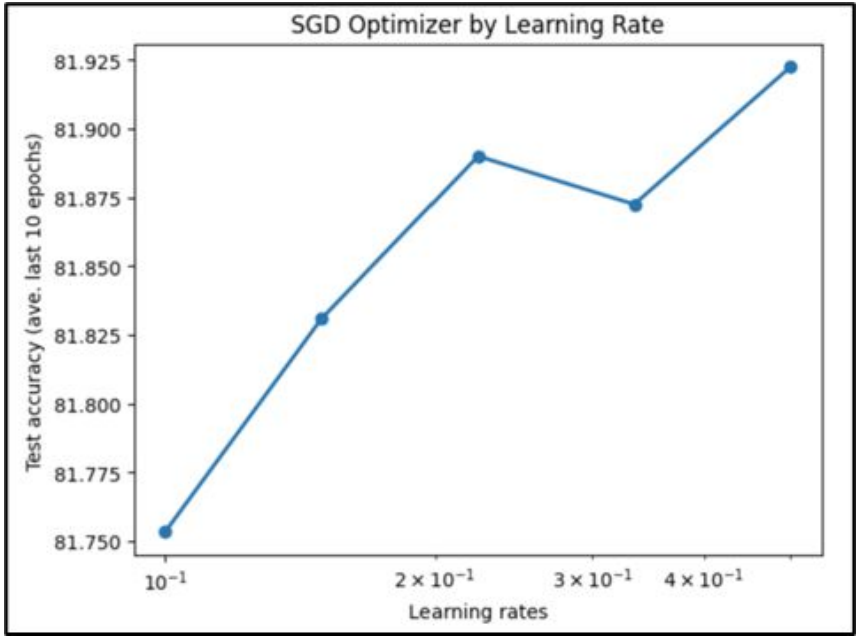
RMSprop shows unstable behaviour with high learning rates
May be OK with much lower learning rates
Ultimately chose not to further investigate this optimizer

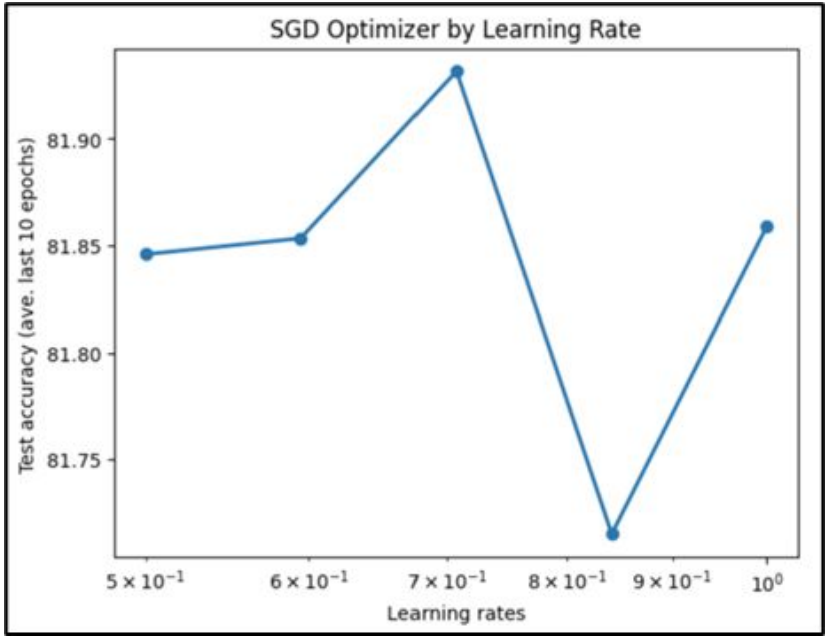
Experiment 12 - Optimizers



Adam shows excellent performance properties with both loss curve and accuracy tracking
However, SGD does a slightly better job so will continue to use SGD

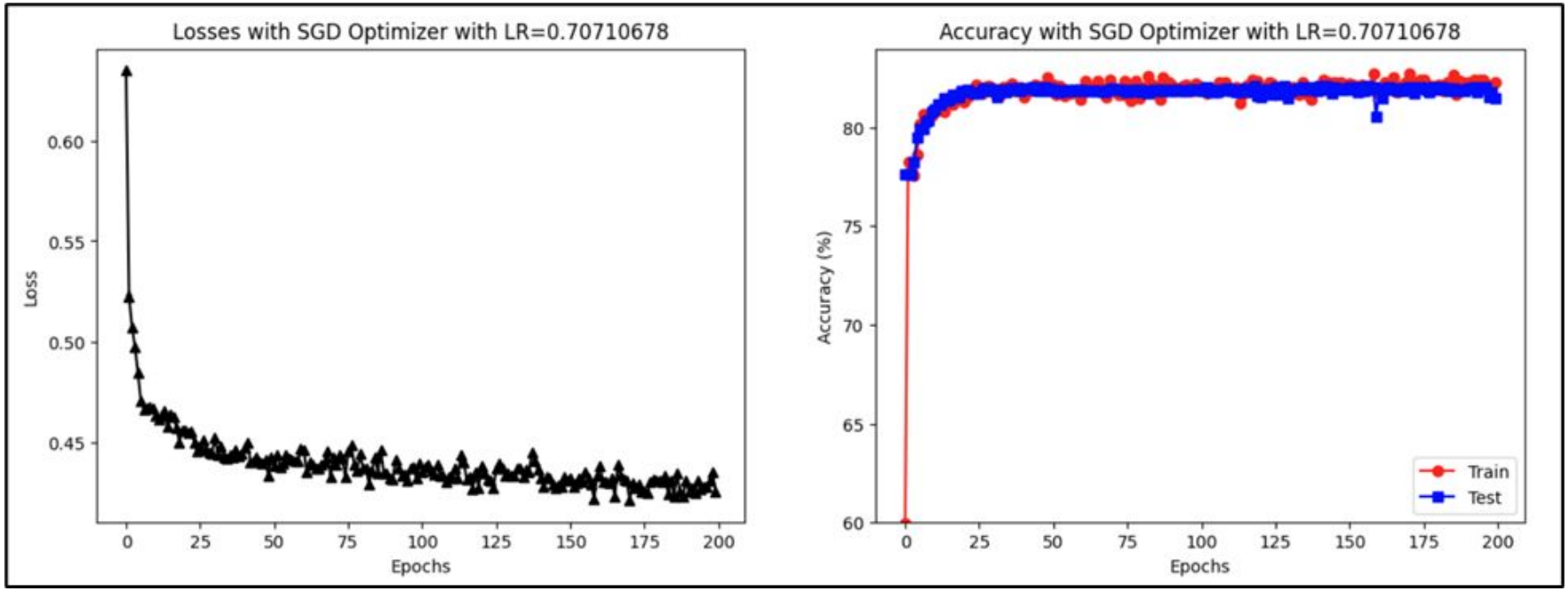
Experiment 13 - Learning Rate Revisited





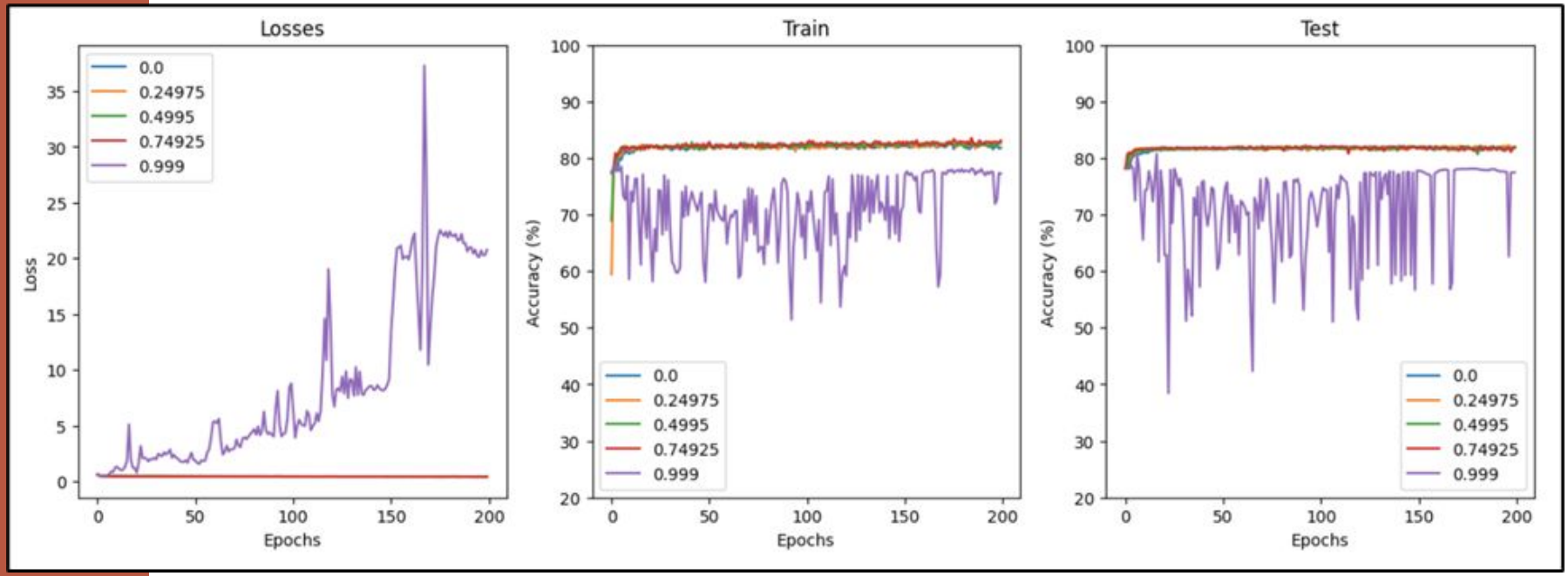
A learning rate of approximately 0.7 provides the optimal accuracy

Experiment 13 - LR Cont'd



Rerunning the algorithm using the optimized learning rate creates a stable model. Unfortunately, it looks like there is less opportunity to improve accuracy. We will try one more experiment

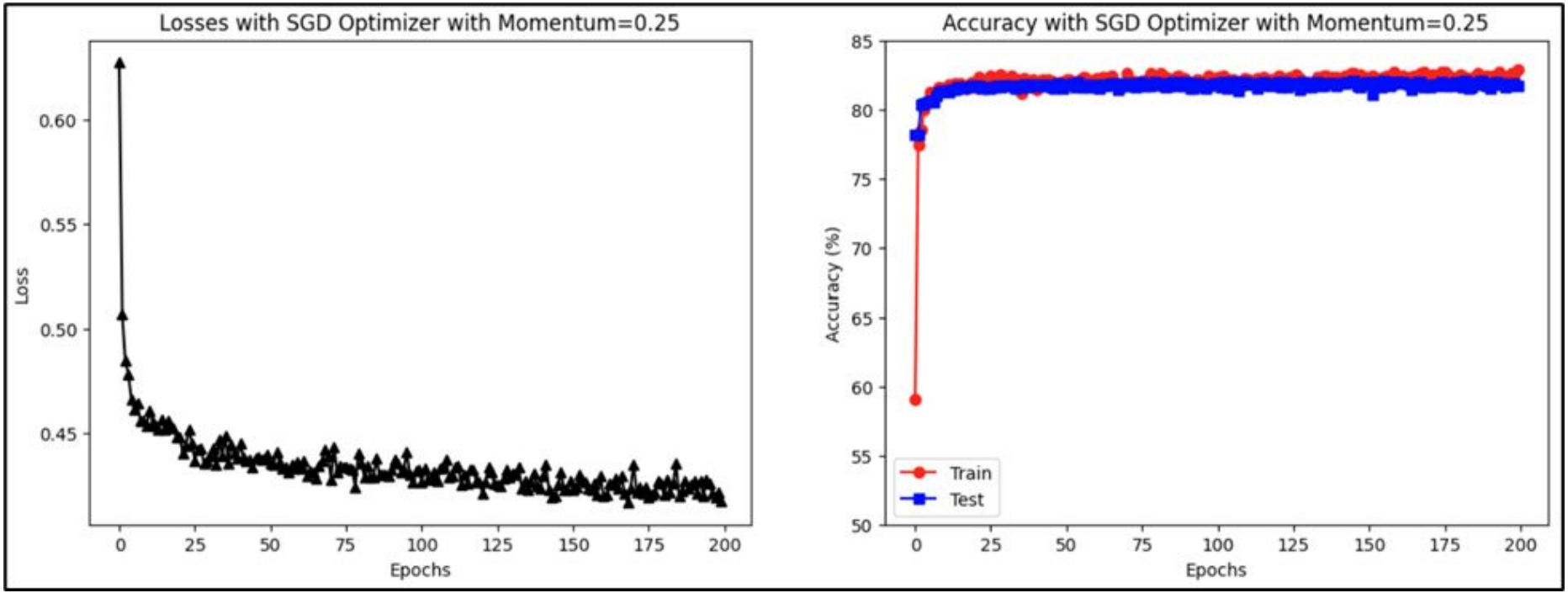
Experiment 14 - Momentum



Higher momentum rates provide poor models with exploding gradients. Lower momentum rates provide closer tracking accuracies

The momentum in the middle start to generalize poorly

Experiment 14 - Momentum Cont'd



This may provide the most optimized ANN model.
There is not much opportunity to increase the accuracy further.

Conclusions

- After about 13 experiments, the model was able to:
  - Become more generalized
  - Faster to train
  - Slightly more accurate
- Some experiments decreased accuracy significantly but ultimately most experiments provided marginal benefits
- The actions and experiments that provided significant improvements were:
  - Data normalization
  - Increasing complexity through breadth and depth
  - Optimizing learning rate

Challenges

- Properly calculating accuracy
  - Seemed to consistently be measuring 77.88% until fixing accuracy calculations
- Understanding when a model is overfit or underfit vs just right.
  - Ultimately chose to evaluate models by making sure testing accuracy tracked well with training data
  - This makes most sense to me since the test data is unseen by the model and the testing size was relatively large
    - If the model did not have a good fit or not generalized well, the model's testing accuracy should deviate significantly from training accuracy
- Training time - had to consistently reduce epoch sizes to speed up experimentation
  - This holds more error which reduces accuracy

Future Work

- Feature selection using methods like PCA and Chi-squared analysis to reduce dimensionality of dataset
- Compare my custom ANN performance against other popular machine learning algorithms
- Measure other performance indicators to better understand the model
    - Confusion matrix
    - ROC and AUC
    - Precision, and Recall
    - etc.

Bibliography

Begüm , Ç., & Ünal, D. (2019). Comparison of Data Mining Classification Algorithms Determining the Default Risk. *Scientific Programming*, 5.