# Assignment 6 - Neural Networks

**Executive Summary:**

Our assignment was to evaluate machine learning technologies for image recognition using neural networks as a benchmark study for an institution. The purpose of the study is to explore alternative network architectures, activation functions, optimization methods, and hyperparameter settings against classification performance accuracy and processing time. Given the complexity and size of the data, it is important from a management perspective that the predictive accuracy of models be weighed against cost constraints, time of model development and implementation, as well as overall performance.

**Research and Design:**

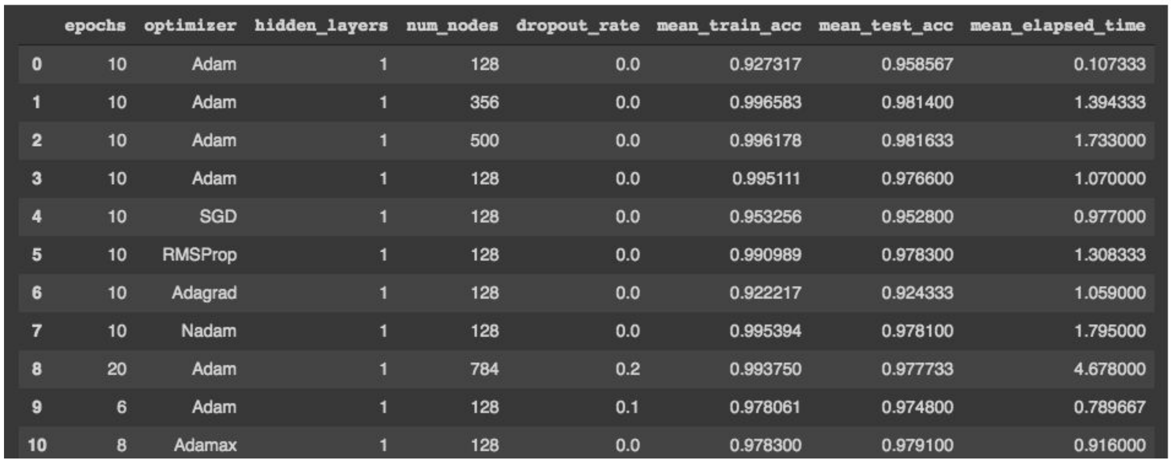
Similar to the prior week, we utilized data from the MNIST database of handwritten digits as a platform for experimenting with neural networks and character recognition. This data includes 70,000 observations (images of numbers) and 784 pixels (features). We split the data into a 60,000 train and 10,000 test sets. A number of models were built using neural networks and classification performance accuracy of the model (ability of model to correctly predict the correct handwritten digit (0-9) along with processing times of the models were recorded for both the training and test set to examine how changing these selection criteria's impacted model scores.

**Technical Overview:**

We used Python's Keras API within the TensorFlow package for implementing neural networks on the MNIST data set. It's important to recognize upfront that pixels can take a range of values from 0 to 255. We applied scaling on the features to bring all features to the same level of magnitude so all variables are uniformly weighted in the model. The model can then be built and compiled according to our specifications using training and test set accuracy score and processing times as the basis for model comparison. We assessed the training accuracy scores relative to the test accuracy scores to assess for a model overfitting or underfitting. For our analysis, we experimented with the number of layers and number of nodes within each layer used when building the model architecture. In addition to the architecture changes, we explored the number of epochs run, dropout rates, type of optimizer and learning rate within the optimizer as the criteria we felt most likely to impact model scores as a function of both model accuracy and processing time to run the models. Final scores were averaged over the course of 3 runs to account for randomization factor when modeling neural networks.

**Findings:**

As part of the design of the experiment, each of our team members selected few variables and tested its impact on processing time, and train / test accuracy scores. Below are results from 10 models, each model varied in choice of optimizer, number of nodes, epochs, and drop out rate.



| | epochs | optimizer | hidden_layers | num_nodes | dropout_rate | mean_train_acc | mean_test_acc | mean_elapsed_time |
|---|---|---|---|---|---|---|---|---|
| 0 | 10 | Adam | 1 | 128 | 0.0 | 0.927317 | 0.958567 | 0.107333 |
| 1 | 10 | Adam | 1 | 356 | 0.0 | 0.996583 | 0.981400 | 1.394333 |
| 2 | 10 | Adam | 1 | 500 | 0.0 | 0.996178 | 0.981633 | 1.733000 |
| 3 | 10 | Adam | 1 | 128 | 0.0 | 0.995111 | 0.976600 | 1.070000 |
| 4 | 10 | SGD | 1 | 128 | 0.0 | 0.953256 | 0.952800 | 0.977000 |
| 5 | 10 | RMSProp | 1 | 128 | 0.0 | 0.990989 | 0.978300 | 1.308333 |
| 6 | 10 | Adagrad | 1 | 128 | 0.0 | 0.922217 | 0.924333 | 1.059000 |
| 7 | 10 | Nadam | 1 | 128 | 0.0 | 0.995394 | 0.978100 | 1.795000 |
| 8 | 20 | Adam | 1 | 784 | 0.2 | 0.993750 | 0.977733 | 4.678000 |
| 9 | 6 | Adam | 1 | 128 | 0.1 | 0.978061 | 0.974800 | 0.789667 |
| 10 | 8 | Adamax | 1 | 128 | 0.0 | 0.978300 | 0.979100 | 0.916000 |

The best model would be the one that gives us the best ratio between training and test accuracy averaged across 3 runs for each model against the processing time. Models 2 & 3 yielded the best mean test accuracy scores, however, the training accuracy is much higher than the test accuracy which is a potential sign of overfitting. If new data were to be incorporated it could negatively impact future test accuracy, and increase processing time. Model 10 with Adamax optimizer is best model as the training and test accuracy scores are very close and processing time remained low.

**Conclusion and Recommendation:**

The benchmark study provided us with insights on basic neural networks and the impact of changing variables on model accuracy and processing time. The baseline default variables are provide a startinging point when building a model, when adjusted learning rate is too high or too low the model never converged to a good accuracy score. Type of optimizer and associated learning rates selected are important factors as it significantly impacted time the model took to converge. The number of epochs should be experimented with after optimizer selection based on train/test accuracy scores. Increasing nodes and layers helped improve accuracy of model, but negatively impacted processing time. Applying activation and dropout rate values were much quicker alternatives impacting accuracy scores. To summarize, iterating with choice of optimizer and number of epochs provided us a final model. However, optimizing towards accuracy comes at an additional cost and time, and often requires management to identify constraints and requirements when determining the optimal model needed for the dataset. To expand on this study, we should experiment with more activation functions.