COMP551 Assignment 3 Group 16

Rohan Agarwal, Eric Chao, Meilin Lyu

November 2022

1 Abstract

In this paper, we implemented different architectures of Multi-Layer Perceptron on a classification task with the Fashion MNIST benchmark data set. We examine the performance impact of network depth and training epochs, considering training time and classification accuracy. We compare the performance of our feed forward implementation with Tensorflow's Convolutional Neural Network. We found that increasing the number of layers reaches convergence in fewer epochs, but increases training time per epoch. ReLU and Leaky ReLU activations outperform Hyperbolic tangent, L2 Regularization noticeably improves accuracy for a fixed number of epochs, normalized data showed a significantly better accuracy than unnormalized data, and CNN proved to perform significantly better than a MLP architecture.

2 Introduction

We investigated the performance of different types of Multi-Layer Perceptron models (MLP) on the Fashion-MNIST data set. The data set contains 60,000 training examples along with 10,000 testing examples each assigned to one of 10 labels. We computed results using a multitude of models - zero, one and two layer using normalized data. We added L2 regularization along with an attempt at unnormalized data. While ReLU was the common choice in the implementations above, we also tested a two layer model using TanH and Leaky ReLU functions. With this information, we attempted to come up with our own best implementation as well. To compare our results, we created a ConvNet which produced a training accuracy of 94.7% and testing accuracy of 91.66%, suggesting that a CNN is by far the superior choice for image classification tasks. New research using architectural search [4] found that using a fine-tuned DARTS model, which at it's core is a CNN model, returns an accuracy of 96.91%. Another approach using a recurrent neural networks (RNN) by Phong & Ribeiro [3], produced an accuracy of 95.92%.

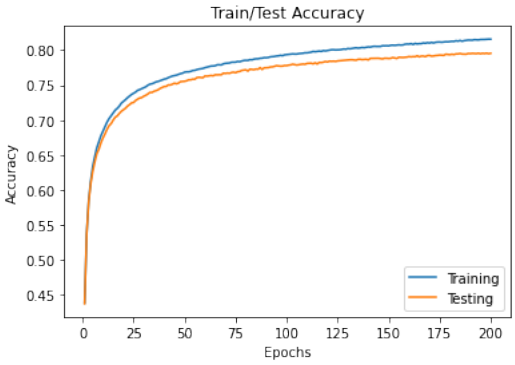
3 Datasets

For our experiments we used the modern Fashion-MNIST data set of greyscale images, each of dimensions 28 by 28, associated with one out of 10 labels for different types of clothing. The set is split into a training set containing 60,000 examples and a testing set with 10,000 examples [5]. Fashion-MNIST is meant to be used as a replacement for the widely-known MNIST data set used for classifying handwritten digits. Though MNIST is a benchmark to validate machine learning algorithms, the data set saw accuracies up to 99.7%. Hence, Fashion-MNIST poses a more realistic challenge for real world image classification scenarios.

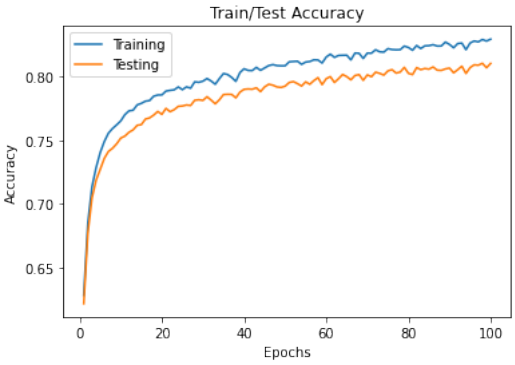
4 Results

4.1 Multi-Layer MLP

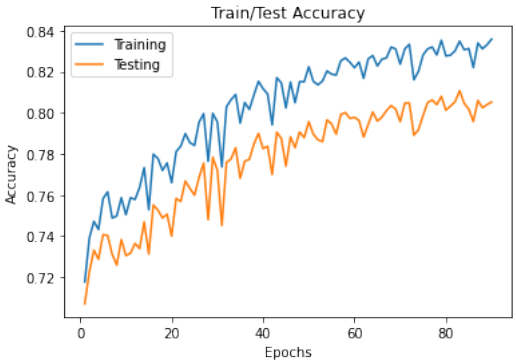
We trained MLP models with 0, 1 and 2 hidden layers, all having 128 units each, using Stochastic Gradient Descent with batch size 500. This size was chosen empirically to get the best tradeoff



((a)) Accuracy over epochs, 0 layers



((b)) Accuracy over epochs, 1 layer



((c)) Accuracy over epochs, 2 layers

between training time and numerical stability with a learning rate of 0.001. Bigger batch sizes resulted in overflow in the forward pass, while smaller sizes were too slow to reasonably wait for, given our development environment on Google's Colaboratory.

Our first experiment showed that increasing the number of hidden layers impacts classification accuracy favorably.

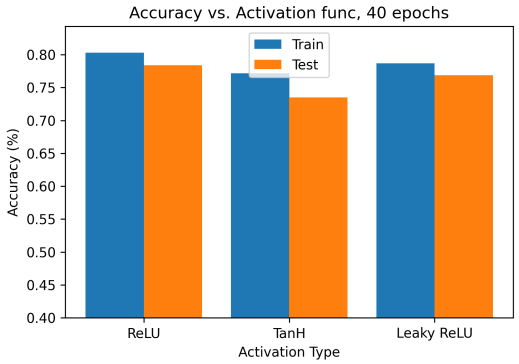
Hidden layers	Epochs	Seconds/epoch	Train %	Test %
0	200	3.92	81.56	79.56
1	100	32.49	82.96	81.06
2	90	53.04	83.94	81.23

The results are consistent with the idea that each layer in the feed-forward MLP adds its own level of non-linearity that could not be produced with one single layer, and adding more layers gives a network more "capacity" to extract abstract features from data that do not linearly map to the output, such as image data. However, by increasing the depth we also increase sequential computation since a layer waits for the previous one. This translated into much higher training times with deeper models. We also note that accuracy score increases more erratically over epochs as we increase the layers, which indicates a greater gradient accumulating over layers, making bigger updates to the weights at a time.

4.2 Activation Types

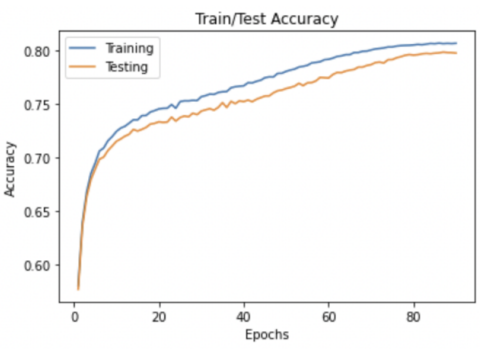
For the next experiment we explored the performance of activation functions among the popular Rectified Linear Unit (ReLU), Hyperbolic Tangent (TanH) and Leaky ReLU for 40 epochs each and a 0.01 learning rate. ReLU and its Leaky variant performed similarily, the former achieving 78.38% and 76.9% the latter. TanH showed slower convergence and reached a lower accuracy of 73.49%. This result is expected since it is a saturating activation function (i.e. it squeezes all inputs in a range, from -1 to 1). Saturation restricts the capacity of the neural network to update its weights [1]. The

gradient produced is smaller, hence the accuracy over epochs curve is smoother, but it also convergence more slowly and exhibits poorer generalization than non-saturating activations. Additional figures are included in the Appendix.

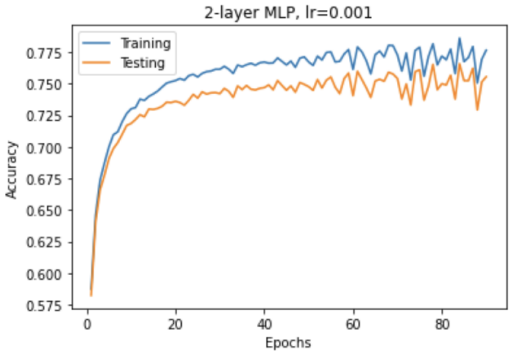


4.3 L2 Regularization

L2-regularization in all three layers setting lambda = 0.05, training for 90 epochs with a 0.001 learning rate yields a testing accuracy of 0.7974. The testing accuracy is better than the previous 2-layer MLP model without regularization, which had a testing accuracy of 0.7657. We chose our lambda value based on the testing accuracy. Using 5 epochs, a choice of lambda = 0.1 with the same settings yields an accuracy of 0.6924; a choice of lambda = 0.3 with the same settings yields an accuracy of 0.6702; and a choice of lambda = 0.05 with the same settings yields an accuracy of 0.6954.



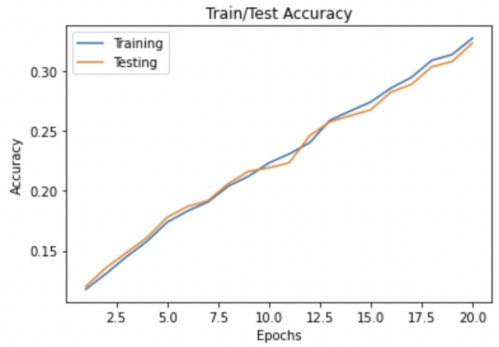
(a) With L2 Regularization



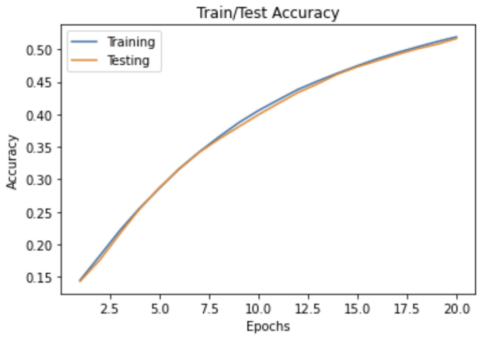
(b) No Regularization

4.4 Training with Unnormalized Data

Unnormalized data has values in range (0, 255), thus to avoid numerical issues during training, we used TanH activations and used Kaiming initialization for the weights, which is a zero-centered Gaussian with standard deviation of $\sqrt[2]{2/n}$ [2] where n is the number of input units to the layer.



(a) Unnormalized Data



(b) Normalized Data

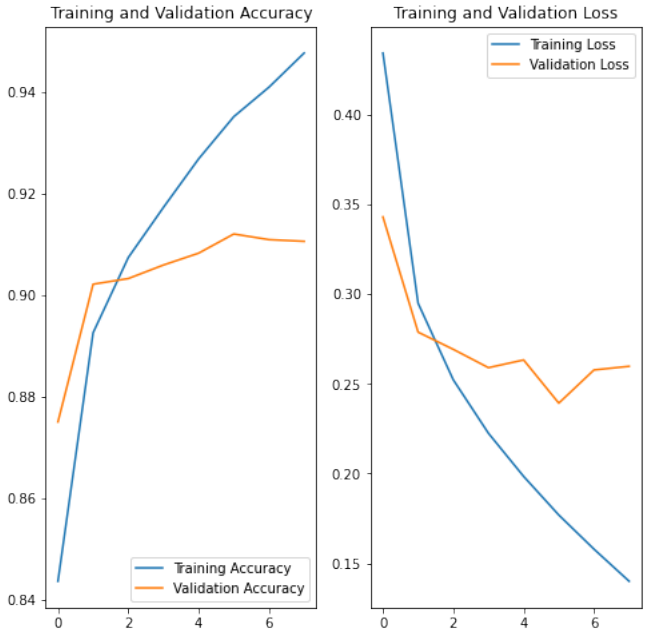
As expected, using Tanh activation yields a low accuracy of 0.3257. Nevertheless, we can observe the significant difference in accuracy between using normalized data (b) and unnormalized data (a).

4.5 ConvNet

A convolutional neural network was formed using tensorflow and sklearn. Intially, a model was created using 32 filters, size of 3 and 16 filters, size of 2 in each respective layer. This was an arbitrary choice of parameters which produced a training accuracy of 93%. In order to produce the most accurate prediction, the grid search function from sklearn was used to find the best parameters for the model. The search grid was set up as follows:

1. Layer 1 Filter Options: [8, 16, 32] Sizes: [2, 3]
2. Layer 2 Filter Options: [16, 32] Sizes: [2, 3]

After fitting the 24 different models, the model with 32 filters of size 2 and 32 filters with size 3 produced the best accuracy. After re-fitting the model with these new parameters it returned a training accuracy of 94.7%. The training and validation loss and accuracy for the same has been graphed below:



It is clear that a convolution neural network is the superior choice when it comes to classifying images. Our simple CNN returned a testing accuracy of almost 91.66% when the best accuracy we received from an optimized MLP was $< 85\%$.

4.6 MLP Architecture Grid Search

To find the best MLP architecture we utilised grid search from sklearn to explore different configurations. Our model was able to choose from 3 different learning rates $(0.01, 0.001, 0.0001)$ and 3 different layer sizes with varying number of hidden inputs, totaling in 30 different combinations. At the end, something quite similar to our 2 layer model was returned. It was found that a 2 layer model with 128 hidden units each with a learning rate of 0.0001 returned the best training accuracy of 85.3% amongst all. To see all possible results you may refer to our Colab file. Coincidentally, it also returned a testing accuracy of 85.3%.

5 Discussion and Conclusion

Our experiments compared the performances of different neural network architectures and the effect of hyper-parameters choices. For MLP models, we found that models with 2 hidden layers achieved a better testing accuracy using less epochs than models with lower hidden layers. Additionally, different activation functions also have significant impact on model performance. We discovered ReLU activation function gives the best model performance, following closely by its variation leaky ReLU. However, TanH gives a significantly lower accuracy due to the function's saturating nature. We discovered that using L2 regularization on a model with two hidden layers increases the performance by about 3% which is a noticeable effect. Unnormalized data requires a smaller variance in the initialization of layer weights and the accuracy grows slower compared to training with normalized data. Our convolution model gave a remarkable 5% improvement in accuracy. The best MLP model configuration found using grid search gave an accuracy of 85.3%, comparable to our best MLP model which yielded an accuracy of 81.23%. The only difference in hyper-parameter between the two models was in learning rate. The best model had a learning rate of 0.0001, while our previous model used a learning rate of 0.01. Since using 0.001 learning rate gave an accuracy of 76.59%. We showed that the relationship between learning rate and testing accuracy is non-linear. This relationship should be further investigated by future researches.

6 Statement of Contributions

1. Meilin - Experiments using L2 regularization and unnormalized data, write up
2. Eric - Implementation of Stochastic Gradient Descent, experiments 1, 2, contributions to 3 and 6
3. Rohan - ConvNet, best MLP architecture implementation, write up

Bibliography

[1] Anna Bosman and Andries Engelbrecht. "Measuring Saturation in Neural Networks". In: (Dec. 2015), pp. 1424–1430. DOI: 10.1109/SSCI.2015.202.

[2] Kaiming He et al. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. DOI: 10.48550/ARXIV.1502.01852. URL: https://arxiv.org/abs/1502.01852.

[3] Nguyen Huu Phong and Bernardete Ribeiro. "Rethinking Recurrent Neural Networks and Other Improvements for Image Classification". In: (Mar. 2021). URL: https://arxiv.org/pdf/2007.15161v3.pdf.

[4] Muhammad Suhaib Tanveer, Muhammad Umar Karim Khan, and Chong-Min Kyung. "Fine-Tuning DARTS for Image Classification". In: (June 2020). URL: https://arxiv.org/pdf/2006.09042v1.pdf.

[5] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: (Aug. 2017). URL: http://arxiv.org/abs/1708.07747.

7 Appendix

7.1 Experiment 1

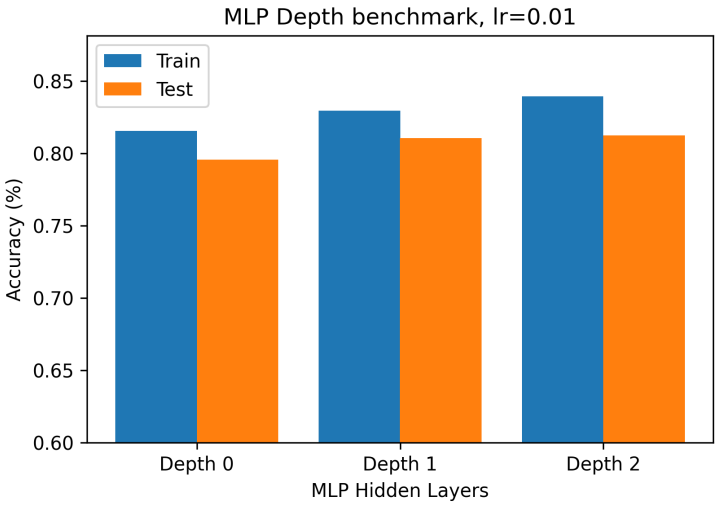
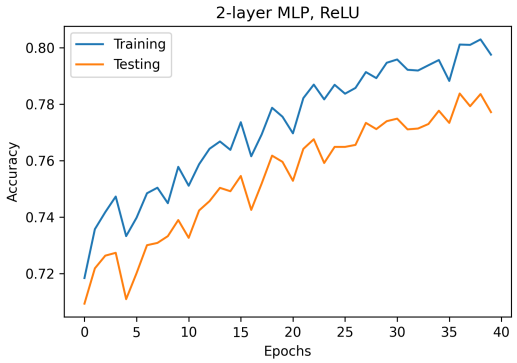
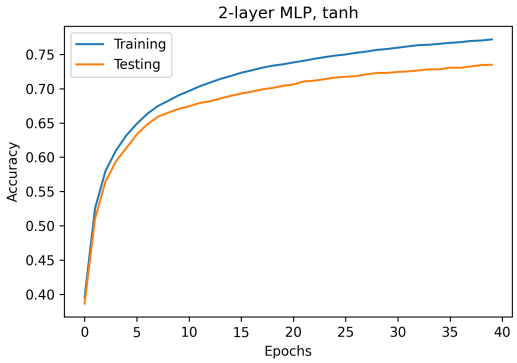


Figure 1: MLP accuracy by hidden layers

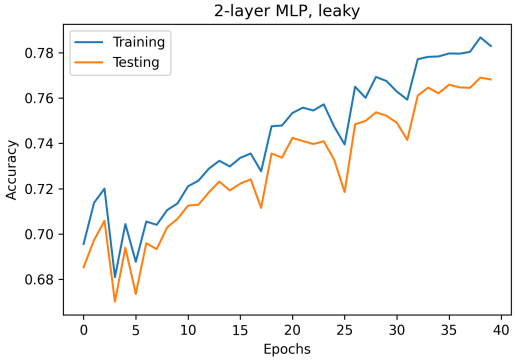
7.2 Experiment 2



((a)) ReLU activation, lr=0.01

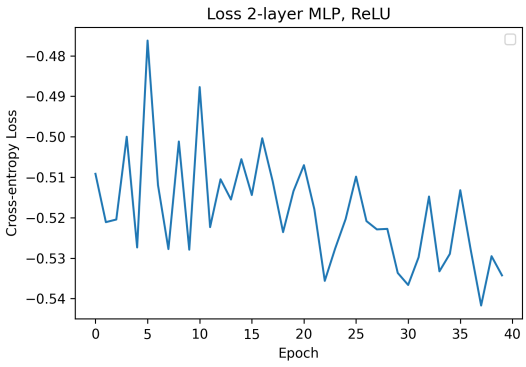


((b)) TanH activation, lr=0.01

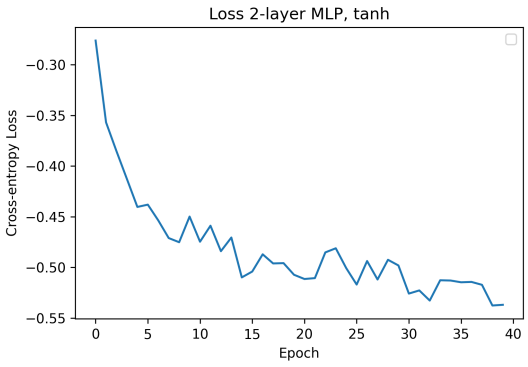


((c)) Leaky ReLU activation, lr=0.01

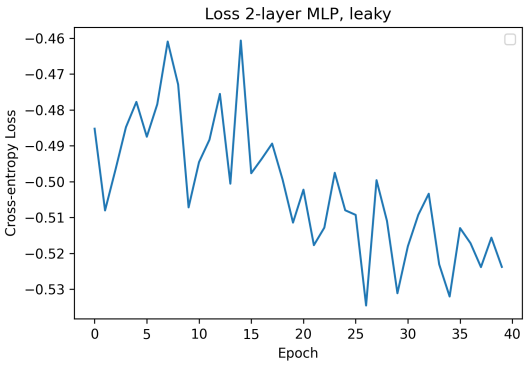
Figure 2: Accuracy over epochs across activation functions



((a)) ReLU activation, lr=0.01



((b)) TanH activation, lr=0.01



((c)) Leaky ReLU activation, lr=0.01

Figure 3: Loss of different activation networks