COMP 551: MiniProject 3

Harms Simon, Madrigal Ariel, and Quinsey Michael

April 4, 2022

Abstract

In this project, we implemented a multi-layer perceptron from scratch and examined its accuracy in classifying the image data from the FASHION-MNIST dataset. We tested our model's accuracy with various different hyperparameters, activation functions, and regularization techniques.

We found that our model performed best when using 1 hidden layer with 300 nodes and a learning rate of 0.1. When compared to a convolutional neural network implemented using PyTorch, our model's performance was better, with an accuracy of 0.8916 compared to the CNN's accuracy of 0.76.

1 Introduction

In this project, we examined the accuracy of a from-scratch implementation of a multi-layer perceptron (MLP) model. We experimented with various aspects of this model to try and achieve the best accuracy, including: the number of hidden layers, the number of units in each layer, different activation functions between the layers, and the number of epochs through the training data. We then compared our model's accuracy to the accuracy of a convolution neural network (CNN) implemented using PyTorch.

We tested this model on the FASHION-MNIST dataset, a labelled collection of clothing images commonly used to benchmark neural networks. Because it is a more sophisticated version of the MNIST dataset, it offers basic neural network image classification benchmarking while also providing a more challenging classification task than normal MNIST [1].

Other experiments have been conducted on deep neural networks and the FASHION-MNIST dataset, and it appears that convolutional neural networks usually offer better results than basic MLPs. For example, Tang et al were able to achieve an accuracy of 96.21% on this dataset in their neural network model [2]. This was achieved through a pyramidal neural network (a CNN based on increasing width as depth increases) and various data optimization techniques. This model utilizes a large number of convolutional layers.

Another comparative article with a different dataset is from Medina et al, who found that CNN performed better than MLP (99.4% and 95.7% accuracy) when classifying images of underwater pipelines to detect if they had algae [3].

The results of our experiments showed that our self-made model performed best when using 1 hidden layer with 300 units and a learning rate of 0.1 which achieved an accuracy score of 0.8916.

Comparatively, the PyTorch CNN implementation gave an accuracy score of 0.76, which is a lower result than we expected.

2 Datasets

We used the FASHION-MNIST dataset for benchmarking our MLP implementation. FASHION-MNIST is a collection of PNG images of fashion items from Zalando, each with category labels. It acts a direct and more sophisticated replacement for the original MNIST benchmark dataset [1].

After loading in the training data, we reshaped the 28x28 image into a 784-length array and recorded the raw data before normalizing it the datasets. We then examined the raw and normalized testing and training data to see the effect of our normalization before fitting it to the model.

3 Results

For Task 3.1, we compared the performance of the MLP across different number of hidden layers (Fig 1). First, we created a model with 0 hidden layers and evaluated the performance on the test set across various learning rates (1, 0.1, 0.01, 0.001) (Fig 1A). We observed that a learning rate of 0.1 offered the best performance, giving us an accuracy of 0.84. We repeated this procedure for 1 hidden layer (Fig 1A) and 2 hidden layers. We observed that a learning rate of 0.1 gave us the best performance for the 3 models. Then, we compared the performance across different number of hidden layers (Fig 1D). For this comparison(and all subsequent in this report), we selected the best model based on the performance varying the learning rate. The performance was very similar between the model with 1 hidden layer (0.8797). and 2 hidden layers (0.8853), with a slight improvement for the 2 layer version.

From these results, we observed that the addition of non-linearity had a higher improvement on accuracy than the increase of 1 layer depth. We reasoned that because the 0 hidden layer model could be considered as a softmax regression (no hidden bases), the addition of these hidden bases was crucial for the improvement of learning. These results were in accordance to our expectations (that nonlinearity and higher depth improve performance).

We also evaluated how the model performance changes across epochs. To illustrate this, we showed the performance of the 2-layer model with learning rate 0.1 (Fig 1E). We observed that around 15 epochs were necessary to reach a plateau for the test accuracy.

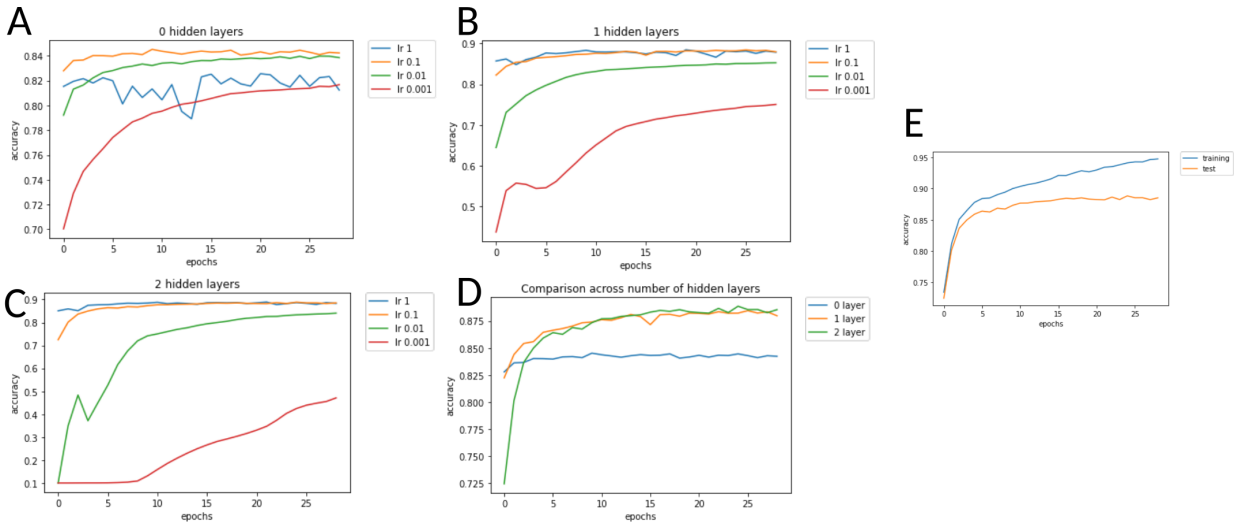

Figure 1: Task 3.1. **A**. Test accuracies of MLP with 0 hidden layers across various learning rates. **B**. Same as A but for 1 hidden layer. **C**. Same as A but for 2 hidden layer. **D**. Comparison across number of hidden layers for the best model across learning rates. **E**. Training and test accuracies of MLP with 2 hidden layers and learning rate of 0.1.

For Task 3.2, we measured the performance of the 2-layer model with the hyperbolic tangent (tanh) and Leaky-ReLU functions (Fig 2). We first selected the best model varying the learning rate for each activation function (Fig 2 A,B).Then, we compared the performance between the 3 activation functions tested (Fig 2 C). We observed that the performance for ReLU (0.8853) and leaky ReLU (8848) was very similar, and that the tanh function performed slightly lower (0.8802) in comparison ReLU and leaky ReLU. Using tanh could lead to vanishing gradient issues, which should be ameliorated with ReLU and fixed with leaky ReLU. Technically, the best activation function should be Leaky ReLU, but in our experiments we observed almost equal performance between ReLU and leaky ReLU. We believe that one of the reasons why we don't have a vanishing gradient issue in this experiment is that the network depth is small.

For Task 3.3, we evaluated the performance of including Dropout in our MLP using various probabilities of keeping the full layer (0.25,0.5,0.75) (Fig 3). As previously, we selected the best model for each probability (Fig 3 A,B). Then, we compared the performance across the defined proportions (Fig 3 C), including a probability

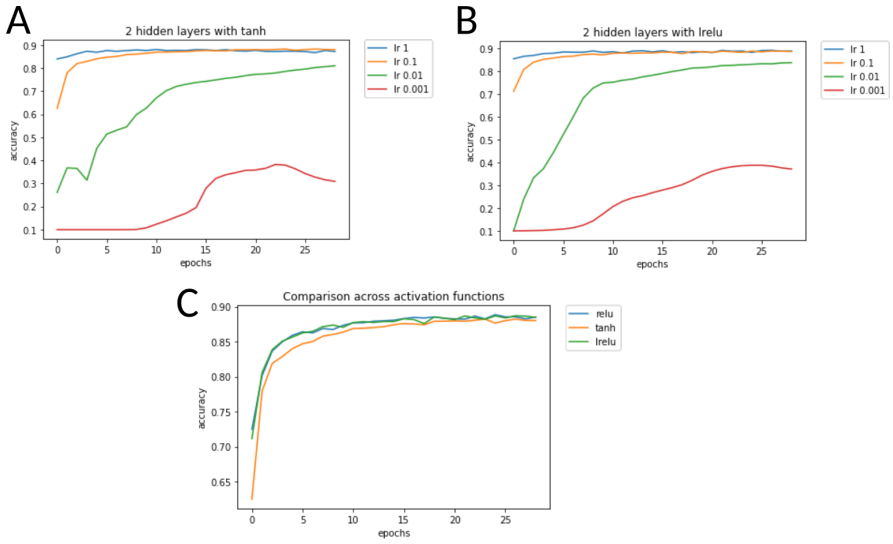

Figure 2: Task 3.2. **A**. Test accuracies of MLP with tanh across various learning rates. **B**. Same as A but with leaky relu. **C**. Comparison across various activation functions for the best model across learning rates.

of 1, which would translate to no dropout. We observed that p of 0.25 was the worst(0.854), while the others were very similar; p of 0.5 was 0.8789 and p of 0.75 was 0.883. Adding regularization to a MLP is generally desirable because it would prevent overfitting. For our experiment, we observed slightly decrease in accuracy performance when including dropout. We believe that the reason for this is that the model we used might not be very complex, and therefore we still observe good performance when using p of 1. One possibility to test this would be to increase the expressiveness of the model (e.g. increase the depth or width) and then compare across various p.

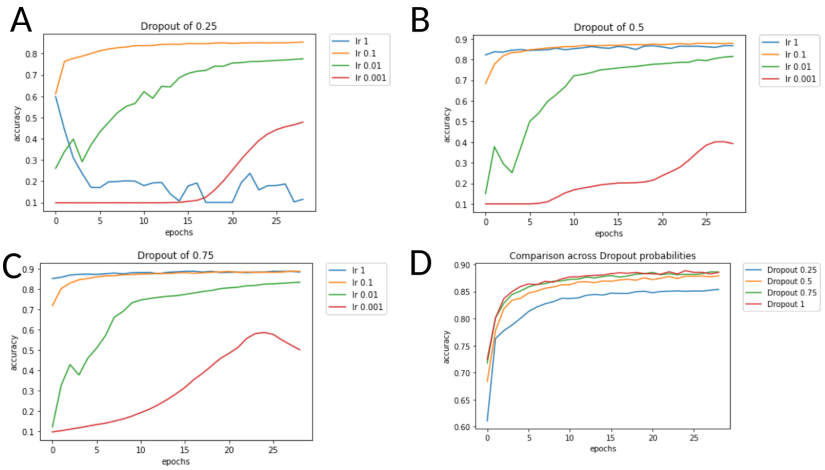

Figure 3: Task 3.3. **A**. Test accuracies of MLP with dropout proportion of 0.25 across various learning rates. **B**. Same as A but with dropout of 0.5. **C**. Same as A but with dropout of 0.75. **D**. Comparison across varying dropout proportion for the best model across learning rates.

For Task 3.4, we trained the 2 hidden layer MLP with unnormalized data (Fig 4). We first selected the best model across various learning rates (Fig 4A). Interestingly, the best learning rate was 0.01, which was different from all our previous experiments. Then, we compared normalized versus unnormalized (Fig 4B) data. We were expecting lower performance for unnormalized data, but surprisingly, we observed very similar performance. We think that the weight and biases in our MLP are adapting to deal with the scaling of the input variables, and that would explain why applying the MLP with unnormalized data still offers good performance.

For Task 3.5, we trained a Convolutional Neural Network (CNN) from the PyTorch package with two hidden convolutional layers and two hidden fully-connected layers on the same dataset as the previous experiments.

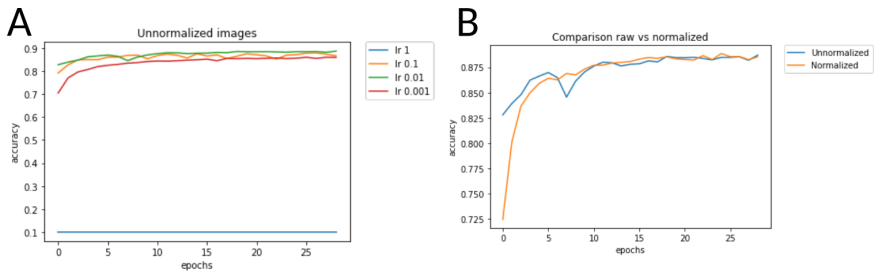

Figure 4: Task 3.4.**A**. Test accuracies of MLP with unnormalized data across various learning rates. **B**. Comparison of MLP performance using unnormalized versus normalized data.

Each fully-connected layer had 128 nodes and each convolutional layer used a 5 by 5 kernel size and a stride 1, which we determined gave the best result. Each layer used ReLU activation functions. The network was trained using a SGD optimizer from torch.optim with a learning rate of 0.01 and the built-in back-propagation of PyTorch's cross entropy loss function. The CNN ended up with an training accuracy of 0.76 and a testing accuracy of 0.77 after 100 epochs of training (Fig 5), which is worse performance than all other MLP experiments gave.

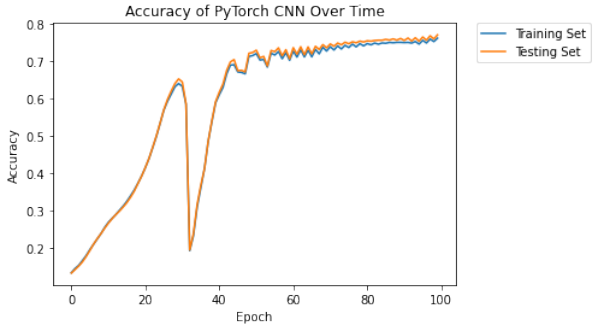

Figure 5: Task 3.5. Shows the training and test accuracy of the CNN over 100 epochs.

For Task 3.6, we varied our MLP architecture (Fig 6). Based on our previous experiments, we chose to kept ReLU as our activation function and a probablity p of 1 of keeping the full layer. We tested a model with: 1 layer of 300 units, 2 layers of 10 and 10 units, 2 layers of 30 and 30 units, 2 layers of 128 and 128 units, 2 layers of 128 and 10 units, 2 layers of 200 units and 2 layers of 300 units. For each of these architectures, we varied the learning rate to obtain the best model. As example, we only show the performance across learning rates of 2 models (Fig 6A,B). The best performance was achieved by the model with 1-layer and 300 units (0.8916), followed by a 2-layer of 300 units each(0.8864) We observed that increasing the width enabled higher performance, although this came with the cost of increase in the computational time during optimization. In theory, increasing the depth (higher than 2) could also enable us to achieve better performance, and should have a similar effect to increasing the width of a 1-layer network.

4 Discussion and Conclusions

In this project we implemented a MLP and evaluated its performance on the Fashion-MNIST dataset. Our best performing model was a 1 layer MLP with 300 units (0.8916). Across our experiments, we observed that a learning rate in the range of 0.1 and 0.01 achieved better performances. To potentially increase the performance, we could test more learning rates within this range. Also, we trained our model using mini batch stochastic gradient descent with momentum. We used batch size of 32 and b (momentum) of 0.9. We kept these hyperparameters constant, but potentially we could optimize them to obtain better results. Other venues of possible improvement in performance could be to increase the depth of the MLP and/or increase width.

The CNN we trained performed worse than the MLP for reasons that are not fully clear. It was expected

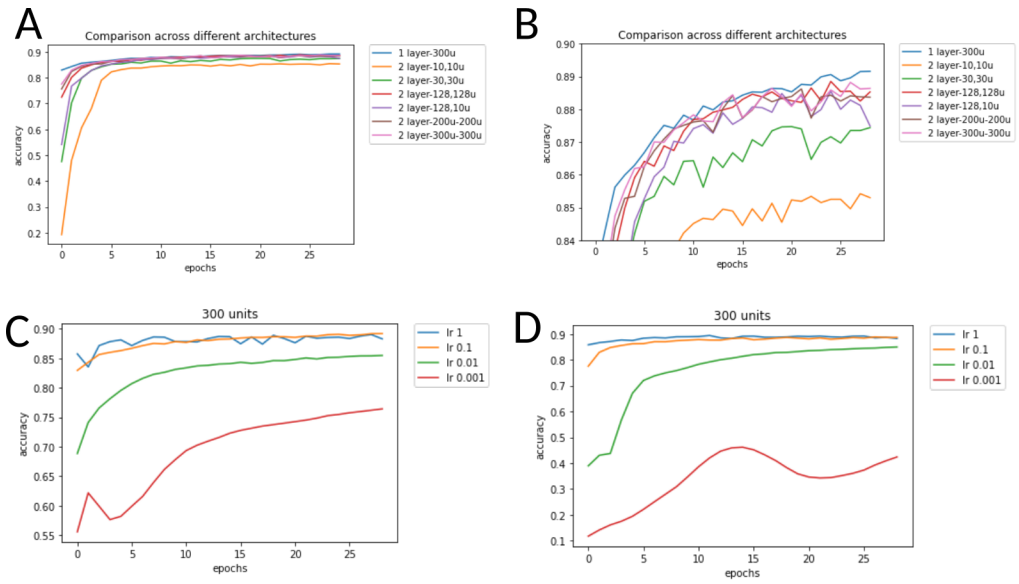

Figure 6: Task 3.6. **A**. Comparison of different MLP architectures showing test accuracies for the best model across learning rates. **B** Zoom of A to enable distinction of performance in the range of accuracy 0.84-0.9. **C** Test accuracies for a MLP with 1 hidden layer and 300 units across various learning rates. **D** Test accuracies for a MLP with 2 hidden layers (300,300) units across various learning rates.

that for this task, a CNN would perform well because image classification problems lend themselves well to CNNs. It's possible that we need more or fewer convolutional layers to achieve good performance with CNNs for this problem or that CNNs simply aren't a good fit for this specific image classification problem, although that seems highly unlikely given that this is a fairly standard image classification problem which, as stated before, CNNs perform well on. It's also possible there was some issue with our implementation of PyTorch's CNN package that caused some inaccuracy or inefficiency in training, though in that case we would expect the accuracy to be much worse than 0.77, so this explanation is also unlikely. In either case more work could be done to experiment with CNNs with different configurations of layers and more finely tuned hyperparameters to explore their performance compared to MLP in more detail.

5 Statement of Contributions

AM and MQ worked on the MLP. SH worked on the CNN. All authors wrote the report.

References

[1] H. Xiao, K. Rasul, and R. Vollgraf. "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms." arXiv: `cs.LG/1708.07747 [cs.LG]`. (Aug. 28, 2017).

[2] Y. Tang, H. Cui, and S. Liu, "Optimal design of deep residual network based on image classification of fashion-mnist dataset," *Journal of Physics: Conference Series*, vol. 1624, no. 5, 2020.

[3] E. Medina, M. R. Petraglia, J. G. R. C. Gomes, and A. Petraglia, "Comparison of cnn and mlp classifiers for algae detection in underwater pipelines," pp. 1–6, 2017. DOI: `10.1109/IPTA.2017.8310098`.