1 Abstract

In this assignment, we analyzed the performance of different neural networks – a Multilayer Perceptron Model with different architectures and a Convolutional Neural Network – on the Fashion MNIST data set. We found that all MLP and the CNN model classify the images of the data set with a high degree of accuracy, with CNN being more accurate at classifying the images of the Fashion MNIST data set. To get the best possible performance metrics we compared the accuracy of different learning rates and we compared the performance of our MLP with a different number of hidden layers, different activation functions, L2 regularization to the network weights, different number of hidden neurons, and the use of unnormalized images.

2 Introduction

The main tasks of this assignment were to acquire the Fashion MNIST dataset, and center, normalize and flatten the images for input to our MLP. This dataset contains 60 000 labeled training examples for us to train our MLP as well as 10 000 testing examples to evaluate the performance of the model. The second goal was to implement a Multilayer Perceptron Model from scratch by creating class objects for the layers of our Neural Network(Input, Hidden layers), activation functions, and softmax function for the output layer. With this MLP implementation, we were tasked with running a series of experiments, in which 1) we determined the optimal hyperparameters for our models by performing grid search and we compared the training and validation accuracy for different architectures of our MLP. Examples of the main MLP architectures that we examined were an MLP with no hidden layer, MLP with 1 hidden layer with 128 units an MLP with 2 hidden layers with 128 units, etc.. 2) For the MLP with 2 hidden layers, we also examined its performance with different activating functions(ReLU, LeakyReLU, Tanh). 3) We also examined the performance of the 2 hidden layers of MLP when adding L2 regularization. 4) We analyzed the performance of our 2 hidden layer MLPs with unnormalized images. 5) We compared the performances of all the different MLP architectures with a CNN consisting of 2 convolutional layers, 2 fully connected layers with 128 units, and a ReLU activating function. 6) After carefully examining which MLP architectures and hyperparameters performed best we were able to come up with a new MLP architecture that outperformed all the previous ones. We explored how adding more width and depth to MLP increased expressive power we also investigated (and implemented) early stopping methods for neural networks, such as patience. We looked at other papers using this dataset, where people used several different techniques to achieve 96% accuracy such as Fin Tuning Differential Architecture Search (DARTS) which uses fixed operations to correct imperfect approximations [TKK20] and Sharpness-Aware Minimization (SAM) [For+20] which seeks parameters that lie in neighborhoods having uniformly low loss.

3 Datasets

In the Fashion MNIST dataset, there are a total of 70 000 labeled 28 by 28 black and white images, split into a training set of size 60 000 and a testing set of size 10 000. This dataset was intended as an alternative to the famous original MNIST dataset of handwritten numbers and is what we used to evaluate the accuracy of our MLP. There are a total of ten classes, each corresponding to a type of clothing, however, there are some similarities between classes, namely the sneaker and ankle boot categories. For each set, we do find that the labeled data is evenly distributed, with 6000 members of each class in the training set and 1000 members of each class in the training set.

4 Results

(1) We examined the effect of the number of hidden layers (of 128 neurons) on the prediction accuracy with zero, one and two hidden layers using RELU as an activation function, shown in the table below:

hidden layers	*validation accuracy*	*percentage*
0	0.8287	82.87%
1	0.8573	85.73%
2	0.8798	87.98%

Firstly, after implementing the model we started with a small perturbation check to see if our gradient calculation and cross entropy loss function were implemented correctly. We obtained a small perturbation calculation value of 1.4468266349237225e-11 (see last line of colab), which confirmed our model implementation before we started training and testing.

So, we can observe that, as we keep the other hyperparameters -- such as the number of nodes per layer, the batch size (128), and the activation function -- there is definitely an upward trend when we increase the number of hidden layers. This naturally makes sense, as since we improve the complexity of the model, we are better able to capture the details, and thus create better predictions. For hyperparameter tuning of the three models, we performed a grid search on the training data to determine the optimal learning rate. In Figures 5, Figure 6, and Figure 7 you can see that a learning rate of 0.1 outperformed 0.05, 0.001, and 0.005 for all three models. We therefore used a learning rate of 0.1 for each one.

As we know, the model with zero layers is essentially multiclass regression model – therefore it makes sense to see that the non-linearity will increase the accuracy, as it will further be able to address the complexity of the problem.

In Figure 1 and Figure 2 you can see the results for the zero hidden layer MLP where inputs are mapped directly to the output. We have two plots, Figure 1 contains the training and validation accuracies as a function of the epoch and Figure 2 is a plot representing the loss as a function of the number of total iterations. We used Figure 1 to determine the optimal amount of epochs to train the model on to avoid overfitting. We observed that the validation accuracy peaked at 2 epochs before falling. In Figure 2 we see a nice convergence, showing the model learning properly.

In Figure 3 and Figure 4 we have the same plots as in Figure 1 and Figure 2 respectively, except for the one hidden layer MLP with 128 hidden units and using RELU as an activation function. In Figure 3, we observe that the validation accuracy peaks at 2 epochs before falling, just like in the 0-layer MLP. In Figure 4 we also see a nice convergence, showing the model is learning properly.

In Figure 8 and Figure 9 we see the same plots for the two hidden layer MLP with 128 hidden units and using RELU as an activation function. In Figure 8, we observe that the validation accuracy peaks at 7 epochs this time before falling. In Figure 9, just like 0-layer and 1-layer MLPs, we see the cross entropy loss converging to a local/global minimum.

(2) After seeing the performance of the 2 layer ReLU MLP -- as a direct follow up -- we decided to examine the effect of using different activation functions on the accuracy of the models, using tanh and leakyRELU on the models and evaluating their performance:

activation function	*validation accuracy*	*percentage*
tanh	0.8759	87.59%
RELU	0.8798	87.98%
leakyRELU	0.881	88.1%

These models were all trained with 2 hidden layers and an optimal learning rate of 0.1, and we can see that, at least in this case, the performance between the different models are fairly similar. From these tests we cannot say that one is strictly outperforming the others, so we should simply

choose the one that has the most advantages for our situation. For example, LeakyReLU and ReLU are less computationally expensive than TanH and addresses the vanishing gradient problem. LeakyReLU in our case performs the best by a small margin -- but it's main advantage over ReLU is that it addresses the dying ReLU problem where gradients "die" and become 0 (which leads to dead neurons). In Figure 10 and Figure 11 you can see the cross entropy loss as a function of iterations, showing convergence of both models.

(3) Then, we wanted to examine the effects of L2 regularization on the accuracy of our model. Since λ is tuneable hyperparmater, we performed a grid-search and observed how the test accuracy changed as shown in the following table:

L2 regularization factor	*validation accuracy*	*percentage*
0.001	0.8742	87.42%
0.01	0.8587	85.87%
0.1	0.779	77.1%
1	0.1	10%

In Figures 12, 13, 14 and 15 you can observe the results of the MLP consisting of 2 hidden layers having 128 units and ReLU activation function but with L2 regularization added to the network weights. The λ's that used for the L2 regularization were [0.001, 0.01, 0.1, 1]. We know that regularization prevents the model form overfitting, which limits the complexity that the model can capture. In our case, overfitting is not the limiting factor, and thus we can see that as we reduce the complexity of the model, we are failing to capture the details that are necessary to make an accurate prediction.

(4) We then were tasked with training the model on unnormalized images. When training on these unprocessed images, we observed a lower accuracy of 75.07%. Although the pixel values are relatively close, in this case we see that normalization is necessary to achieve optimal classification performance. The main reason to explain this drop in accuracy is that the unnormalized images have large pixel values (up to 255) which makes the calculations more computationally expensive and difficult to handle for our model. This can be seen as we had to reduce the learning rate to 0.0001 to avoid nan values. Normalizing the pixel values allows for easier computation and faster convergence for the model.

In Figures 16 and 17 we observe the results of the 2 layer ReLU MLP trained using unnormalized images. Figure 16 is a plot of the training and validation accuracies in terms of epochs, where we observe that the validation accuracy peaks at 2 epochs. Figure 17 then shows a plot of cross entropy loss vs iterations, where smooth convergence of the model can be observed.

(5) Using a Convolutional Neural Network with two convolutional layers and two fully connected hidden layers (each with 128 hidden neurons) we found the accuracy to be 90.95% on the testing set, after tuning and optimizing the hyperparameters of the model, we had a test accuracy of 91.4% as well as a loss of 0.350. For the hyperparameter tuning of our CNN model, we focused on getting the optimal accuracy for 3 main hyperparameters: filter size, number of filters, and dropout rate. To do this we performed two grid searches -- one on filter size and number of filters and the other solely on dropout rate. What we observed from the first grid search is that a 8 filters, each size of 3 pixels gave us the best test accuracy of 87.58%. This can be seen in Figure 21, where the mean score is the highest for 8 filters of size 3. We took these filter hyperparameters and did a second grid search where we observed that a 0 dropout rate achieved the best mean score. This can be see in Figure 22. We can see the training and validation accuracy as a function of epochs for our CNN in Figure 18

We observe that our CNN model performs better then our previous MLP models with the same number of hidden layers and connected nodes, as expected. This makes sense as CNN's are

designed for use for applications in image and speech recognition. Its built-in convolutional layer reduces the high dimensionality of images and takes into account the local connectivity.

The color map in Figure 19 shows the accuracy of the prediction of each class. So for all classes with exception of 6, the model predicts with high accuracy (85% and above). However for class 6, the model often makes an incorrect prediction and is often mistaken for classes 0, 2, and 4. If we examine the actual labels, we see that class 0 is a Tshirt/Top, class 2 is pullover, class 4 is coat, and class 6 is shirt. We can see that there are obvious visual similarities between these classes, even a human could misclassify these categories of clothing, which we can see in 20, which are all similar despite belonging to different classes. For example, the outline of a pullover is very similar to that of a shirt, so there is good reason for the model to perform poorly at classifying these examples (same goes for coats and t-shirts/tops).

(6) In attempt to find the best possible MLP architecture to go head to head with the CNN we decided to test the effects of making our MLP deeper (by adding more layers). Our best performing MLP model from the earlier experiments is the 2 layer LeakyReLU MLP with 128 hidden units, with a test accuracy of 88.1%. When adding another layer we observed a test accuracy of 88.2%, a marginal increase in performance, but an increase nonethelesss. The training and validation accuracy as a function of epoch plot can be seen in Figure 23, where the validation accuracy plateaus at epoch 2 but steadily increases until 10th epoch. The cross entropy convergence can be seen in Figure 24.

We attempted to push the depth a bit more with a 4 layer LeakyReLU MLP, but our performance dropped to 87.46%. This goes to show that added complexity doesn't always increase the performance of the model. The accuracy vs. epoch and cross entropy loss vs. iteration can be seen in Figures 25 and 26 respectively.

We also tested the effect of the number of hidden units in a hidden layer on the test accuracy. In order to analyze this effect, we examined the accuracy of a 2 hidden layer MLP with 32, 64, 128, 256 hidden units respectively. The 4 models had the same activating functions and hyperparameters The accuracy for the model with 32 hidden units was 86.41% , the model with 64 hidden units was 87.56% , the model with 128 hidden units was 87.98% and the model with 256 hidden units had an accuracy of 88.21%. As we can see, as the number of hidden units in a hidden layer increases our test accuracy increases as well.
From these results we can see that added complexity (in terms of depth and width) increases the expressive power of the multilayer perceptron. However, even though making our MLP deeper and wider had a marginal increase in test performance, the MLP model still fell short to the performance of our optimal CNN by roughly $\sim 3\%$. It is also important to note that increasing the parameters also increased the training time of the models.

(7) When implementing our MLP model, one of our main focuses was to make sure that our model was not overfitting. We made sure that we trained our models using a validation set before training with the entire training dataset, but we wanted to incorporate an early stopping element to our model to monitor the convergence of the validation set. One additional hyperparameter we added to our MLP implementation was the 'patience' argument. The 'patience' of our MLP represents the maximum number of iterations of observed increasing validation cross entropy loss before the training is completely stopped. This early stopping allows for our model to notice when the model's generalisation is becoming worse on "unseen" data and starting to overfit the training data. We decided to use a patience of 10 iterations to incorporate early stopping in our MLP implementation.

5 Discussion and Conclusion

We can see a definite trend in the improvement of MLP model accuracy when we increase the number of hidden layers, especially between 0, 1, 2, and 3 for the Fashion MNIST dataset.

However, when we introduce the fourth hidden layer, we see that the accuracy of the prediction goes down, which implies that we are overfitting when we add a fourth layer. As expected, the linear model with zero layers performs the worst, as it cannot capture the complexity of the data. When we perform the same experiments, except with different activation functions, we see that there is marginal increase in performance when applying different functions, which is completely reasonable, as all activation functions tested are commonly used. Addressing the L2 regularization, we observed a notable decrease in accuracy of our model as we increase our regularization factor. The same effect is seen when we train the model on data that is not normalised, which results in a drop to 75.05% accuracy, which highlights the importance of preprocessing the data before training. Lastly, we see that a well tuned ConvNet performs better than our MLP, but its accuracy can be approached by tuning the performance of our MLP.

6 Statement of Contribution

- Willie Habimana (260987793): Part of MLP & CNN experiments, and report
- William Huang (260972252): Acquire and prepare data, report.
- Dgebe Nicoals (260867207): Worked on MLP Implementation, MLP & CNN experiments, and report.

7 Appendix

Figure 1: 0 Layer ReLU MLP Training/Validation Accuracy vs. Epoch

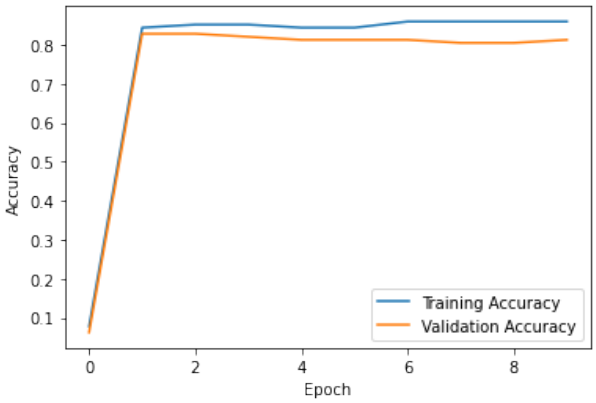


Figure 2: 0 Layer ReLU MLP CE Loss vs Iteration

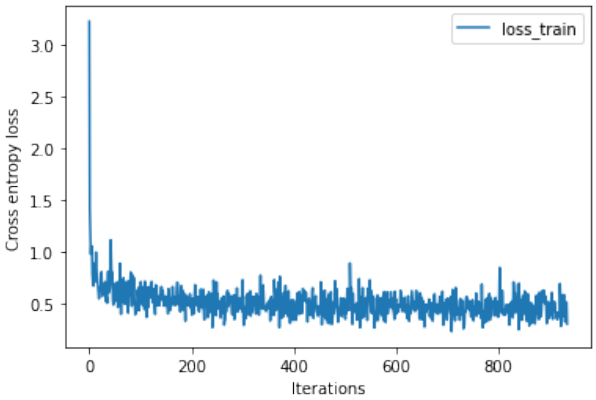


Figure 3: 1 Layer ReLU MLP Training Accuracy vs. Epoch

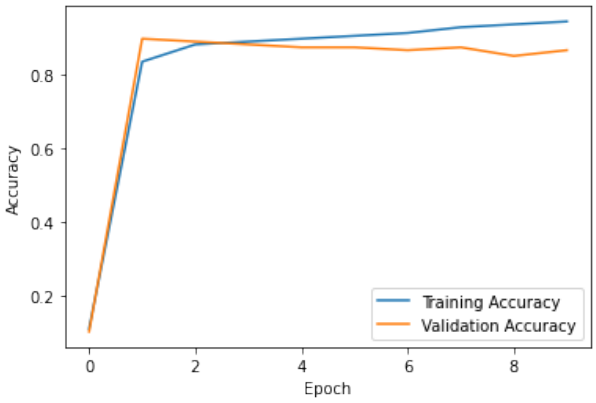


Figure 4: 1 Layer ReLU MLP CE Loss vs Iteration

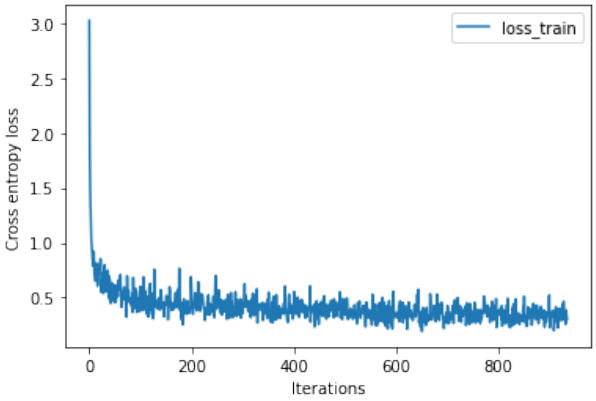


Figure 5: 0 Layer ReLU MLP Mean Score vs. Learning Rate

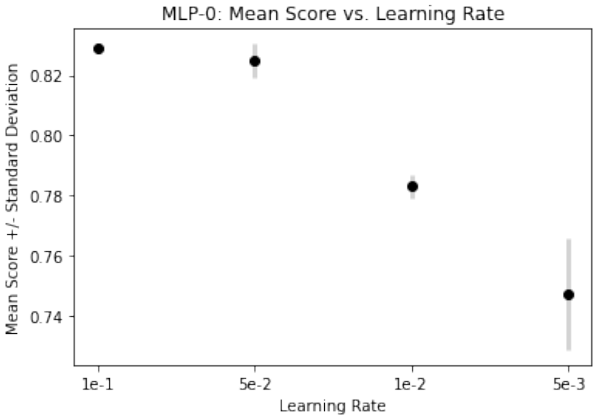


Figure 6: 1 Layer ReLU MLP Mean Score vs. Learning Rate

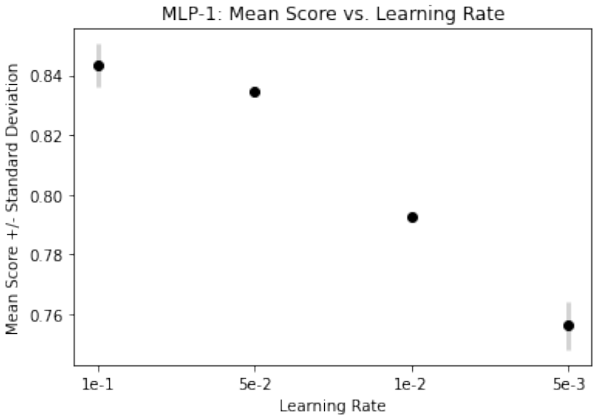


Figure 7: 2 Layer ReLU MLP Mean Score vs. Learning Rate

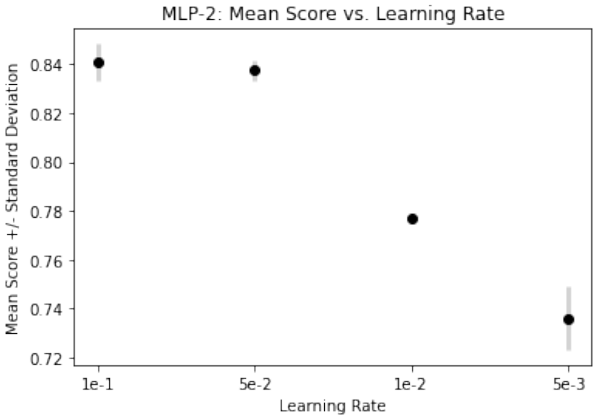


Figure 8: 2 Layer ReLU MLP Training/Validation Accuracy vs. Epoch

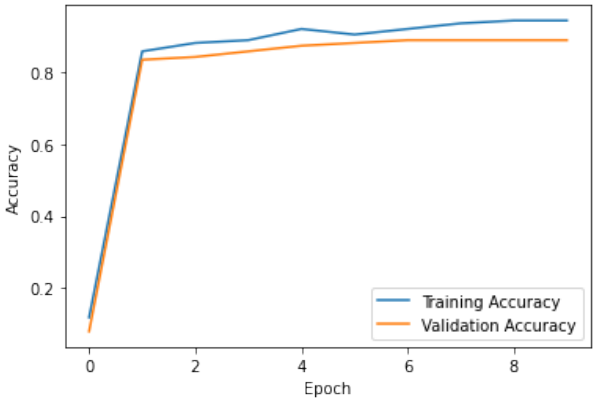


Figure 9: 2 Layer ReLU MLP CE Loss vs Iteration

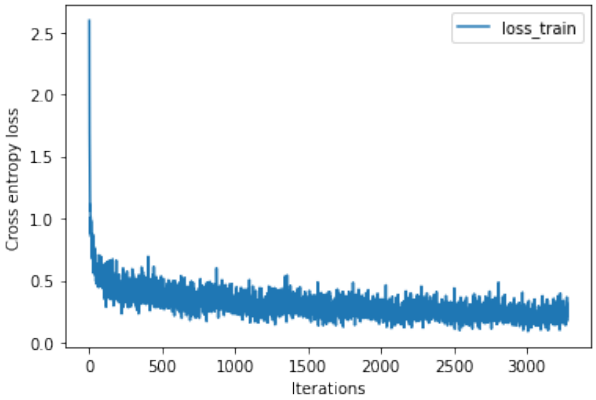


Figure 10: Loss as a function of iterations of the MLP with two hidden layers of 128 and tanh as an activation function

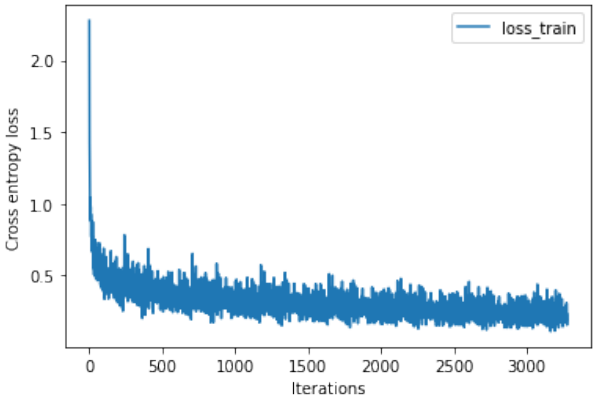


Figure 11: Loss as a function of iterations of the MLP with two hidden layers of 128 and leakyRELU as an activation function

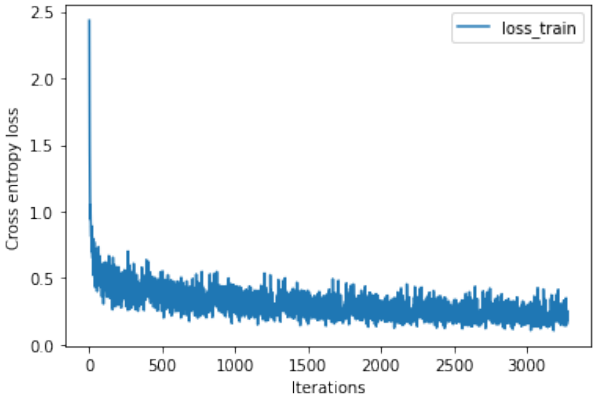


Figure 12: cross entropy as a function of iterations when applying a 0.001 L2 regularization factor

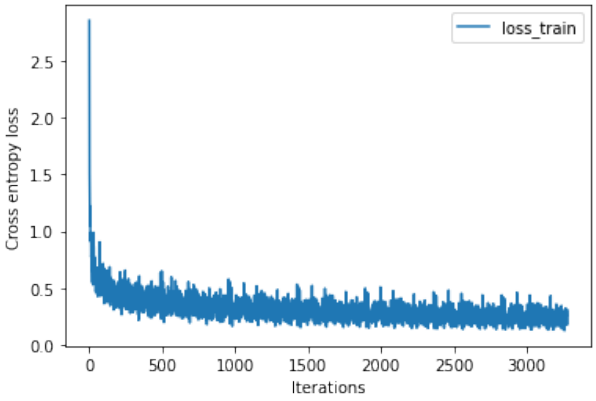


Figure 13: cross entropy as a function of iterations when applying a 0.01 L2 regularization factor

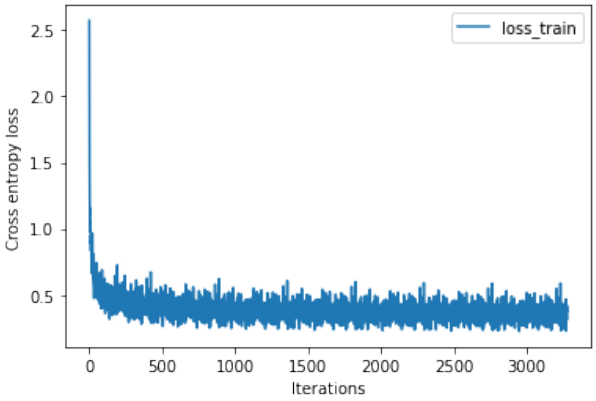


Figure 14: cross entropy as a function of iterations when applying a 0.1 L2 regularization factor

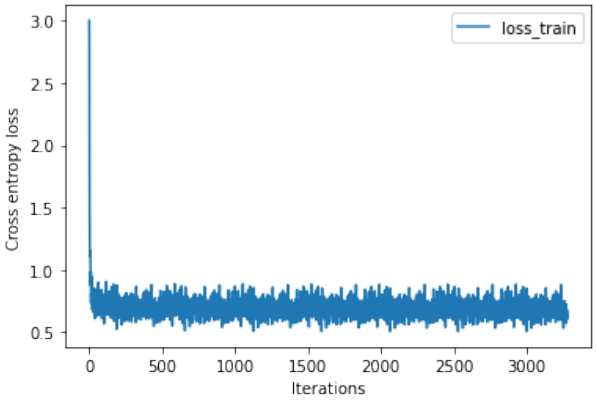


Figure 15: cross entropy as a function of iterations when applying a 1 L2 regularization factor

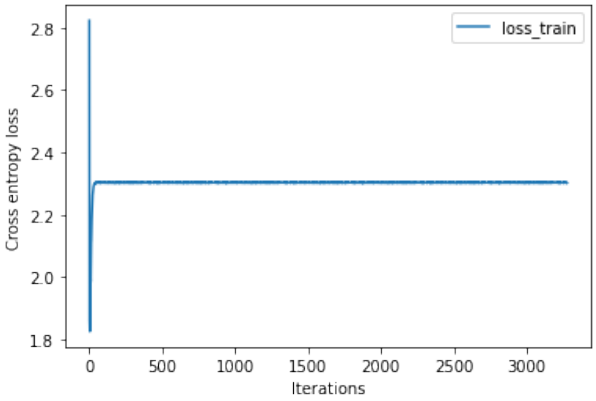


Figure 16: Training Unnormalized Images: Accuracy vs. Epoch

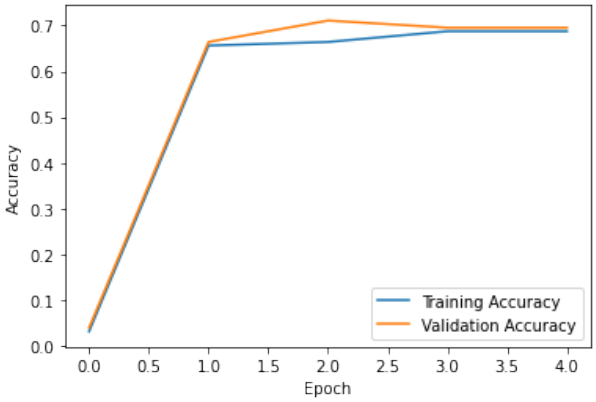


Figure 17: Training Unnormalized Images: CE loss vs. Iteration

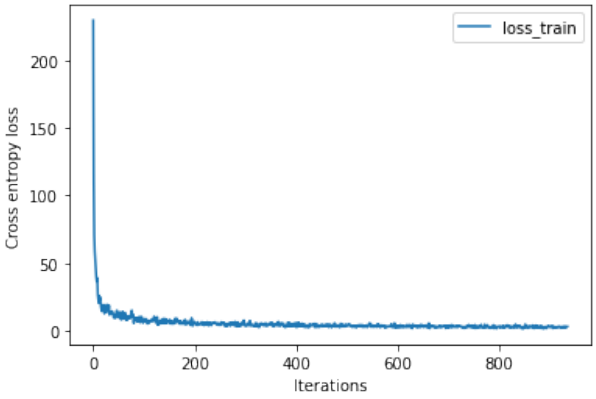


Figure 18: Training and validation accuracy vs epochs for our CNN model.

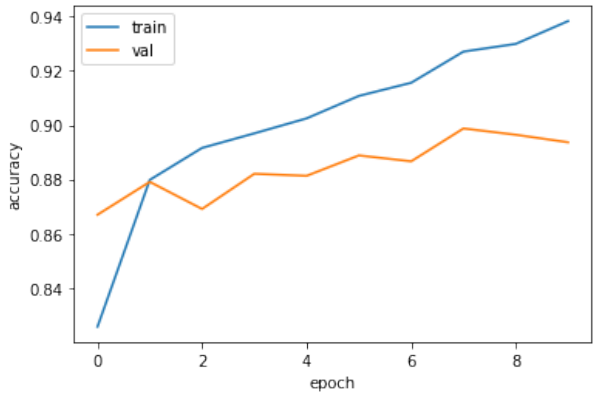


Figure 20: Samples that look like the same class, however are from four distinct classes

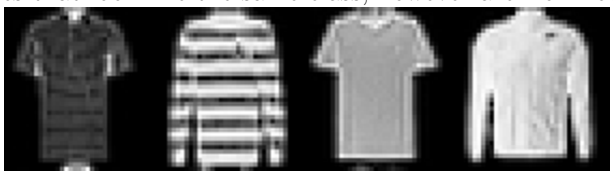


Figure 19: Predicted vs Actual performance of the CNN on the test set

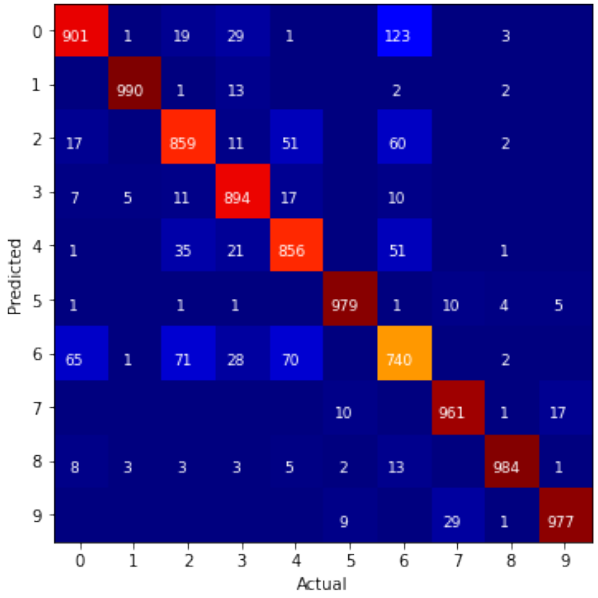


Figure 21: CNN: Mean Score vs (Filter Size, Number of Filters)

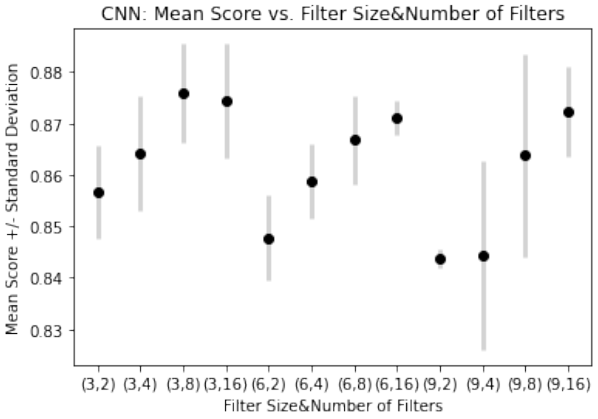


Figure 22: CNN: Mean Score vs Dropout Rate

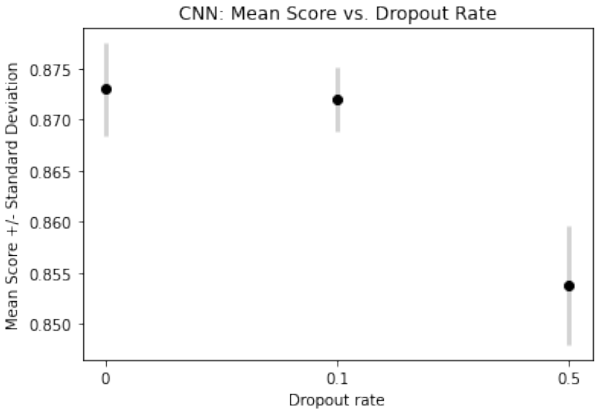


Figure 23: 3 Layer LeakyReLU MLP Training/Validation Accuracy vs. Epoch

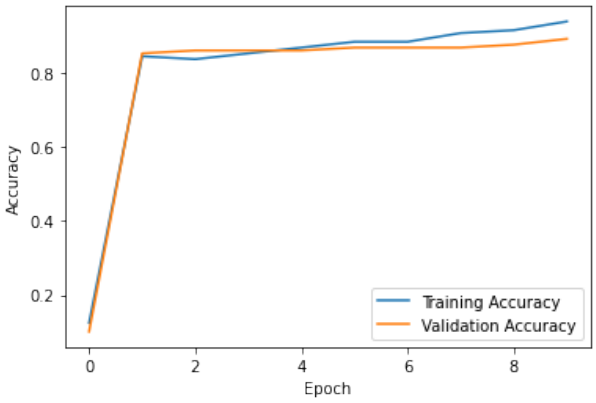


Figure 24: 3 Layer LeakyReLU MLP CE loss vs. Iteration

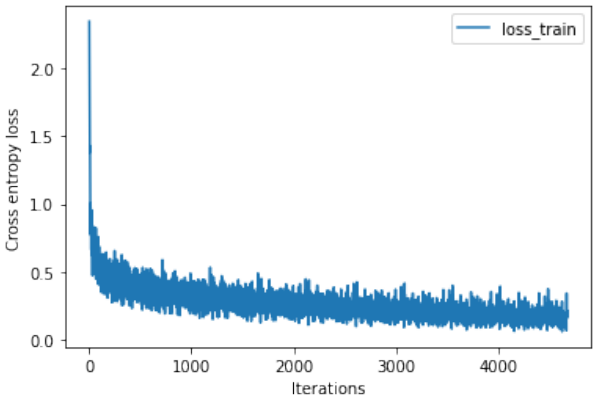


Figure 25: 4 Layer LeakyReLU MLP Training/Validation Accuracy vs. Epoch

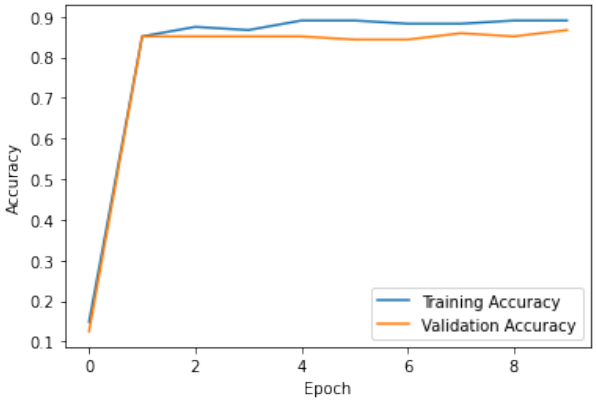
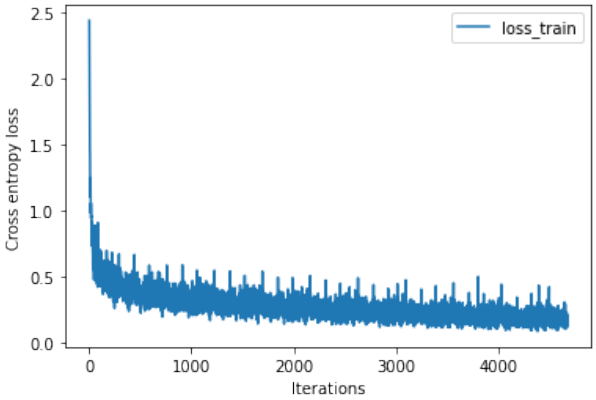


Figure 26: 4 Layer LeakyReLU MLP CE loss vs. Iteration



References

[For+20] Pierre Foret et al. *Sharpness-Aware Minimization for Efficiently Improving Generalization*. 2020. DOI: 10.48550/ARXIV.2010.01412. URL: https://arxiv.org/abs/2010.01412.

[TKK20] Muhammad Suhaib Tanveer, Muhammad Umar Karim Khan, and Chong-Min Kyung. *Fine-Tuning DARTS for Image Classification*. 2020. DOI: 10.48550/ARXIV.2006.09042. URL: https://arxiv.org/abs/2006.09042.