Mini Project 3 Report

Frida-Cecilia Acosta-Parenteau, Daphné Ducharme, Celeste Groux

1 Abstract

The goal of this project was to implement different MLP and CNN model architectures in order to study their accuracy on classification tasks. The MLP model experiments showed that normalizing images was imperative, one layer was better than 0 or 2, and tanh was the best activation function. As well, the number of hidden units in a layer was found to not affect the model performance, and similarly the difference due to adding dropout was very small. Early stopping on the other hand was found to be useful as a regularization strategy to improve test accuracy. The CNN model experiments showed that max pooling was better compared to average pooling, while more channels also increased the model performance. However, surprisingly, additional convolutional layers were found to decrease the testing accuracy. Overall, we found that our best CNN model outperformed the best MLP model as expected, but the difference was very small.

2 Introduction

The main objective of this project was to study the impact that altering the training phase could have on the accuracy of classification made by neural networks models. The dataset that was used was Fashion-MNIST consisting of Zalado's fashion article images [1]. We implemented MLP models that differed in their parameters to study their effect and also implemented a CNN model to then compare to our best possible MLP model. We obtained results that varied greatly for the MLP models, and by selecting the best hyperparameters and training options given our experiments, we ultimately built an MLP model that was almost as accurate as the CNN one. Training these types of models proved to be quite challenging since neural networks can have a very high variance and thus tend to overfit. Many researchers, such as Guoqiang Peter Zhang in his paper Neural Networks for Classification: A Survey [2], have tried to address this problem. In the paper, Zhang explains some of the model choices that can be done to improve the performance of neural networks in classification. Indeed, he mostly discusses that, in general, the best way to improve an MLP model is by reducing the optimization done during the training, and this lead us to try a few experiments to decrease the expressiveness of our model, such as using early stopping or dropout for regularization.

3 Dataset

The data used for this project is the Fashion-MNIST dataset. This data consists of 60,000 train and 10,000 test 28x28 grayscale pixel Zalando article images of fashion items. These fashion items are split into 10 classes: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot. The features of this dataset are the pixel darkness, which are represented by a value between 0 and 255. This dataset is known as a benchmark dataset and is commonly used in the machine learning community to verify the performance of an algorithm. For this reason, it is a good dataset to test our models against.

Given the large size of the dataset and the limited computational resources, the size of the training set, validation set, and test set were reduced to 1000, 1000, and 400 instances respectively to use with the MLP model. This selection of a small subset of data is due to issues related to Google Colab. Indeed, even with the extra credits, running the project seemed to never end; it would run and never stop. Thus, because of time constraints, we simply used Colab and reduced the dataset size. The CNN model however was applied to the entire dataset. The data was also normalized for input to the MLP model. This meant that every pixel feature was normalized separately. For the CNN model, the data was normalized using PyTorch transforms toTensor() and Normalize().

4 Results

Note that unless stated otherwise, the reader may assume that the MLP models in this section have 2 hidden layers, 128 units, no dropout regularization, data normalization, and ReLu as the activation function. The MLP model was also trained over 4000 epochs unless otherwise stated.

4.1 Three different MLP models

In this experiment, we studied the effect of depth on the accuracy of an MLP model on a classification task. We compared the accuracy of models with 0, 1, and 2 hidden layers.

Number of hidden layers	0	1	2
Accuracy	0.795	0.81	0.6125

Table 1: Accuracy of MLP Models with Differing Number of Hidden Layers

From the results in Table 1, we see that the error loss is slightly decreased from 0 to 1 hidden layer, but it increases considerably from 1 to 2. This is surprising because, typically, increasing the depth of an MLP model is known to improve the prediction accuracy as it allows for the model to be more expressive and better fit to the given data. Perhaps the addition of the second layer however leads to too much expressiveness and so this model overfits, which could explain why it seemed to perform worse.

4.2 Activation Functions

For this task, we studied the effect of different activation functions on an MLP model. The three activation functions compared are ReLu, tanh, and Leaky-ReLu with parameter 0.01.

Activation function	ReLu	tanh	Leaky-ReLu
Accuracy	0.6125	0.7625	0.69

Table 2: Accuracy of an MLP Model with Different Activation Functions for two hidden layers

According to the results from Table 2, Tanh provides the best prediction accuracy while Relu performed the worst out of all three. Leaky-ReLu stands in the middle, but it has the potential of offering a better accuracy by trying different parameter values, which is why we decided to test it in experiment 4.6. It is not surprising that leaky-ReLu performed better than ReLu because the former do not squash all the negative values passed to it to 0 like ReLu, but instead it multiplies them by some small parameter value.

4.3 Dropout Regularization

In this section, we experimented on the effect that dropout regularization might have on an MLP model. As is commonly done, a mask was applied to each layer excluding the bias, where each neuron of a hidden layer had a 50% chance of being dropped, while each neuron of the input layer had a 80% chance of being kept.

Dropout	Inactive	Active
Accuracy	0.6125	0.365

Table 3: Accuracy of an MLP Model with and without Dropout Regularization

We see that the performance of the model seems to worsen when we include dropout regularization. Perhaps, the dropout as implemented removed too much from the expressiveness of the model, and made it harder to learn how best to classify the images. Perhaps dropout would have been more helpful if we had used a larger set of training samples since this might have balanced the reduced expressiveness of the model by giving it more training data to learn from.

4.4 Unnormalized Images

In this section, we compared the accuracy of the model with normalized and unnormalized data. As can be seen in Table 4, normalization is evidently very important to the MLP model performance. The difference in accuracy is very large at almost 52%. Normalization is very important since it forces all the features to share the same mean and variance which can lead to a faster convergence.

For instance, it can be noted that for the unnormalized data the training iterations stopped after epoch 100 because the loss overflowed to infinity. This did not happen for normalized data.

Data	Normalized	Unnormalized
Accuracy	0.6125	0.09

Table 4: Accuracy of an MLP Model with and without Data Normalization

4.5 CNN

Various hyperparameters were tested to see the effect on final performance of the model. These models were also all trained on 150 epochs. The three hyperparameters varied were the number of convolutional layers, max pooling versus average pooling, and the number of channels. The CNN architecture is structured first with convolutional layers with 3 channels, Relu activation function and pooling, then 2 linear layers of 128 units. The structure of the first part is altered here. First, we tested maximum versus average pooling. The CNN had 2 convolutional layers, with pooling after each layer. The CNN model with max pooling was found to perform best, with a small improvement over average pooling. Average pooling smooths out the image features, while max pooling brings out sharper image features. Commonly max pooling is used, and this choice is supported by the results we obtain, where bringing out sharper image features seem to perform best.

Next, we tested different number of convolutional layers. The networks tested had either 2, 4, or 6 convolutional layers. Given the small input size of 28x28 and these additional layers, pooling was applied only to the last layer. Max pooling was selected as the pooling type given the results of first test. We found that the model with 2 convolutional layers performed best, with performance decreasing as the number of convolutional layers increased. In fact, the performance dropped off considerably for the six layer CNN model. Looking at the respective loss versus accuracy plots as a function of epochs, we find that the 4 layer CNN spends about 25 epochs at a loss of around 2.3 before dropping off and increasing steadily in accuracy, while the 6 layer CNN accuracy remains constant over training, and the loss remains at around 2.3 for all epochs. This is surprising to see, as one would expect some improvement during training, and more comparable results to the other compared networks. This result is perhaps due to the CNN not being able to converge given the much higher complexity of the model with much more weights to optimize. These results also show that a simple 2 convolutional layer CNN can still perform very well.

Finally, we tested the effect of the number of channels used. We compared using 3 channels versus only one, using the model architecture of the two layer CNN in the previous experiment. We found that using 3 channels instead of 1 improved the performance by almost 3%. Most CNN models shown in class used multiple channels, so these results support this choice also. Perhaps the use of additional channels perhaps acts similarly to ensembles in aiding predictions.

Pooling Type	Maximum	Average	-
Accuracy	83.35	82.6	
# Convolutional Layers	2	4	6
Accuracy	85.33	84.46	10.0
# Channels	1	3	-
Accuracy	82.65	85.33	

Table 5: CNN Hyperparameter Experiment Results

4.6 Different Parameter Values for Leaky-ReLu as an Activation Function

In this experiment, we explored the effect of different parameters for the parametric leaky-ReLU activation function. The parameters considered and the respective MLP model accuracy are shown in Table 6 and in Figure **??**.

Parameter value	0	0.01	0.5	0.75	1	2	3	5
Accuracy	0.7325	0.685	0.7575	0.775	0.775	0.7425	0.635	0.73

Table 6: Accuracy of an MLP Model with Different Parameter Values in the Leaky-ReLu Activation Function

The best model accuracy was found using parameter values 0.75 and 1, both with an accuracy of 77.5%. Parameter value 0.5 was not far behind with an accuracy of 75.75%, and the worst accuracy of 63.5% was obtained with parameter value 3. Usually, most programs will use 0.01 as the parameter value because it provides a small non-zero slope for the negative values. However, in our case, 0.01 returned the second worst accuracy. But it is known that leaky-ReLu is not guaranteed to be better than ReLu for all problems and datasets, so perhaps this is just not the best option for our dataset, especially when it has such a small number of instances to train on.

4.7 Early Stopping for Regularization

In this section, we studied the effect of using early stopping for regularization in order to increase the accuracy of the MLP model. Indeed, we used a validation set of the same size as the training set on which we stopped the iteration early if the difference of loss was lower than 0.01. For our other experiments, we used 4000 as our number of iterations, but early stopping provided us with a new value of 2100 that we used on the same model to compare the two accuracies.

Early stopping	Non-active	Active
Accuracy	0.6125	0.6475

Table 7: Accuracy of an MLP Model with and without the Use of Early Stopping in its Training Phase

As a result of using early stopping, we observe a small increase in the accuracy of the model. One reason for this could be that our model was overfitting, and this regularization technique managed to decrease the variance for better generalization, while still being able to fit well to the training data.

4.8 Different Number of Hidden Units per Layers

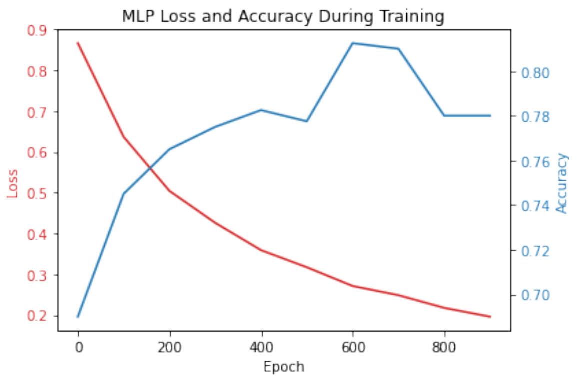
In this experiment, we tested the MLP with 64 hidden units in each layer.The results in Table 8 are similar to the results with 128 hidden units. We have seen that increasing width rather than depth does not change the accuracy as much.

Number of hidden layers	0	1	2
Accuracy	0.78	0.81	0.6325

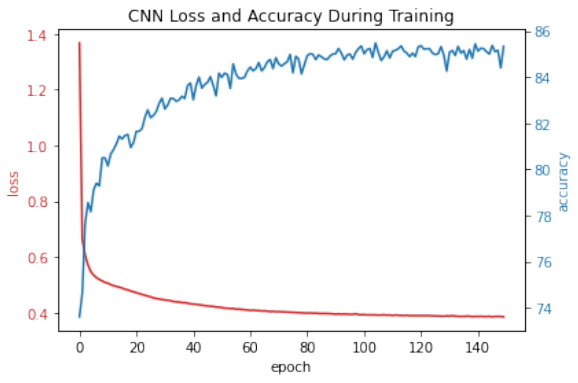
Table 8: Accuracy of an MLP Model with 64 hidden units per layer

4.9 Best MLP and CNN Architecture

Given the previous experiments, the best MLP model architecture was selected as follows. It was chosen to have 128 hidden units in a single hidden layer. We stopped the epochs at 1000 since the accuracy did not improve further as we can see in Figure 1a. We chose these hyper-parameters according to their performance in our experiments. Further, ReLu had a better performance compared to TanH for one hidden layer, which is not shown in the results. This can be justified by the fact that ReLu does not have the vanishing gradient problem. We also decided to omit early stopping because, even though the results we presented in section 4.7 show a slight increase in accuracy, we deemed this increase to be too small to consider it, especially since the project results in general seemed to vary a lot on each run.



(a) MLP



(b) CNN

Figure 1: MLP and CNN Loss and Accuracy during Training

For the CNN model, the best model architecture was selected as the two convolutional layer model with max pooling after the second layer, and two linear layers of 128 units. This model was found to have the best performance given CNN hyperparameter tests, achieving a final accuracy of 85.33% after 150 epochs. The results of training loss and test accuracy over the number of training epochs for both MLP and CNN models are shown in Figure 1b.

5 Discussion and Conclusion

In MLP, we found that there was a lot of variation in the accuracy results for all the experiments. The range of variation in accuracy results was around 20%. However, the two hidden layers accuracy never went above the one layer result. This contradicts the experimental data in theory, which says that having more layers is better for accuracy for multi-class classification. However, in our case, these results could be a consequence of overfitting, since we used a small training dataset of 1000 images. Therefore, a very complex model with 128 hidden units in two hidden layers probably overfitted. Further, we noticed that the loss decreased a lot slower than with one layer. For the same number of epochs (4000), we got losses less than 0.1 for one hidden layer whereas in two we got at the minimum 0.9.

The results of the hyperparameter experiments for the CNN model were also found to support common CNN architecture choices of multiple channels and max pooling. It was surprising to see that the two convolutional layers CNN performed best compared to additional layers, but perhaps this model is already expressive enough to classify the images well, and the addition of more layers possibly leads to overfitting.

Comparing CNN and MLP, we find that the best performance achieved for both are 85.33% and 81% respectively. As expected, CNN performs best. CNN, unlike MLP, is able to capture spatial patterns which is very helpful for the task of image classification. However, it should be noted that the difference between CNN and MLP is very small. One might expect the gap to be much higher, but finding the best model architectures to best make use of these models can be difficult. Overall, it shows that a simple CNN model trained over a small number of epochs can easily outperform the MLP model for the task of image classification.

6 Statement of Contributions

Frida-Cecilia worked on implementing the MLP model and the experiments. Daphné worked on implementing some experiments. Celeste worked on loading and processing the data, as well as the convolutional neural network experiments.

References

[1] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[2] G.P. Zhang. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462, 2000.