COMP 551: Mini-Project 3

Antoine Bonnet
260928321
McGill University

Dragos Secrieru
260923248
McGill University

Cyril Saidane
260706157
McGill University

Abstract

This paper explores the power of multilayer perceptron (MLP) models to perform image classification on the Fashion-MNIST dataset. We considered multiple MLP models by varying the number of layers, activation functions, dropout rate, image normalization and comparing its optimal performance with a Convolutional Neural Network (CNN). We found that the optimal MLP architecture consists of 1 hidden layer with a ReLu activation function which achieved a test accuracy of 88.17%. We also found that CNN outperformed all MLP models with a test accuracy of 92.17%.

1 Introduction

1.1 Motivation and findings

Our image classification models were all tested on the Fashion-MNIST dataset. Our goal was to find the optimal MLP model for that dataset, and compare its performance with a CNN model. We found that even with extensive amounts of hyperparameter tuning on the MLP, the CNN model still vastly outperformed it. No hyperparameter tuning was performed on the CNN model. We also found that normalizing our images and adding a dropout parameter increased our validation accuracy. Nonetheless, all of the improvements resulting from hyperparameter tuning were marginal at best, as the MLP model had a plateau at around 87-88% accuracy, a slight improvement to the standard depth 0 model which achieved 85 % accuracy.

1.2 Relevant work

We first explored the literature on MLP and CNN models to inform our architecture decisions. A paper (3) explores MLP hyperparameter tuning on the fashion MNIST dataset and obtained an accuracy of approximately 87 % (which is similar to our performance) for a depth 3 architecture of hidden layer widths 256, 128 and 100. Most of the papers we found were using CNN models, which makes sense since they are inherently more adapted to image classification. For example a CNN model similar to ours was used in (1), achieving a test accuracy of 92% with 2 convolutional layers and 2 dense layers.

The most interesting addition they introduce is called batch normalization, which normalizes the data between every single layers in the CNN. The paper found that the model converged around 4 times faster. Although we did not implement batch normalization, it could be the subject of future experiments.

Another paper (2) also used a CNN model to classify Fashion-MNIST and had a more complex model which included the use of pooling layers. They found a 4x improvement in training times. Pooling layers reduced the input size and thus the number of parameters to learn. For that reason, we included a pooling layer in our CNN model.

2 Dataset

The Fashion MNIST dataset is a dataset composed of 70,000 gray-scaled images showing articles of clothing. The images are low resolution and encoded into 28x28 matrices of pixel values. They are labelled with one of 10 classes (dress, coat, trousers, etc.). The data is split with 60K instances for training and 10K for testing.

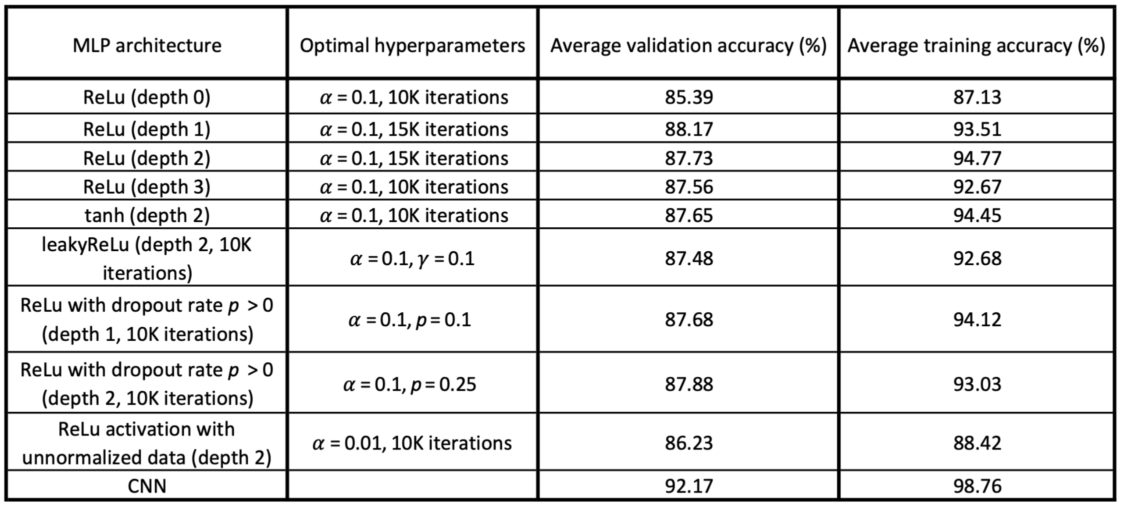
2.1 Pre-processing features

To pre-process the data, we started by vectorizing each image into a one dimensional array, turning the 28×28 image matrices into a 784×1 vector. The training set becomes a $60,000 \times 784$ matrix and the testing set a $10,000 \times 784$ matrix. This is done to format the data for input into a multi-layer perceptron. We then normalized each vectorized image so that each pixel value has standard normal distribution over the training set to improve training times by smoothing out the inputs for gradient descent. We also transformed the labels using one-hot encoding to use the multi-class cross-entropy loss, thus turning each of the 10 unique labels into a unit vector of length 10.

3 Results

The performance of each architecture with a given set of hyperparameters was obtained through a K-fold cross-validation function which was used in most of our experiments. For every architecture, we selected the optimal hyperparameters with highest validation accuracy. The corresponding average validation and training accuracies of each optimal architecture are listed in Figure 1.

In all MLP experiments, we used mini-batch gradient descent on the cross-entropy loss to optimize the weights and biases of the network with batch size 32. Convergence was reached when all gradient norms were below $\varepsilon = 10^{-5}$. All hidden layers were of width 128. As all MLPs were designed to perform multi-class image classification, each model comprised an output layer with softmax activation function. Weights were initialized with $\mathcal{N}(0, 0.0001)$ distribution and biases were initialized with Exponential distribution with parameter $\lambda = 0.01$.



MLP architecture	Optimal hyperparameters	Average validation accuracy (%)	Average training accuracy (%)
ReLu (depth 0)	$\alpha = 0.1$, 10K iterations	85.39	87.13
ReLu (depth 1)	$\alpha = 0.1$, 15K iterations	88.17	93.51
ReLu (depth 2)	$\alpha = 0.1$, 15K iterations	87.73	94.77
ReLu (depth 3)	$\alpha = 0.1$, 10K iterations	87.56	92.67
tanh (depth 2)	$\alpha = 0.1$, 10K iterations	87.65	94.45
leakyReLu (depth 2, 10K iterations)	$\alpha = 0.1, \gamma = 0.1$	87.48	92.68
ReLu with dropout rate $p > 0$ (depth 1, 10K iterations)	$\alpha = 0.1, p = 0.1$	87.68	94.12
ReLu with dropout rate $p > 0$ (depth 2, 10K iterations)	$\alpha = 0.1, p = 0.25$	87.88	93.03
ReLu activation with unnormalized data (depth 2)	$\alpha = 0.01$, 10K iterations	86.23	88.42
CNN		92.17	98.76

Figure 1: Optimal hyperparameters and corresponding accuracy for each MLP architecture

3.1 3 MLP models with ReLu activation

We first explored the performance of the standard MLP architecture with ReLu activation functions with number of hidden layers (depth) ranging from 0 to 2. We performed hyperparameter tuning on the learning rate $\alpha \in \{.001, .01, .1\}$ and the maximum number of gradient descent steps $m \in \{5000, 10000, 150000\}$. The resulting validation accuracies are shown in plots 2, 3 and 4. It is important to note that for the first two models we performed 5-cross fold validation and for the last one 2-cross fold validation because deeper models result in higher running times. The models achieved optimal validation accuracies of 85.39%, 88.17% and 87.73% respectively. For comparison, we trained a MLP model of depth 3 with learning rate $\alpha = 0.1$ and maximum 10000 iterations and obtained an average validation accuracy of 87.56, which is slightly worse than depth 1 and 2 models.

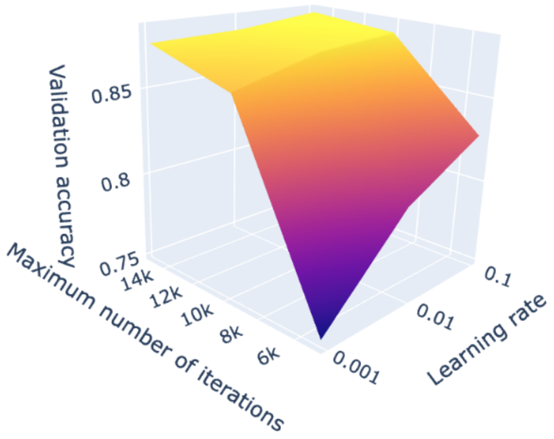


Figure 3: MLP with Relu activations and depth 1

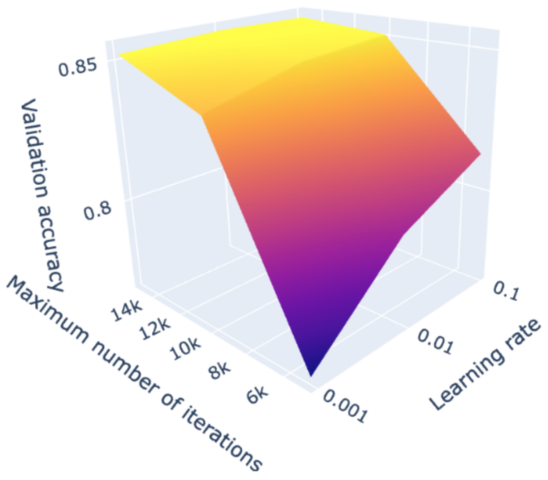


Figure 2: MLP with Relu activations and depth 0

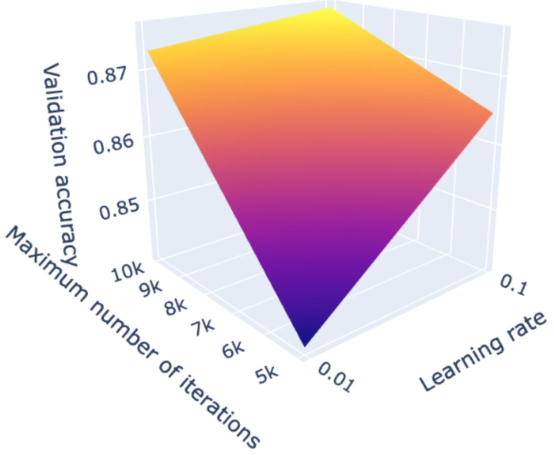


Figure 4: MLP with Relu activations and depth 2

First we see that all models performed best with the highest learning rate $\alpha = 0.1$. Also, the depth 0 model had both the lowest training and validation accuracy of all. This was expected since models of greater depth are more complex and have more expressive power. However, the MLP with 2 hidden layers overfit the data more than the one with a single hidden layer, since it yielded higher training accuracy but lower validation accuracy.

Although we expected the model with depth 3 to yield slightly higher training accuracy and lower validation accuracy than both depth 1 and 2, we were suprised to find it produced both lower training and validation accuracies. Perhaps the number of iterations used was too small to reach convergence on the depth 3 model. Moreover, the variance introduced by the small cross-validation number K might be at fault in these results. As we increased the depth, we observed more variations in accuracy, perhaps because greater depth might yield more local optima.

3.2 Tanh and Leaky ReLu activations

We then explored other activations functions for the MLP architecture such as the tanh and leaky ReLu functions with constant depth 2. For the tanh model, we performed hyperparameter tuning on the learning rate α and the maximum number of gradient descent steps m. For the leaky ReLu model, we fixed the maximum number of iterations to 10000 and performed hyperparameter tuning on the learning rate $\alpha \in \{.01, .1\}$ and the $\gamma \in \{.1, .5\}$ parameter of the leaky ReLu function. The resulting 2-fold cross-validation accuracies are shown in Figures 5 and 6.

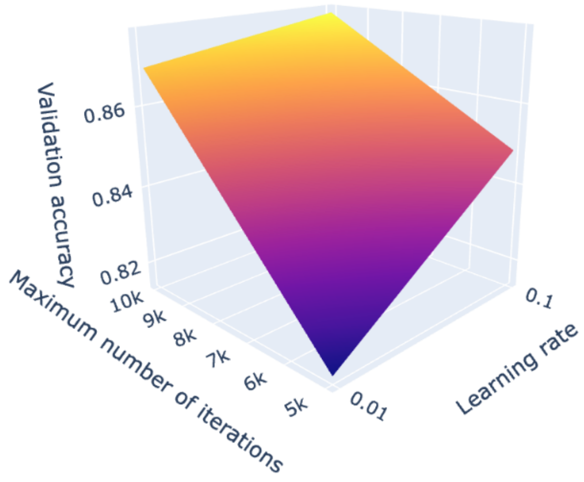


Figure 5: MLP with tanh activation and depth 2

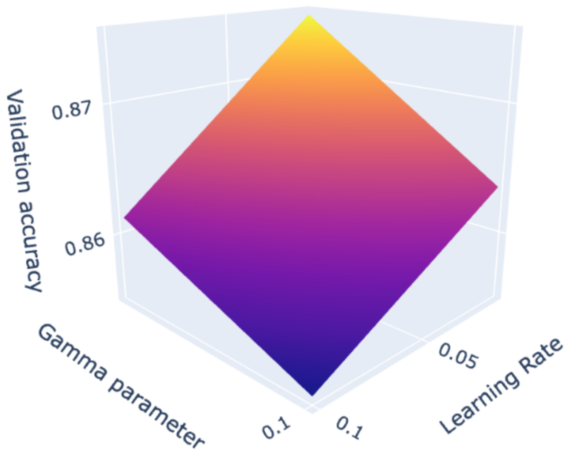


Figure 6: MLP with leaky ReLu activation and depth 2

The tanh model reached an optimal validation accuracy of 87.65% with hyperparameters $\alpha = 0.1$ and 10 000 iterations for a corresponding training accuracy of 94.45%. The leaky ReLu model reached a slightly lower optimal validation accuracy of 87.48 with hyperparameters $\alpha = 0.1$ and $\gamma = 0.1$, for a lower training accuracy of 92.68. We found that tanh performs slightly better then Leaky-ReLU. We also found that ReLu performed better than both of them. This is not that suprising since ReLu is the most widely used activation function, making it the default choice in many models.

3.3 Dropout regularization

We then explored the incorporation of weight dropout in the MLP architecture with ReLu activation and depths 1 and 2. We performed hyperparameter tuning on the learning rate α and the dropout rate $p \in \{.1, .25, .35, .5\}$ (corresponding to the proportion of connections set to zero in the network) with a maximum number of iterations of 10000. The resulting 2-fold cross-validation accuracies of the depth 2 model are shown in Figure 7.

For the model of depth 1, the optimal hyperparameters were dropout rate $p = 0.1$ and learning rate $\alpha = 0.1$, with validation and training accuracies of 97.68 and 94.12 on average. In this case, both accuracies were lower than the model of depth 1 without dropout.

For the model of depth 2, the optimal hyperparameters were dropout rate $p = 0.25$ and learning rate $\alpha = 0.1$, yielding a slight increase of the validation accuracy over the model of depth 2 without dropout from 87.73% to 87.88%, but a reduction in training accuracy from 94.77% to 93.03. This was expected, as dropout is a regularization strategy that reduces overfitting, thus reducing the training accuracy but increasing the validation accuracy. This was our second best performing MLP.

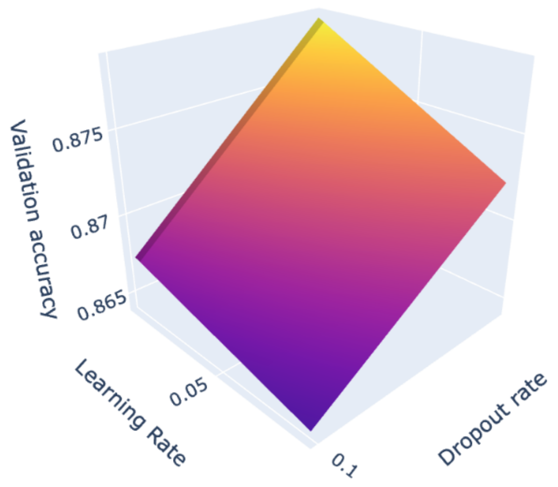


Figure 7: MLP with dropout and depth 2

3.4 Unnormalized images

Finally, we explored the effect of input normalization by training a MLP model with ReLu activations and depth 2 on unnormalized images. We performed hyperparameter tuning on the learning rate $\alpha \in \{.01, .1\}$ and the maximum number of iterations $m \in \{5000, 10000\}$. The best performance was achieved at $\alpha = 0.01$ and $m = 10000$. This was our second worst MLP model, with both validation and training accuracies being reduced from 87.73% to 86.23% and from 94.77% to 88.42% with regards to the normalized MLP with depth 2. This was expected, as normalization is necessary to smooth out the optimization landscape to help the gradient descent algorithm find the global optimum.

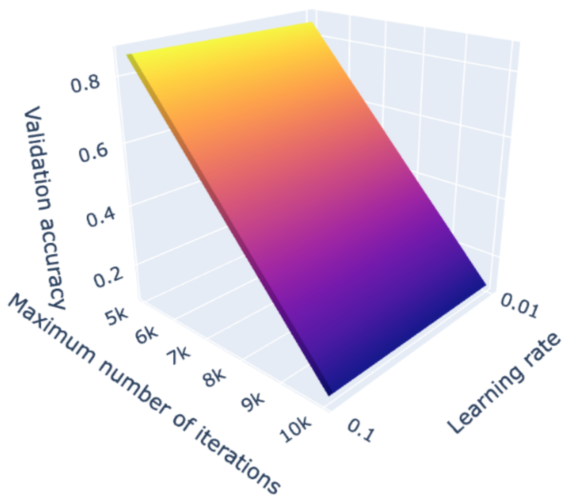


Figure 8: Unnormalized MLP model with depth 2

3.5 Convolutional Neural Network

We trained a CNN model with 2 convolutional and 2 fully connected layers. The two convolutional layers had a convolution window size of 3x3. We also used a pooling layer of size 2x2. We found that, even without tuning, the CNN model vastly outperforms all of our MLP models. We decided to train the model for 10 epochs. As shown in Figure 1, we obtained an extremely high training accuracy of 98.76%, for a final test accuracy of 92.17%. We expected the CNN to outperform the MLP model, because convolutional layers are specifically designed to use spatial relations in an image to extract more information than simple MLPs. This comparison will be further explored in the Discussion..

To illustrate the effect of training on the CNN model, we plotted the CNN test accuracy as a function of the number of training epochs, as shown in Figure 9. We observed that the test accuracy reached a plateau of approximately 92% after only 2 training epochs. Thus, the CNN reached convergence at faster than the MLP models.

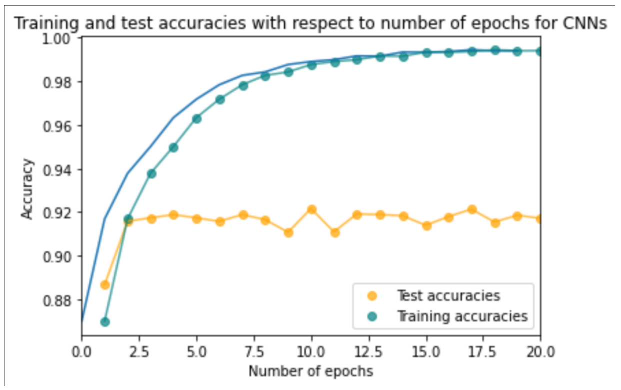


Figure 9: CNN performance

3.6 Optimal MLP

Our optimal MLP was chosen as the architecture with highest validation accuracy in Figure 1. We chose the following model: a 1 hidden layer, ReLu activation MLP without dropout, normalized data, batch size of 32 and a total of 15K gradient descent iterations. 15K gradient steps with batch size 32 corresponds to 8 epochs.

We then trained this model on the whole training set during 8 epochs with a training accuracy of 92.02%. We then ran this model on the test dataset (which had not been used before) to obtain a final MLP test accuracy of 87.66%. To illustrate the effect of overfitting, the test and training accuracy in terms of epoch size of this model are shown in Figure 10. As expected, we observed that the training accuracy generally increases as the number of training epochs grows, while the testing accuracy increases only slightly or not at all.

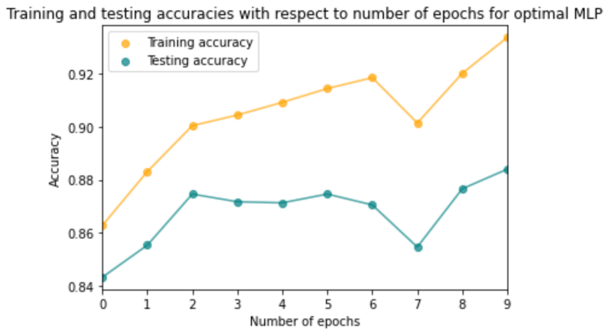


Figure 10: Optimal MLP performance

4 Discussion and Conclusion

A CNN corresponds to a special case of MLP with convolutional layers that consist of only local activations, meaning that a MLP is inherently a more complex model than CNN. And yet, we found that a CNN reached a test accuracy of 92.17% on the Fashion MNIST dataset, thus vastly outperforming even the optimal MLP model that reached a peak at 88.17% test accuracy. Thus, we obtained an increase of 4% in accuracy by choosing a CNN model over a MLP model.

In theory, while training an MLP it is possible by pure luck that it chooses the same local weights as a CNN model, and so it could reach a test accuracy as high as 92%. An explanation for why that does not happen might be that the mini-batch gradient descent optimization algorithm does not favour local weights.

All of the architecture choices that we explored such as normalization, dropout regularization and various activation functions only yielded marginal improvements in the MLP model. For a huge leap in improvement we have to be able to cut down the complexity of the hypothesis space drastically without hurting its performance. This is exactly what a CNN does. The inductive bias on which relies a CNN is that spatial relations in images can be used to guide classification. Clearly, it makes much more sense to view an image as a two-dimensional object rather than a one-dimensional vector. CNN use that heuristic, making a convolutional layer have less weights than a dense layer.

For further experiments, our code was designed in such a way as to permit us to train a MLP with any numbers of hidden layers, possibly greater than 2. Greater depths and varying widths could be explored to see if they yield improvements in accuracy on the MLP model. Furthermore, other version of gradient descent could be used to improve training times, such as Adam.

5 Statement of Contributions

The workload was evenly distributed among the 3 team members. Here are their respective contributions:

Dragos Secrieru: Worked on data pre-processing, built the CNN and contributed to the optimal MLP architecture. Wrote section 1 and 3 of the report and created the visual plots.

Antoine Bonnet: Implemented the Multi-layer perceptron, mini-batch gradient descent algorithm, feed-forward and backpropagations algorithms, hyperparameter tuning method and dropout regularization. Ran simulations and wrote section 4 and 3 of the report.

Cyril Saidane: Loaded data sets, worked on pre-processing, designed and ran some of the experiments. Wrote section 2 and 5 of the report.

References

[1] Bhatnagar, S., Ghosal, D., amp; Kolekar, M. H. (2017). Classification of fashion article images using Convolutional Neural Networks. 2017 Fourth International Conference on Image Information Processing (ICIIP). https://doi.org/10.1109/iciip.2017.8313740

[2] Agarap, A. (2017). An Architecture Combining Convolutional Neural Network (CNN) and Support Vector Machine (SVM) for Image Classification.

[3] Mingyu Bae, Optimizing the Hyper-parameters of Multi-layer Perceptron with Greedy Search, American Journal of Computer Science and Technology. Volume 4, Issue 4, December 2021 , pp. 90-96. doi: 10.11648/j.ajcst.20210404.11