COMP 551 - WINTER 2022

APPLIED MACHINE LEARNING

MiniProject 3 Report

Multiclass Classification on Fashion-MNIST with Multi-Layer Perceptron and Convolutional Neural Network

Group 71

George Qiao 260779462

Atia Islam 260737946

Vraj Patel 261022581

School of Computer Science, McGill University

April 5, 2022

Abstract

In this project we compared the performance of MLP and CNN on Fashion-MNIST dataset [1] in a multi-class classification task. We explored different network architecture for both of the models and extensively tuned their hyperparameters. Our best self-implemented MLP model achieves 89.3% test accuracy while for CNN model suing TensorFlow framework we managed 90.0%. We found that having the right amount of model complexity is critical, that tanh is the best activation function, and that regularization techniques such as dropout is effective in improving training efficiency and reducing generalization error.

Introduction

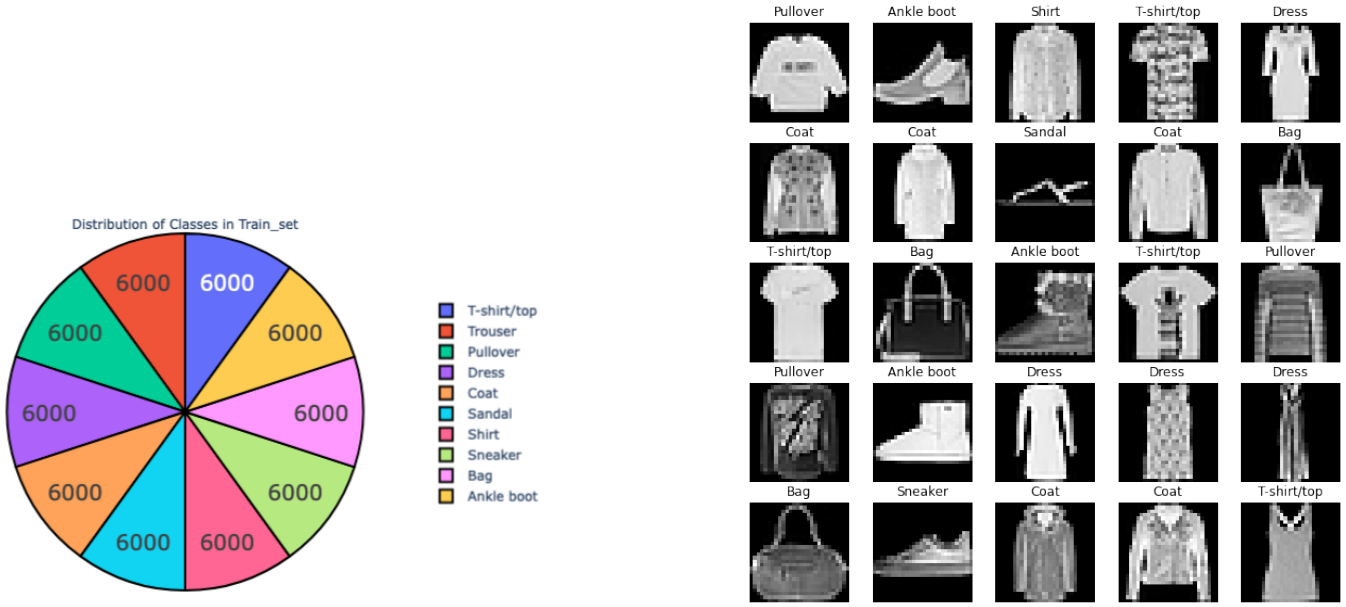
The goal of this project is to implement from scratch a multi-layer perceptron (MLP) classifer on the Fashion-MNIST dataset and investigate the effect of its various hyperparameters. The resulting model will also be compared to an implementation of convolutional neural network (CNN) using the TensorFlow deep learning framework. The key aspect of this project is tuning the variety of hyperparameters in MLP and CNN, which include network architecture, activation function, learning rate, and ones related to regularization techniques like dropout. Our final MLP and CNN models both achieved satisfactory test accuracy, with CNN slightly edging out MLP (90.0% vs. 89.3%).

Dataset

Fashion MNIST-dataset is presented by Zalando Research (Xiao et al., 2017 [1]) as an upgrade from the original MNIST dataset. It consists of 60,000 28x28 greyscale images in the training set and 10,000 in the test set. Each label is one out of 10 classes like MNIST. We applied normalization on the input images for getting the dimensions so that each pixel lies in [0, 1] interval, while also saving a copy of unnormalized inputs for comparison. We applied the one-hot encoding technique for the labels to allow for the use of 'softmax' function. Our train-set has balanced classes (see Figure 1a).

Ethical Implications

Fashion-MNIST is a popular benchmark dataset for multiclass classification. There are no ethical concerns in using this dataset for testing model performance itself, but one must be aware of the limitation of such testing (e.g. high performance on Fashion-MNIST does not necessarily transfer onto other real world datasets which can be more complex).



(a) Class distribution

(b) Some samples

Figure 1: Fashion-MNIST Dataset

Results

Due to the shear amount of computation required, we did not tune every hyperparameter extensively nor did we manage to run the models for too many epoches (many models could use more training to improve performance). We hereby acknowledge these limitations.

Due to limited space, please check ”MLP_final.ipynb” for all plots on MLP training and performance.

Gradient descent methods, batch sizes and learning rates

We tested our model for different optimizer on ReLU with two hidden layers each having 128 units, but other than Adam no other optimizer converges faster and taking longer time to give results, so here we present only Adam results with $\beta_1 = 0.9$ and $\beta_2 = 0.99$.

For two hidden layer MLP with ReLU as an activation function we tried for small and large batch sizes and found out that for the batch size of 10000 our MLP performs really well if we increase or decrease the batch size from this the performance changes drastically. Also, the case for different learning rates, we found out that at lower learning rates such as 0.01 and 0.001 we got higher accuracy but if we increase learning rate to 0.015 our error increases drastically for the same batch size for each experiments.

Number of hidden layers, activation function, dropout regularization, and normalized vs. unnormalized

The results of the following MLP experiments are summarized in Figure 2. For each experiment, all hyperparameters other than the one investigated are kept constant.

ReLU activation	No layers	One layer	Two layers
Train accuracy	87.3	93.8	92.5
Test accuracy	85	88.4	87.7

Learning rate = 0.01 number of iterations = 100 tolerance=0.0001 batch size = 10000

(a) Varying number of hidden layers

2 hidden layers	ReLU	tanh	Leaky ReLU
Train accuracy	92.5	95.1	94.1
Test accuracy	87.7	87.7	88.1

Learning rate = 0.01, number of iterations = 100, tolerance = 0.0001
batch size = 10000, No. of units in the middle layers = 128

(b) Varying the activation function

ReLU activation	Without Dropout Regularization	Without Dropout Regularization	
		rate = 0.8	rate = 0.7
Train accuracy	92.5	90.8	88.7
Test accuracy	87.7	87.8	86.6

Learning rate = 0.01, number of iterations = 100, tolerance = 0.0001 batch size = 10000
No. of units in the middle layers = 128

(c) Using dropout regularization

ReLU activation	Normalized learning rate = 0.01	Unnormalized learning rate = 0.001
Train accuracy	92.5	93.9
Test accuracy	87.7	88.2

number of iterations = 100, tolerance = 0.0001 batch size = 10000,
No. of units in the middle layers = 128

(d) Using unnormalized images

Figure 2: Summary Tables

We implemented no-layer, one layer and two layer MLP with ReLU activation function and keeping all other hyperparameters constant. MLP with one and two layers performs slightly better compared to no layers. However, one layer MLP model performs a little better if we compare it with two layers, probably due to insufficient training on the two layer model. Increase in network depth is supposed to make the models more expressive (and thus more prone to overfitting). We do see a larger gap between test and train accuracy.

We experimented with different activation functions for the two layer MLP with 128 units at each layers having the same parameters. Results shows that all the activation functions performs nearly the same but Leaky ReLU performs slightly better compared to the other two. ReLU due to its zero value and gradient on the negative side performs slightly worse but trains significantly better due to feature selection

Overall, with dropout regularization, model test accuracy is not affected by very much, but the models train slightly better and seem to have less overfitting. In addition, we repeated the tests with different activation functions and same phenomenon was observed.

For unnormalized images we need to lower our optimized learning which we used in normalized images because if we use the same learning rate weight updated at each and every iteration is higher which

was causing divergence. Hence, we used 0.001 as a learning rate which led to convergence and similar performance as using 0.01 learning rate on normalized images. Lowering initialized weights also helps with reducing the effect of feature scaling.

CNN models using TensorFlow

A typical ConvNet takes input image, pass it through a set of layers consisting of convolution, pooling and fully-connected layers [2]. For our experiment, our base model had 2 convolutional and 2 fully-connected layers with ReLU activation for all layers except 'softmax' at output layer. We decided to use 32 filters for first two layers, 64 filters for the second convolutional layer and used kernel size (3,3). We used batch normalization to solve a major problem called internal covariate shift, allowing a higher learning rate. Dropout is applied. The final network architecture is shown in the Appendix.

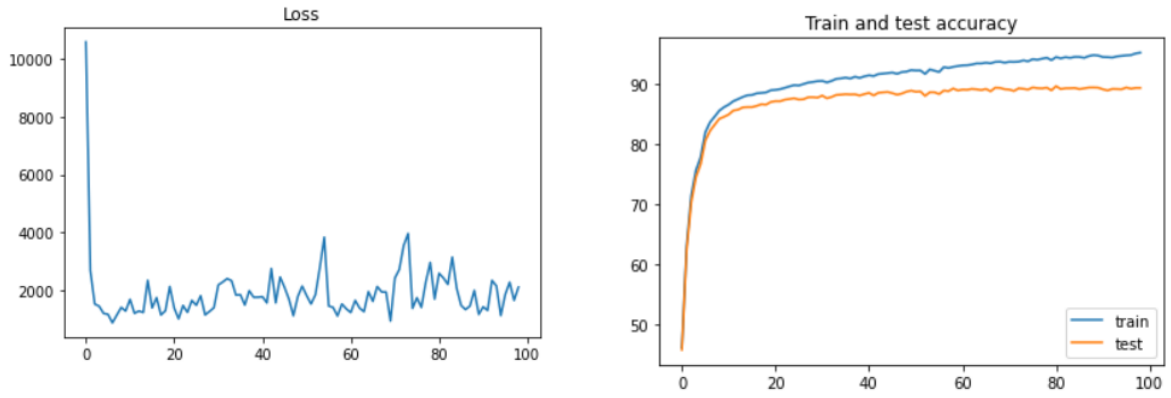
After tuning, we ended up using 256 batch size and max epoch of 50. We used Adam like we did in MLP to improve training. We used 25% dropout for the convolution layers and 50% for fully-connected layers. We also experimented with adding l2-regularization and bias regularization of 0.01. We briefly experimented with data augmentation as a regularization technique. Using ImageDataGenerator, we applied random rotation, shift, shear and flip to the dataset for richer data experience for our model to train and better generalization. Finally, we obtained a test accuracy of 90.0%, which is slightly better than all our previous MLP models. It is as expected since CNN usually performs better on large image datasets since convolutional layers can capture correlation between pixels better.

Please see the Appendix for the detailed CNN network architecture and more explorations of CNN tuning.

Best performing MLP model

We ran different experiments on MLP with different activation functions, learning rate, keep rate for dropout regularization and different architectures. We found that the best performing MLP model is the one with two hidden layers, tanh activation function, Adam as the optimizer, and using dropout regularization.

The hyperparameters for this model are: First layer 256 units, second layer 128 units, learning rate 0.01, keep rate 0.75. This achieved a test accuracy of 89.3%. Figure 3 shows the evolution of loss and train/test accuracy over 100 epochs.



(a) Loss (b) Train and test accuracy

Figure 3: MLP optimal model training and performance

Discussion and Conclusion

Key takeaways from the project:

- Compared to MLP, CNN is more well suited for image classification task on large datasets; CNN slightly improves performances over MLP.
- Hyperparameter tuning is key for both MLP and CNN training.
- MLP is more difficult and slower to converge than CNN which makes its hyperparameter tuning difficult and tedious
- Adam does seem the best gradient descent optimizer and has its place as the infustry standard.

Possible directions for future investigation:

- Investigate the effect of adding more hidden layers in MLP on performance.
- Compare MLP and CNN for regression tasks.
- Compare MLP and CNN for other non-image classification tasks.
- Adding pooling, striding, etc. in CNN.

Statement of Contribution

Atia Islam: Data normalization, CNN, write up

George Qiao: MLP implementation, write up

Vraj Patel: MLP implementation, write up

References

[1] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.

[2] K. Meshkini, J. Platos, and H. Ghassemain, "An analysis of convolutional neural network for fashion images classification (fashion-mnist)," in *International Conference on Intelligent Information Technologies for Industry*, pp. 85–95, Springer, 2019.

Appendices

Additional Figures

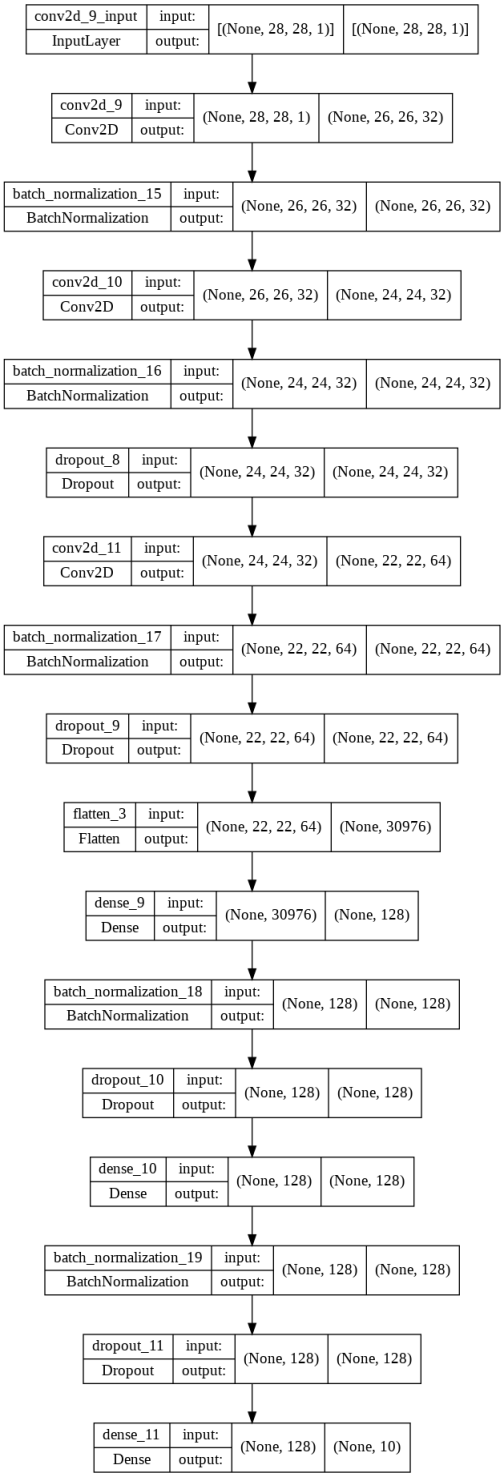
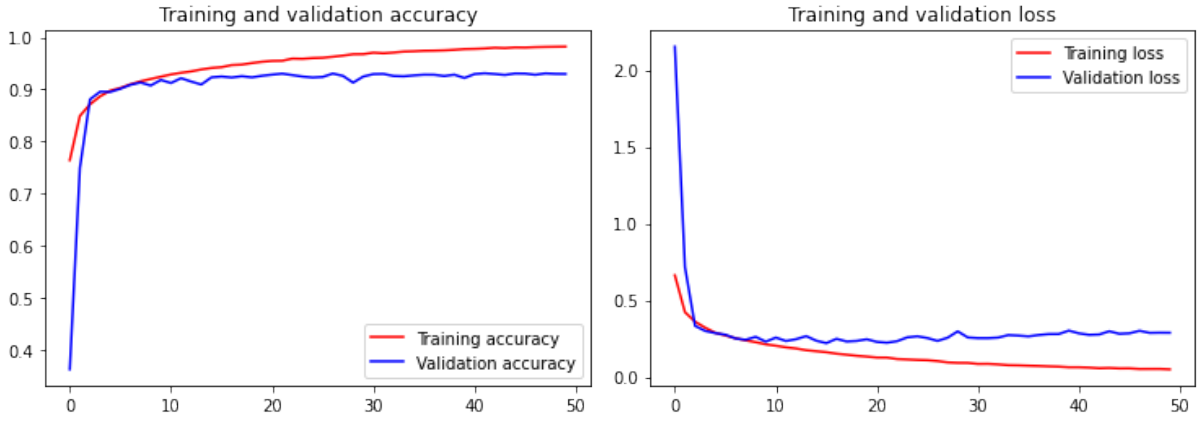


Figure 4: CNN architecture



(a) accuracy vs epoch

(b) loss vs epoch

Figure 5: Training (red) and test (blue) accuracy and loss for best CNN model

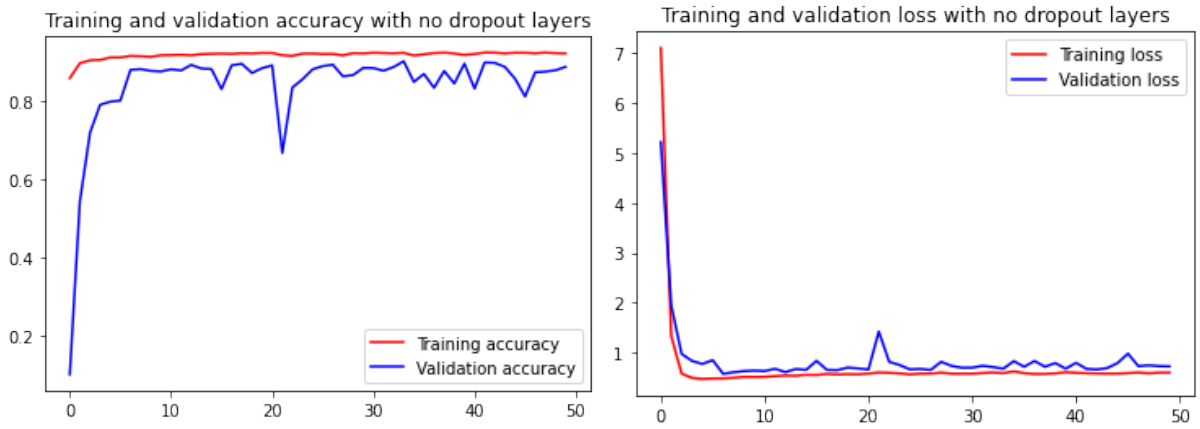
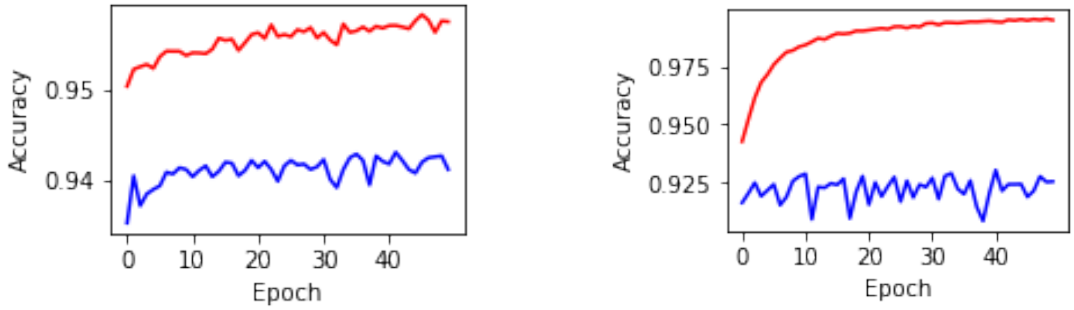


Figure 6: CNN training and performance, no dropout layers



(a) SGD

(b) RmsProp

Figure 7: CNN Training (red) and Validation (blue) accuracy with SGD and RmsProp optimizers.