Introduction to Machine Learning Project Report (MS2)

Arda Çınar Demirtaş, 384868 - Hojoon Kim, 385098 - Yiğit Narter, 384870

1. Introduction

This project aims to implement several machine learning methods to classify images from the FashionMNIST dataset, which contains grayscale images of various fashion items. Our objective is to develop and evaluate the performance of multiple neural network architectures, including Multi-Layer Perceptrons (MLP), Convolutional Neural Networks (CNN), and Transformer models. Additionally, we will utilize Principal Component Analysis (PCA) to reduce the dimensionality of the dataset and assess its impact on MLP performance.

2. Method

2.1 Data Preparation

We implemented a validation set and principal component analysis (PCA) for data preparation. When the argument "test" is not set, which means it's not a test, we create a validation set from the training set. We also added an argument as "val_ratio" that determines the ratio for the validation set.

To reduce the dimensionality of the dataset, principal component analysis (PCA) was implemented. The process involved computing the mean of the data, centering the data by subtracting the mean, and creating the covariance matrix. Eigenvalues and eigenvectors of the covariance matrix were then calculated, and the top 'd' eigenvalues (the value is taken from the user) and corresponding eigenvectors were selected. The data was projected onto this new subspace to reduce its dimensionality, while also the explained variance is computed, which is a measure of how much of the information is retained when projecting the data. This method is to be used for MLP.

Also, for CNN, standard min-max normalization was applied to the data as it was observed to be better for the model's performance.

2.2 Methodology

We implemented a Vision Transformer (ViT). The model divides images into patches, which are then flattened and embedded into vectors. A classification token is added, and positional embeddings are applied, here we chose learnable positional embeddings instead of sinusoidal embeddings. The ViT consists of transformer blocks, each with layer normalization, multi-head self-attention, and feed-forward neural networks, incorporating dropout and residual connections. The main hyperparameters include the number of patches, transformer blocks, hidden dimension size, and attention heads. The chosen hyperparameters are as follows: learning rate: 10^{-4}, epochs: 20, number of patches: 4, number of blocks: 8, hidden dimension: 64, and number of heads: 4.

We also implemented Multi-Layer Perceptron (MLP). The MLP model was implemented using PyTorch and designed to classify the images after they were flattened into 1D vectors. The architecture consists of an input layer, several hidden layers with ReLU activation functions, and an output layer with a softmax activation function for classification. The layers are fully connected, meaning each neuron in one layer is connected to every neuron in the next layer. The input layer takes input vectors of size 784 (28x28 pixels flattened). As an addition, we added extra parameters to the MLP class such as the hidden layers' size and number, also activation function type (e.g. ReLU or sigmoid), all of which can be chosen by the user through the command line.

Convolutional Neural Network (CNN) implementation consists first of four consecutive convolutional layers, each with 3x3 filters and the number of filters beginning with 32 and increasing twofold through the first three layers up to 128. Also, each convolutional layer is followed by a max pooling layer. Then, the resulting tensor is flattened to be passed through two fully connected layers as this is a classification task. Batch normalization and dropout layers were implemented sparsely to ensure faster convergence and prevent overfitting.

For every model, regularization techniques like dropout and layer normalization are employed to enhance generalization. The Adam optimizer is used for training.

2.3 Hyperparameter Search and Visualization

Another addition was to visualize the hyperparameter evaluations for all three methods. We added an argument as "plt" that enables this visualization.

To achieve optimal performance, various hyperparameters such as learning rate, batch size, number of epochs, and model-specific parameters (e.g., number of blocks or heads) are tuned.

For the three models, our current implementation iterates learning rates in a logarithmic scale. However, we can easily modify the plotting code to test other hyperparameters.

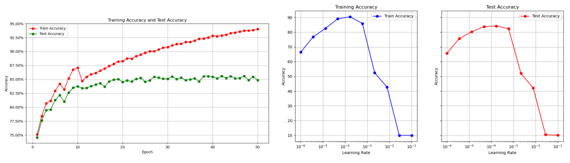
For the other hyperparameters that are not plotted such as number of hidden layers, activation function, batch size, or dropout rate, when architectures were in the design phase the best values for most of them were decided on empirically (manual search), then they were verified via grid search as in learning rate.

3. Experiment and Results

We conducted an evaluation of the methods on the FashionMNIST dataset.

Given the extensive hyperparameter space for the ViT, we tested numerous combinations and identified the optimal values for reduced training time and enhanced accuracy. We fixed the hyperparameters and subsequently tested how each parameter individually affects performance.

To determine the appropriate number of epochs and learning rates, we varied epochs from 1 to 50, learning rates from 10^{-6} to 10^{-1} and visualized the accuracy.

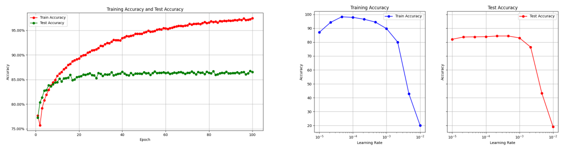


[Figure 1&2: Accuracy by epochs & lrs for ViT]

As depicted in Figure 1, the training accuracy improves continuously with an increasing number of epochs. However, the test accuracy plateaus at approximately 85% after 20 epochs, indicating that beyond this point, the model begins to overfit rather than generalize. Consequently, we selected 20 epochs as it strikes a balance between training time and model performance.

Figure 2 illustrates that learning rate of 10^{-4} yields the best performance for both the training and test datasets. A smaller learning rate results in prolonged training times, while a larger learning rate hinders fine-grained learning. Therefore, 10^{-4} was chosen as the optimal learning rate.

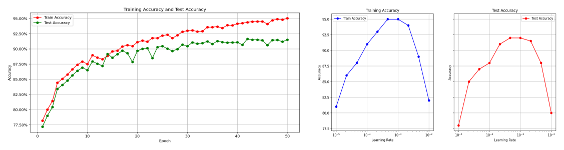
The MLP model also involves numerous hyperparameters. After running many tests to determine the optimal ones for accuracy and speed, we selected the activation function as ReLU, the batch size as 128, the number and size of the hidden layers is 5 and 512 respectively. Similar to ViT, the accuracy for MLP was tested first by iterating over a range of learning rates, while the epoch number is 25. To plot the MLP with a fixed value of the learning rate, we varied the number of epochs between 1 and 100.



[Figure 3&4: Accuracy by epochs & lrs for MLP]

The plots show that the best performance is again observed around 10^{-4}, hence it is chosen as the optimal learning rate for MLP. With the learning rate fixed, it can be seen that after 25 epochs, the model starts overfitting the training data while the test accuracy remains roughly the same. Therefore a learning rate of 10^{-4} and 25 epochs, along with the other fixed hyperparameters yielded a satisfactory result.

The chosen hyperparameters for CNN are a learning rate of 10^{-3}, a batch size of 64, dropout rates of 0.3, 0.4 and 0.25, and filter numbers of 32, 64, and 128. The max pooling had a size of 2x2 and a stride of 2. Each layer has "same" padding to preserve the size of the image before pooling. The model was run for 50 epochs. As can bee seen from figure 5, The train and test accuracy both increase together, with train accuracy increasing faster. Figure 6 shows optimal learning rate around 10^{-3}.



[Figure 5&6: Accuracy by epochs & lrs for CNN]

4. Discussion and Conclusion

4.1 Performance

In conclusion, Tables below show the performance for the training set and the validation set of our final model with optimal hyperparameters.

Metric	MLP	CNN	ViT
Acc.	93.800% (eps:25 | lr:10^{-4})	95.786% (eps:50 | lr:10^{-3})	91.636% (eps:20 | lr:10^{-4})
F1	0.937748 (eps:25 | lr:10^{-4})	0.958884 (eps:50 | lr:10^{-3})	0.915893 (eps:20 | lr:10^{-4})
Elapsed time	478.9381 sec	2540.03 sec	3486.6085 sec

[Table 1: training set performance]

Metric	MLP	CNN	ViT
Acc.	86.022% (eps:25 | lr:10^{-4})	91.807% (eps:50 | lr:10^{-3})	84.944% (eps:20 | lr:10^{-4})
F1	0.858227 (eps:25 | lr:10^{-4})	0.91345 (eps:50 | lr:10^{-3})	0.848949 (eps:20 | lr:10^{-4})
Elapsed time	3.9037 sec	18.9732 sec	18.8582 sec

[Table 2: validation set performance]

After the determination of the optimal parameters, we also tested how the PCA affects the MLP performance by trying out different values for d (new dimensions of the reduced data, which are normally vectors with size 784 for MLP input layer). Table 3 shows the results after applying PCA.

Metric	d=5	d=10	d=20	d=50	d=100
Train Acc/ time	70.374% 310.19 s	80.511% 326.21 s	86.250% 318.91 s	90.772% 319.53 s	92.567% 347.14 s
Valid. Acc/ time	67.967% 2.12 s	77.429% 1.95 s	82.219% 2.02 s	83.733% 2.53 s	85.274% 2.36 s
Exp. var.	55.49%	65.89%	74.21%	83.11%	89.01%

[Table 3: PCA performance for MLP]

4.2 Model Comparison

Comparing the accuracies both on the training and validation sets, it can be seen CNN model is the most effective model for classifying images in the FashionMNIST dataset, thanks to their ability to extract and utilize spatial features efficiently. In terms of speed, MLP outscores the other two with the shortest elapsed times for training and prediction, due to the simplicity of its architecture and the absence of computationally intensive operations like convolution. With the PCA applied, the MLP's speed increases even more, while the accuracy decreases, demonstrating the trade-off between computational efficiency and accuracy. As d is increased, the accuracy also increases due to more components capturing the most variance and preserving the information (as shown by the explained variance), while the training and prediction speed decreases. With d=100, the accuracy is very close to that without PCA, with shorter elapsed time. The ViT model, which is the slowest and has the least accuracies despite their potential, requires more computational resources and longer training times due to its complexity.

Comparing the generalizability of the models, looking at figures 1, 3, and 5, it can be seen that CNN is the one that generalizes the best. After some time, the test accuracies of MLP and ViT hit a plateau while the training accuracy keeps increasing, whereas in CNN both accuracies increase at very close rates, meaning the MLP and ViT overfit after a while. The success of CNN in this metric can be attributed to a better usage of dropout and batch normalization layers.

4.3 Discussion

We experimented with other hyperparameters. Here, we will describe the hyperparameters that we found particularly interesting. For ViT, MLP and CNN, we examined the batch size. We observed that smaller batch sizes tended to enhance performance. Larger batch sizes appeared to make it difficult for the model to learn the information of each token effectively. We believe this is due to the low resolution of the FashionMNIST dataset, which is 28x28 pixels.

For MLP, choosing sigmoid or tanh as the activation function decreased the accuracy while also requiring more time. For CNN, increasing the number of filters for each layer twofold increased the expressivity of the neural network and contributed positively to accuracy.

4.4 Summary

Overall on the validation set, CNN performed the best with the highest accuracy while also becoming the second fastest model. When we submitted to run with the test data, the test accuracy was 92.2% and the F1 score was 0.914. Due to providing the best performance for image classification in this context, we chose CNN as the best method. Future work to improve the model could involve more hyperparameter tuning and data augmentation.