Introduction to Machine Learning Course Project

Milestone 2

by

Oussama Ghali[*] & **Zachary Doll**[†]
Supervised by Professor. Mathieu Salzmann

Submitted June 2, 2024

[*]Student ID: 341478, Email: oussama.ghali@epfl.ch
[†]Student ID: 356458, Email: zachary.doll@epfl.ch

Introduction

This report explores deep learning techniques for classifying items in the Fashion-MNIST dataset. We implemented and evaluated three models: Transformer, Multilayer Perceptron (MLP), and Convolutional Neural Network (CNN), using PyTorch. Additionally, Principal Component Analysis (PCA) was applied to the MLP model to assess the impact of dimensionality reduction on performance and efficiency. Our objective is to compare the effectiveness of these models and the role of PCA in fashion item classification.

Method

Data Preparation

The Fashion-MNIST dataset, consisting of 70,000 grayscale images of 10 categories, was split into 60,000 training and 10,000 test images, each 28x28 pixels. We normalized pixel values to [0, 1] and shuffled the data for unbiased evaluation. Hyperparameter tuning, optimizing parameters like learning rate and layer count, was essential for enhancing model performance. Through systematic experimentation and random sampling, we identified the best models for classifying Fashion-MNIST items.

PCA

Principal Component Analysis (PCA) was used to reduce data dimensionality while preserving significant variance by transforming the data to a lower-dimensional space, projecting it onto directions that maximize variance. For tuning PCA, we experimented with different numbers of principal components to find the optimal balance between dimensionality reduction and explained variance. We tested several configurations and selected the one that provided the highest accuracy on the validation set while retaining significant variance. This method effectively minimized information loss and significantly improved computational efficiency when applied to the MLP model.

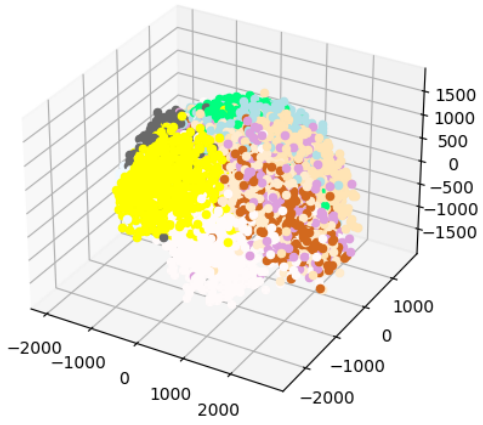


Figure 1: PCA Visualization of Test Data

Transformers

Transformers use self-attention mechanisms to capture long-range dependencies in data. Each block includes self-attention layers and feed-forward neural networks. We trained the Transformer model using backpropagation and gradient descent. Due to time constraints, we tested 25 hyperparameter combinations using random sampling, including variations in the number of blocks, hidden dimensions, attention heads, and learning rates. This approach allowed us to efficiently optimize the model's performance.

MLP

The Multilayer Perceptron (MLP) is a classic neural network that takes a vector as input. Each neuron applies a weighted sum to its inputs, which is then passed through an activation function. We trained the MLP using backpropagation and gradient descent, minimizing the loss function by adjusting weights and biases. Hyperparameters were optimized through cross-validation to find the best configuration. We tested 20 random parameter combinations, namely the number of layers, hidden layer dimensions, and learning rate.

CNN

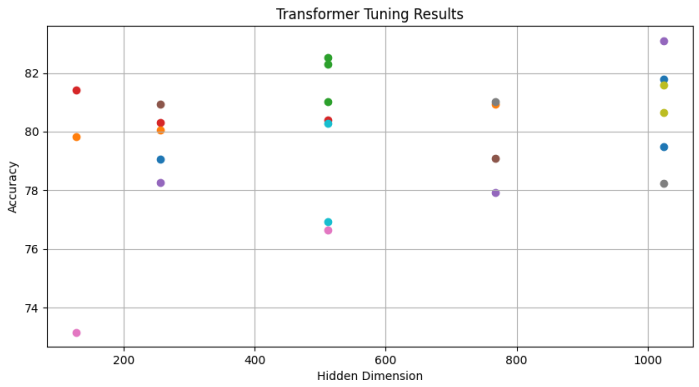
The Convolutional Neural Network (CNN) architecture includes convolutional layers with ReLU activation and max pooling. Configured with input channels and filters of sizes 64, 128, and 256, the network increases depth and feature extraction. The model, containing 962,314 parameters, was trained with a batch size of 32 using the ADAM optimizer (learning rate 0.001) over 50 epochs. To prevent overfitting, a dropout method with a 0.3 dropout rate was employed. Data augmentation techniques, such as random horizontal flips, rotations, and random crops with padding, were used to enhance data diversity. For tuning, we experimented with different filter sizes, learning rates, and iterations, using random sampling to test 20 parameter combinations.

Experiment/Results

We present visualizations of the model performances with their respective hyperparameters and discuss the results obtained. For each model, random sampling was used to test 20-25 parameter combinations due to time and computational constraints. All training of our models were performed on Google colab's TPU v2 and GPU T4.

Transformers

The best configuration for the Transformer model was 2 blocks, 1024 hidden dimensions, 4 heads, and a learning rate of 0.0001, achieving an accuracy of 83.108%. Training the Transformer model took $\approx$ 54 seconds, and tuning the hyperparameters took $\approx$ 18 minutes and 46 seconds.



MLP

Principal Component Analysis (PCA) was applied to the MLP model to reduce data dimensionality while preserving significant variance. With 106 components, we captured approximately 90% of the cumulative explained variance, reducing dimensions from 1024 to 106 and significantly improving computational efficiency (Fig.3).

Experiments compared the performance and training time of the MLP model with and without PCA. Without PCA, training took ≈ 1 min 19 s, with a total parameter tuning time of around 30 minutes (T4 GPU), achieving 84.800% accuracy. With PCA (106 components), training took ≈ 18 seconds, with a total tuning time of 13.5 minutes (TPU V2), achieving 85.442% accuracy.

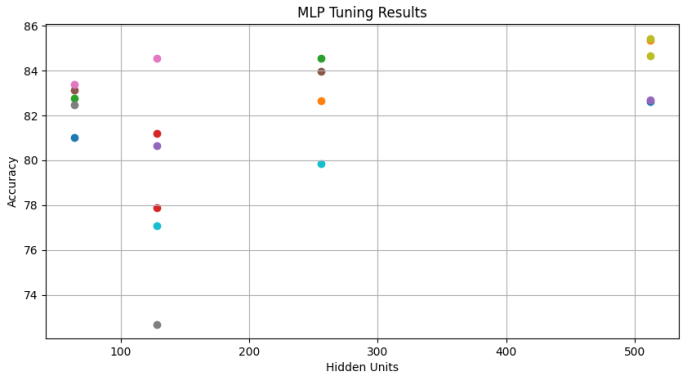


Figure 2: MLP random sampling tuning

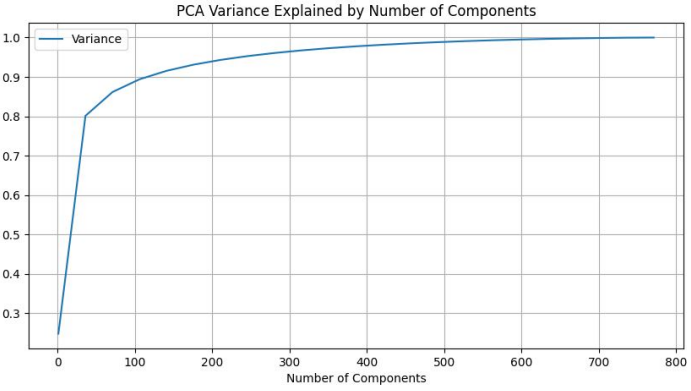
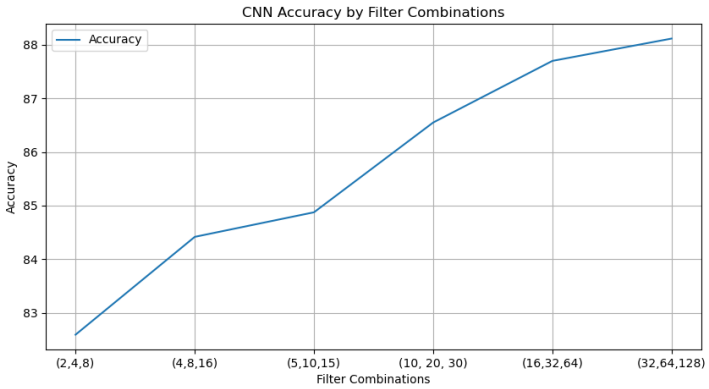


Figure 3: PCA analysis results

While applying PCA significantly reduced training time, as expected, it did not seem to have any significant impact on the accuracy of the model.

CNN

Optimizing the CNN hyperparameters initially involved iterative experimentation, focusing only on filter sizes.



The graph shows accuracy variations with different filter combinations from a deprecated version of our experiments. We then experimented on optimizing filter sizes, learning rates, and iterations combinations. Although we did not generate a new graph due to time constraints, testing 20 random parameter combinations, we found that using a (64, 128, 256) filter configuration, 5e-4 learning rate, max iterations of 20 and dropout of 0.3, we achieved 96.175% (90.4% test) accuracy and an F1 of 0.962 (0.912 test) on our training set.

Key findings from hyperparameter tuning include:

- The number of convolutional layers and filter sizes significantly impact final predictions.
- The best filter combination was (64, 128, 256) with a learning rate of 0.0005 and max iterations of 20.

Total training time with the best filter combination was approximately 6 minutes and 13.619 seconds.

Discussion And Conclusion

This study evaluated MLP, CNN, and Transformer models for classifying fashion items in the Fashion-MNIST dataset. PCA reduced data dimensionality effectively. The MLP achieved 85.442% accuracy, the CNN 93.942%, and the Transformer 83.108%, making CNN the best performer.

We used random sampling for hyperparameter for quick and efficient results. More robust methods like grid search, or optimizing batch size and activation functions may have also enhanced results and performance.

We developed high-accuracy classifiers, balancing performance and computational efficiency, highlighting the importance of hyperparameter tuning and dimensionality reduction in improving deep learning models for image classification.