REPORT :

This report presents the methodology, experiments, and results of training and evaluating different neural network architectures, including Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), and Transformer models, for image classification tasks.

CNN :

Method

We tried 2 different *architectures*. One inspired by LeNet (2 Conv. Layers & 3 FC layers, see *Image 1*) and the other inspired by the CNN exercise (3 Conv. Layers & 2 FC Layers, see *Image 2*). Both use cross entropy loss, as usual for classification. Both were first tested on basic hyperparam. values. For the better one we tried different optimizer and learning rate combinations, as well as different channels per layer.

Results

* All values for CNN were measured on the validation set, which contains **a third** of the **shuffled dataset**, after 10 epochs of training.

LeNet: Accuracy = 58% in 44s

Inspired by exercise: See *Table 1* and *Table 2*

Notes **:** 1. Both Batch- and Z-Normalization didn't really influence the values below.

2. Increasing epochs to 20 for the best model (*Table 2*) didn't change the values much.

Discussion

LeNet yields significantly worse results than the other architecture. Adam seems to be the better choice for the criterion. While LeNet seems to be too shallow (not good results / not enough epochs), the one inspired by the exercise does better, the more channels we add. This can be seen by comparing values of *Table 1* and *Table 2*. The fact that increasing the epochs on the best model doesn't change its performance and that the model performs better on the training set than on the test set indicates that we are overfitting on the training data. This could potentially be counteracted by reducing the complexity of the model.

TRANSFORMER :

Method

The transformer model used in this project is based on the Vision Transformer (ViT) architecture. The methodology involved several key components to process and classify images. First, the input images were divided into smaller patches, each treated as a token. These flattened patches were then mapped to the hidden dimension size using a linear mapper. A learnable classification token was added to the sequence of patch tokens, and positional embeddings were incorporated to retain positional information. To identify the optimal configuration, we try various hyperparameters, including different learning rates, batch sizes, and the number of epochs.

Results

After varying the hyperparameters: the learning rate of 1e-2 was the most effective, balancing the speed of convergence and stability during training. Increasing epoches take a lot of time to train but have better results, same for the number of batch sizes.

Discussion

As we can see from the table, the test accuracy and f1 score of the transformer model were both influenced by the learning rate, number of epochs, and batch size. A high learning rate caused overshooting, while a low rate slowed convergence; the optimal rate balanced speed and precision. Too few epoches led to underfitting, and too many led to overfitting; the optimal number avoided both. Small batch sizes caused noisy gradients, and large sizes led to poor generalization; the optimal size balanced efficiency and gradient quality. We finally managed to reach ≈ 83% of accuracy.

MLP & PCA:

Method

The MLP architecture that we used for this project consisted of three fully connected layers: an input layer matching the number of input features(the argument input_size), two hidden layers with ReLU activation, and an output layer corresponding to the number of classes . The key hyperparameters were selected through experimentation, resulting in a learning rate of 1e-5 and 100 training epochs. And also the loaded FashionMNIST dataset was split into training and test sets, with images flattened into vectors for compatibility with the MLP input. For PCA, the number of principal components was set to 100, retaining a significant portion of the variance.

Results

MLP without PCA

The MLP without PCA had 235,146 parameters. It achieved a training accuracy of 92.19% and a validation accuracy of 84.240% with a final F1-score of 0.842665 on the validation set. The training time was approximately 89.45 seconds for 100 epochs. (ref. table 4)

MLP with PCA

The PCA-reduced MLP had 60,042 parameters. It achieved a training accuracy of 89.06% and a validation accuracy of 83.475%, with a final F1-score of 0.834067 on the validation set. The training time was significantly reduced to approximately 68.47 seconds for 100 epochs. (ref. table 4)

Discussion

Applying PCA before training the MLP significantly reduced the number of parameters and training time while only slightly decreasing the accuracy and F1-score. This can be considered as a trade-off between computational efficiency and model performance and the results suggest then that PCA is an effective dimensionality reduction technique, particularly useful when computational resources are limited. While the MLP without PCA achieved higher accuracy, the PCA-augmented MLP offered a faster and more efficient alternative with a very small performance reduction.

Annex

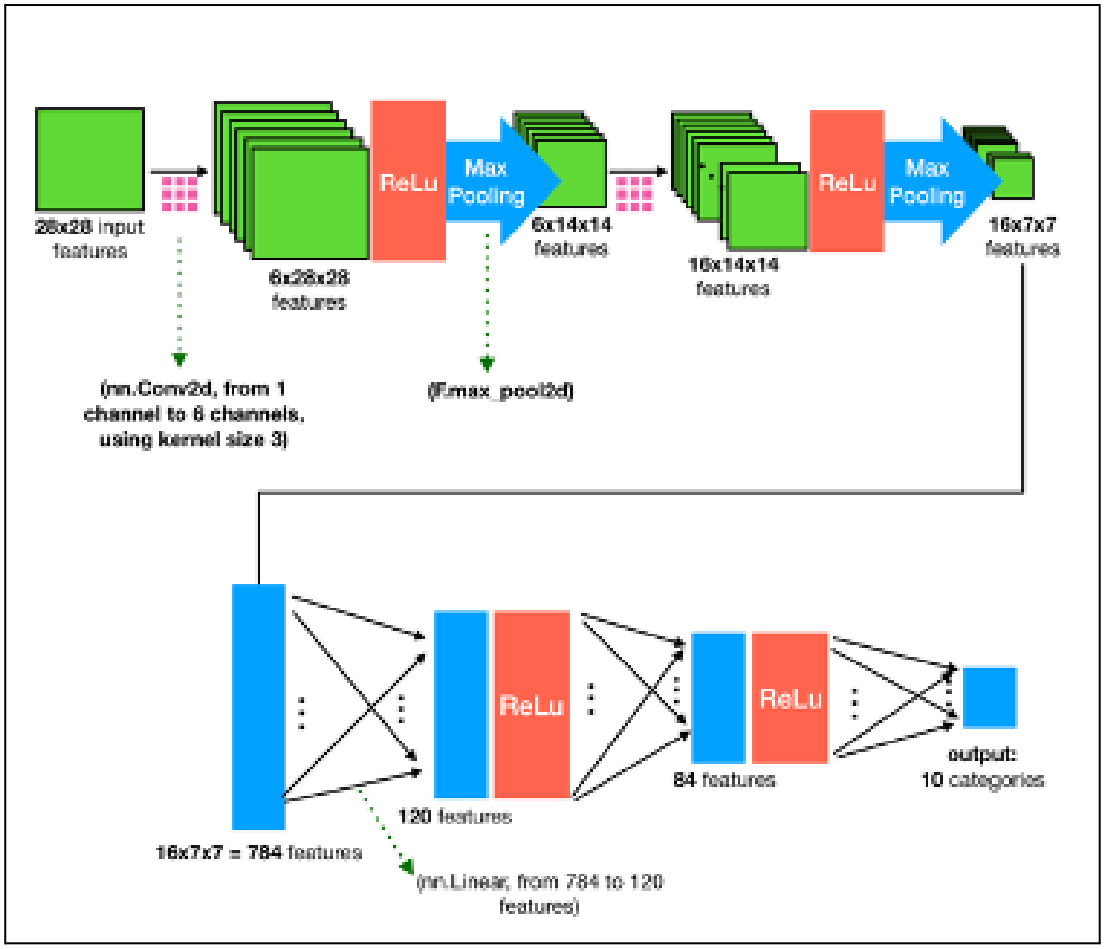

Image 1: CNN first architecture (LeNet)

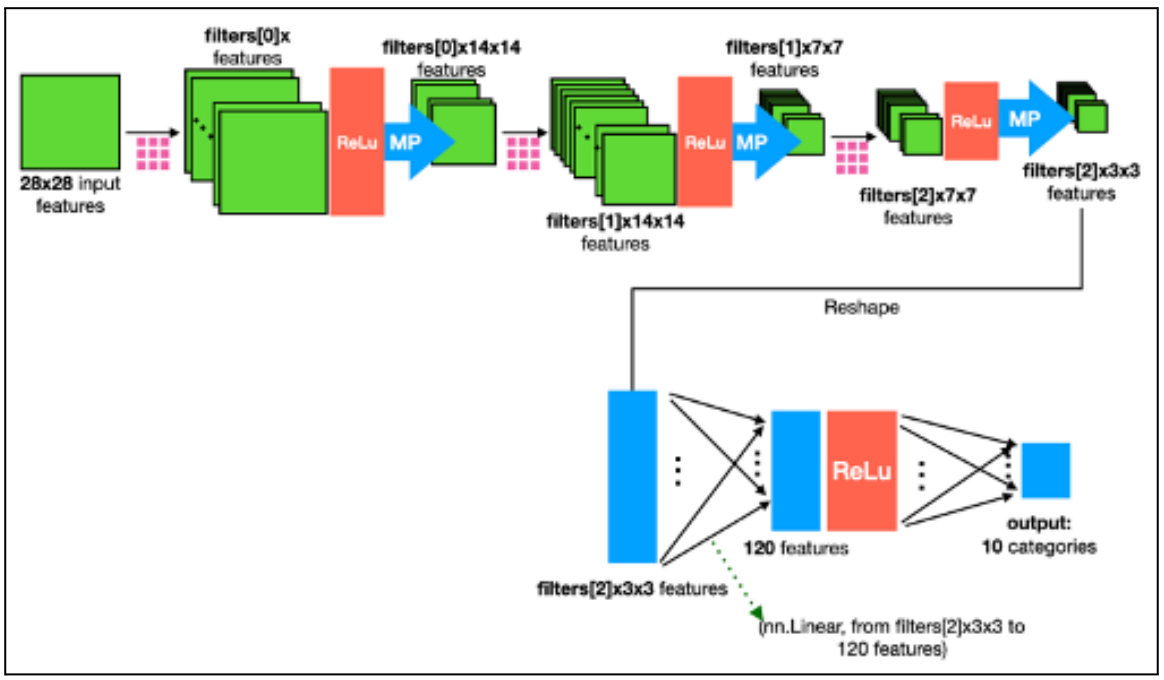

Image 2: CNN second architecture (inspired by CNN exercise)

	Lr = 1e-5	Lr = 1e-4	Lr = 1e-3	Lr = 5e-2	Lr = 5e-3	Time
SGD optimizer (Training)	-	Acc=22.5%, F1=0.136	Acc=71.6%, F1=0.701	-	-	85s
SGD optimizer (Validation)	-	Acc = 22.8%, F1=0.137	Acc=71.6%, F1=0.701	-	-	-
Adam optimizer (Training)	Acc=74.3%, F1=0.734	Acc=84.3%, F1=0.839	Acc=94.2%, F1=0.942	Acc=90.7%, F1=0.903	Acc=90.4%, F1=0.863	90s
Adam optimizer (Validation)	Acc = 74.37%, F1 = 0.736	Acc=83.1%, F1=0.827	Acc=89.0%, F1=0.890	Acc=86.4%, F1=0.863	Acc=87.4, F1=0.871	-

Table 1: CNN Results on channels (16, 16, 64)

Lr	1e-5	1e-4	1e-3	5e-3	5e-2	Time
Results Train (Adam)	Acc=74.6%, F1=0. 733	Acc=84.8%, F1=0.841	Acc=95.3%, F1=0.953	Acc=91.6%, F1=0.915	Acc=9.9%, F1=0.018	117s
Results Test (Adam)	Acc = 75.2%, F1 = 0.737	Acc=84.5%, F1=0.838	Acc=89.3%, F1=0.893	Acc=86.3%, F1=0.863	Acc=10.2%, F1=0.019	-

Table 2: CNN Results with channels (16, 32, 64) and only Adam optimizer

Batch Size	Epochs	Learning Rate	Accuracy (%)	F1 Score
16	40	0.01	81.2	0.77
32	40	0.01	**83.49**	0.80
32	40	0.005	79.10	0.76
32	20	0.01	66.28	0.61
16	20	0.01	56.07	0.44
32	10	0.01	69.54	0.65

TABLE 3: Transformer results with Adam optimizer

Model	***Parameters***	***Training Time (s)***	***Train Accuracy (%)***	***Validation Accuracy (%)***	***Train F1-Score***	***Validation F1-Score***
MLP	*235,146*	*89.45*	*92.19*	*84.240*	*0.889065*	*0.842665*
MLP with PCA	*60,042*	*68.47*	*89.06*	*83.475*	*0.862406*	*0.834067*

TABLE 4 : Experiment and results for the MLP (with and without PCA)