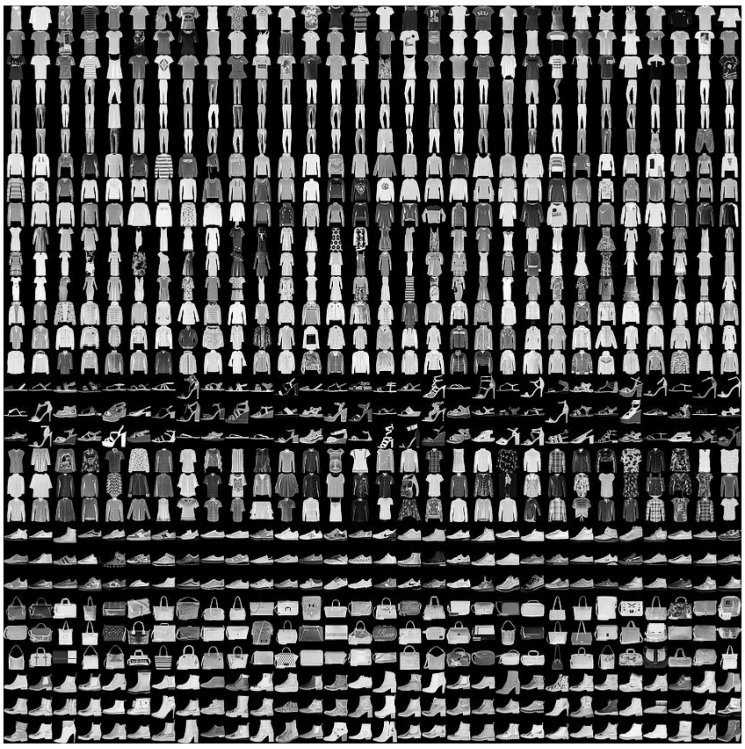
Sunday 2nd June, 2024

Ecole Polytechnique Fédérale de Lausanne

Introduction to Machine Learning : CS-233

Project report "Milestone 2 : Fashion MNIST"



Authors:
Cyrielle Manissadjian (345312)
Samuel Steullet (363873)
Tom Jeannin (345859)

Professor: Matthieu Salzmann

Table of Contents

1 Introduction

MNIST Fashion dataset includes 70,000 28x28 pixel grayscale images of 10 clothing and accessories. The objective is to classify images with 3 different algorithms: a Convolutional Neural Network (CNN), a Multi-Layer Perceptron (MLP), and a Vision Transformer (ViT) into categories such as T-shirts, trousers, or sneakers.

MLPs, although less specialized for images than CNNs, can perform well with preprocessing like Principal Component Analysis (PCA) to reduce dimensionality. CNNs are suited for image classification as they capture spatial hierarchies using convolutions, pooling, and non-linear activations to recognize patterns like edges and textures. ViTs break images into patches, using Multi-Head Self-Attention (MHSA) in a Transformer encoder to weigh patch importance, capturing global context, in general more effectively than traditional convolutional methods.

2 Methods and experiments

For all the algorithms that follow, we began by normalizing our data. Since a test set is unavailable we create a validation set by excluding 20% data from the training set. This is possible because we dispose of 60,000 homogeneous samples in the training dataset [1], enough to split our data in a well-representative way. Concerning the architecture of our models, we originally started with the architecture of the Jupyter Notebook exercises. We modified them to enhance scores, avoid overfitting, and decrease computational complexity and execution time.

The code has been updated with 9 new parser arguments for command line usage to adjust hyperparameter values (`n_patches`, `n_blocks`, `hidden_d`, `n_heads`, `filters`), validation set fraction (`val_set`), optimizer selection between SGD and Adam (`optimizer`), and model dropout fraction (`dropout`). All the different graphs are automatically saved in a folder `"graph_scores"`. The confusion matrix of the model is printed in the terminal to verify our results.

We chose ADAM optimizer over SGD for big datasets and high-dimensional input space due to its efficiency. It converges faster, less sensitive to initial learning rate, but is prone to local minima. It can also overfit more easily, which might explain some of the overfitting of our models.

2.1 MLP and PCA

The first architecture was made of 3 hidden layers (256, 128, and 64 inputs each). We then modified it by adding two dropouts to help avoid overfitting (changeable from the command line) ; two batch normalization to avoid internal covariate shift and change in distribution of layer inputs during training ; and one flattening. This leads to 243'530 parameters. With PCA, keeping the 81 first components, the model drops to 63'562 parameters. The total variance explained by the first 81 principal components is 87.23% and we also see in Figures [11(b)], [16] and [11(a)] the mean-variance, the 10 first principal components and the cumulative explained variance.

The Figure [5] summarises the different models we ran for the MLP with accuracies and F1 score during training. The ROC curve for our best MLP model is in Figure [6(a)].

2.2 CNN

We've implemented 3 convolutional filters followed by a linear fully-connected neural network made of one hidden layer of size 120. We then added a batch normalization between each convolutional filter and a dropout at the end. With filters of (4, 8, 16), (8, 16, 32) and (16, 32, 64) we have respectively 41'250, 82'554 and 173'802 parameters. All the tests that we did had a dropout of 0.5.

We have tested different learning rates and filters for our CNN model, results are summarised in Figure [8], and the Accuracies, F1 score and ROC curve for out best MLP model are represented in Figures [7(a)].

2.3 Transformer

We started with the original Vit architecture for this algorithm, but it was too demanding in time (we had around 4,5 million parameters). Thus we went with 14 patches, 2 blocks, 128 nodes, and 2 heads.

To find the best hyperparameters, we picked 2 or 3 hyperparameters to tune and fixed the others with standard values. The tunable hyperparameters were chosen carefully, by selecting one related to model optimization and another related to model capacity.

LogSoftmax is used in the final layer of the MLP because of its best numerical stability, logarithmic scale benefits, and simplification of loss calculation.

We can find the different transformers trained in Figure [10], And the accuracies and F1 scores of our best transformer model during training in Figure [9].

3 Discussion

3.1 MLP and PCA

Due to the fully-connected specificity of MLP, there is a high number of parameters and thus, the model has a higher chance of overfitting the data. This explains why the accuracy of the training set is much higher than that of the validation set. Nonetheless, we obtain high scores for our validation data so our model can generalize for unseen datas.

PCA reduces features for simpler, faster model training and noise removal.

	Mean Accuracy (%)	**Standard Deviation (%)**	**Computation Time (s)**
With PCA	85.78	0.73	52 min
Without PCA	85.57	0.35	1 hour 24min

Table 1: Performance comparison with and without PCA, for all our experimental runs

Mean accuracy is similar with or without PCA, but the increased standard deviation may also introduce more variability in model performance. PCA decreases the computation time by almost half. This is due to the number of parameters that are largely reduced.

3.2 CNN

We observed that when we run a CNN with a learning rate of 0.1, no matter what filter we use, we have accuracies of 10%, i.e. random classification. When we look at these models' confusion matrix, we can see that they only predict the same class. The issue can be that 0.1 is a too-big learning rate and at each step, the model jumps forward and back because the steps aren't small enough. The more parameters there are in a model, the more it is sensitive to learning rates. If the learning rate is too large it is more likely to explode and have bad results. This might be the reason why CNN with filters (4, 8, 16) and a learning rate of 0.05 have a reasonable accuracy while CNN with filters (8, 16, 32) with the same learning rate has an accuracy close to randomness. The same reasoning can be applied with filters (16, 32, 64).

3.3 Transformer

The bottleneck of the transformer is the time it takes to train. Several factors can explain that. The one that has the biggest impact is the time complexity of the self-attention mechanism is $O(N^2 \cdot d + d \cdot N^2)$ which is big, knowing that for our model there is 128 hidden nodes (d) and images are splitter into 14 patches (N).(we were time limited to run our model). Transformers implies a wide range of hyperparameters, we surely did not find the right one as Transformers could have led us to better results than all the previous model. We could have used K-fold cross validation to find the rigth hyperparameters, but as the model already took us around 4 hours for 10 epochs to run. Implementing K-fold would have made it impossible if we didn't had another machine.

4 Conclusion

The transformer could have been the best model, but it requires a good architecture to be efficient. We did not find it because it took us too much time. In the end, our best model is CNN. Having many parameters does not imply that the model is good (we learned it at our own fee). Dimensionality reduction using PCA can gain a lot of time with no accuracy reduction. We have some pretty good results, especially if we take a glance to what the images look like in 28 x 28 pixel. The models might have even better results than human recognition.

5 Annex

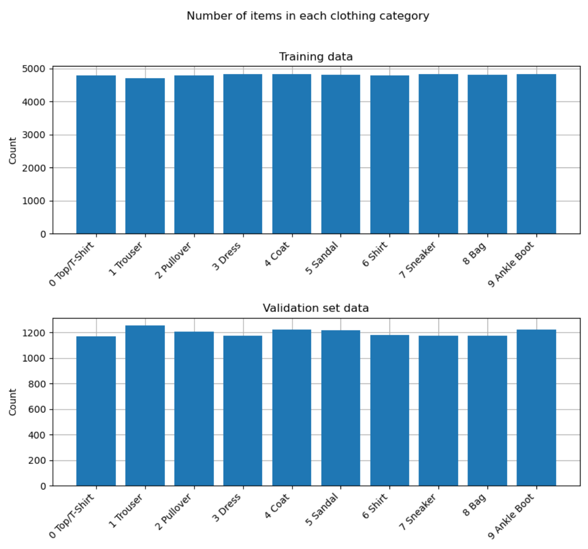


Figure 1: Number of clothes count in each 10 class for train and test data

```
========================================
MLP                             --
Sequential: 1-1               --
    Flatten: 2-1             --
    Linear: 2-2              200,960
    BatchNorm1d: 2-3         512
    ReLU: 2-4                --
    Dropout: 2-5             --
    Linear: 2-6              32,896
    BatchNorm1d: 2-7         256
    ReLU: 2-8                --
    Dropout: 2-9             --
    Linear: 2-10             8,256
    ReLU: 2-11               --
    Linear: 2-12             650
========================================
Total params: 243,530
========================================
```

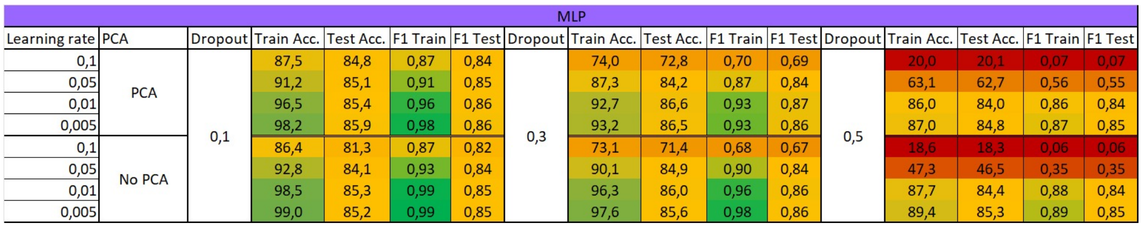
Figure 2: MLP Model architecture

```
========================================
CNN                       --
Conv2d: 1-1              160
BatchNorm2d: 1-2         32
Conv2d: 1-3              4,640
BatchNorm2d: 1-4         64
Conv2d: 1-5              18,496
BatchNorm2d: 1-6         128
Linear: 1-7              147,712
Linear: 1-8              2,570
Dropout: 1-9             --
========================================
Total params: 173,802
========================================
```

Figure 3: CNN Model architecture

```
=======================================
MyViT                           25,344
Linear: 1-1                    640
ModuleList: 1-2                --
    MyViTBlock: 2-1            --
        LayerNorm: 3-1        256
        MyMSA: 3-2            49,536
        LayerNorm: 3-3        256
        Sequential: 3-4       131,712
    MyViTBlock: 2-2           --
        LayerNorm: 3-5        256
        MyMSA: 3-6            49,536
        LayerNorm: 3-7        256
        Sequential: 3-8       131,712
Sequential: 1-3                --
    Linear: 2-3                1,290
    LogSoftmax: 2-4            --
=======================================
Total params: 390,794
=======================================
```

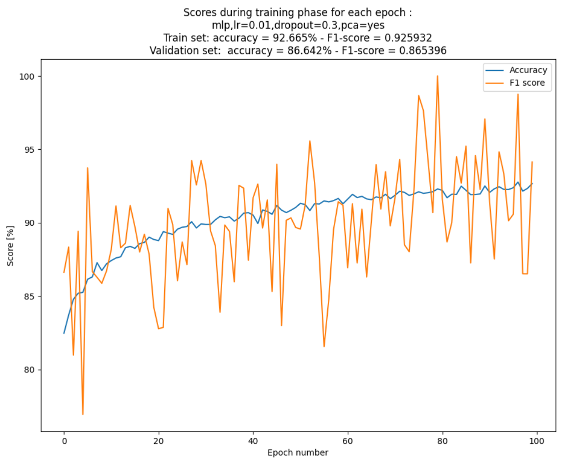
Figure 4: MyViT Model architecture



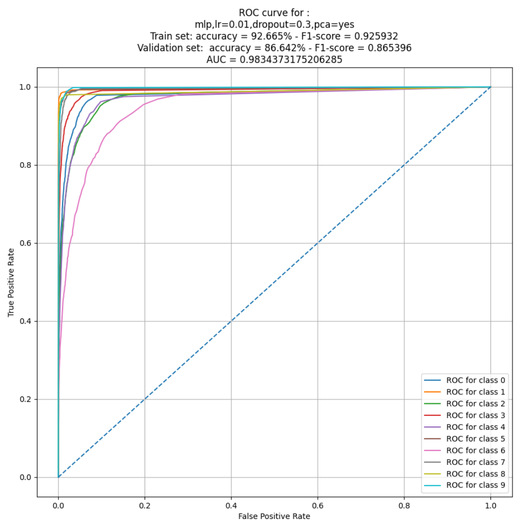
MLP																
Learning rate	PCA	Dropout	Train Acc.	Test Acc.	F1 Train	F1 Test	Dropout	Train Acc.	Test Acc.	F1 Train	F1 Test	Dropout	Train Acc.	Test Acc.	F1 Train	F1 Test
0,1	PCA	0,1	87,5	84,8	0,87	0,84	0,3	74,0	72,8	0,70	0,69	0,5	20,0	20,1	0,07	0,07
0,05			91,2	85,1	0,91	0,85		87,3	84,2	0,87	0,84		63,1	62,7	0,56	0,55
0,01			96,5	85,4	0,96	0,86		92,7	86,6	0,93	0,87		86,0	84,0	0,86	0,84
0,005			98,2	85,9	0,98	0,86		93,2	86,5	0,93	0,86		87,0	84,8	0,87	0,85
0,1	No PCA		86,4	81,3	0,87	0,82		73,1	71,4	0,68	0,67		18,6	18,3	0,06	0,06
0,05			92,8	84,1	0,93	0,84		90,1	84,9	0,90	0,84		47,3	46,5	0,35	0,35
0,01			98,5	85,3	0,99	0,85		96,3	86,0	0,96	0,86		87,7	84,4	0,88	0,84
0,005			99,0	85,2	0,99	0,85		97,6	85,6	0,98	0,86		89,4	85,3	0,89	0,85

Figure 5: Table MLP

Figure 6: Result of the best MLP model got : lr=0.01, dropout=0.3, with use of PCA

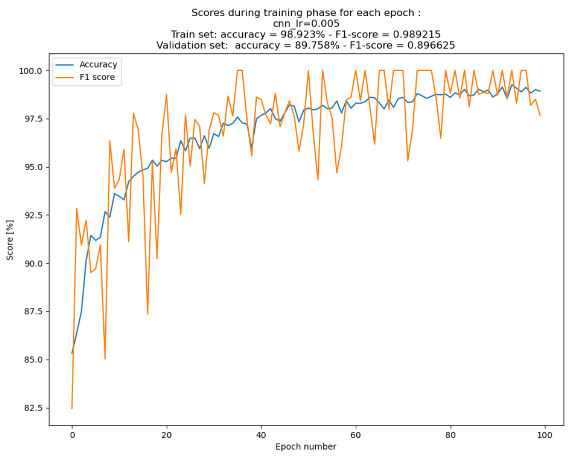


((a)) Accuracy and F1 scores for MLP

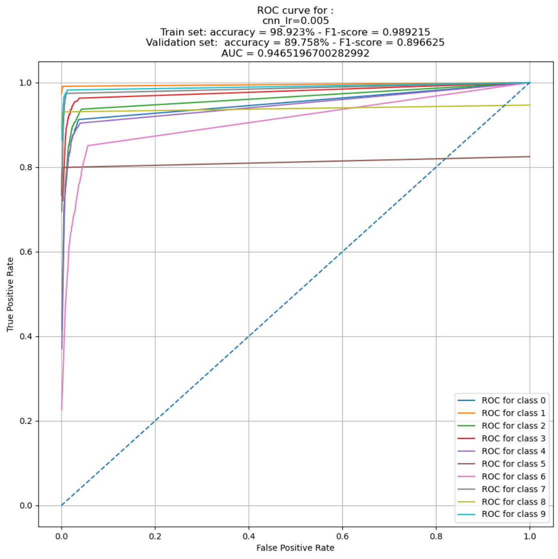


((b)) ROC curve for MLP

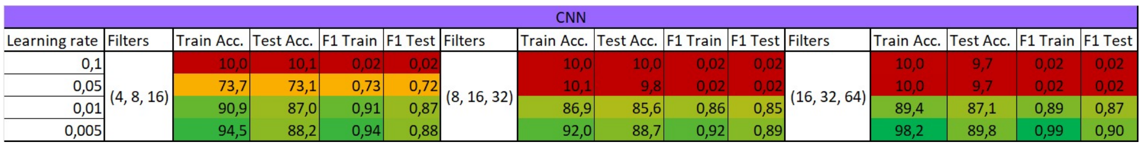
Figure 7: Result of the best CNN model got : lr=0.005, dropout=0.5 and filters = (16, 32, 64)



((a)) Accuracy and F1 scores for CNN



((b)) ROC curve for CNN



CNN														
Learning rate	Filters	Train Acc.	Test Acc.	F1 Train	F1 Test	Filters	Train Acc.	Test Acc.	F1 Train	F1 Test	Filters	Train Acc.	Test Acc.	F1 Train	F1 Test
0,1	(4, 8, 16)	10,0	10,1	0,02	0,02	(8, 16, 32)	10,0	10,0	0,02	0,02	(16, 32, 64)	10,0	9,7	0,02	0,02
0,05		73,7	73,1	0,73	0,72		10,1	9,8	0,02	0,02		10,0	9,7	0,02	0,02
0,01		90,9	87,0	0,91	0,87		86,9	85,6	0,86	0,85		89,4	87,1	0,89	0,87
0,005		94,5	88,2	0,94	0,88		92,0	88,7	0,92	0,89		98,2	89,8	0,99	0,90

Figure 8: Table CNN

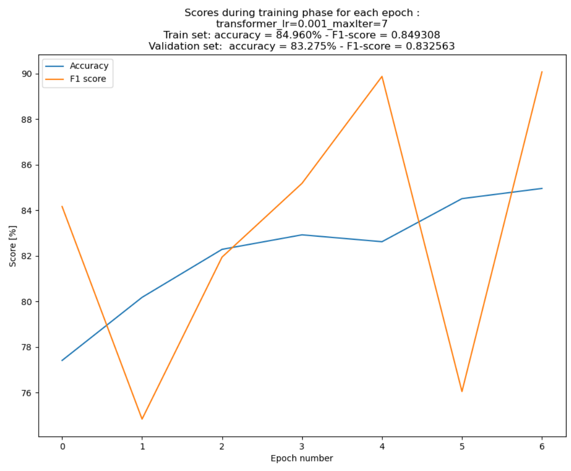
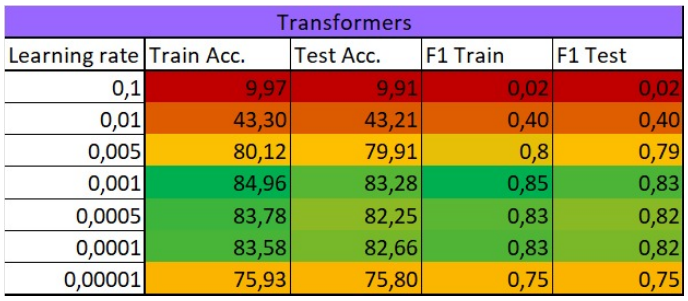


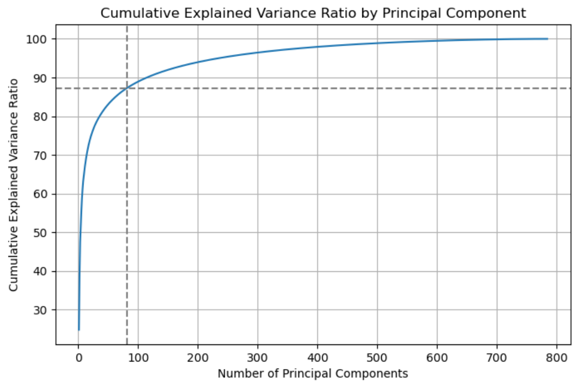
Figure 9: Transformer: Accuracy and f1 score for Transformer (lr=0.001, 7 epochs)



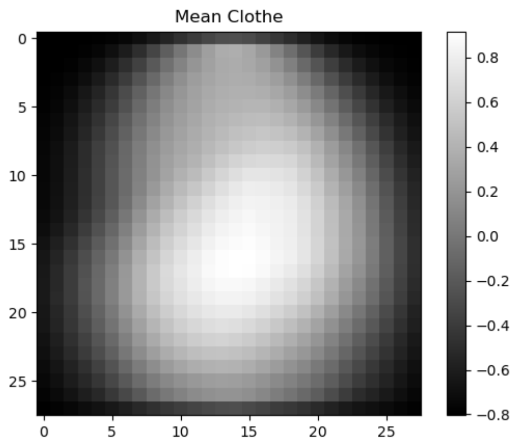
Transformers				
Learning rate	Train Acc.	Test Acc.	F1 Train	F1 Test
0,1	9,97	9,91	0,02	0,02
0,01	43,30	43,21	0,40	0,40
0,005	80,12	79,91	0,8	0,79
0,001	84,96	83,28	0,85	0,83
0,0005	83,78	82,25	0,83	0,82
0,0001	83,58	82,66	0,83	0,82
0,00001	75,93	75,80	0,75	0,75

Figure 10: Table Transformer

Figure 11: PCA visualisation



((a)) cumulative sum



((b)) mean PCA

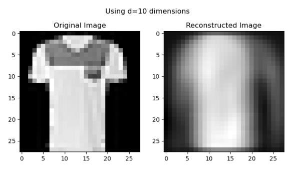


Figure 12: PCA d=10 Dimensions, Explained variance = 65.80%

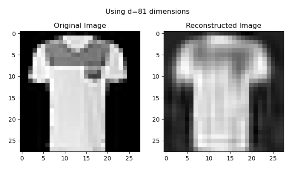


Figure 13: PCA d=81 Dimensions, Explained variance = 87,23%

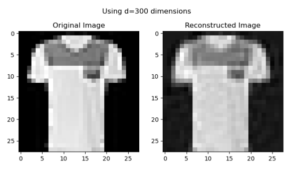


Figure 14: PCA d=300 Dimensions, Explained variance = 96,46%

Figure 15: Visualisation of images before and after PCA

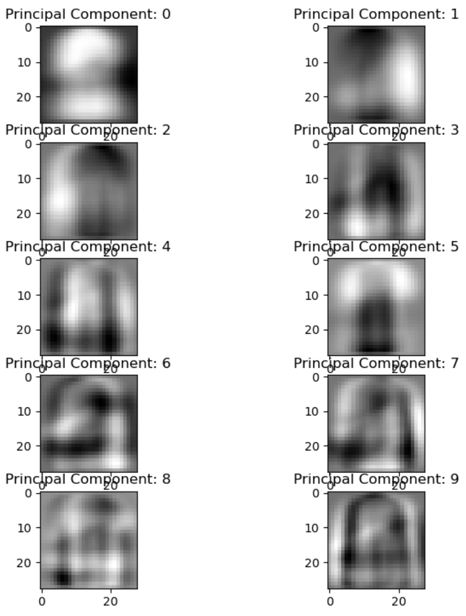


Figure 16: 10 first components