CS233 FashionMNIST MS2

Stanislas Castellana, Alice Cotten, Cyrill Strassburg

1 Introduction and Methods

This report discusses categorization of fashion item images from the Zalando Fashion-MNIST dataset with a multi-layer perceptron (MLP), convolutional neural network (CNN) and a vision transformer (ViT) model. The data contains 10 categories evenly split between 60'000 labeled samples.

Data preprocessing and splitting The provided data includes a set of features and labels, as well as a competition testing feature set, for which no corresponding labels are available. The complete feature/label set is randomly split with a 1:4 ratio to create training/testing sets for our models. The batch size is fixed at a size of 64 data points.

MLP The MLP class takes a variable number of Fully connected (FC) layers, with any number of nodes per hidden layer, followed by a ReLU activation function. At every FC layer it is possible to apply a dropout layer with parameter $p \in [0, 1] \subset \mathbb{R}$. Activating dropout will randomly zero out parameters with probability p. We decided to test two different MLPs with the same hidden layers $[256, 128, 64]$ but differing in dropout use. We also apply principle component analysis (PCA) to both our MLP models.

CNN Similarly to the MLP our CNN class has a variable number of convolutional layers followed by a variable number of FC layers, its architecture inspired by Yann LeCun's LeNet. Padding is added to the input of each convolution layer to avoid information loss, with padding size being determined by $(k-1)/2$ where k is the width and height of the convolution kernel. At each convolution layer, the output is passed to a ReLU activation function and then pooled by maximal value with a 2×2 kernel, dividing the width and height of the convoluted image by a factor of 2. The outputs of the last convolutional layer is then passed to the fully connected layers, then fed into a ReLU activation function at the end. For the experiments in this report, we decided on two convolutional layers with kernel sizes 5 and 3 followed by three FC layers as can be seen in figure 1.

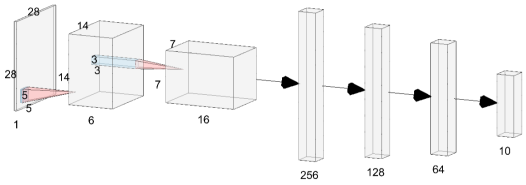


Figure 1: Visualisation of our CNN architecture taking a $Ch, D, D = 1 \times 28 \times 28$ image as input. The size of the fully connected layers corresponds to its outputs.

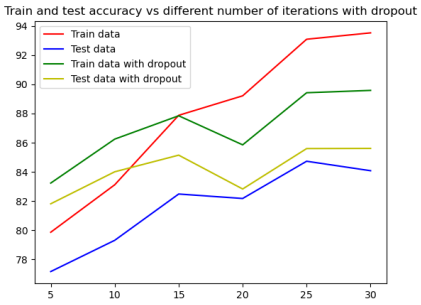
ViT As its name suggests, the transformer we are implementing is adapted to computer vision. Originally developed for NLP, transformers take as inputs a sequence of tokens, and have a notion of attention to give importance to input data at any point in a possibly long sequence. To achieve this same notion of tokens we had to *patchify* our images, i.e convert an image to a sequence of tokens. This is simply achieved by splitting a square image into a sequence of smaller square patches. Our ViT takes as argument the number of patches per image, the number of attention blocks, the number of hidden neurons in the FC layer, the number of attention heads, and finally the output dimension: 10. We decided to experiment by varying the number of blocks and keeping other parameters constant.

2 Results/Discussion

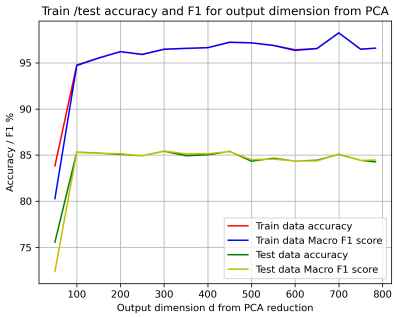
2.1 MLP

Dropout In figure 2 we can see proportionality between accuracy and epoch number with respect to all datasets. Also, the results on the validation data do not differ by more than 1% for epoch numbers above 20. Generally the difference in accuracy between train and validation sets is much smaller when using dropout. This is expected as dropout reduces over-fitting by randomly zeroing out weights. It should be noted that according to literature, learning rates with and without dropout generally are not in the same order of magnitude. We decided to keep the same learning rate to obtain fair experimental results.

PCA The PCA results in figure 2 show that $\forall d \geq 100$ the accuracy and macro f1 scores tend to be around constant for both the training and validation data sets. From this inference, reducing the dimension of the images to 100 pixels encodes approximately all the information in a 784-dimensional space. A possible explanation for similar performance at lower dimensions could be that PCA removes unnecessary information in higher dimensions, e.g a certain bag could have features that other bags do not, and PCA would make it more similar to the other ones. Clearly, the point of interest in the graph is around the region $d = 100$ as it is where there is the biggest change in performance. A part from that, the data is pretty redundant and shows how little PCA affects results for bigger values of d.



(a) Comparison of train and test accuracy of the MLP model for different numbers of training iterations with learning rate 10^{-1}



(b) Accuracy and F1 score vs different output dimension from PCA. Fixed learning rate of 10^{-1}, 30 epochs. Note that there is no dropout in this model

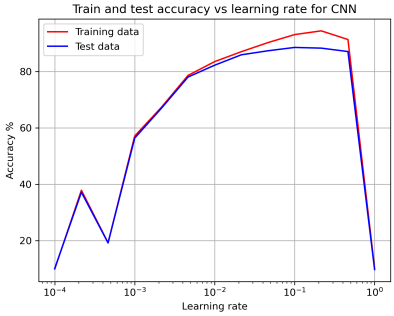
Figure 2: MLP: Dropout and PCA

2.2 CNN

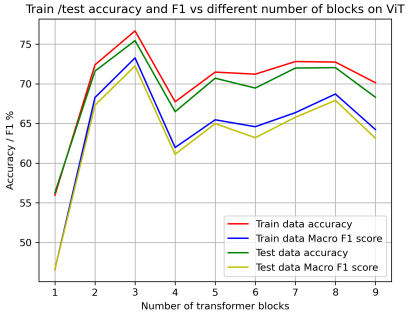
For the CNN we decided to experiment with learning rate. To our surprise we found an optimal learning rate of $lr = 0.1$ as shown in figure 3. We are surprised as it is much greater than what we found in the first project. It would have been interesting to use momentum for the learning rate in Stochastic Gradient Descent (SGD) to possibly accelerate the rate of convergence to a local minima, or maybe a learning rate scheduler to make sure the steps are not too big when being close to convergence, and not too small in the opposite case.

2.3 ViT

In the case of the ViT we are looking at the effect of increasing the number of blocks on the accuracy and macro F1 score of the model. We expected an increasing relationship between the number of blocks and accuracy and macro f1 score but it is not what we obtained. This can be explained by the fact that more blocks corresponds to deeper feature representation and an increased model capacity. Also, models with more blocks have more weights therefore being more computationally expensive to train. Possible reasons for not obtaining a strictly increasing relationship could be that there were not enough training epochs and the random component of SGD. It would have been more insightful to retrain all models 5-10 times to obtain averaged out performance values.



(a) Varying learning rate on test and train accuracy. The best value for learning rate for the validation/test set is $lr = 0.1$. CNN's were run for 20 epochs.



(b) How the number of blocks in a ViT affects performance. The other parameters were fixed to: 7 patches / width of the image, 64 nodes in the sequential hidden layer, and 8 heads

Figure 3: CNN and ViT

3 Conclusion

Globally we found that the decreasing order of performance of the models were MLP (50 epochs), CNN (20 epochs), ViT (12 epochs) with training times of 5, 15, and 2682 seconds per epoch (on CPU). This is not we expected in theory but it is most likely due to the fact that the slower models like the CNN and ViT were not run for enough epochs, and also that our models could be configured to have more weights. We found that the CNN could surpass the 85% limit while the MLP could not. Unfortunately, we did not have the time to train the ViT enough. There are also other caveats, for example the CNN reaches close to $75 - 80\%$ accuracy after one epoch (!) while the MLP takes significantly more. We would also like to continue the project by comparing different architectures for the same model, or compare how the number of weights in a model affects performance / overfitting.