Clothes Classification: recycling networks
Machine Learning Programming assignment

Nicholas Bova
ID number: 544164

A.Y 2023/2024

Introduction

Image recognition identifies a wide range of problems. Image classification is the most simple. It consists in assigning images to one class chosen from a fixed set.
Image recognition is used across a lot of industries and applications. In *medical imaging* it recognizes diseases and anomalies in *X*-rays, MRIs, and CT scans. It finds application in security and surveillance, for instance, to identify individuals (*facial recognition*).

1 Analyze the data

The dataset consists in a total of 70000 images of 10 different kinds of clothes. The ten classes are: *T-shirt, trousers, pullover, dress, coat, sandal, shirt, sneaker, bag* and *ankle boot*.
The images were divided in *training set* (60000) and *test set* (10000). Each image consists in 28×28 pixels, each one containing a value between 0 and 1. The pictures were taken with a low-resolution grayscale camera. In Figure 1 we can see the picture of some training samples of the ten classes.
We observe that the classes are equiprobable. There are, indeed, 6000 samples of each classes.
The goal of this assignment is to build a ten-classes classifier with the highest accuracy.

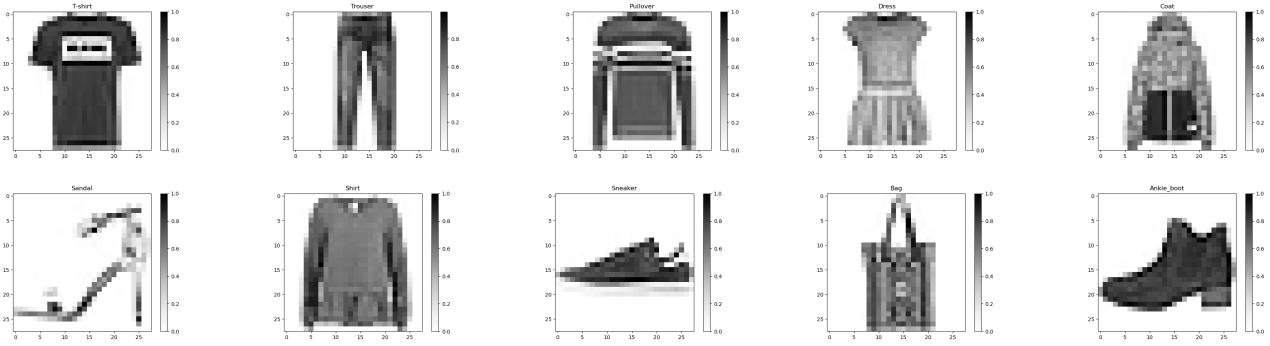


Figure 1: images of some training samples belonging to the ten classes.

2 Selection of the model

We are going to approach the problem with various models. At first we build a baseline model, such as *homoscedastic Gaussian Discriminant Analysis* (GDA). The problem is later explored with more sophisticated models. We are going to focus our attention on *feedforward* networks. A *Multi-Layer Perceptron* (MLP) with different architectures is designed. Our discussion concludes with *Convolutional neural networks* (CNNs). CNNs were designed specifically for image classification and they represent the state of the art for recognition of signals, images in particular.

3 Homoscedastic GDA

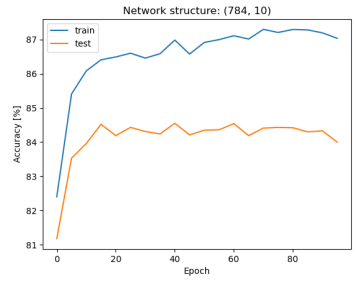
Gaussian Discriminant Analysis (GDA) is a generative method used in machine learning for classification problems. The model assumes that the density function $p(\boldsymbol{x}|y)$ of each class is a Gaussian distribution. In the *homoscedastic* model all the classes are assumed to share the same covariance matrix, meaning they have identical spreads and shapes. This assumption simplifies the model and reduces the number of parameters to estimate.
We trained our model on the training set and we evaluate it on the test set. We obtained a training accuracy of $83.26\,\%$ and a test accuracy of $81.50\,\%$. *Accuracy* is defined as the fraction of samples that are correctly classified. Despite the model simplicity, good performances are still achieved.
This reflects that the underlying assumptions of *homoscedastic GDA* are met in practice with good approximation. This means that every class has the same dispersion around its mean. They all have the same covariance matrix. Moreover the conditional probability $p(\boldsymbol{x}|y)$ is a gaussian with good approximation.

3.1 Minimum Distance classifier

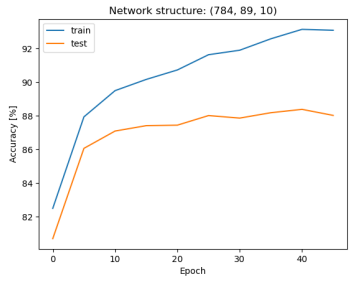
Recall from Section 1 that the classes are equiprobable. It is interesting to buil a *minimum distance classifier*. This model further assumes that components in the feature vectors have uniform variance and they are mutually independent. By training and evaluating the model we get a training accuracy of $68.57\,\%$ and a test accuracy of $67.68\,\%$. These performances can be be read as a confirmation of our hypotheses. Minimum distance classifier does not work well due to non mutually independent features. Consider, for instance, a T-shirt and focus on the sleeve. Suppose we take a pixel with ink on the lower edge of that sleeve. Neighbouring pixels are not independent. It is more likely to have ink in the upper part (pixels forming the sleeve), rather than in the lower one.

4 Multi-layer perceptron

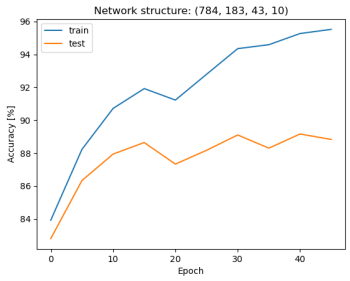
The next step consists in the implementation of a MLP. We tried different network architectures using the PVML library. Firstly we used a MLP with no hidden layers, consisting of 784 input neurons (28×28 pixels) and 10 output neurons (number of classes). After this we included one hidden layer with 89 hidden neurons. It was then created a structure with two hidden layers, containing 183 and 43 hidden neurons, respectively. Given the struc-



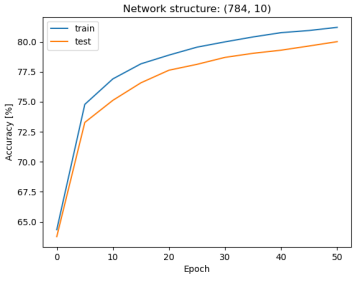
(a) no hidden layers.



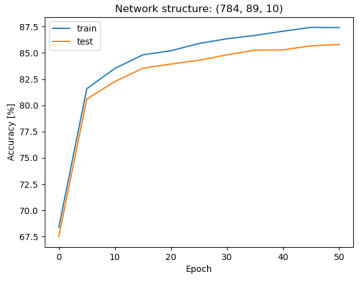
(b) one hidden layer.



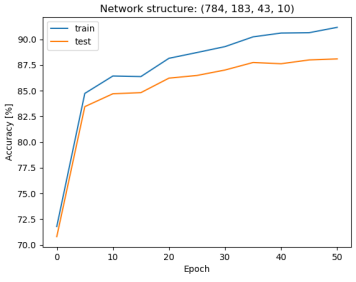
(c) two hidden layers.



(d) L_2, no hidden layers.



(e) L_2, one hidden layer.



(f) L_2, two hidden layers.

Figure 2: training and test accuracies for different architectures of MLP. In the first row we have no normalization; L_2 normalization is implemented in the second row.

ture of the dataset the model was firstly trained and evaluated with no features normalization. The model was then trained and evaluated after L_2 *normalization*. We choose this normalization technique because all the components in a feature vector represent the same kind of information. Pixels carry the same type of information. A

The results are reported in Table 1 and shown in Figure 4.

	no hidden layer		**one hidden layer**		**two hidden layers**	
	Training	*Test*	*Training*	*Test*	*Training*	*Test*
no normalization	87.03	84.00	93.09	88.03	95.50	88.83
$\boldsymbol{L_2}$	81.20	80.01	87.41	85.80	91.16	88.09

Table 1: percentage accuracies obtained for different network architectures.

With L_2 normalization performances are not better. However, notice that the more the hidden layers, the more the model tends to overfit. L_2 normalization seems to limit the overfitting.

4.1 Visualization of the weights

It is useful to analyze the learned weights for the MLP without hidden layers. Here we can appreciate the effect of L_2 *normalization*. The weight tells us how much that pixel contributes positively or negatively to that class.

(a) no normalization

(b) L_2 normalization

Figure 3: weights of the MLP without hidden layers. The positive values are in red, the negatives in blue.

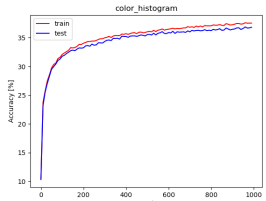
For instance, consider class 1 (T-shirt). Imagine that an image presents ink in the lower half at the edges of the image. Those pixels contributes very negatively to the class 1. Makes sense, since T-shirts have no long sleeves or other shapes.
Notice that central pixels are not very important for classes 1, 3, 5 and 7 (T-shirt, pullover, coat and shirt). They are otherwise relevant for classes 2, 4, 6, 8, 10. (trousers, dress, sandal, sneaker and ankle boot). We dwell on the role of central pixels for class 6. If an image has ink in the central part, it is unlikely classified as a sandal.

4.2 Feature extraction

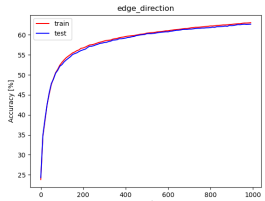
At this point we performed some *low-level features* extraction. Although we are dealing with grayscale images, we extract *color histograms* (CH) and *RGB co-occurrence matrix* (RGBCM) vectors, in addition to *edge direction histogram* (EDH) and *gray-level co-occurrence matrix* (GLCM). Better performances should be obtained with the last two. Observe that even with color images, we would not expect a good performance for CH and RGBCM. This is because it is unrealistic to distinguish a shirt from a pullover by the color.
The extracted vectors are then combined with L_2 normalization. The resulting vectors are taken as input of a MLP. For the sake of brevity we consider a MLP with no hidden layer. Training process and evaluation return the outcomes listed in Table 2.

	CH		**EDH**		**GLCM**		**RGBCM**	
	Training	*Test*	*Training*	*Test*	*Training*	*Test*	*Training*	*Test*
no normalization	37.53	36.81	63.03	62.63	41.64	41.02	31.09	31.02
$\boldsymbol{L_2}$	37.59	36.69	71.05	70.28	47.60	46.87	38.07	37.88

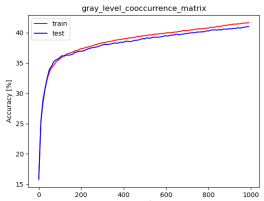
Table 2: percentage accuracies obtained for a MLP without hidden layers. The input consists in low-level features vector normalized with L_2 normalization.



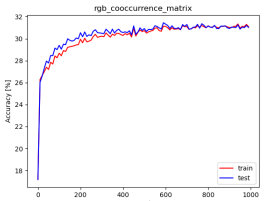
(a) CH.



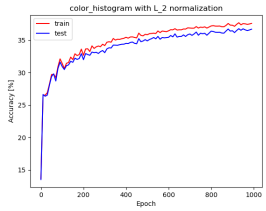
(b) EDH.



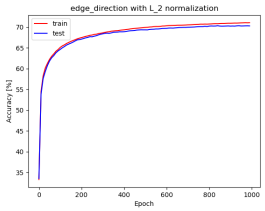
(c) GLCM.



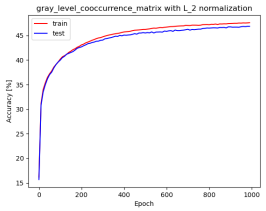
(d) RGBCM.



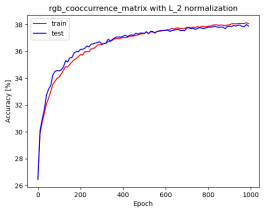
(e) CH + L_2.



(f) EDH + L_2.



(g) GLCM + L_2.



(h) RGBCM + L_2.

Figure 4: training and test accuracies for different *low-level features* vectors with and without L_2 normalization.

As expected *color histograms* and *RGB co-occurrence matrix* don't fit well. The model is clearly underfitting. Accuracy increases with *edge direction histogram*. Notice that GLCM does not perform as expected. We suppose this is due to low- resolution images, such as 28×28 pixels.
With this kind of images GLCM is less effective than EDH. Texture patterns are harder to discern indeed. On the other hand EDH focuses on the shape of the object. It relies on more prominent features like edges. This makes it suitable for low-resolution images.

5 Convolutional Neural Networks

In order to improve the performances of our model we used *Convolutional Neural Networks*. They consist in a MLP where the linear operators are replaced with *convolutions*.

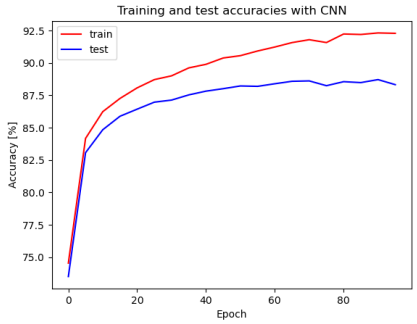


Figure 5: training and test accuracies using a CNN

We created a CNN using the PVML library with the following code:

```
network = pvml.CNN([1, 12, 32, 48, 10], [7, 3, 3, 3], [2, 2, 1, 1], [0, 0, 0, 0])
```

The CNN was trained given the parameter in Table 3. To be accurate this *hyperparameters* should have estimated with some *model selection*, using, for instance, a *grid search*. The same applies for the hyperparameters of previous models. In this assignment we did not perform model selection due to computational cost.

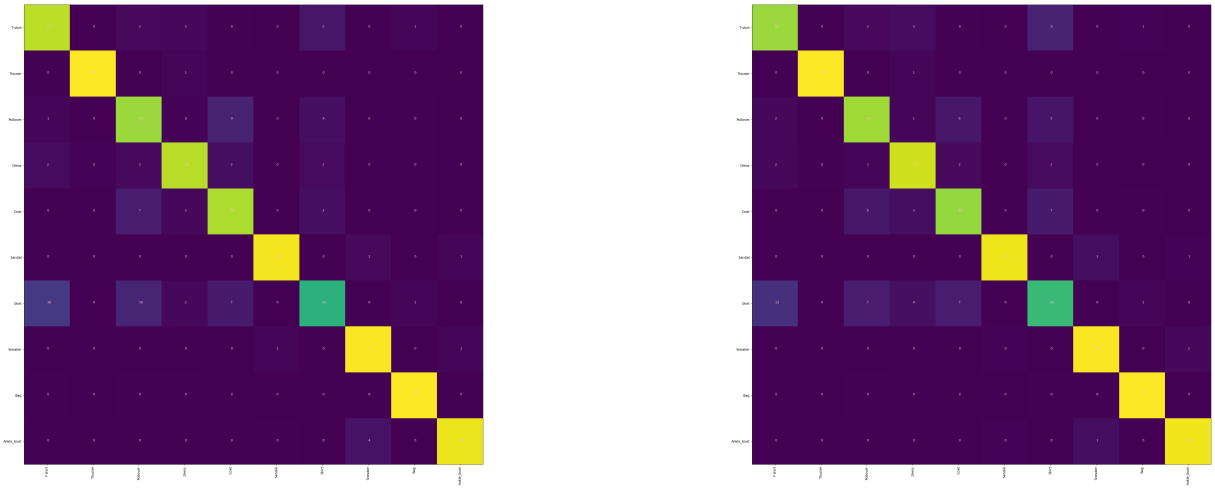
batch size	learning rate	epochs
100	0.0001	100

Table 3: parameters used in training CNN.

Training and evaluation give $92.98\,\%$ training accuracy and $88.75\,\%$ test accuracy. The performances could be optimized with a better and larger training set. Figure 5 displays training and test accuracies over number of epochs.

6 Analysis

We computed and printed the *confusion matrix* for a MLP with two hidden layers and for the CNN. See Figures 6a and 6b. All models struggle with the *shirt* class. Notice that shirts are wrongly classified as *T-shirts*, *pullover* and *coat*. We could have expected this mistakes by looking at Figure 1. For instance, many people would get confused from pullover and shirt.
Notice that these models do not confuse footwear with clothes. It is comforting, since this garments need to be recycled. It would be unpleasent to recycle an ankle boot together with T-shirts.



(a) MLP with two hidden layers.

(b) CNN.

Figure 6: confusion matrices for different models.

We tried to identify the test samples that are misclassified even if the classifier predicted their label with high confidence. For this purpose we used CNN. The outcomes are shown in Figure 7. Accordingly to the confusion matrix 6b, we can see that the model wrongly classified a shirt with a coat, a sneaker with a sandal, an ankle boot with a sandal. The latter reflects what was said in Section 4.1. Moreover, the model wrongly classified an anckle boot as sneaker, a trouser as dress and a shirt as T-shirt. It seems reasonable to classify the last sample as a T-shirt. Even a person would make that mistake. Furthermore the trouser in Figure 7 has the shape of a dress.

Conclusion

In this study we had to solve an image recognition problem. The dataset consisted in low-resolution grayscale images.
We explored various models, including Gaussian Discriminant Analysis (GDA), Multi-Layer Perceptron (MLP), and Convolutional Neural Networks (CNN). Among these, the CNN and MLP models demonstrated superior performance, achieving a maximum test accuracy of around $89\,\%$. This indicates that deep learning architectures, particularly those designed to capture spatial hierarchies like CNNs, are well-suited for handling image recognition tasks even with low-resolution inputs. We think that CNN did not overperformed MLP due to the training set size and composition.
Future work could focus on further optimizing these models. One could emply transfer learning. Using a pre-trained CNN would improve accuracy and robustness of our model.
Moreover, some model selection could be done in order to take the best values of all the hyperparameter. Maybe we could implement *k-fold cross-validation*.

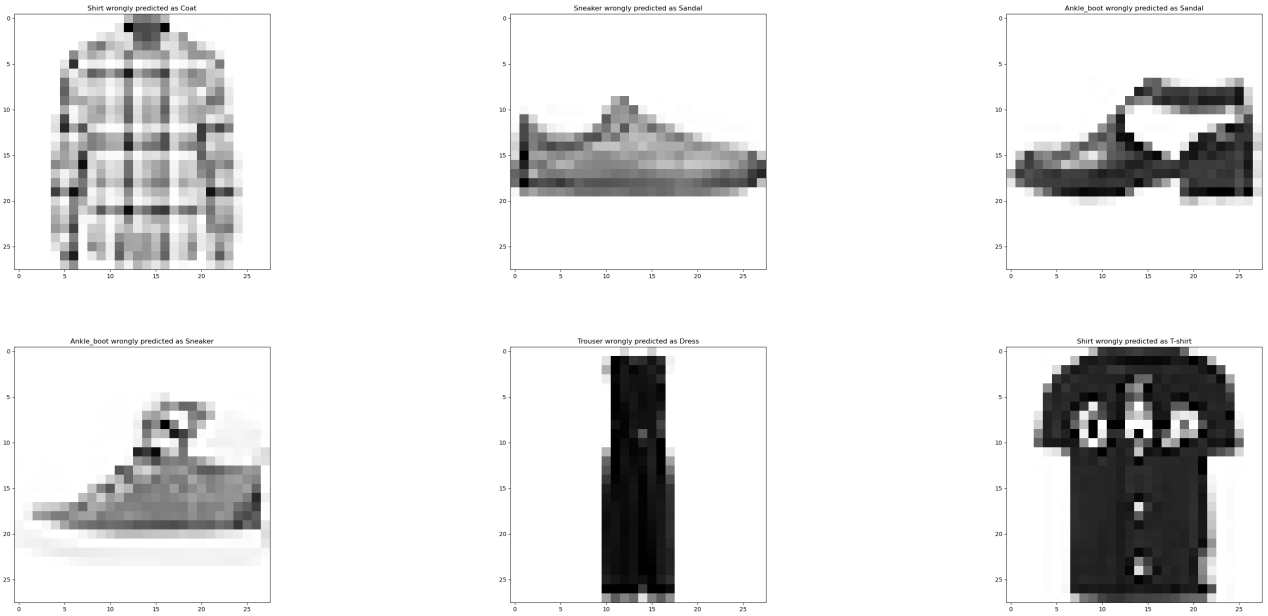


Figure 7: samples that the model misclassified with high confidence.

Declaration

I affirm that this report is the result of my own work and that I did not share any part of it with anyone else except the teacher.