# Advanced Deep Learning and Kernel Methods

## Challenge 1:
## A kernel methods / DL pipeline for the FashionMNIST dataset

Emanuele Ruoppolo
*Università degli Studi Trieste*
emanuele.ruoppolo@studenti.units.it

## 1 Introduction

Image classification is one of the most common and important applications of Artificial Intelligence. In this context, the FashionMNIST dataset is a dataset of grayscale garment images consisting of 60,000 training images and 10,000 test images. Each image is $28 \times 28$ pixels in size and belongs to one of 10 classes of garments. The goal of this challenge is to build a classification pipeline that leverages both kernel and Deep Learning methods to classify the images in the FashionMNIST dataset.

The project starts by using a hybrid approach between unsupervised and supervised learning. Assuming no true labels for the data, the goal is to define ten clusters (as much as the initial categories), train classification models, and compare the classification results with real ones. After testing three different Kernel Principal Component Analysis (KPCA) techniques for dimensionality reduction, the technique that apparently returned a better separation of reduced-dimensionality classes was chosen, and clusters were constructed on the reduced data. Each cluster was assigned the label of the most frequent element within it and three classification models, Support Vector Machine (SVM), Fully Connected Neural Network (FCNN) and Convolutional Neural Network (CNN) were trained to classify the data.

The results were then confornted with the real ones through the use of different classification metrics.

## 2 The dataset

The dataset used is a reduced version of the original. Only 8,000 samples were considered, including 6,000 for training and 2,000 for testing. The figure 1 shows the distribution of classes within the training dataset, which appears to be uniform so that a-posteriori balancing techniques do not have to be considered.

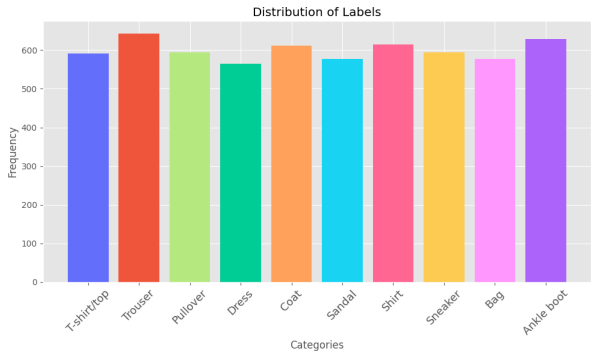


Figure 1: Distribution of training data.

Before applying a dimensionality reduction to the data, a second analysis was conducted for the purpose of evaluating its intrinsic dimension. Therefore, both the eigenvalue spectrum and cumulative variance were calculated as a function of the number of principal components considered. In figure 2 we can observe both the logarithmic spectrum and the cumulative variance explained as a function of the number of principal components. We can observe that 95% of the variance is explained by about the first 75 principal components, where there is actually the knee of the two curves. In table 2 is an extract

of the cumulative variance values for the first 150 principal components, we can see that the increase in cumulative explained variance from 75 to 150 components is only four percentage points.

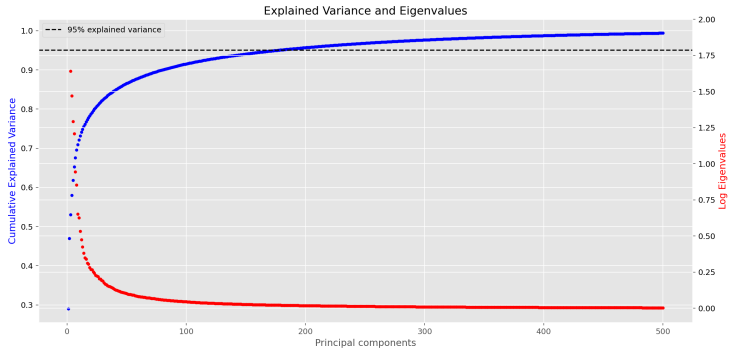


Figure 2: Explained variance and logarithmic spectrum of the data principal components.

| P.C. | Exp. Variance |
|---|---|
| 1 | 29% |
| 15 | 76% |
| 75 | 90% |
| 120 | 93% |
| 150 | 94% |

Table 1: Explained variance some of the first 150 principal components.

In addition, the intrinsic size of the data was calculated using the Two-NN algorithm from the library `skidm.id` which estimates the local intrinsic dimension of data using only the distances to the two nearest neighbors of every point. The result was a value of 15, so even lower of the supposed knee-point of the explained variance curve.

## 3 Understanding data geometry

The first step in the pipeline was to understand the geometry of the data. To do this, three different KPCA techniques were tested: the linear, cosine and radial basis function (RBF) kernels. The goal was to find the kernel that best separates the classes in the reduced space. The main evaluation was qualitative in nature, observing class separation in both a two- and three-dimensional plane. Also on a qualitative basis, the value of $\gamma$ for the RBF kernel was chosen, which was set at $\gamma = 0.01$. Figures 3 and 4 show the 2D and 3D representations of the reduced data with the three kernels. It is first interesting to note that, as one would expect, the cosine kernel returns a completely different separation than the other two techniques. In analyzing the three techinques, it was evaluated that the RBF kernel returned a sharper separation of classes, and thus was chosen for cluster construction. Excluding, in fact, the label 'coat' which turns out to be very overlapping with 'pullover,' the separation of the other classes, although using only three dimensions, turns out to be by separate bands. In contrast, it can be observed that using the other kernels the separation is less clear and the classes are more overlapping and more difficult to distinguish. Therefore, the RBF kernel with $\gamma = 0.01$ was chosen for cluster construction.

## 4 Bridging unsupervised and supervised

Once the RBF kernel for dimensionality reduction was chosen, the clusters were constructed. To do this, a new reduction to 10 principal components was, first of all, performed so that we could have a better explanation of the variance and, therefore, hopefully a better separation of classes. A Dirichlet Process Gaussian Mixture Model (DPGMM) was used to construct the clusters. The advantage of using a DPGMM over a classical model such as KMeans lies in its flexibility. Whereas KMeans requires one to specify the number of clusters a priori, the DPGMM can accommodate a variable number of components (clusters), thanks to the use of a Dirichlet process that allows one to assign an infinite number of possible

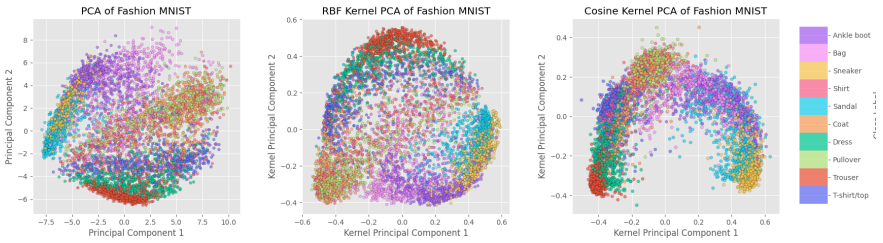


Figure 3: 2D representation of the data using different KPCA techniques.

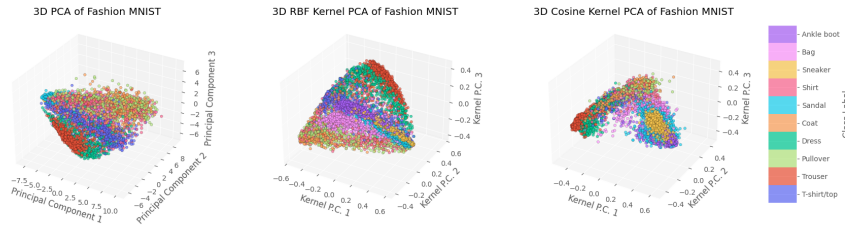


Figure 4: 3D representation of the data using different KPCA techniques.

clusters, adding new ones only when necessary. While not directly exploiting this property in this case, DPGMM was preferred because it uses Gaussian distributions to model each cluster, allowing greater flexibility in cluster shapes and sizes than KMeans, which assumes spherical clusters of equal size. The maximum number of clusters was set to 10, and the model was trained on data reduced to 10 principal components. The result of the model is 10 clusters whose label was assigned initially at random. In the figure 5 we can see the mapping between the clusters and the actual classes. We can see that some clusters are very well separated, such as cluster 2 which contains mainly elements of the class 'trouser', or cluster 9 which contains only elements of the class 'ankleboot'. Others are much more overlapping, in particular cluster 1 combines elements representing low shoes, 'sandal' and 'sneaker,' with a majority of the second class. To further analyze the distribution of data in clusters, we can look at the figure 6. We notice that clusters 3-4-6 are very overlapping and contain items from many the classes, while clusters 0-2-5-9 are much more homogeneous and contain mostly items from only one class. It can be seen, however, that despite the overlap between some of the clusters, each still contains a majority of items of a specific and different class, so it was decided not to proceed with further clustering or data balancing techniques and to label, for the next steps each cluster with the most frequent class within it.

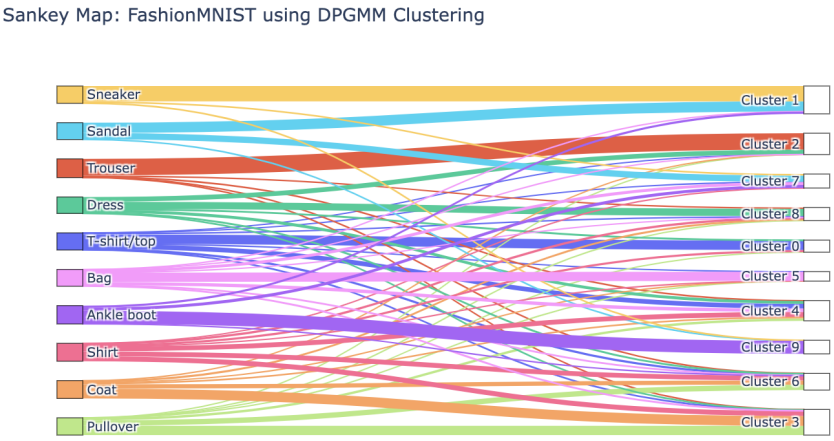


Figure 5: Sankey map of the clusters.

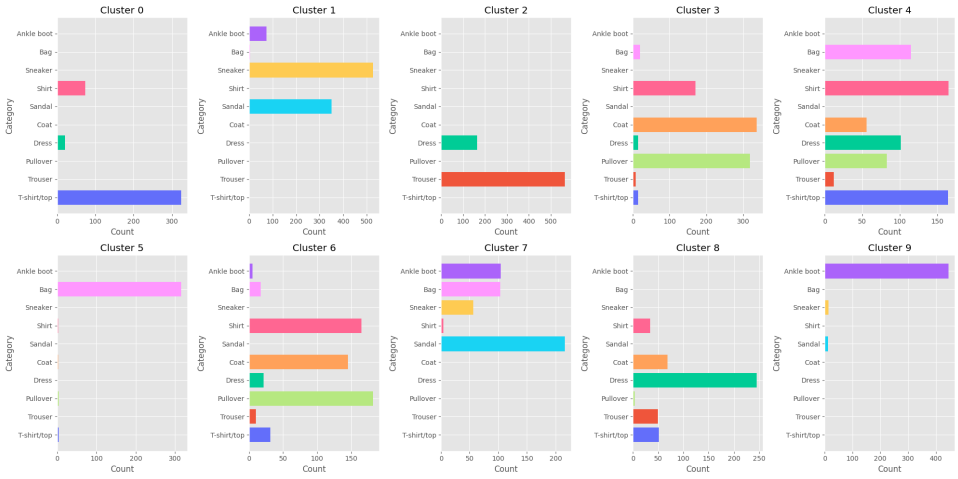


Figure 6: Data distribution among the clusters.

## 5 (Supervised) Classification

The next step was to build classification models on the data labeled with the classes, randomly assigned to the clusters. Three models were built: a Support Vector Machine (SVM), a Fully Connected Neural Network (FCNN), and a Convolutional Neural Network (CNN). The models were trained on the original images, not reduced to 10 principal components. Parameter selection for each model was performed by grid search, using, instead of static train/test split, 3-fold cross-validation. The results were evaluated by four different metrics: weighted accuracy, precision, recall and F1-score. Elementary architectures were chosen for the two neural networks, in the case of FCNN a single hidden layer, while in the case of CNN two convolutional layers, both followed by max pooling and a fully connected layer. Both networks have as input size the image size (784) and as output the number of possible classes (10). Training was performed on 20 epochs using Adam as the optimizer and crossentropy as the loss function. Below in the tables 2, 3, 4, are the parameters used for grid-search and in bold highlighted those that returned the best accuracy results, selected for the rest of the work. In addition, table 5, show the results obtained from the three models.

| **SVM** | | |
|---|---|---|
| kernel | 'linear' | **'rbf'** |
| C | 0.1, 1, 10 | 0.1, 1, **10** |
| gamma | | 0.01, **0.1**, 1 |

Table 2: SVM grid-search.

| **FCNN** | |
|---|---|
| activation | **ReLu**, tanh |
| hidden size | 128, **256**, 512 |
| learning rate | **0.01**, 0.001 |
| batch size | 32, **64**, 128 |

Table 3: FCNN grid-search.

As expected the CNN outperforms the other two models, with an accuracy of 94.6% and an F1-score of 0.945. The SVM model, despite being the simplest, also performs well with an accuracy of 93.6% and an F1-score of 0.936. The FCNN model, on the other hand, has a slightly lower accuracy of 87% and an F1-score of 0.867. The results are in line with the expectations, as the CNN model is the most complex and the one that can best exploit the spatial information of the images, while the SVM model, thanks to the gaussian kernel, is able to exploit the separation of classes obtained by the KPCA. The FCNN model, on the other hand, is the simplest, having a single layer and has the worst performance within the models, but still having satisfactory results.

| CNN | |
|---|---|
| first conv. channel size | **16**, 32 |
| second conv. channel size | 16, **64** |
| fully connected neurons | 128, **256** |
| batch size | 32, **64** |
| learning rate | 0.01, **0.001** |
| activation | ReLu, **tanh** |

Table 4: CNN grid-search.

| | **SVM** | **FCNN** | **CNN** |
|---|---|---|---|
| Accuracy | 0.936 | 0.870 | 0.946 |
| F1-score | 0.936 | 0.867 | 0.945 |
| Precision | 0.937 | 0.874 | 0.947 |
| Recall | 0.936 | 0.870 | 0.946 |

Table 5: Result of the models

However this results need now to be compared with the real labels of the data, to understand the clustering quality.

## 6 Wrap up!

In this section, we compare the results of the models previously trained on the test data with their actual data labels. To do so, we used the sameprevious metrics: weighted accuracy, precision, recall, and F1-score. In addition, to better visualize the comparison between the real and predicted labels, a confusion matrix was constructed for each model.

First, the cluster labels were mapped to the real labels so that the results could be compared. To do this, a majority voting method was used, assigning each cluster the most frequent label within it by assigning a different label to each cluster. Then the models were tested on the data and comparison metrics were calculated. The results are shown in table 6.

| | **SVM** | **FCNN** | **CNN** |
|---|---|---|---|
| Accuracy | 0.551 | 0.561 | 0.558 |
| F1-score | 0.556 | 0.564 | 0.564 |
| Precision | 0.596 | 0.610 | 0.605 |
| Recall | 0.551 | 0.561 | 0.558 |

Table 6: Results of the comparison.

The lowering of results is evident, and it can be seen that it affects all models similarly. If, in fact, in the previous case one could see a difference, albeit small, between the various models, in this case this difference is almost nonexistent. Clearly the main reason for this is the quality of the clustering, which although it allowed a discrete separation of the data but still not optimal. More generally, it is probably the very approach of reducing the dimensionality of the data that fails to capture all the necessary information of the geometry of the data and separate them optimally. While informally trying approaches such as t-SNE makes this separation much more apparent, and such an approach in more general context may be recommended if a priori knowledge of the data labels is lacking. Also, as seen initially the intrinsic data size suggested by the Two-NN model is 15, while the knee of the eigenvalue spectrum seems to be there for about 70 principal components, so probably the reduction to 10 principal components is not sufficient to capture all the information needed to enable optimal cluster separation.

Nevertheless, a level of accuracy of 60% is still not to be underestimated, as it outperforms both dummy and random classification models.

To inspect the results more closely, the confusion matrices of the models were constructed, shown in figure 7. We see some common classification issues for all models, such as the confusion between 'pullover' and 'coat', 'sandal' and 'sneaker' or 'shirt', 'pullover', 't-shirt' and 'coat'. In all of this cases we have more data misclassified than correctly classified. This reflects the initial clustering, where, for example, both clusters 3 and 6 contained approximately the same number of instances of the 'coat' and 'pullover'

classes, and the same for 'sandal' and 'sneaker' in clusters 1 and 7. Anyway, since this seems reasonable since these two classes could be considered as the most similar in the dataset, it is not surprising that the model had difficulty in distinguishing and separate them in the clustering phase. If we had to perform a more accurate analysis we could have tried ro re-balance the clusters after this first comparison, but this was not done in this case.

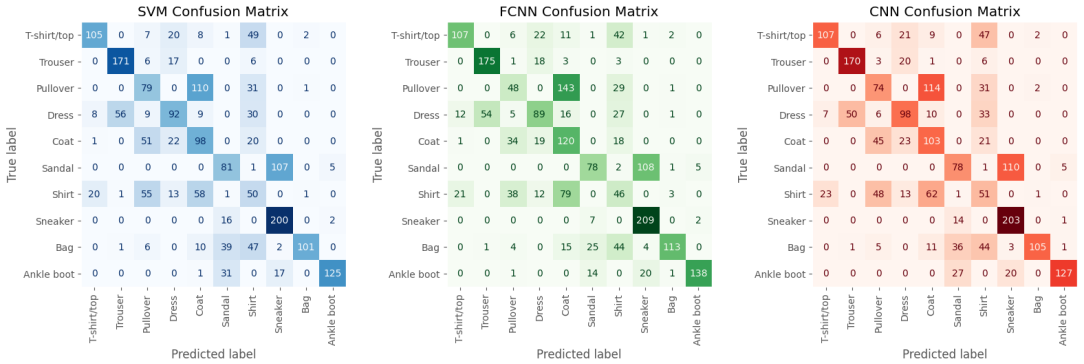


Figure 7: Confusion matrices for labelling comparison.

## 7 A fully-supervised approach

In this last section we repeat the classification process, but this time using the real labels of the data. The models, in the same architectures and hyperparameter configuration, were trained on the original train images, not reduced to 10 principal components, then tested on the test data. The results are shown in table 7.

| | **SVM** | **FCNN** | **CNN** |
|---|---|---|---|
| Accuracy | 0.888 | 0.822 | 0.897 |
| F1-score | 0.888 | 0.831 | 0.897 |
| Precision | 0.889 | 0.822 | 0.898 |
| Recall | 0.888 | 0.819 | 0.897 |

Table 7: Results of the fully supervised approach.

Here we can assess the difference in the two approaches. The fully supervised approach, as expected, outperforms the previous one. The CNN model, as in the previous case, is the one that performs best, with an accuracy of 89.7% and an F1-score of 0.897. The SVM model, on the other hand, has an accuracy of 88.8% and an F1-score of 0.888, while the FCNN model has an accuracy of 82.2% and an F1-score of 0.831. The results are in line with the expectations, with the CNN model being the best one, as in the previous case. For sure a fully supervised approach is always preferable, but in the case of lack of labels, the hybrid approach used in this work can be a good alternative, knowing that to reach fine results, a very accurate work in all the phases is needed.

Looking at the confusion matrices in figure 8, we can visualize the performances of the models. Clearly the misclassification cases are much less than in the previous case, but it is interesting to notice that even in this case the main misclassifications happened between 'pullover' and 'coat', and 'shirt', 'pullover', 't-shirt' and 'coat', while not for 'sandal' and 'sneaker'. This could confirm the intrinsic similarity of the classes, but also, maybe, the fact that the images of the dataset are not very high resolution, so the models could have difficulty in distinguishing the details of the garments.

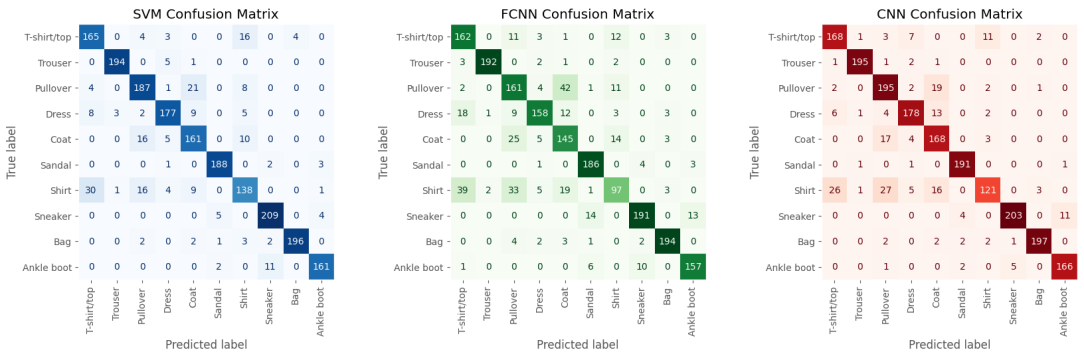


Figure 8: Confusion matrices for fully supervised approach.

Test Set Accuracy: 0.8880 Test Set F1 Score: 0.8878 Test Set Precision: 0.8886 Test Set Recall: 0.8880