Advanced Deep Learning and Kernel Methods Challenge01

Alessandro Minutolo

1 Introduction

This report aims to conduct a detailed analysis for some exercises regarding kernel methods. In the following chapters, we will see the evaluation of several approaches for the fashionMNIST dataset in order to reduce its complexity, understand the data and fit some models.

2 Understanding the Geometry

After downloading the fashionMNIST dataset, I performed some pre-processing. First of all I took only a shuffled subset of the data because it would have been really time and computationally demanding. Then I rescaled the data using a StandardScaler() from scikit-learn, which is always useful in a machine learning problem. The fashionMNIST dataset consists of 70000 greyscale images of clothing items, divided into 10 categories, each image is 28x28 pixels, resulting into 784 features per image.

The first approach I tried was to perform a usual PCA (Principal Component Analysis) to the scaled dataset. It actually didn't result to be a efficient option, as we can see from Figure 1, it can't separate the categories. Then I tried some kernel-PCA, first with a polynomial and then a Gaussian kernel. Trying to tune the parameters degrees for pylinomial and gamma for Gaussian, the most effective values were $degree = 2$ and $gamma = 0.005$. As we can see in Figure 2 and Figure 3, the most effective transformation was performed by the Gaussian-PCA, where we can clearly distinguish at least the categories 'Trouser' and 'Sneaker', but we still have problems with the other cateogories.

3 Bridging unsupervised and supervised

After performing a Kernel-PCA of 10 principal components, I tried to assign 10 labels to the resulting datapoints, applying Agglomerative Clustering, which is a hierarchical clustering algorithm that builds a tree-like structure (called a dendrogram) to represent the data's hierarchical relationships. The goal here was to leverage the reduced features to automatically identify groups of similar

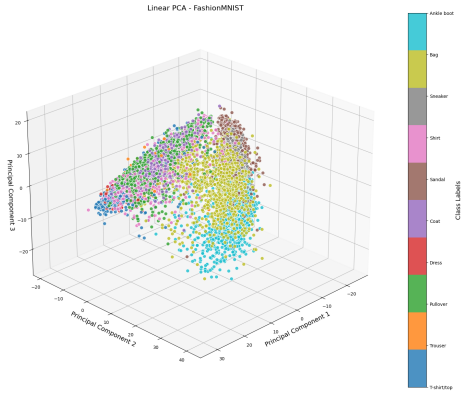

Figure 1: Linear PCA

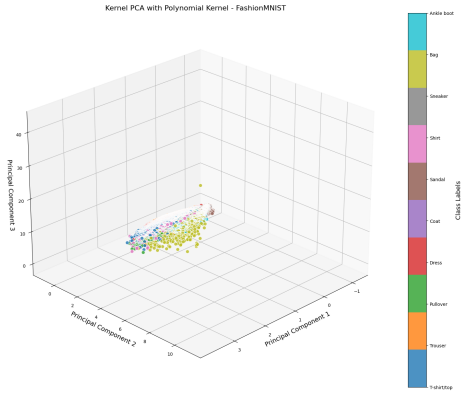

Figure 2: Kernel-PCA with polynomial kernel

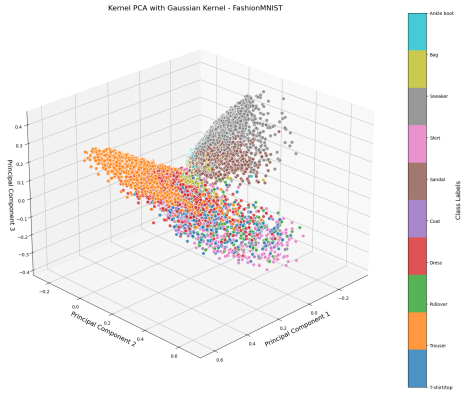

Figure 3: Enter Caption

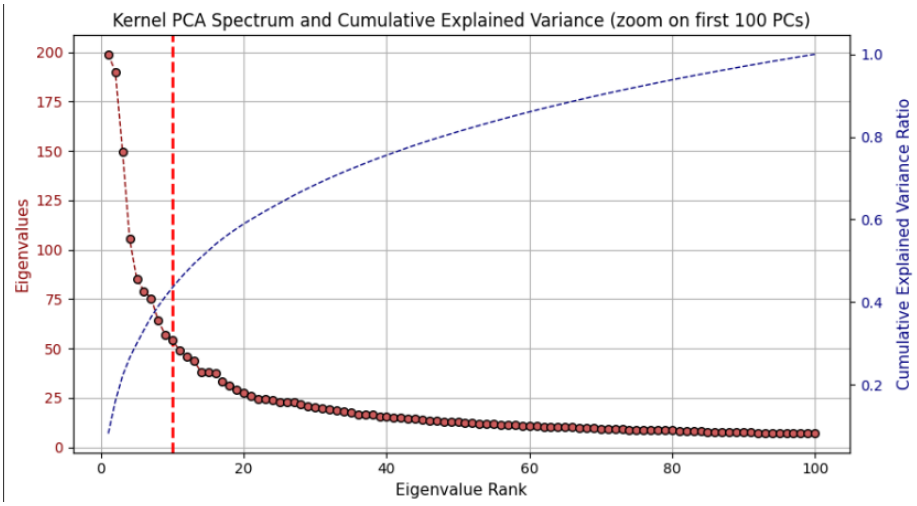

Figure 4: Eigenvalues and cumulative explained variance

images, assuming the data contained inherent groupings, hoping that Agglomerative Clustering algorithm could recognize each possible class of images (such as t-shirts, trousers, shoes, etc.).

To further evaluate the effectiveness of KernelPCA, I created a plot (Figure 4) to visualize the eigenvalue spectrum and cumulative variance explained. The eigenvalues, represented by the dark red dashed line, indicate how much variance each of the principal components explains in the data. Each point on the x-axis corresponds to a specific principal component, while the y-axis shows the magnitude of the eigenvalue for that component. Larger eigenvalues represent more important components that capture greater variance in the data, while smaller eigenvalues represent less significant components.

From the plot, we can see that the first few components have much larger eigenvalues compared to the later ones. This suggests that the most significant features of the data are captured by the first few components, and subsequent components contribute less to explaining the variance.

The cumulative explained variance is plotted as a navy blue line on the second y-axis. This curve shows the cumulative proportion of the total variance in the dataset that is captured by the first few components. The plot suggests that the first 10 components capture a substantial portion of the total variance. The line gradually increases as more components are considered, and the increase slows down as we approach the 10th component.

The vertical red dashed line at $x = 10$ marks the point where 10 principal components are selected. We can observe from the plot that by this point, a significant amount of the variance has already been captured, indicating that the selected 10 components represent the majority of the data's variability.

4 (Supervised) classification

With the new dataset composed of the original images and the new labels clustered, I implemented and evaluated three machine learning models: Support Vector Machine (SVM) and a Fully Connected Network (FCNN). As regards the SVM model, it was built choosing a Gaussian kernel with $gamma = 0.005$ and $c = 10$. It's important to notice that the gamma chosen is the same one used for the Gaussian PCA. I implemented different types of architecture for Fully Connected Networks and the best one was with ReLu as non-linearity, learning rate = 0.0005 with Adam optimizer and batch size = 64. For the CNN i used ReLu as activation function, Max Pooling with a kernel size of 2x2 and a stride of 2 for the pooling strategy, a learning rate of 0.001 and batch size = 32

The results show the highest accuracy for the SVM (0.94), which could outperform both FCNN (0.90) and CNN (0.89). Probably the reason is because SVM is using a Gaussian kernel, which was the same used for reducing the dimensionality of the data to get the labels. Even if SVM is better, we can always rely on Neural Networks in particular when we have to deal with a much bigger problem and SVM is really hard to fit.

5 Overall Evaluation

In this section we want to evaluate the overall accuracy of the pipeline comparing the prediction of the models to the real labels. In order to do so, firstly, I had to assign the true label names to the clustered categories. I sampled multiple subset of the clusters assigning one label by direct visual inspection. By doing so, I could empirically see if the clustering was done in an effective way. While most of the categories were well distinguishable, there were some like 'Bag' and 'Sandal', which were very hard to understand; eventually, I assigned them by exclusion. By performing this test of accuracy, we can see how the model doesn't work very well. The accuracy of the three models are 0.36 for SVM, 0.36 for FCNN and 0.35 for CNN. Figure 5 shows the confusion matrix for SVM. Looking at it, the categories that appear to be predicted relatively well are:

- "T-shirt/top": high true positive value (106) and relatively low misclassifications.
- "Ankle boot": true positives (191) are much higher than the false positives or false negatives.

The categories that seem to be more challenging to predict accurately are:

- "Pullover": there are several instances where "Pullover" is misclassified as other labels, and the true positive value (64) is not as high as for the well-predicted categories.
- "Trouser": similar to "Pullover", there are more misclassifications than true positives (64) for this category.

In the case of the FCNN, the categories that appear to be predicted most accurately are:

- "T-shirt/top": high true positive value (110) and low misclassifications.
- "Ankle boot": true positive value (99) much higher than false positive or false negative

The categories that seem more challenging to predict accurately include:

- "Pullover": there are several instances where "Pullover" is misclassified as other labels, and the true positive value (63) is not as high as for the well-predicted categories.
- "Sandal": similar to "Pullover", there are more misclassifications than true positives (40) for this category.

As regards the CNN, the categories that appear to be predicted most accurately are:

- "T-shirt/top": high true positive value (105) and low misclassifications.
- "Ankle boot": true positive value (104) is much higher than the false positives or false negatives.

The categories that seem more challenging to predict accurately include:

- "Dress": there are several instances where "Dress" is misclassified as other labels, and the true positive value (36) is not as high as for the well-predicted categories.
- "Bag": similar to "Dress", there are more misclassifications than true positives (29) for this category

In summary, the categories that seem to be predicted most accurately across the three models are "T-shirt/top" and "Ankle boot". The categories that appear to be more challenging to predict accurately include "Pullover", "Sandal", "Dress", and "Bag". These results reflect what we expected from the label assignment of before, with more time and resources we could change our strategy for the dimensionality reduction and for the clustering. It's clear that that clustering method is not able to understand the features of the categories.

6 Fully-supervised approach

Finally, I tried a fully-supervised approach fitting the three models built before (SVM, FCNN and CNN) giving them the original data. From the confusion matrices in Figure 8, 9 and 10, we see that

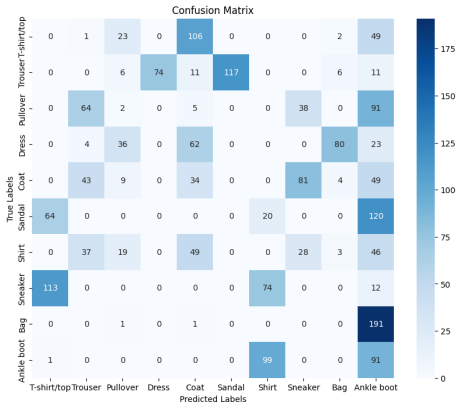

Figure 5: EOverall Evaluation for SVM

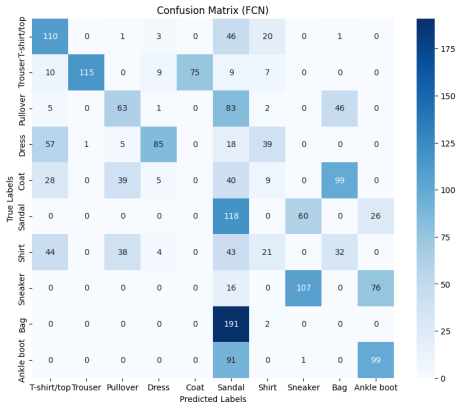

Figure 6: Overall Evaluation for FCNN

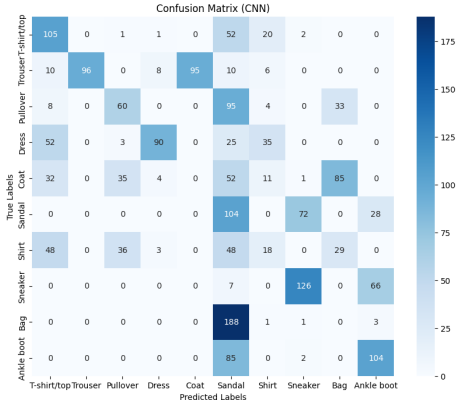

Figure 7: Overall Evaluation for CNN

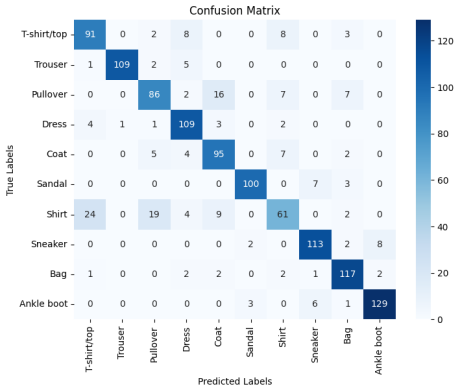

Figure 8: Fully supervised SVM

- "Pullover" category has relatively high misclassifications, with the true positive values being lower compared to some of the better-predicted categories.
- "Dress" category also exhibits higher misclassifications, with the true positive values being lower than desired.
- "Bag" category seems to be another challenging one, with more instances of it being misclassified as other labels.
- "Sandal" category has a higher rate of false positives and false negatives in the CNN and FCN models, indicating it is not being predicted as accurately as some other categories.

On the other side, the models seem to perform better on categories like "T-shirt/top", "Ankle boot" across the different confusion matrices.

These results show that the new models have the same difficulties in predicting certain categories as in previous hybrid models

The new models demonstrated significantly higher performance compared to the previous ones with accuracy respectively of 0.84 for SVM, 0.85 for FCNN and 0.87 for CNN. This result is not particularly surprising and can be attributed to two primary factors. First, in the fully supervised approach, the model was fed with 728-pixel images, which inherently carry far more information than the 10-dimensional vectors used at the very beginning of the hybrid pipeline. Second, the fully supervised approach involves training the models directly on the data, whereas the hybrid pipeline involves multiple stages, each introducing potential sources of error.

In conclusion, the hybrid pipeline was a very fascinating strategy, which I could have spent more time working on. The results are not very satisfying, but I am sure that tuning some parameters and performing other strategies, we could have a far better result.

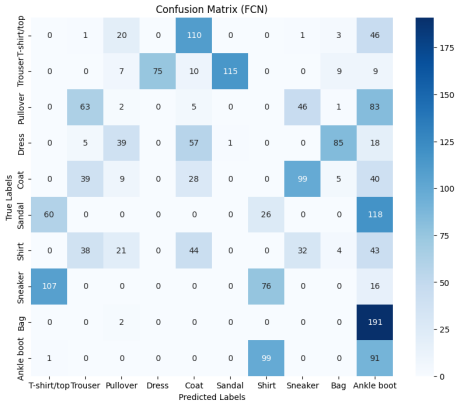

Figure 9: Fully supervised FCNN

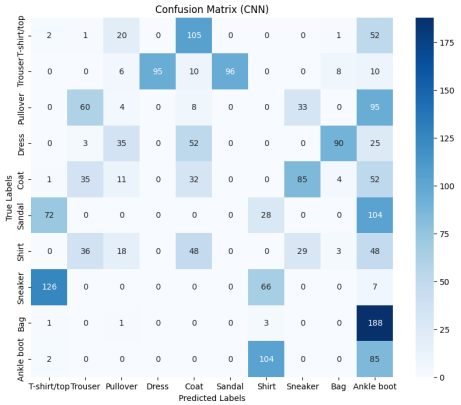

Figure 10: Fully supervised CNN