IT UNIVERSITY OF CPH

Machine Learning

BSMALEA1KU

Project Report

Jakub Mráz	Malthe Nielsen	Mark Assejev
jamr@itu.dk	lamn@itu.dk	maass@itu.dk

03/01-2024

Contents

Abstract

This report presents a project in machine learning, focusing on the classification of images of clothing. Utilizing a dataset of 15,000 greyscale images from Zalando, the project employs a variety of classifiers and dimensionality reduction techniques.

Key efforts include the manual implementation of a Linear Discriminant Analysis function and a Naive Bayes classifier using Python and NumPy, alongisde the training of an SVM and Machine Learning model.

The report then compares the three classifier models and delves deeper into the impacts of the presence of a hard to isolate class.

Introduction

Image classification is an important field within Machine Learning and one with numerous real-world applications, ranging from simple pattern recognition to facial recognition software.

In this project, we will focus on the classification of images of clothing into 5 classes given by our dataset. The dataset itself is a set of 15000 greyscale 28x28 images from the website Zalando (Graversen 2023).

Our main goal is to train 3 classifiers and combine feature extraction with dimensionality reduction techniques to achieve the best accuracy. It is mandatory for us to implement a Linear Discriminant Analysis dimensionality reduction function and a Naive Bayes classifier from scratch, using only Python and NumPy.

Our personal goal is to gain a deeper understanding of Machine Learning models, the differences between them, and the nuances of image classification.

Dataset Description

The dataset came pre-split into a training and test set consisting of 10,000 and 5,000 images respectively.

The dataset comes with labels, of which there are 5 classes, given as integers from 0-4. These classes represent the type of clothing these images depict. The classes are as follows:

Class	Description
0	T-shirt/Top
1	Trousers
2	Pullover
3	Dress
4	Shirt

The dataset is split evenly among the classes, meaning 2,000 training images and 1,000 test images per class.

Each image is 28x28 pixels and in gray-scale. The colours of the pixels are given as integers with values from 0-255. Below, we show a sample of the images alongside their corresponding classes, as provided in the project description (Graversen 2023). Note that in the actual dataset, all images have a black background.

Figure 3.1: A random sample of 10 images from the training dataset.

We normalised the colour values to range from 0-1 instead for our Neural Network, since it is usually good practice to do so (Brownlee 2020). When we applied the same normalization to our other classifiers, the results slightly worsened; therefore, we retained the original values in those instances. This discrepancy might arise because Neural Networks are generally more sensitive to scaling (Brownlee 2020). In contrast, other models do not significantly benefit and might even lose precision due to floating point calculations.

Exploratory Data Analysis

The images themselves were given as a flat 1-dimensional array in the .npy file format. We used a script to convert the array elements into .jpg images in order to visually examine them. We noticed that class 4 (Shirt) often resembled other classes, making it difficult to distinguish. This similarity would end up posing a challenge for our classification models as well.

Next, we used a script to generate a mean image for each class by averaging all images, resulting in 5 representative images. These are shown below.

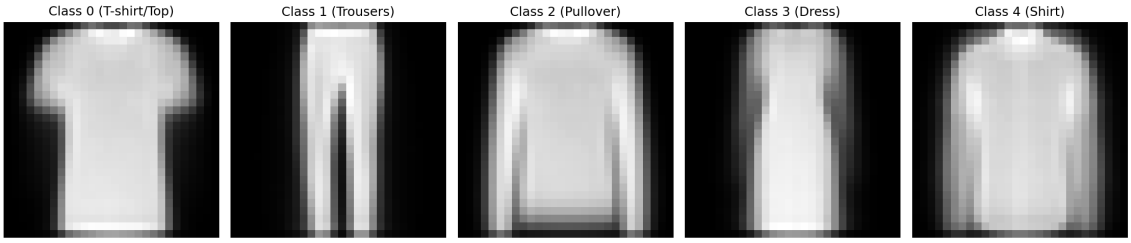

Figure 4.1: Mean image of each class.

We found that each of the first 4 classes has a distinct shape, while the Shirt class visually blends into the T-shirt/Top and Pullover classes. One can spot that the short sleeve pattern is brighter on the Shirt image, indicating that this class contains pieces with both short- and long-sleeved shirts. What generally separates shirts from T-shirts is "a collar, sleeves with cuffs, and a full vertical opening with buttons or snaps" (Wikimedia Foundation 2023b). Unfortunately, due to the low resolution of the images, details like buttons or cuffed sleeves were not visible, and collars were visually similar across Shirt and T-shirt classes.

An immediately distinctive feature is that the Trousers class has a large black gap in the middle, due to the nature of the clothing type. This ended up as one of our features extracted for classification.

Although individual clothing images may exhibit noticeable patterns, these characteristics are not apparent in the mean images. Therefore, shape analysis was crucial for classifying these images.

Feature Engineering and Selection

As previously mentioned, our features ended up consisting mostly of values centered around the shapes of the clothes.

5.1 Features

5.1.1 Eigenshapes

The first set of features we used were the 'eigenshapes,' derived from the dataset's individual pixels. Derived from 'eigenface', which is the result of a commonly-used technique used for facial recognition (Acar 2021), these eigenshapes are the resulting eigenvectors, from a PCA operation applied to the pixels, which capture the most variations within the shapes. Combining these images in various proportions can be used to reconstruct the original images.

The number of eigenvectors we selected for the output was 50. Our rationale behind this was that it was already a steep reduction from the 784 (28x28) original pixel values, and, while this could be reduced further, we would eventually perform another round of dimensionality reduction on the combined feature set, so we did not want to lose precious information during this step. Below is the visualisation of the first couple of eigenshapes:

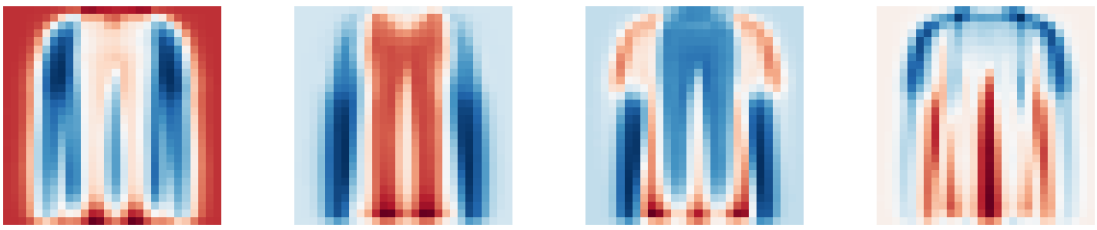

Figure 5.1: Eigenshapes (principal components) visualised.

The red pixels indicate positive deviations while the blue pixels indicate negative deviations, with white being the neutral field. With this in mind, we can interpret the first image as giving shape to trousers while removing the other parts that would indicate a top piece. The second image is an even stronger shape of trousers while the third image seems to give shape to the short sleeved pattern. To reconstruct the original image, each vector is assigned a weight, amplifying or diminishing features to create the final image.

5.1.2 Pixel Differences from Mean Image

Another feature we used was the difference between the given image and the mean image for each class. Certain classes have a distinct shape, so this feature was intended to further solidify the divisions between the classes and help the models differentiate between them.

5.1.3 Histogram of Oriented Gradients

A common feature in computer vision and image processing, HOG features effectively capture the form and structure of local areas within images (Tyagi 2021).

5.1.4 Ratio of Black Pixels Alongside the Y Axis

Devised during our exploratory data analysis, this feature captures the ratio of black pixels along the Y-axis in the middle of the X-axis, representing the gap between the trouser legs.

5.1.5 Other features

Besides those mentioned, we also experimented with using contour features and Fourier descriptors, but those ended up not making a difference in the classifiers' accuracy, so we decided to remove them.

5.2 PCA

With our full feature set extracted, we decided to do another round of PCA on all the features, since it was mandatory for us to compare the results of PCA and LDA with scatterplots. This time, the number of principal components selected was 10.

5.3 LDA

We implemented the Linear Discriminant Analysis feature reduction function from scratch using NumPy, later verifying it with the Sklearn library. We examined the scatterplots of both and found them to be identical, barring the rotation, which was due to random eigenvector selection. The number of linear discriminant values was required to be 2, so we did not experiment with different values.

To manually implement LDA, we calculated the mean feature vectors for each clothing category, which laid the groundwork for class-based feature analysis. We then calculated the two scatter matrices, which capture the variance within and between each class. After, we solved the eigenvalue problem for $(S_W^{-1} S_B)$ to identify optimal class-separating directions.

Finally, we selected two eigenvectors based on the two highest eigenvalues to form a new feature space, which we then projected our data onto.

5.3.1 PCA and LDA comparison

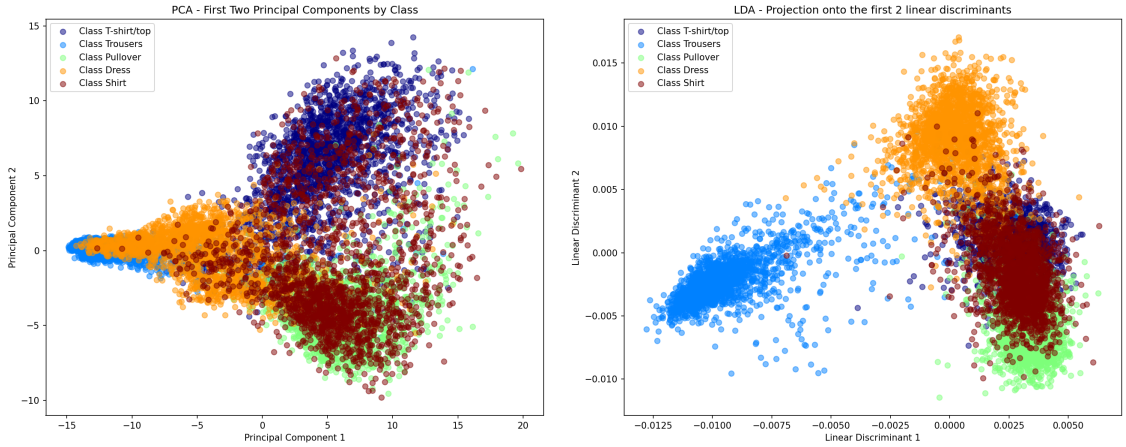

Figure 5.2: Scatterplots of the first two Principal Components and Linear Discriminants respectively.

In the two plots above, the different classes are clearly clustered together, but not as perfectly as one would like. The PCA plot shows us that the orange (Dress) class is heavily overlayed on the blue (Trousers) class, implying the first two principal components are not very useful for differentiating between the two. This is not the case in the LDA plot, where the blue and orange classes are clearly separated. In contrast, the PCA plot shows that the dark blue (T-shirt/Top) and green (pullover) classes are clearly separated, while in the LDA plot, they are not as far apart. Both plots however show us that the red class (Shirt) is heavily overlayed over dark blue (T-shirt/Top) and green (pullover), echoing our own analysis from EDA. The two plots are available in bigger sizes in the Appendix.

Following this comparison, we explored the results of sequentially applying PCA followed by LDA to the features. The result seemed to be the best of both worlds for us, where the red class was still heavily overlayed over others, but the other 4 classes were all separated from each other.

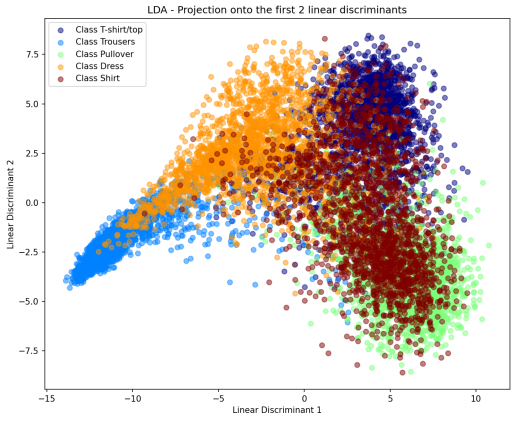

Figure 5.3: Scatterplot of both PCA and LDA applied one after other.

Model Development

6.1 Naive Bayes' Classifier

The Naive Bayes classifier is built upon the Naive Bayes' theorem with an assumption of independence among predictors. In other words, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

We implemented this classifier from scratch using only NumPy and employed sklearn.metrics functions to evaluate the model's accuracy. The resulting metrics were cross-validated against the GaussianNB classifier from Sklearn, yielding identical outcomes. We used the Gaussian version in both cases since our features are continuous.

To implement the classifier manually, we started by computing the prior probabilities for each class based on their frequency in the training dataset. We then calculated the mean and variance of features to model the feature distributions in each class. Afterwards, we implemented a PDF in to estimate the likelihood of a feature value given its class mean and variance based on the Gaussian distribution.

The log probabilities and posterior probabilities, alongside the PDF were then used to classify the images, with the class with the highest probability being chosen.

The primary advantage of the Naive Bayes classifier is its speed, attributable to simplified calculations under the assumption of feature independence. However, this assumption often does not apply in real-life scenarios, typically resulting in the model not being the most accurate (Wikimedia Foundation 2023a).

This model was trained on the feature set with PCA and LDA applied, shrinking the features to 10 and 2 components respectively. This was required by the project.

6.2 Support Vector Machines

Support Vector Machines (SVM) is a powerful and versatile supervised machine learning algorithm used for both classification and regression. Our implementation utilizes the sklearn library, which includes an SVM module.

SVMs work by finding a hyperplane that divides a dataset into two or more predefined classes, aiming for the widest possible margin between them.
Before training, the SVM model is initialized with specified parameters, such as the kernel type (linear, rbf, etc.), regularization parameter C, among others.

Initially, we trained the SVM model on the features with the same transformations as the Naive Bayes classifier; but after testing, the full feature set without these two dimensionality reductions ended up having the best performance. This is

likely due to the SVM model being well-equipped for dealing with multi-dimensional datasets.

We experimented with each kernel and the hyperparameters and found that in our case, the highest success rate was achieved with the RBF kernel. We then turned the C parameter up, which controls the tolerance to overfitting, and found the sweet spot to be at 2, since moving up to 3 produced negligible results. To make sure we were not overfitting, we measured and compared the accuracies for both the training and test datasets. The model's accuracy on the training dataset was higher by 0.04. It being higher was to be expected and we found the value to be low enough to conclude the model was not overfitted.

6.3 Neural Network Classification

As our third classification method, our team decided to utilize the TensorFlow library and explore the performance and viability of a neural network for our dataset.

Neural network classification is a method in machine learning where a neural network is used to assign class labels to input data.

It usually consists of:
- An input layer (matching the data features)
- One or multiple hidden layers (for feature extraction and learning)
- An output layer (for class predictions)

The training of a Neural Network consists of 3 steps, which are repeated multiple times (generations) in the next order:
- Forward propagation (passing data through the network)
- Loss calculation (measuring prediction error)
- Back propagation (adjusting weights to minimize error)

The model we chose is a typical neural network model structured for image classification problems (Al-Saffar, Tao, and Talab 2017). Our model includes two convolutional layers; the first serves as the input layer with 32 filters, followed by a second layer with 64 filters. The first layer extracts the most basic features, while the second one works on a higher level to detect more complex features, building upon the features detected by the first layer. One MaxPooling layer reduces the spatial dimensions (width and height) of the input, which helps in reducing the computational load and memory usage. One Flatten layer is necessary because the next dense layers expect input in a flat, 1D format, so it converts the data from two dimensions to one. Lastly, two Dense layers (one with 128 neurons and one with five neurons as the output layer) perform complex computations, essentially making the final decision based on the learning done.

Since the Neural Network finds its reasoning for classification on its own, it was trained on the raw (normalised) images, without any feature extraction or dimensionality reduction.

Model Evaluation and Comparison

Model	Accuracy	Precision	Recall	F1 Score
Naive Bayes	0.6828	0.6666	0.6828	0.6671
SVM	0.8708	0.8746	0.8708	0.8722
Neural Network	0.89	0.89	0.89	0.89

Table 7.1: Performance Metrics of Different Models

Judging by the metrics, the finely tuned Neural Network method emerged as the most suitable solution for the classification problem. Before identifying the optimal layer and neuron configuration, the best accuracy achieved was close to random guessing, or 0.5. After experimenting with various layer combinations and epochs, we achieved an accuracy of 0.89.

The Naive Bayes classifier performed the worst, followed by the SVM-rbf classifier, whose performance was very close to that of the Neural Network.

In addition to those four measures, we also looked at the confusion matrices for each model. Note that in all these figures, the classes are numbered as x+1:

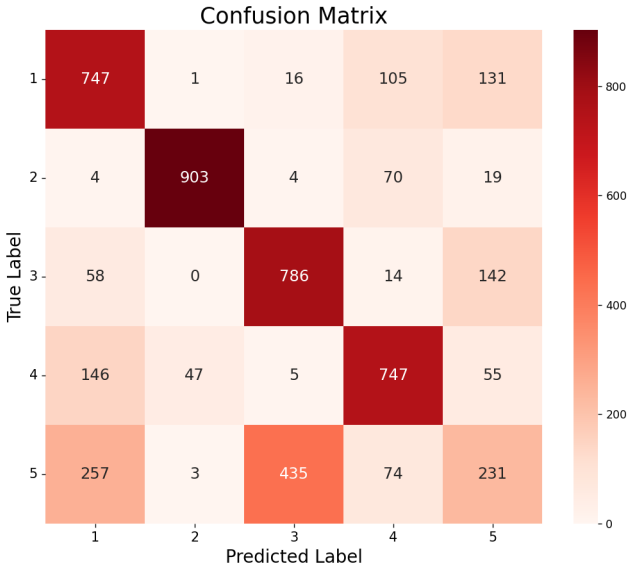

Figure 7.1: Naive Bayes classifier confusion matrix.

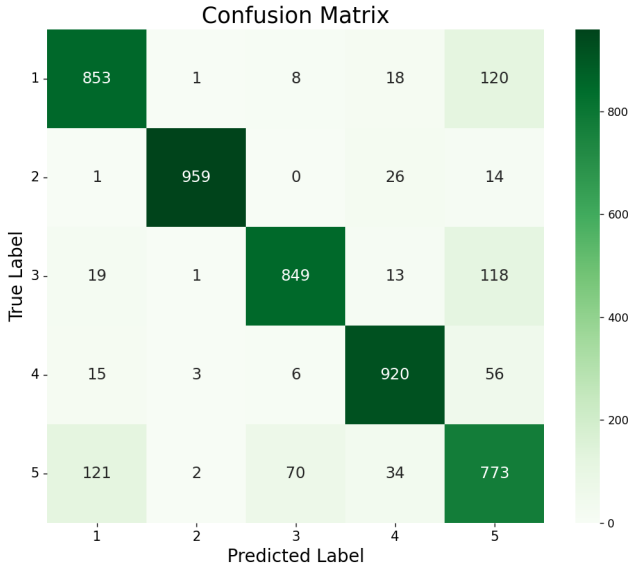

Figure 7.2: SVM classifier confusion matrix.

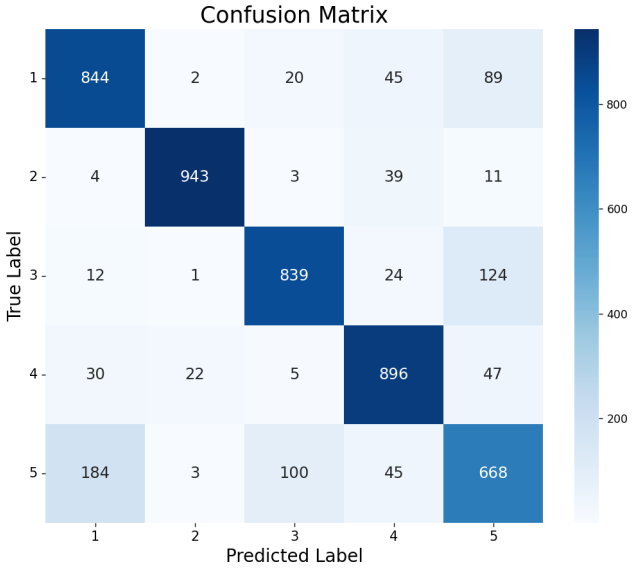

Figure 7.3: Neural Network classifier confusion matrix.

From these figures, we can see that all 3 classifiers performed the best on the Trousers class and the worst on the Shirt class, with the Naive Bayes classifier predicting more Shirts as Pullovers than as Shirts. This ended up being the downfall of this model, since from its matrix, we can see that its accuracy in predicting the other classes was around 75% (1000 images per class; ˜750 correctly predicted yields 75%).

One reason for the divergent results could be the differently processed feature sets going into each model. To make sure this wasn't the case, we tested both the SVM and Neural Network models using the same PCA $\rightarrow$ LDA transformation. The F1 of SVM came out as 0.78, while the F1 of the Neural Network came out as 0.87. This means the performance ranking stayed the same but both models ended up performing worse.

Discussion

Throughout the whole project, we noticed that the issue of the Shirt class being too similar to the T-shirt/Top and Pullover classes kept coming back to haunt us. First, we could not think of a good feature to extact to differentiate this class from the rest like we did with Trousers; next, not even the eigenshape extraction and features such as HOG and average shape difference were able to fully distinguish it, as revealed by the PCA and LDA scatterplots. This ultimately led to the Naive Bayes classifier underperforming. The SVM and Neural Network classifiers dealt with this issue much better, though it was still their weak point.

Considering we ourselves had issues with differentiating this class from the rest, we came to the conclusion that there was not much more we could have done about this and that better resolution images would be needed for a more accurate classification due to the inherent similarities between the three types of clothing. Perhaps a Neural Network could be used to extract some kind of Shirt-exclusive feature.

We think that the reason the Naive Bayes model performed worse overall is that the assumption of feature independence did not end up holding true in this scentario, as is often the case. The SVM and Neural Network models are equipped to capture complex relationships between features and thus performed better in every scenario.

Something that crossed our minds when looking at the training and test datasets was that the ratio was 2:1, which seemed a bit high on the test side. An argument could be made here for using a portion of the test data in the training set instead to further improve the models' accuracy, though we did not do this.

We believe that further research on this topic should definitely focus on identifying features unique to the Shirt class, which could potentially improve the accuracy of some of the models above 90%.

Conclusion

In this project, our goal was to fit machine learning models to classify images of clothing. This was done on a dataset consisting of 28x28 pixel images spread evenly across 5 classes. We extracted features and utilised PCA and LDA for dimensionality reduction.

We trained 3 models: a manually implemented Naive Bayes model, a Support Vector Machine (SVM) utilizing the RBF kernel, and a neural network model developed with TensorFlow. The Naive Bayes model ended up performing the worst (F1 score: 0.66), though still better than a coin toss. In contrast, the SVM (F1 score: 0.87) and the neural network (highest accuracy and F1 score of 0.89) were both able to reliably classify the majority of the images.

The biggest obstacle we encountered during this project was distinguishing the Shirt class from the T-shirt/Top and Pullover classes. We were unable to properly distinquish this class from the rest and our models suffered as a result. We believe this area most warrants further research.

To conclude, this project provided us with a valuable experience and insight into image classification and machine learning as a whole. We feel that we now have a deeper understanding and appreciation for AI technologies and their power to provide automated solutions to real-life problems.

Appendix

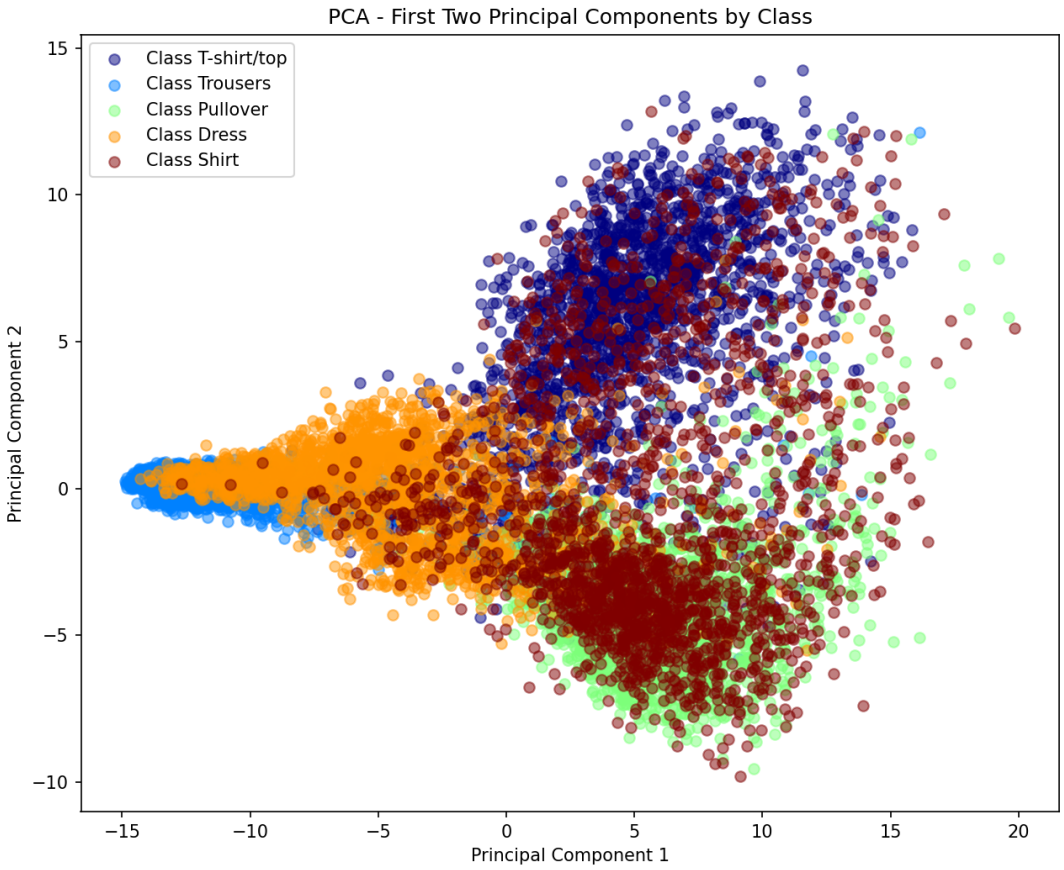

Figure 10.1: Scatterplot of the first two Principal Components.

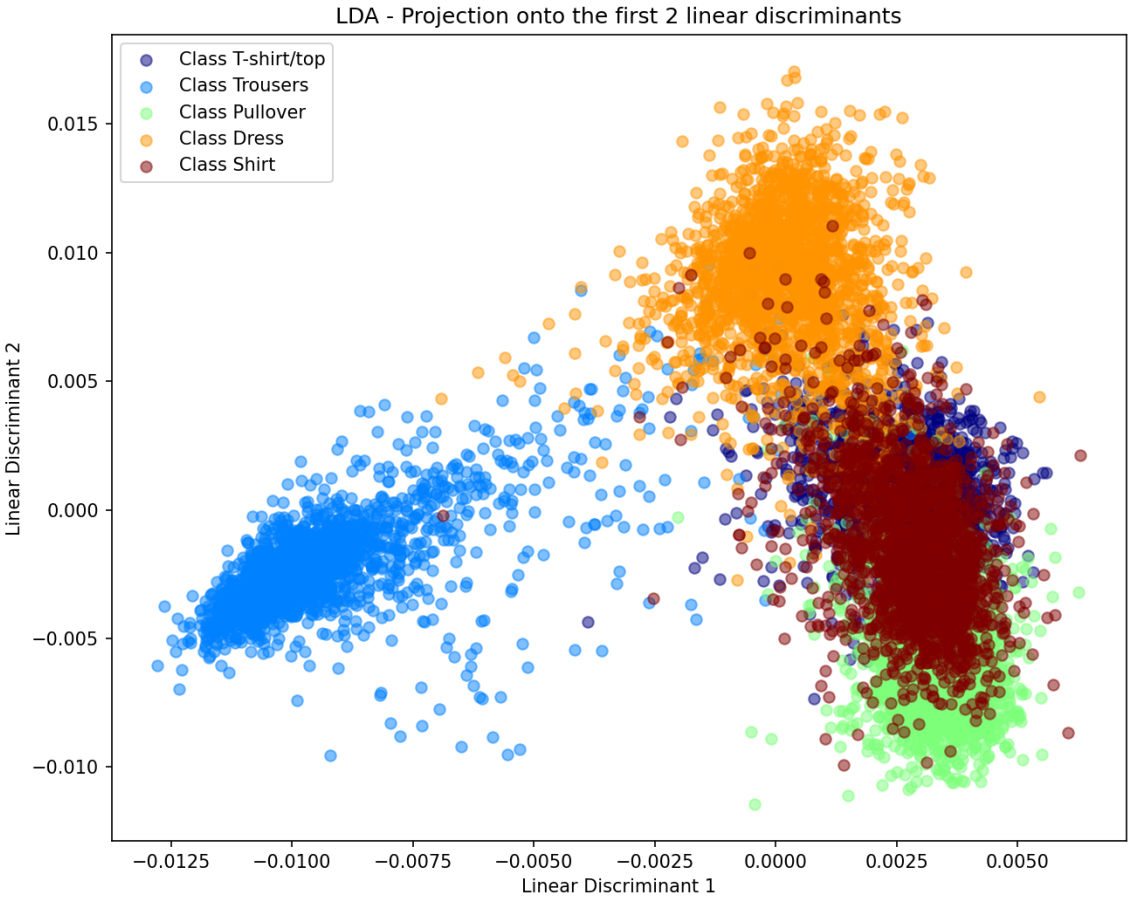

Figure 10.2: Scatterplot of the first two Linear Discriminants.

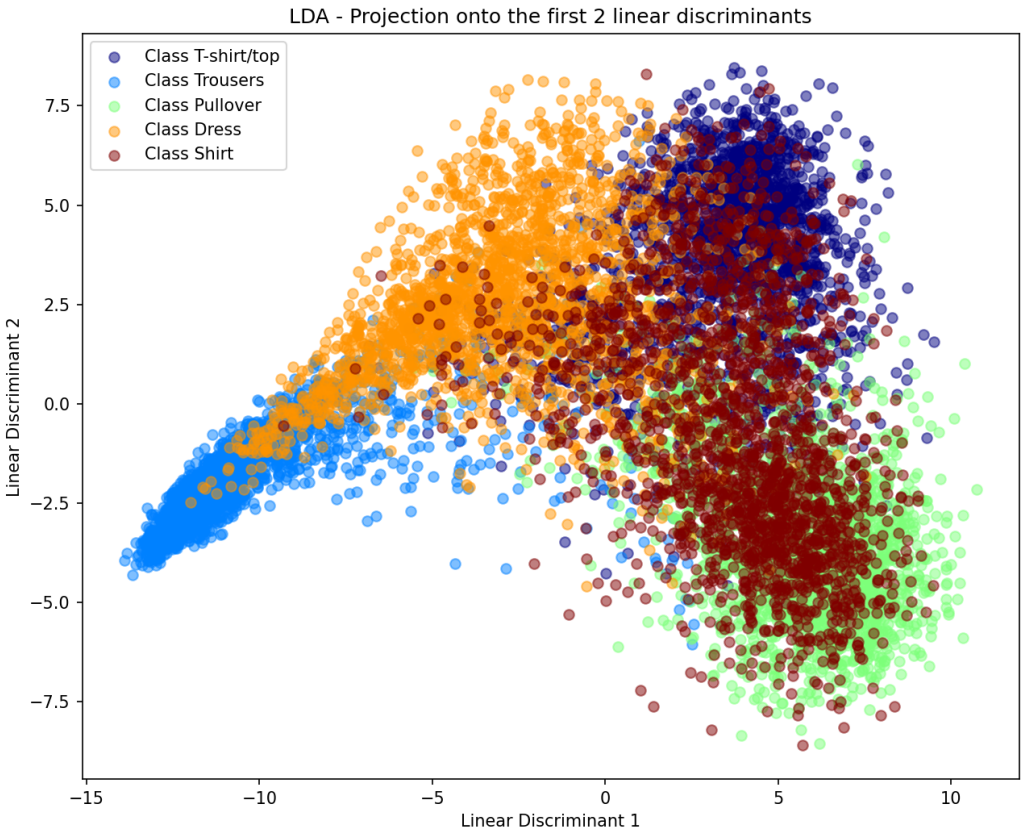

Figure 10.3: Scatterplot of both PCA and LDA applied one after other.

Bibliography

Acar, Nihat (Aug. 2021). *Eigenfaces: Recovering Humans from Ghosts*. Medium. URL: https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184.

Brownlee, Jason (Aug. 2020). *How to Use Data Scaling to Improve Deep Learning Model Stability and Performance*. MachineLearningMastery.com. URL: https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/.

Graversen, Therese (2023). "Exam Assignment: Machine Learning (BSc Data Science)". IT University of Copenhagen, Fall 2023 semester.

Al-Saffar, AAM, H Tao, and MA Talab (2017). "Review of deep convolution neural network in image classification". In: *2017 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*. IEEE. Jakarta, Indonesia, pp. 26–31. DOI: 10.1109/ICRAMET.2017.8253139.

Tyagi, Manish (July 2021). *HOG (Histogram of Oriented Gradients)*. Medium. URL: https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f.

Wikimedia Foundation (Dec. 2023a). *Naive Bayes Classifier*. Wikipedia. URL: https://en.wikipedia.org/wiki/Naive_Bayes_classifier.

– (July 2023b). *Shirt*. Wikipedia. URL: https://en.wikipedia.org/wiki/Shirt.