IT UNIVERSITY OF COPENHAGEN

# Clothing Image Classification

January 3rd, 2024

**Raivis Lickrastins**
rail@itu.dk

**Tobias Heine Zschiegner Michelsen**
tomi@itu.dk

**Daniil Strelan**
dastr@itu.dk

This report is presented for the course
BSMALEA1KE, Machine Learning
3rd semester of BSc, Data Science
IT University of Copenhagen
Denmark

# 1 Introduction

Since the 1960s, image classification in the context of machine learning has been researched and improved. Lawrence Robert is credited as the founder of this technique, having written his doctoral thesis on the matter [1]. One of the topics researched on the authors doctoral thesis, and the most relevant for this report, was how a 3-dimensional representation could be converted into a 2-dimensional one. In this day and age, image classification plays an impactful role in the lives of the generalized public. Some examples of how it is used in an every day context include the defense, retail/e-commerce, healthcare, security and many other industries. The following report investigates image classification through the lens of the Fashion-MNIST dataset. The goal of the report is to:

1. Determining the category of clothing from an image by applying methods including Principal Component Analysis and Linear Discriminant Analysis for a dimensionality reduction of the dataset.

2. Implementing three classifiers; Naive Bayes, K-Nearest Neighbours, and Convolutional Neural Network

First and foremost, this report provides an overview of the data, followed by a section on Exploratory Data Analysis, and the implementation of the three classifiers subsequently. Finally, the results are interpreted and discussed.

# 2 Data Description

In the following sections, an overview of the dataset, including its origin, characteristics, and quality of the data will be provided. Additionally, the modifications applied to the data through data augmentation and normalization will be discussed.

## 2.1 Data Source and Characteristics

The dataset used stems from the Zalando website and consists of 15,000 labeled images of clothing. Each image is a grayscale 28x28 picture of either a t-shirt/top, trousers, a pullover, a dress, or a shirt. The images are split into two subsets: a training set containing 10,000 images and a test set containing 5,000 images. Both subsets are provided as NPY formatted files where the rows represent the different pieces of clothing. The first 784 columns are the pixel values of the corresponding piece of clothing, each taking an integer value between 0 and 255. The last column, number 785, contains an integer between 0 and 4, representing the category of clothing.

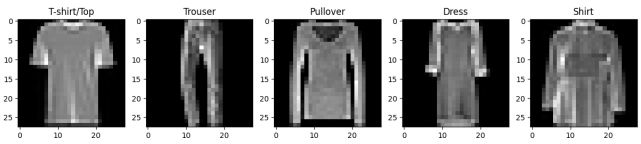



Figure 1: Overview of the Clothing Categories

## 2.2 Data Quality

Furthermore, the quality of the dataset is satisfactory. There is no missing data in the dataset, as every piece of clothing has 784 pixel values present, along with an integer indicating the clothing category. The distribution of clothing categories is relatively uniform, as each category in the training set has around 2000 entries. T-shirts/tops, classified as Category 0, is the most frequently represented item in the dataset, having 2033 entries. On the other hand, Trousers, classified as Category 1, is the least prevalent, with a count of 1947 entries (Figure 1.). None of the categories are significantly overrepresented or underrepresented, which is vital to avoiding biases towards a particular category.

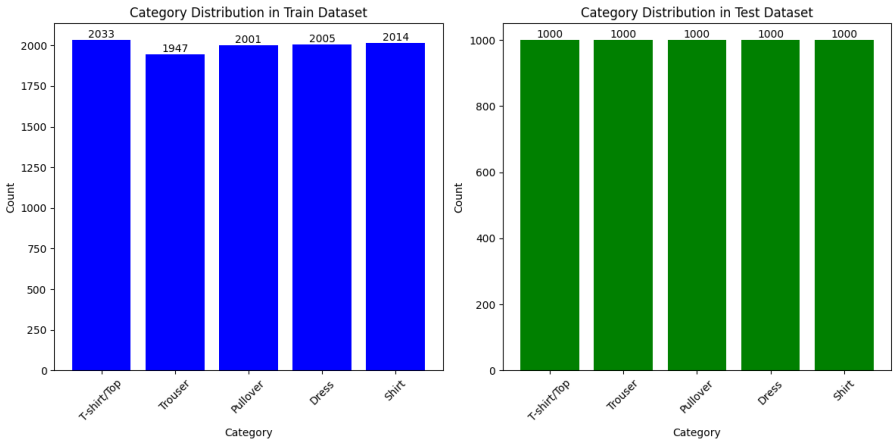



Figure 2: Histograms of the Category Distributions

## 2.3 Data Augmentation

With the intention of creating a more robust model capable of accurately classifying the given piece of clothing, the data has been augmented. Specifically, data augmentation is being applied in the context of training a Convolutional Neural Network. The images, or rather the layouts of the pixels, are slightly altered according to these parameters:

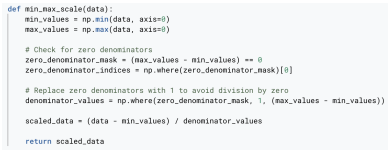
1. Horizontal Flipping: The image can be randomly flipped horizontally during training.
2. Rotation Range: The image can be randomly rotated within a 10 degree range in either direction. This accounts for a scenario, in which the clothing item is oriented differently.
3. Zoom Range: This parameter randomly zooms up to 20% inside the images. This accounts for images that were taken with varying proximities to the piece of clothing.
4. Shear Range: The image gets distorted by changing the perspective. This accounts for pictures of clothing that were taken from different angles.

The intention behind these changes to the images is to improve the machine learning model by enlarging the dataset and by reducing potential overfitting, where the model would perform significantly worse on the testing data compared to on the training data. In addition, the parameters try to simulate real-world conditions, in which the images of clothing are not always taken in the same consistent state.

## 2.4 Data Normalization

Another important step in the preprocessing of the dataset is to normalize the data. By normalizing data, you achieve a standardized data format, ensuring a consistent scale across all features. In our case, the data consists of pixel values ranging from 0 to 255. By normalizing the data, the values are scaled down to be between 0 and 1. This has several advantages. First of all, by having a consistent range, features with a larger numerical range will not disproportionately influence the model's training process. Secondly, neural networks use gradient-based optimization for testing. By having normalized data, the gradient is the same across all features, allowing for faster and more accurate training. Lastly, normalization reduces the model's sensitivity to the scale of features. For the K-Nearest Neighbours (KNN) algorithm it was only important when we used PCA components and LDA variables for classification where the distance between data points is even more essential with regard to finding the nearest neighbor. Having different features on differently scaled ranges would affect the distance computation and therefore also the accuracy and reliability of a model.

Originally, the images were represented by a 1-dimensional vector containing 784 pixels. However, as training a Convolutional Neural Network requires the data to be formatted as a 4-dimensional tensor, the group reshaped the data.



```
def min_max_scale(data):
    min_values = np.min(data, axis=0)
    max_values = np.max(data, axis=0)

    # Check for zero denominators
    zero_denominator_mask = (max_values - min_values) == 0
    zero_denominator_indices = np.where(zero_denominator_mask)[0]

    # Replace zero denominators with 1 to avoid division by zero
    denominator_values = np.where(zero_denominator_mask, 1, (max_values - min_values))

    scaled_data = (data - min_values) / denominator_values

    return scaled_data
```

Figure 3: Min-Max Scaling

```
# Standardize the data
scaler = StandardScaler()
scaler.fit(pixels_train)
```

Figure 4: Standard Scaler

# 3 Dimensionality Reduction

As each image in the dataset is represented by 784 pixels, the dataset is high-dimensional. High-dimensionality is a problem for machine learning algorithms due to the curse of dimensionality [2]. In other words, when the dimensionality, or the number of features increases, the volume of the space will increase exponentially. Consequently, the data becomes sparse and therefore difficult to draw conclusions from. Furthermore, high-dimensionality can be problematic in terms of computational complexity and overfitting of the data. By reducing the number of features, machine learning algorithms can perform more efficiently and visualizations can be plotted in 2 or 3 dimensions. This begs the question, how can the features of the dataset be reduced while retaining the significant information? In the subsequent sections, the paper will investigate two methods for dimensionality reduction; Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA).

## 3.1 Principal Component Analysis (PCA) - Method Overview

Principal Component Analysis (PCA)[3] is a technique that can be utilized to reduce the dimensionality of large datasets, while preserving most of the information. PCA can be broken down into several steps. The first step is to standardize the data, as to prevent features with larger ranges from disproportionately influencing the model. The next step is to compute a covariance matrix, which can help to identify correlations between features. Covariance is a measure that indicates how two features are correlated. The covariance of two features is defined as the expected value of the product of their deviations from the individual expected values, or in simpler terms:

$$\operatorname{cov}(X, Y) = E\left[(X - E[X])(Y - E[Y])\right]. \tag{1}$$

When two features change in the same direction, the covariance is said to be positive. Reversely, if one feature decreases while the other increases, the covariance is negative. These correlations are important, as they can show how similar the information is between two features. By removing highly correlated features, the dimensionality of the data can be reduced without losing much information. The third step is to compute the eigenvalues and eigenvectors of the covariance matrix to identify the principal components. The principal components are new variables created through linear combinations of the original variables. These components carry more information, allowing for a reduction of dimensionality. They represent the directions of the data that explain a maximal amount of variance. The eigenvectors are beneficial to PCA as they signify the direction of the axes with the most variance, whereas the eigenvalues signify the amount of the variance affiliated with the principal component. Lastly, the eigenvectors are ranked by their corresponding eigenvalue. The eigenvector with the highest eigenvalue will be the most significant principal component. The number of principal components will equal the number of dimensions. In our case, we are interested in a 2-dimensional space and are therefore only interested in the two highest eigenvalues. To sum it up, the dataset is projected back onto the principal components. By utilizing PCA, the dimensions are reduced and the most important information is kept.

## 3.2 Principal Component Analysis (PCA) - Application

The group's implementation of PCA follows the previous procedure. Two approaches were used, Z-score normalization and Min-Max scaling. For Z-score normalization, the "StandardScaler" library function was used by removing the mean and scaling to unit variance. This results in the average value of every feature to be 0, centering the data around the origin. In the second approach, Min-Max

scaling was applied using the "MinMaxScaler" library function by adjusting features to a given range. This ensured that each features contributed proportionally to the final result. In the next steps, the covariance matrix is computed to analyze the interdependencies among features. Then the eigenvalues and eigenvectors are calculated of the covariance matrix. The eigenvectors are then ordered according to their eigenvalues and the top two components were chosene. Finally, the standardized data was projected onto these principal axes by computing the dot product of the standardized data matrix and the selected eigenvectors. As mentioned, we set the variable n_components to 2, as we are interested in a 2-dimensional visualisation of the data. (See visualisation under Section LDA / PCA Visual Comparison)

### 3.3 Linear Discriminant Analysis (LDA) - Method Overview

Linear Discriminant Analysis (LDA) is a supervised classification technique that is also used for dimensionality reduction. It is used to find a linear combination of features that separates two or more classes. In our case, the classes are the different categories of clothing and the features are the 784 pixels the images consist of. The objective of LDA is to project all the features in higher dimension space to lower dimensions. This is done by calculating the between-class variance, within-class variance, and lastly constructing the lower-dimensional space. The between-class variance calculates the separability between the categories of clothing. The variance quantifies how far apart the means of the different clothing categories are from each other. LDA finds the mean of every category and then how much the mean differs from the mean of all categories combined. In other words, LDA finds the average image for each category of clothing and then compares that image to the average image of all categories combined. By doing so, LDA identifies the unique features (pixels) for each category that make it stand out from the average looking image. The bigger the difference is in the mean of every category to all categories combined, the more separated the categories are.

The within-class variance calculates the distance within each category of clothing. Let us take the category 'Trousers' as an example. For each image, the pixels of the trousers are being averaged. In addition, the pixels of all trousers are averaged. The within-class variance then calculates the deviation between the average pixel count for every trouser and the average pixel count of all trousers combined. The deviations are further squared and summed up to quantify the overall distance between images in that specific category. This results in a measure representing the spread or scatter of the images in the category. The higher the deviation, the more different the images in the category are.

Finally, a lower-dimensional space is created based on Fisher's Criterion, which maximizes the ratio of the between-class variance to the within-class variance. The objective is to create a separation in which the different classes are relatively far apart from each other, while keeping the individual classes grouped. By default, LDA separates classes into N -1 dimensions, however, the number of dimensions can be specified. LDA will choose the dimensions carrying the most information. Now that a general understanding of LDA has been established, the paper will proceed to detail the group's application of LDA.

### 3.4 Linear Discriminant Analyis (LDA) - Application

First, the data is normalized by the *min_max_scale function*, which takes the dataset *data* as input and applies min-max[4] scaling to each feature. It checks for features where the minimum and maximum are the same and replaces all zero denominators with 1, as to avoid division by zero. The data is then rescaled by min-max normalization. To continue, the function *LDA* is defined, which takes four parameters: *X, y, n_components, regularization. X* is the input data matrix that consists of rows representing the individual images and columns representing the pixels. *y* is the category for every image (ie. 0). *n_components* specifies the number of components (ie. dimensions). *regularization* is used to avoid singularity in scatter matrices. Furthermore, the mean of every category and the mean of all images in X is computed. The function proceeds to calculate the within-class and between-class scatter matrix. Then the eigenvalues and eigenvectors are found through the generalized eigenvalue problem, where the within-class matrix gets subtracted from the between-class matrix. These eigenvalues and eigenvectors are further sorted in descending order. It proceeds by selecting the top components (not

specified). The eigenvectors are then normalized and the ratio of every eigenvalue divided by all eigenvalues is computed to see how much variance each component explains. Finally, the *LDA* function is applied to the scaled pixels with *n_components*set to 2. The data is transformed and further visualized.

### 3.5 LDA and PCA - Visual Comparison

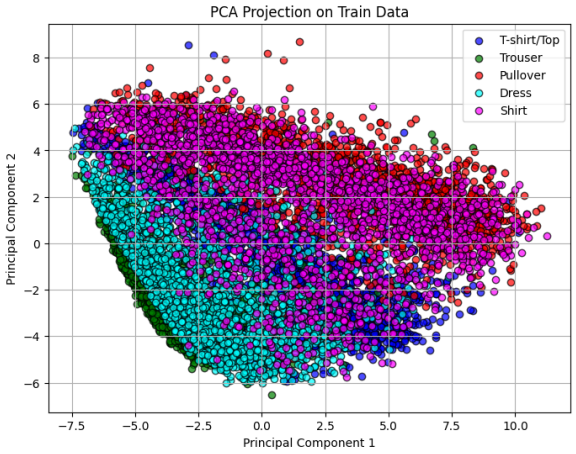



Figure 5: PCA - Train

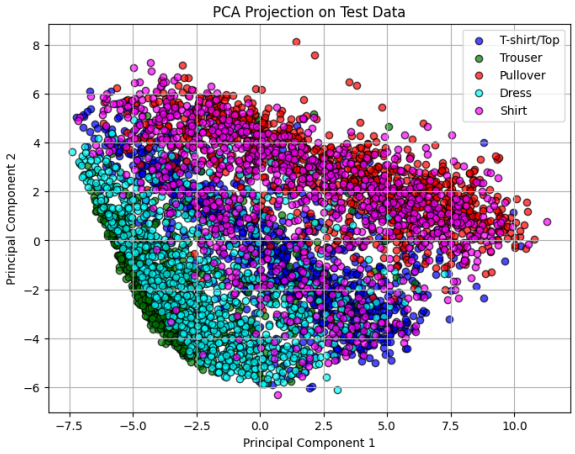



Figure 6: PCA - Test

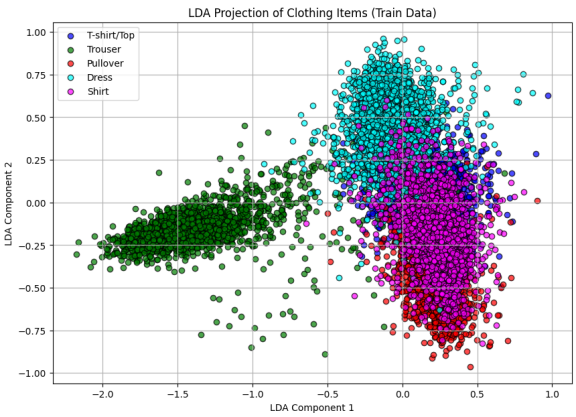



Figure 7: LDA - Train

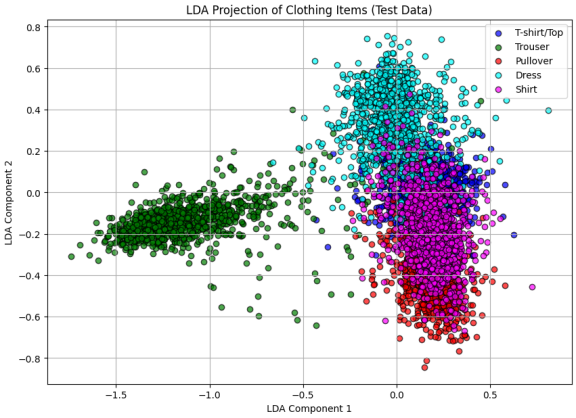



Figure 8: LDA - Test

The scatter plots of the two different dimensionality reduction techniques offer insight into the nature of the data. We will investigate each plot separately and then compare the two.

In the scatter plot belonging to the PCA projection, the two principal components that carry the most information position the data points in one cluster. The PCA1 (X-axis) carries the most significant information, whereas PCA2 (Y-axis) carries the second most significant information. It also implies that the Principal Component 1 explains more variation than Principal Component 2. Additionally, the data is the most dense around the origin, which is normal for PCA projections. There are some outliers. For example, there's one data point belonging to 'Trousers', which is clustered with 'Pullover', even though the clusters are opposite to each other. Lastly, the plot visualizes the classes as being relatively distinctly grouped, although with some overlap. The overlap represents some of the variance in the data points.

As for the second scatter plot belonging to LDA, the data points are grouped into relative distinct clusters. The most distinct cluster is the 'Trouser' category, which has very little overlap to the other categories. The other four categories are much more intermingled, although still relatively grouped together. This suggests that the features of the different categories can be separated, however, they still have similarities. Furthermore, certain groups are more dense compared to others, signaling degrees of within-class variance. The LDA projection shows a number of outliers that could be images of clothing which were mislabeled or simply very different looking from the usual image in that category.

When comparing the two scatter plots, there are several key differences. As LDA is a supervised

dimensionality reduction technique, the category separation is more well defined than in the PCA plot, where the different categories are more intertwined. In general, each plot has their own takeaway. The PCA plot is telling of the variance and its ability to be compressed into a smaller dimension. LDA indicates how well the categories can be divided. This is important for the choice of classifier.

# 4 Classification

Now that the exploratory data analysis has been concluded, the report will use the findings from the LDA and PCA dimensionality reduction techniques to accurately classify the clothing categories. The following sections will investigate three different classification methods and their implementations in the report.

## 4.1 Naive Bayes Classifier

The Naive Bayes classifier is founded on the basis of Bayes' Theorem [5], which describes the probability of an event occurring, when taking conditions related to the event into consideration. Mathematically, Bayes' Theorem is expressed as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2}$$

where A and B are events and B cannot be 0. $P(A|B)$ is a conditional probability, meaning it's the probability of event A happening, if event B has happened. $P(B|A)$ is the opposite conditional probability, and $P(A)$ and $P(B)$ are prior probabilities; the probability of either event happening without any given conditions. Furthermore, the classifier works on the assumption that the continuous values of the features follow a gaussian (normal) distribution. The classifier is a supervised machine learning algorithm, meaning it knows the category or label of the image. For every image in the dataset, the algorithm calculates the probability $P(A|B)$ for every category of clothing using the Gaussian probability density function:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3}$$

where x is the feature of the image. This is done for every feature of every image. The classifier assumes that all features are independent of each other and computes the probability of the feature x occurring given the category B. Lastly, the classifier combines the probability of every feature with the prior probability of the category to find the posterior probability. The category which ends up maximizing this posterior probability will then end up being the category label.[6]

## 4.2 Naive Bayes Classifier - Application

In the implementation of the Naive Bayes' classifier, the group started with splitting the training data by category. This is important as Naive Bayes' requires the prior probability of every category and the likelihood of each feature for every class.

The code finds every unique category label and adds the corresponding feature vectors to that category. Next up, the code introduces Kernel Density Estimation with a Gaussian kernel to estimate the likelihood of feature values of every category. This is done to get a probability distribution.

```
def kernel_density_estimate(x, data, bandwidth):
    return np.sum(norm.pdf((x - data) / bandwidth)) / (len(data) * bandwidth)
```

Figure 9: Kernel Density function

After that, the classifier is created by combining the prior probability with the likelihoods of the observed feature values to make predictions.

```python
def naive_bayes_classifier(x, class_data, bandwidth):
    probabilities = {}
    for label, data in class_data.items():
        prior_prob = len(data) / len(X_lda_train)
        p_x1_given_y = kernel_density_estimate(x[0], data[:, 0], bandwidth)
        p_x2_given_y = kernel_density_estimate(x[1], data[:, 1], bandwidth)

        #Naive Bayes
        posterior_prob = prior_prob * p_x1_given_y * p_x2_given_y
        probabilities[label] = posterior_prob

    # Predict the class with the highest posterior probability
    predicted_class = max(probabilities, key=probabilities.get)
    return predicted_class
```

Figure 10: Naive Bayes function

Finally, we predict the class of newly unseen data and quantify the accuracy by finding the ratio of correct predictions to all predictions.

```python
nb_y_pred = [naive_bayes_classifier(x, class_data_train, bandwidth=0.1) for x in X_lda_test]

nb_accuracy = accuracy_score(labels_test, nb_y_pred)
print("Accuracy:", nb_accuracy)
```

Figure 11: Naive Bayes function usage for prediction

### 4.3 K-Nearest Neighbors (KNN)

The K-Nearest Neighbours (KNN) algorithm is a supervised classifier based on the concept of similarity.[7] It is non-parametric making it a flexible tool in real-world scenarios where the data may not be normally distributed. Simply speaking, it evaluates the label of a new datapoint by computing the distance between that data point and already labeled data points. The label gets chosen based on the shortest distance to other labeled data points. In the context of the fashion-MNIST dataset, a datapoint is an image which is graphed according to two features. If the image is closer to data points belonging to the category 'Trousers' than to 'T-Shirts', the image will be labeled or classified as 'Trousers'. KNN uses a hyperparameter K, which represents the number of 'neighbors' to consider. If K = 9, the algorithm will consider 9 of the new data point's neighbors. If the majority of the neighbors are 'T-Shirts', then the data point will be assigned to the 'T-Shirt' category.

### 4.4 KNN - Application

The KNN classifier was implemented through the sklearn library. K was set to 9. As the number of neighbors increases, it becomes computationally inefficient to calculate. Increasing the number of neighbors could potentially reduce overfitting. We tested both 3, 5 and 9 neighbors as parameters and the results were almost the same, we chose the latter to be as accurate as possible which also yielded a smoother decision boundary. Furthermore, the classifier was evaluated through the test data.

### 4.5 Convolutional Neural Network (CNN)

The last classifier which was applied to the data set is the Convolutional Neural Network. CNNs are useful for image classification as they are capable of distinguishing spatial features such as edges, textures and shapes.[8] A CNN consists of one to many modules, each of which can be dissected into three steps:

1. Convolution: The greyscale input image is represented by a 28x28 matrix, where each entry is a pixel value between 0 and 255. A convolution filter, which is a small matrix with weights

that the algorithm learns during training, scans over the image. For every position in the input image, the filter and the filtered part of the input image get multiplied to yield an output feature map. The output feature map is an alternative representation of the original input, designed to highlight features of significance. The more output feature maps the algorithm produces, the more features it detects. Earlier layers capture the more superficial features, whereas later layers capture more intricate details.

2. ReLU: After every convolution, a Rectified Linear Unit (activation function) transformation is applied to the convolved feature, which introduces nonlinearity to the model. The data is more often than not nonlinear, and by applying this transformation, the model improves at learning these relationships. ReLU sets all x values bigger than 0 to x and all values equal to or less than 0 to 0.

3. Pooling: The last step is to reduce the amount of data with the intent of reducing the computational complexity. Here, the convolved feature is reduced to only include the highest entry based on two parameters:

    - The size of the filter
    - The stride, which determines how large a part of the input matrix you "pool". So a stride of 3 would mean that you reduce every 3x3 (9 entries) into 1 entry.

At the end, you will have a CNN that is fully connected. The final layer will have a softmax activation function applied to it, scaling the matrix values into probabilities of classes. These probabilities are then used to classify the image.

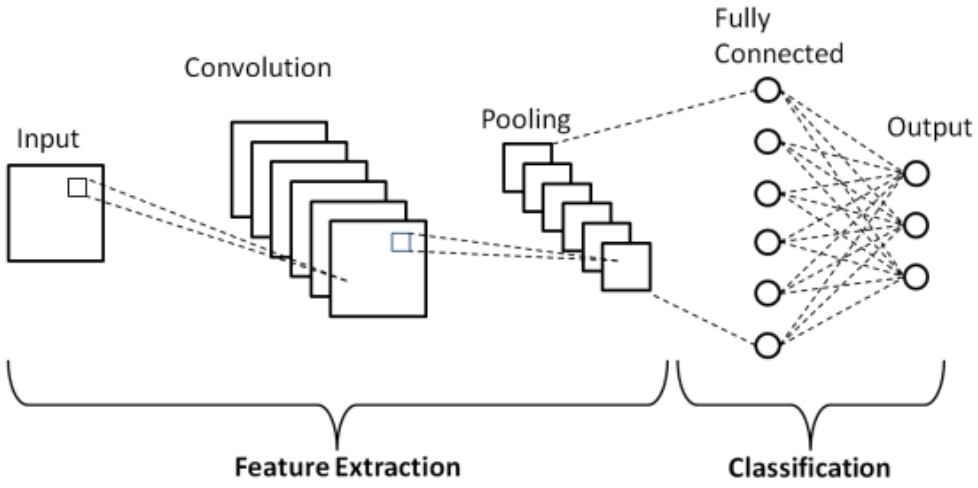



Figure 12: Example CNN

### 4.6 Convolutional Neural Network (CNN) - Application

CNN was implemented through the use of Keras, a library for neural networks in python. Our base CNN model was set up like so; first a convolutional layer followed by a max-pool layer, a second convolutional layer followed by a max-pool layer.

Each layer is responsible for identifying different levels of patterns. One highlights the main features, whereas the other finds more intricate details. These layers are then further flattened in order to convert the data to a one-dimensional vector.

Finally, the vector is classified through the dense layer and a probability between 0 and 1 is output via a softmax activation function:

The learning rate parameter for keras optimizer, in our case "Adam", uses a default of 0.001, which we did not change throughout our training process for our different CNN models. Categorical cross

```
# First conv. layer
model.add(layers.Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
# Second conv. layer
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
```

Figure 13: First two convolutional layers

```
model.add(layers.Flatten())
```

Figure 14: Flattening layers

```
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(num_classes, activation='softmax'))
```

Figure 15: Dense layers

entropy was used for our multiclass classification task, also known as softmax loss. It uses softmax activation plus a cross-entropy loss, hence the name.

The final CNN model is an improved version of the base CNN model. It got more complex as we added another convolutional (hidden) layer to the model, as well as included various other techniques to make the model robust such as batch normalization (standardizing inputs to a layer to each batch), dropout layers (randomly deactivating a fraction of neurons during training). To add to it, the final model also used augmented data which was discussed in one of the previous sections. The augmentations were random rotations, zooming, shearing, and horizontal flipping of the images.

The confusion matrix of our final mode which can be seen in figure 17 proves that classifying images into "Shirt" class was the hardest task, since it got classified wrongly both to "T-shirt/top" (222 times) and "Pullover" (199 times) out of a total 1000. "Trousers" class was the easiest to classify accurately, followed by "Pullover", "T-shirt/Top" and "Dress".

## 5 Results

The classifiers were trained on the training set and subsequently tested on the given testing set. The following section will deal with the results of the different classifiers and their accuracies:

Seeing as the objective of the report is to accurately classify the different categories of clothing, it is expected to see Naive Bayes, using LDA (74.64%) for dimensionality reduction, outperform Naive Bayes utilizing PCA (48.16%). LDA is better at setting the decision boundaries between classes, as that is what it is programmed to do. As seen in the scatter plot of the PCA projection, the dimensions do not align well with category boundaries, instead creating more overlap between categories. The variance portrayed by PCA does not appear to accurately reflect the characteristics of the different clothing categories. As Naive Bayes assumes a Gaussian distribution, and both of the PCA and LDA-transformed datasets meet that requirement, a difference in the fundamental distribution can be ruled out as the underlying reason for the varying accuracy.

Our KNN classification model achieved an accuracy of 82.46% which was better than expected. When we applied PCA for reducing dimensionality, we decided to test how well would KNN classify using only the two PCA components and there was a significant drop in accuracy - from 82.46% to just 56.84%. That suggested to us that PCA's unsupervised dimensionality reduction may not be useful for such a classification task. Using LDA's variables for classification improved the accuracy to 73.84%. Although this approach did not reach the accuracy levels of the original KNN classifier,, it outperformed the version utilizing PCA components. This improvement is due to the fact that LDA aims to maximize class separability which enhances the effectiveness of the KNN classifier.

The first augmented CNN model did not show notable improvement compared to base CNN models, however it did outperform both Naive Bayes classifiers, as well as all variations of KNN. All of our CNN models were trained on 15 epochs and the training progress can be seen in the figure 17. Although

```
Confusion matrix for test data
------------------------------
[[901   2  36  21  40]
 [  4 972   5  15   4]
 [ 17   0 933  12  38]
 [ 42   7  32 888  31]
 [222   0 199  25 554]]
```

Figure 16: Confusion matrix for final augmented CNN model

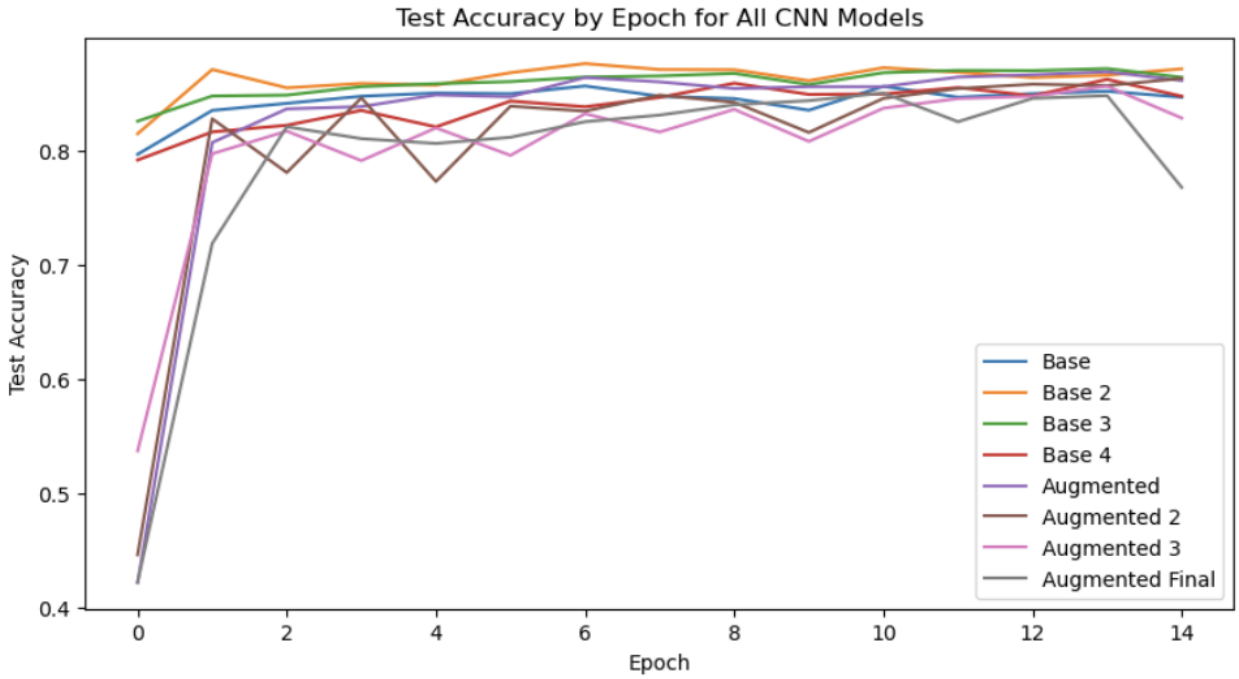



Figure 17: Training process for CNN models

the accuracy for the final augmented CNN model, which was using multiple augmentation filters, including horizontal flipping, random rotations, zoomins and shearing of the images, slightly dropped compared to our first model with augmented data (from 87.66% to 85.08%), we chose that as our best model for this classification task due to the fact that it demonstrated robustness, meaning it could still handle images just as good as the base CNN models even after multiple transformations on the training images were done. The final model helps capture a wider range of patterns which improves generalization of the model and reduces overfitting. It outperforms all of our previous methods and is complex enough to tackle variations of images that would be closer to real-life scenarios.

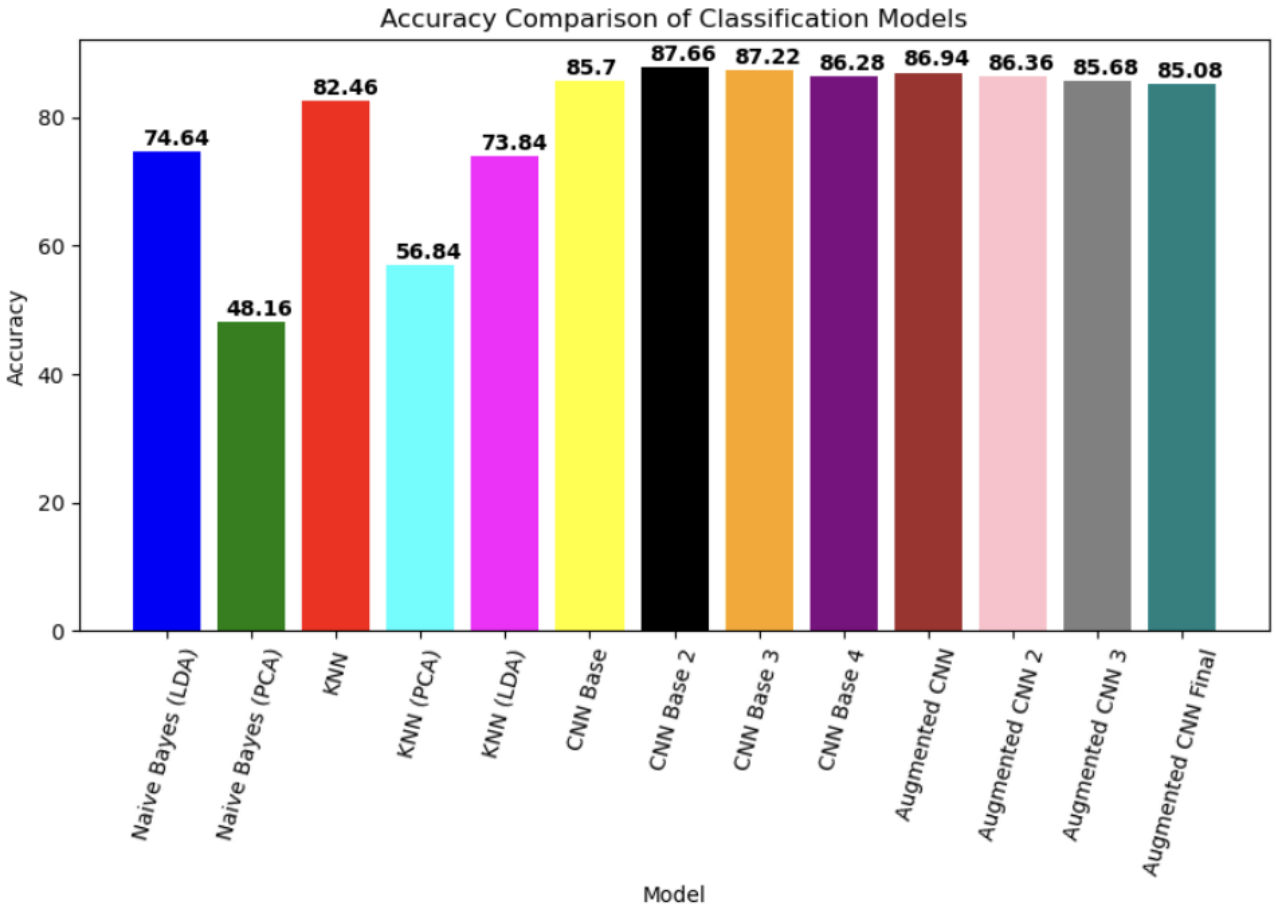



Figure 18: Accuracy comparison of all classifiers

## 6 Conclusion

In conclusion, we explored various different classification methods for the task of greyscale clothing image classification. The comparison of Naive Bayes classifiers using PCA components and LDA variables for dimensionality reduction clearly showed the superiority of LDA in this context by showing a higher accuraacy of 74.64 percent, compared to PCA's 48.16 percent, clearly performing better at class separability.

The K-Nearest Neighbors (KNN) model further showed the limitaions of PCA in capturing the essential features for accurate classification, as shown by a decrease in accuraacy when reducing the dataset to only two PCA components. However, utilizing LDA variables for the KNN model showed a notable improvement in accuracy, although it did not overperform the original KNN model. This shows again the importance of choosing appropriate dimensionality reduction methods in alignment with the classifier being used.

Finally, our exploration of Convolutional Neural Networks (CNN) both without and with data augmentation revealed a new understanding of model complexity and generalization of data. While the first augmentation did not clearly show an improvement compared to the base CNN models, it nevertheless outperformed all the previously mentioned classifiers. Our final augmented model, however, despite a slight decrease in accuracy compared to the first augmented model, was selected as the optimal classifier for this task. It was robust enough after various transformations on the training data, showing stronger ability to adapt and generalize data when it is closer to real-life scenarios, when pictures might be taken from a different angle, zoomed in and so on.

In summary, our experiments underscored the critical importance of selecting appropriate classifiers and preprocessing techniques in image classification tasks. The comparative analysis of various models

revealed that while Convolutional Neural Networks (CNN) appeared to be the optimal choice initially and indeed outperformed other classifiers, effective preprocessing, thoughtful standardization, and careful hyperparameter tuning can significantly enhance the performance of alternative models. This insight highlights the nuanced nature of machine learning tasks, where the application of preprocessing strategies and model selection plays a pivotal role in achieving high-quality results.

## References


[1] "Image Recognition Technology," Trendskout, [Online]. Available: `https://trendskout.com/en/solutions/image-recognition-technology/`. [Accessed: Jan. 3, 2024].

[2] "Curse of Dimensionality," Wikipedia, [Online]. Available: `https://en.wikipedia.org/wiki/Curse_of_dimensionality`. [Accessed: Jan. 3, 2024].

[3] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*. New York: Springer, 2013, pp. 499.

[4] "Feature Scaling," Wikipedia, [Online]. Available: `https://en.wikipedia.org/wiki/Feature_scaling`. [Accessed: Jan. 3, 2024].

[5] "Bayes' Theorem," Wikipedia, [Online]. Available: `https://en.wikipedia.org/wiki/Bayes'_theorem`. [Accessed: Jan. 3, 2024].

[6] "Naive Bayes Classifiers" Geeksforgeeks, [Online]. Available: `https://www.geeksforgeeks.org/naive-bayes-classifiers/`. [Accessed: Jan. 3, 2024].

[7] "K-Nearest Neighbor(KNN) Algorithm" Geeksforgeeks, [Online]. Availabe: `https://www.geeksforgeeks.org/k-nearest-neighbours/`.[Accessed: Jan. 3, 2024]

[8] "Image Classification Using CNN: Introduction and Tutorial" Datagen, [Online]. Available: `https://datagen.tech/guides/image-classification/image-classification-using-cnn/`. [Accessed: Jan. 3, 2024]