Institute of Telecommunications
TU Wien

Project

Barbaro Francesca

02 - 2024
389.204 - Machine Learning Algorithms (WS 2023)

Institute of Telecommunications
TU Wien

Contents

1 Introduction

The main idea of this project is to implement machine learning techniques to perform binary classification and contrastive learning. The used dataset is CelebA, that contains images depicting the faces of various celebrities.

2 Binary Classification & Model Explainability

The CelebA dataset, for the binary classification task, is partitioned into a training set containing 20.000 images and a test set comprising 1000 images. The objective is to classify whether a celebrity is wearing glasses or not.

Various machine learning techniques are employed to achieve this classification.

2.1 Baselines

In this section, the employed methods are implemented as functions called from the *sklearn* package, namely Ridge Classifier, Logistic Regressor, and SVC. The performance of these methods is analyzed to gain insight into the accuracy achieved.

The data is read using the *tensorflow* function *image dataset from directory* and resized to dimensions of (96, 96, 3).
For the implementation of the following methods, a smaller portion (30%) of the data is selected to facilitate quicker training.

2.1.1 Ridge Classifier

The Ridge Classifier is a linear classification algorithm that introduce a regularization in the form of L2 regularization. The Ridge Classifier aims to find a hyperplane that separates the classes while minimizing the sum of squared weights. The regularization term helps prevent overfitting by penalizing large weights in the model. This regularization can be controlled by a hyperparameter, commonly denoted as alpha. A higher alpha value results in stronger regularization, leading to a simpler model with smaller weights.

I set the value alpha to 10 and both trained the classifier on Standardized data and not. In Table (1) it's possible to see the accuracy result on the test set.
The accuracy metric is defined as:

$$Accuracy = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{1}$$

Configuration	**Accuracy**
Not Scaled	0.7342
Scaled	0.7542

Table 1: Accuracy Comparison for Ridge Classifier for $\alpha = 10$, test set

Then I set alpha equal to 100 to have a stronger regularization and I observed slightly improvements in the accuracy of the scaled data. Results are reported in Table (2).

Configuration	Accuracy
Not Scaled	0.7342
Scaled	0.7973

Table 2: Accuracy Comparison for Ridge Classifier for $\alpha = 100$, test set

In Table 4 is reported the confusion matrix with the general form reported in Table (3):

	Predicted Negative	**Predicted Positive**	
Actual Negative	TN	FP	Total Predicted Negative
Actual Positive	FN	TP	Total Predicted Positive
	Total Actual Negative	Total Actual Positive	

Table 3: Confusion Matrix

Where Negative is the class with glasses and Positive is the class without glasses.

	Predicted with glasses	**Predicted without glasses**
Actual with glasses	117	38
Actual without glasses	23	123

Table 4: Confusion Matrix Ridge Regression with Scaled data, $\alpha = 100$

2.1.2 Logistic Regressor

Logistic Regressor is another popular linear classification algorithm for binary classification tasks. This method models the probability of a binary outcome by applying the sigmoid (or logistic) function to a linear combination of input features. The sigmoid function transforms the output into a range between 0 and 1: the probability of belonging to a particular class. Like Ridge Classifier, Logistic Regression can also include regularization terms, such as L1 or L2 regularization, to control the complexity of the model and prevent overfitting.

I selected the default L2 regularization. Results are showed in Table (5).

Configuration	Accuracy
Not Scaled	0.8439
Scaled	0.8605

Table 5: Accuracy Comparison for Logistic Regressor with and without Scaling

	Predicted with glasses	**Predicted without glasses**
Actual with glasses	134	21
Actual without glasses	21	125

Table 6: Confusion Matrix for Logistic Regressor Scaled data

2.1.3 SVC

Support Vector Classification (SVC) is a classification algorithm that leverages the principles of Support Vector Machines to find an optimal hyperplane for separating classes in the input feature space.

The objective of SVC is to find the optimal hyperplane in the feature space, maximizing the margin between different classes. This margin is defined by support vectors, which are the data points closer to the decision boundary. SVC can utilize the kernel trick, allowing the algorithm to implicitly map the input features into a higher-dimensional space. This is useful when dealing with nonlinear relationships between the input variables.
The regularization parameter C controls a squared L2 penalty.

I used the RBF kernel with a regularization parameter C set to 1. The accuracy results are reported in the Table (7).

Data	**Accuracy**
Not Scaled	0.8439
Scaled	0.8638

Table 7: Accuracy Comparison for SVC with and without Scaling

	Predicted with glasses	**Predicted without glasses**
Actual with glasses	133	22
Actual without glasses	19	127

Table 8: Confusion Matrix for SVM Classifier Scaled data

2.1.4 Comparison

In all methods, it can be observed that the highest accuracy is achieved when the images are standardized. Additionally, we note that the Ridge Classifier performs less optimally compared to the Logistic Regressor and SVC, which exhibit very similar performance levels.

From the Table (4), displaying the confusion matrix for the Ridge Classifier, we observe an asymmetry, indicating a greater difficulty in recognizing the presence of the glasses. In contrast, when examining the confusion matrices for Logistic Regression (6) and SVC (8), we notice that they are practically symmetrical and very similar.

2.2 Neural Networks

In this section of the project, binary classification is performed using neural networks. Two distinct types of neural networks are employed.

2.2.1 CNN model

I started with a convolutional neural network (CNN) structure.

The network begins with an input layer where the image data, in the specified shape, is fed into the model.

The subsequent layers are convolutional layers, designed to extract features from the input images. The first convolutional layer employs 32 filters with a 3x3 kernel size, applying the Rectified Linear Unit (ReLU) activation function to introduce non-linearity. This is important for the model to capture

complex patterns in the data.
Following this, a max-pooling layer with a 2x2 pool size is applied to down-sample the spatial dimensions.
The second convolutional layer is similar to the first layer, but employs 64 filters.

After the convolutional layers, a Global Average Pooling (GAP) layer is introduced. This layer condenses the spatial information into a single value per feature map. This reduction helps the model focus on the most relevant features.

Two Dropout layers with a dropout rate of 0.5 are employed. These layers randomly deactivate a proportion of neurons during training, acting as a regularization technique to prevent overfitting.

Following the Dropout layers, a fully connected classifier layer is implemented with 32 units and ReLU activation. This layer integrates the learned features and prepares them for the final classification step.

The output layer is a dense layer with a single unit and a sigmoid activation function. Sigmoid is employed for binary classification, squashing the output between 0 and 1 to represent the probability of the positive class (without glasses).

In terms of model compilation, the Binary Crossentropy loss function is chosen for its suitability in binary classification tasks. The Adam optimizer is employed, and accuracy is selected as the evaluation metric.

The global number of parameters is 21.505, high enough to allow the model to capture the important patterns in the dataset but not too high that could lead to overfitting problems.
The accuracy reached on the test set is of 0.9452, showing a good performance of the network.

The learning curves are reported in Figure (1).

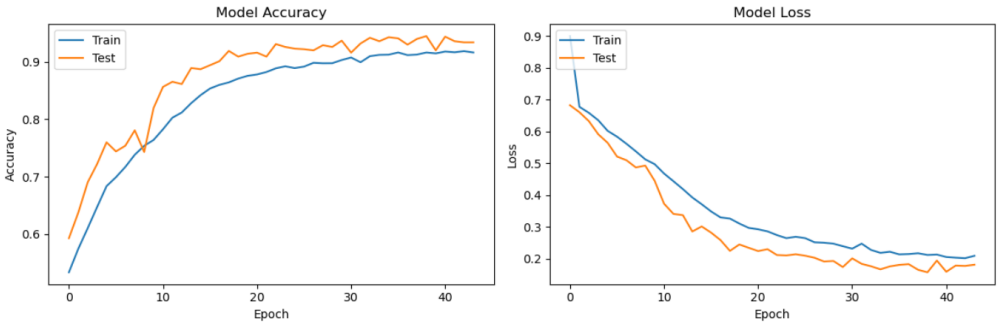

Figure 1: Accuracy and Loss Function for train and test set of CNN

The training stopped after 44 epochs, employing the EarlyStopping technique to avoid overfit. Even if the test set curves show oscillatory behaviours, the general trend of the curves is the same: they are increasing in the same way for the accuracy, and decreasing for the loss function.

Metric	**Train**	**Test**
Loss	0.2121	0.1577
Accuracy	0.9166	0.9452

Table 9: Performance Metrics for the CNN Model

The slightly worst performance in the train set can be due to the more variability presented in the dataset.

2.2.2 ResNet18

As advised, I analyzed the structure of ResNet18, an 18 layers deep convolutional neural network architecture tailored for image classification tasks. It uses special techniques that allows to construct a deep neural network without incurring in the vanishing gradient problem.
For downsampling, ResNet18 employs strided convolutions and pooling operations, effectively reducing spatial dimensions.
Global average pooling is applied at the end of the network, reducing spatial dimensions to a single value per feature map.
It also include skip connection between layers, that facilitates passing features in the network.

I proceeded to custom ResNet18 keeping the same structure but decreasing the number of layers, in order to decrease the big number of parameters.
The total number of parameters is 158.209, a big increment with respect to the CNN.
The accuracy on the test set reached 0.9422, very similar to the previous one.
In Figure (2) are plotted the learning curves for the Train and the Test set.

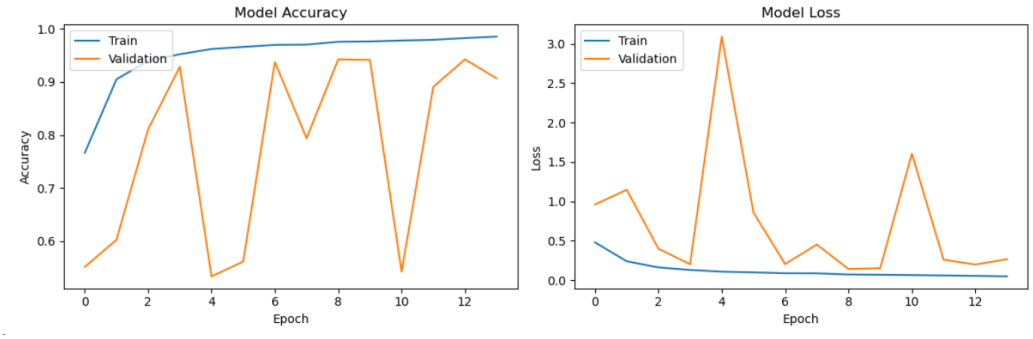

Figure 2: Accuracy and Loss Function for train and test set of custom ResNet

It's possible to notice that the training ended only after 12 epochs, since I employed the EarlyStopping technique to avoid overfitting. The curves related to the test set exhibit highly oscillatory and unstable movements, which led me to prefer the performance of the other network. However, it is important to note that this model is highly efficient as it achieves the same accuracy with significantly fewer epochs.

Metric	Train	Test
Loss	0.0533	0.1969
Accuracy	0.9824	0.9422

Table 10: Performance Metrics for the ResNet Model

2.3 Integrated gradients

In this section, I extended my analysis on the CNN 2.2.1, since it achieves good results with a much simpler structure compared to the second model 2.2.2.

The idea, of this part of the project, is to use the Integrated Gradients (IG) method to understand the impact of each feature on the predictions of the neural network.

I implemented the IG method as showed in the provided *tensorflow* tutorial, and adapted it for my dataset. This feature highlights the pixel in the image that are important for the classification decision. In Figure (3) and (4) some examples are reported, two for each class.

Figure 3: Example of people with glasses

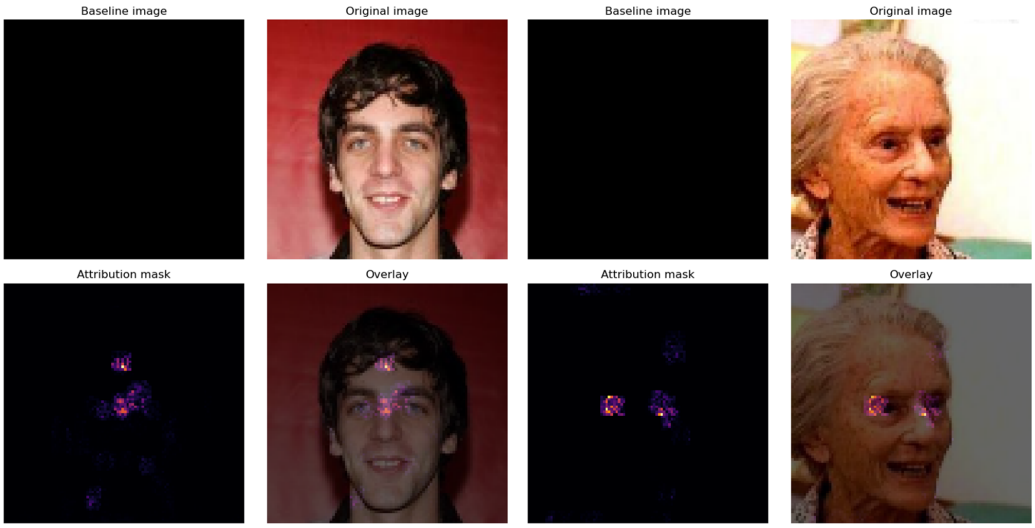

Figure 4: Example of people without glasses

The baseline is an input image used as a starting point for calculating feature importance, I selected a fully black image whose pixel values are all zero.
In the attribution mask we observe some highlighted pixel, these are the important pixel for the classification task. In the overlay the original image is employed as a background for the attribution mask. This makes possible to understand which part of the original image is more representative.

In the case of celebrities wearing glasses (Figure 3), we observe that the representative pixels are predominantly concentrated in the region of the nose, between the eyes, where the bridge of the glasses is located. Interestingly, there is a noticeable absence of pixels in the area corresponding to the lenses and the rest of the eyeglass frame. This is also due to the fact that, selecting a black baseline all the black pixel cannot be considered as important. It's also interesting to notice that the glasses are of diverse nature, both optical and sunglasses, and they exhibit substantial differences. Additionally, glasses can vary significantly in shape and color. However, the common element shared by all glasses is the nasal bridge, where the neural network focuses its attention for classification.

In the case of celebrities who are not wearing glasses (Figure 4), we still observe that the highlighted pixels concentrate once again in the nasal region, where the bridge of glasses would typically be located,

and also around the eyes.

Subsequently, I considered a white baseline to account for black pixels as well. In Figures (5) and (6) there are the results for four different examples.

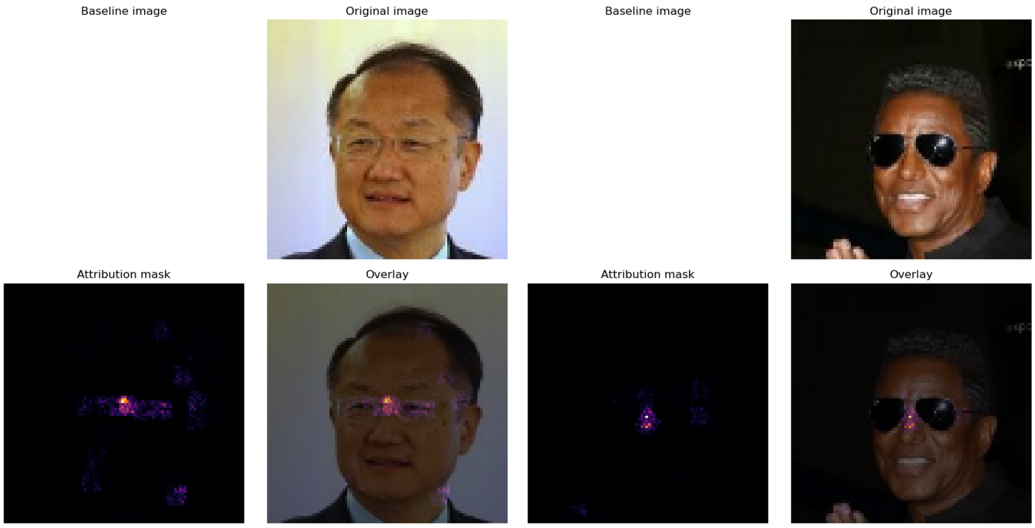

Figure 5: Example of people with glasses, white baseline

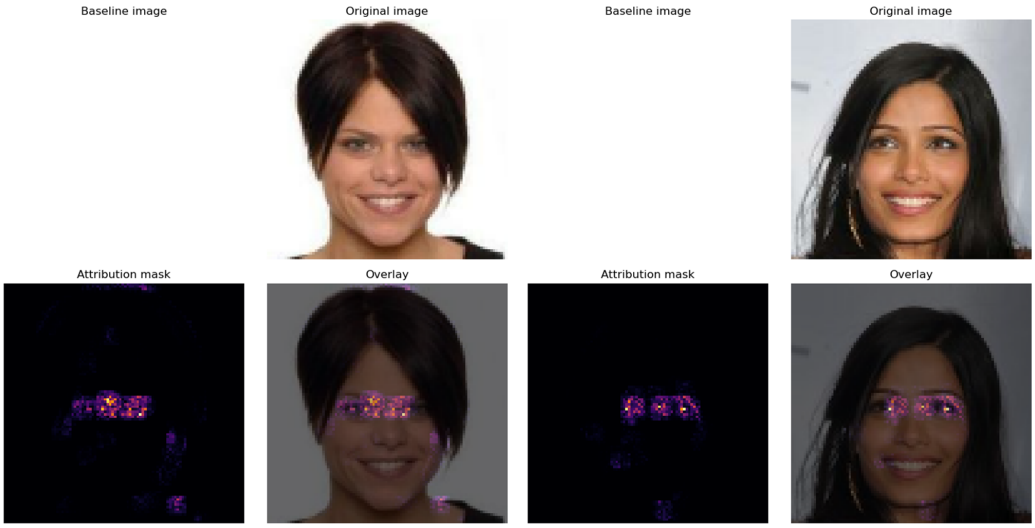

Figure 6: Example of people without glasses, white baseline

Despite the white baseline, in Figure (5) depicting showing wearing glasses, the highlighted pixels consistently correspond to the bridge of the glasses. and not in the surroundings areas. In contrast, in Figure (6) depicting individuals without glasses, the highlighted pixels are concentrated also around the eyes.

2.4 Conclusions

In this initial phase of the project, we evaluated the performance of various machine learning methods for conducting the same binary classification.

It becomes evident that 'standard' methods such as Ridge Classifier, Logistic Regression, and SVC struggle to operate on large datasets, such as CelebA. In fact, to obtain efficient results, it was

necessary to analyze only a subset of the data. The accuracy achieved by these methods is satisfactory, but there is room for improvement.

Neural networks (NN), designed specifically for large datasets, allowed the usage of the entire dataset. However, the training times are relatively long, ranging from 20 to 30 minutes depending on the architecture. The results obtained with NN are significantly superior, reaching much higher accuracy levels. Additionally, it was possible to conduct an explainability analysis of the model using the Integrated Gradients technique. This enabled to understand which parts of the image the NN focuses on for the classification.

3 Face-Recognition with Contrastive Loss Function

In this second part of the work the idea was to implement a triple network that could predict if two pictures are of the same person or not. This is a contrastive learning problem and the advantage of this method is that it's not necessary to retrain the entire model when a new person is added to the system.

The CelebA dataset is divided into a train and test set. The train set includes information on 150 different celebrities, with 30 images provided for each individual. The test set is constructed analogously but with only 5 celebrities.

3.1 Triplet Network

The idea of this method is to learn a mapping function $F(\cdot)$ that embeds input data points x into a shared space $F(x)$, where the similarity between pairs of data points can be easily measured. In this case the function $F(\cdot)$ is defined by an embedding network.

The triple network takes three images as input: an anchor image x_a, a positive image x_+ of the same identity as the anchor, and a negative image x_- of a different identity. The objective is to minimize the distance between the anchor and positive embeddings $d(F(x_a) - F(x_+))$, while maximizing the distance between the anchor and negative embeddings $d(F(x_a) - F(x_-))$. In this work the considered distance $d(\cdot)$ is the Euclidean distance $|| \cdot ||^2$

This architecture encourages the network to learn a representation where similar faces are close to each other, and dissimilar faces are far apart.

3.1.1 Generating triplets

I began by creating triplets that serve as input for the triplet network, using the provided reference code. As previously mentioned, these triplets consist of three images: an anchor x_a, a positive x_+, and a negative x_- image. The positive image represents the same person as the anchor, while the negative image represents a different individual.

In Figure (7) are reported two examples of triplets, one for the train and one for the test set.
The positive image is randomly selected between the one of the same person and the negative image is randomly selected between the images of the other people.

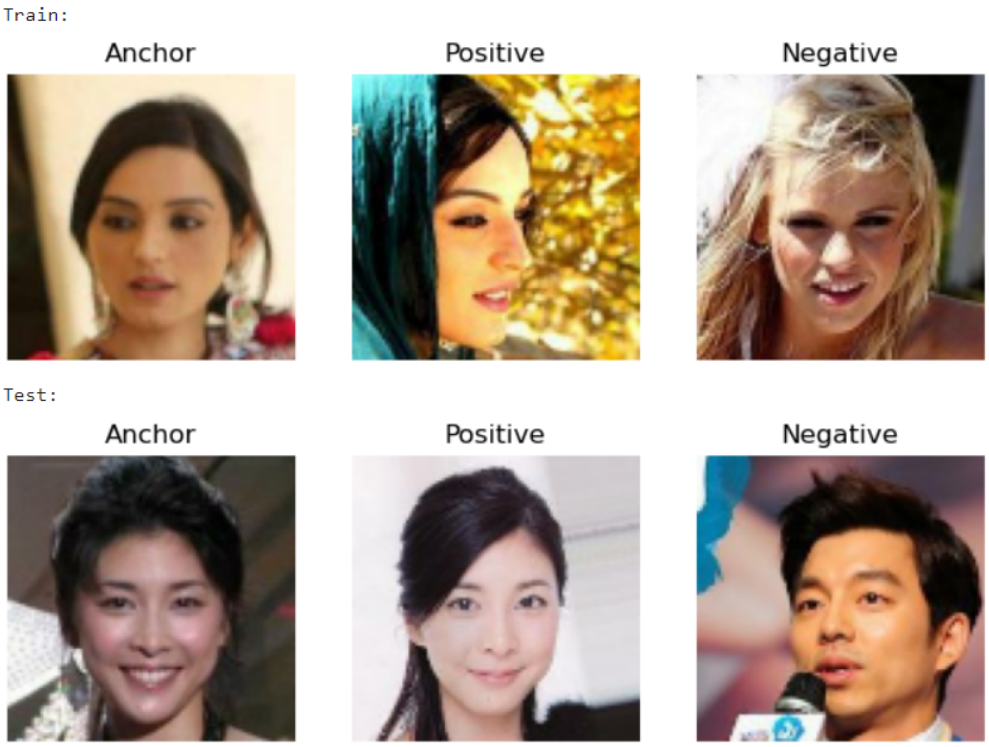

Figure 7: Example of Triplets from train and test set

3.1.2 Triplet Network implementation

To implement the triplet network, it is necessary to define an embedding network that represent the mapping function $F(\cdot)$. I selected a sequential model comprising a convolutional layer with 64 filters, followed by flattening and two densely connected layers: one with 128 units and the other with 20 units.

The embedding network is fed with the set of anchor, positive and negative images. The input will be compressed by the network in a vector on dimension 20, since the last dense layer has 20 neurons.

Subsequently, the triplet network will proceed to compute the Euclidean distance between the embeddings as:

$$Positive\ distance = d(x_a, x_+) = ||F(x_a) - F(x_+)||^2, \tag{2}$$

$$Negative\ distance = d(x_a, x_-) = ||F(x_a) - F(x_-)||^2 \tag{3}$$

The network is trainded in order to correclty classify which of x_+ and x_- is of the same class as x_a. The positive distance should be smaller than the negative distance. To do this the loss function is implemented as:

$$L(x_a, x_+, x_-) = max\{d(x_a, x_+) - d(x_a, x_-) + \alpha, 0\}, \qquad d(x, y) = ||F(x) - F(y)||^2 \tag{4}$$

Where α is the margin term, introduced to force the network not to end up in the trivial solution setting everything to 0. Instead, d describes the distance between the respective embeddings.

It's also necessary to define a custom accuracy that correctly evaluate the behaving of the model. In order to fulfill the accuracy definition (1), this is defined as:

$$mean\{d(x_a, x_+) < d(x_a, x_-)\} \tag{5}$$

Where "Number of correct predictions" corresponds to the number of triplets where the positive distance is smaller than the negative distance, represented by $d(x_a, x_+) < d(x_a, x_-)$.
"Total number of predictions" is the total number of triplets in the dataset. The binary accuracy indicates the proportion of correctly identified triplets: where the model correctly ranked the positive sample closer to the anchor than the negative sample.

I trained the network for 10 epochs and reported the learning curves in Figure (8).

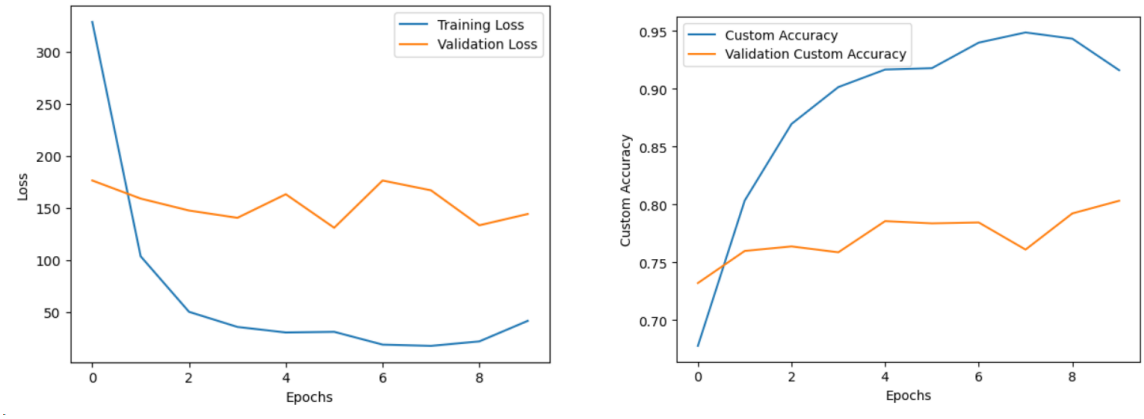

Figure 8: Learning curves: Loss function and Accuracy

An observable difference is noted between the values for the test set, which exhibit lower performance, and those for the training set. This means that the model is overfitting the training data. It would be probably possible to enhance the performance of the model by including an embedding network that better compress the images in a higher dimensional space that can include more information.

Metric	**Train**	**Test**
Loss	41.3000	144.0039
Accuracy	0.9159	0.8031

Table 11: Performance Metrics for the Model

3.2 Visualization of the embeddings

In this section the aim is to analyse the result of the triplet network and visualise the embeddings in a 2D space.

I called the trained embedding network and computed the embeddings $F(x)$ for every image in the test set.

As previously mentioned, the network is designed to embed images in a manner where the Euclidean distance between images of the same person is smaller compared to the distance between images of different individuals. To validate this, I examined the labels of images with the smallest and largest distances, as outlined in Table (12). The smallest Euclidean distance, indicating the highest similarity, is observed for images portraying the same person. Conversely, the largest distance, signifying the lowest similarity, is reported for images of two different individuals.

Distance Category	Corresponding Labels	Euclidean Distance
Smallest	0 and 0	244.97
Largest	2 and 3	7717.90

Table 12: Euclidean Distances and Corresponding Labels

To have a more general understanding I computed two very simple statistics, the mean and the standard deviation for all the Euclidean distances of the embeddings, dividing them between same person or different person. Results are reported in Table (13).

Statistics:	Mean	Standard Deviation
Same Class	2857.83	715.03
Different Classes	3790.10	960.80

Table 13: Mean and SD for Distances Within and Between Classes

To confirm the expected results, when considering images of the same person, the average distances are lower. On the other hand, when considering images of different individuals, the average distances are higher. The standard deviation allows an analysis of how a single values can deviate from the mean, indicating that for some image pairs the distinction is not that clear. In particular, there will be images with similar embeddings representing different individuals, causing the neural network to struggle in recognizing them as different people.

3.2.1 PCA

In order to visualise the embeddings of the test set images in a 2D space was requested to use the Principal Component Analysis (PCA). This is a dimensionality reduction technique that transform the original features of a dataset, in this case the embeddings, into a new set of uncorrelated variables, known as principal components. The PCA identifies the directions (principal components) along which the data varies the most and projects the data onto these directions. It's possible to reduce the dimensionality for visualization considering only the first two principal component and keeping the most essential information.

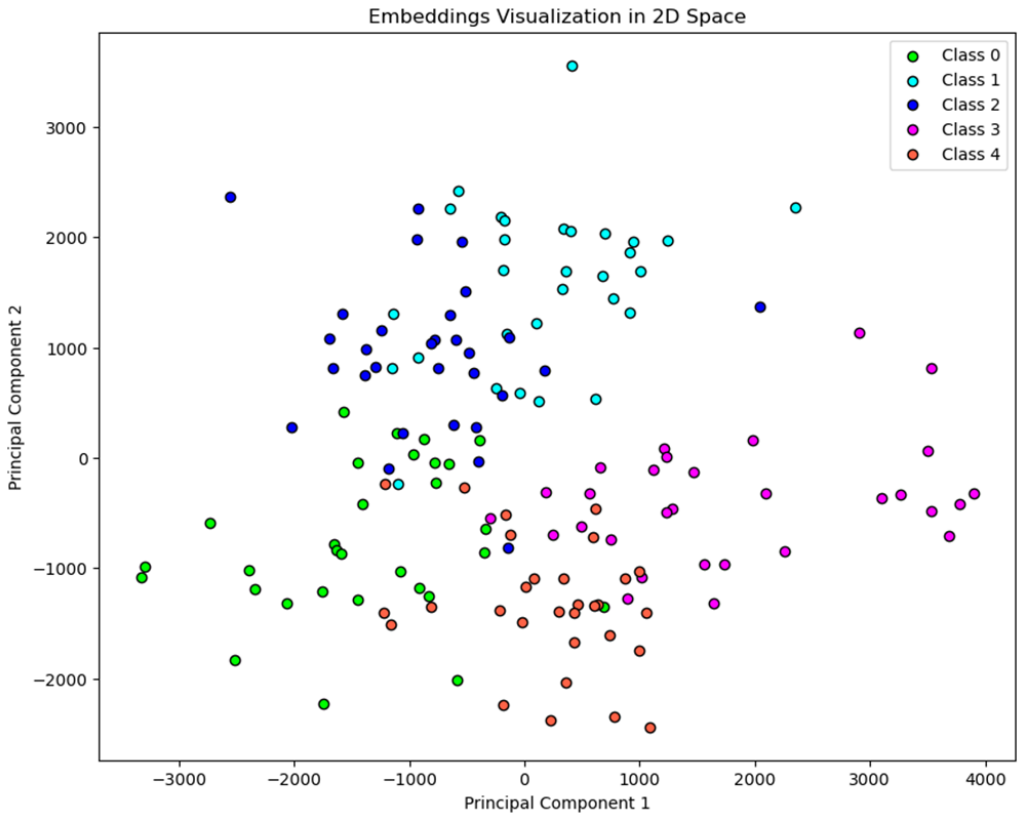

Figure 9: PCA representation of the embeddings

The scatterplot depicted in Figure (9) illustrates the 2D representation. The colors correspond to the images of each individual, revealing a discernible separation between different groups. It's noteworthy that the separation is not perfect, which could be attributed to suboptimal performance of the network. Nevertheless, distinct clusters corresponding to each person are observable. In this context, shorter distances in space indicate greater similarity between images.

3.3 Conclusions

In this second phase of the project, a triplet network was implemented to recognize whether two images depict the same person or different individuals. This architecture enables the training of a neural network (NN) that, once trained, can receive any pair of images as input without requiring additional training. The embedding network can map images into a space where similar images are close to each other. By evaluating the distance between these images, it becomes possible to determine whether they represent the same person or not.

4 References

The material utilized in this project corresponds to the resources outlined in the project proposal.