Using Face Recognition for Authentication

UDACITY

Project Report
Bodziony Rafał

I. Definition

1. Project Overview

Nowadays, internet security is one of the basic requirements of the modern world, security should consist of two or three types of authorization to be safe. Safeness can be described as Something you know, something you have or something you are.[1] The biometric data is unique and impossible to forget e.g. how we look, how we move, or what our eyes look like can be used as an authorization component and increase security. One of the basic biometric features is the face. By appropriately transforming this information, we are able to add an important component to authorization, e.g. checking employees entering the company or as an additional confirmation of payment in the bank to prove the transaction. This problem is called Face Recognition and is an important scientific issue. Chiara Turati et al. showed that even a few days children are able to distinguish a familiar face. The problem of accurate facial recognition and identification can be solved by using neural networks, the task is to extract a feature vector of a face and compare it with the vector which already exists in the database.

The motivation behind the facial recognition app is to take the next step in protecting privacy and increasing online and real-world security. If the face detection algorithm were effective enough to be used commercially, the possibilities could be enormous. From signing messages with a key based on biometric data to simplified multi factor authentications to the automation of security in the real world, e.g. opening doors or locks.

For training and testing purposes and for input/output data as well I will use LFW Face Database. Labeled Faces in the Wild is a public benchmark for face verification, also known as pair matching.[2]

1. Examples of images from LFW Dataset

[1] Brandom, Russell (July 10, 2017). "Two-factor authentication is a mess". *The Verge*. Retrieved July 10, 2017.

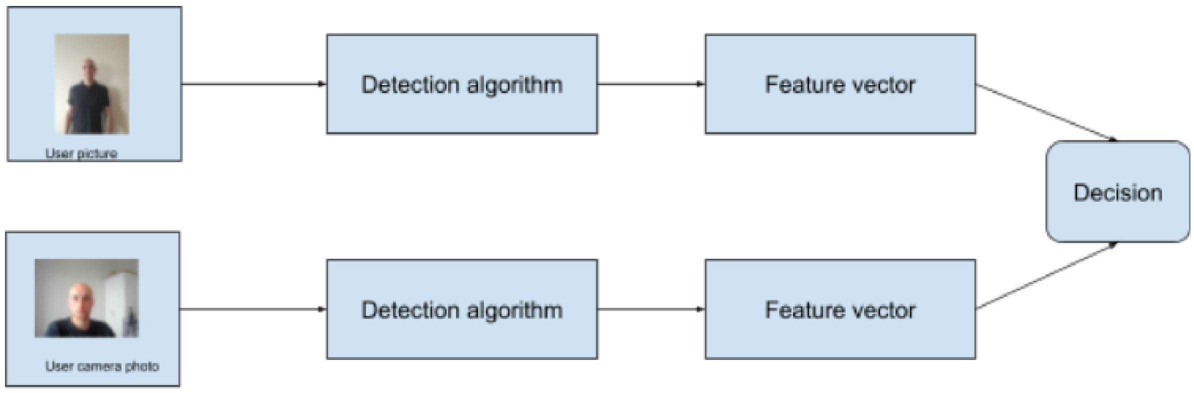
[2] http://vis-www.cs.umass.edu/lfw/

The dataset will be split to train and test sets based on information on the website. Then, I will use face images to distinguish pairs of the same person. I don't assume to recognize the identity information appearing on each image.

The project is assumed to use basic methods such as Eigenfaces to prove the baseline model results and Convolutional neural network-based models. I want to choose one model used in the paperswithcode.com benchmark. Images will serve as input data to the model and throughout the learning process images of the same person will be compared to minimize the distance between feature vectors. Models prepared in this way will be used to the image uploaded by the user and real-time webcam information to decide whether a face is the same or not.

2. Problem Statement

Make an application for user authentication based on facial recognition. In the first step, the user should add a photo of his face, then the algorithm via the webcam should distinguish whether he sees a real face or a photo, and finally determine whether he sees the same user.



2. Basic Model Scheme

3. Metrics

For this project, I will use accuracy as an evaluation metric, suggested by the authors of the LFW dataset. Furthermore, to check potential overfitting over methods I will use F1 to compare final results on validation set.

- Accuracy score is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations

$$Accuracy = \frac{TP+TN}{TP+FN+TN+FP}$$

- F1 score - We compute measure F that uses the harmonic mean (precision and sensitivity) instead of the arithmetic mean because it penalizes (harmonic mean) extreme values more. An F1 result gives equal weight to both measures and will always be closer to the lower precision or sensitivity value.

$$F1 = \frac{2\ x\ precision\ x\ recall}{precision\ x\ recall}$$

II. Analysis

1. Data Exploration

Labeled Faces in the Wild is a public benchmark for face verification, also known as pair matching. The LFW dataset contains more than 13,000 images of faces collected from the web. Each face has been labeled with the name of the person pictured. 1680 of the people pictured have two or more distinct photos in the data set. Dataset was splited to train, test, by researchers. To precise, dataset contains:

- 13233 images
- 5749 people
- 1680 people with two or more images

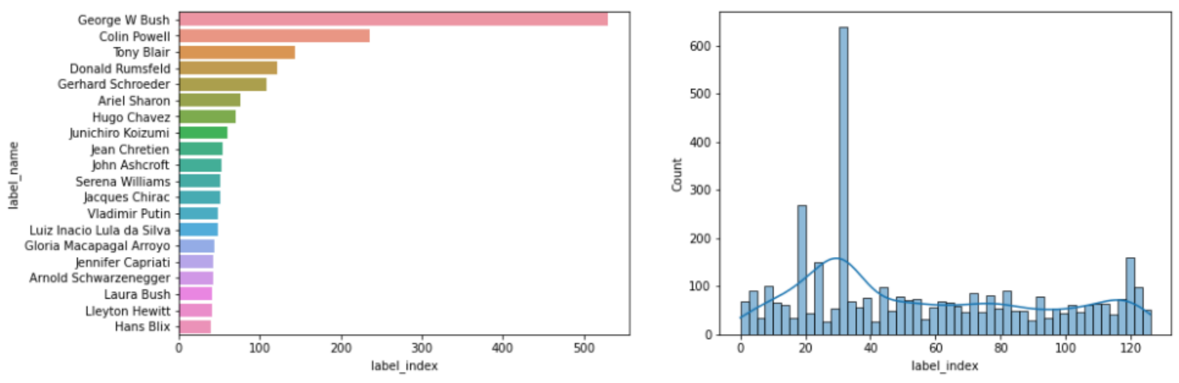
The data will be paired as follows:

- For PCA to [anchor / positive or negative] pairs to increase the number of positive and negative examples
- For Siamese Network to [anchor, positive, negative] pairs, this will allow using triplet loss during training.

Then the data will be saved to local disks and later transferred to S3. Data will be further used for training, test, and validation of results. Furthermore only for training and testing data will be splitted diferentelly, for validation I will use one dataset for PCA and Siamese Network.

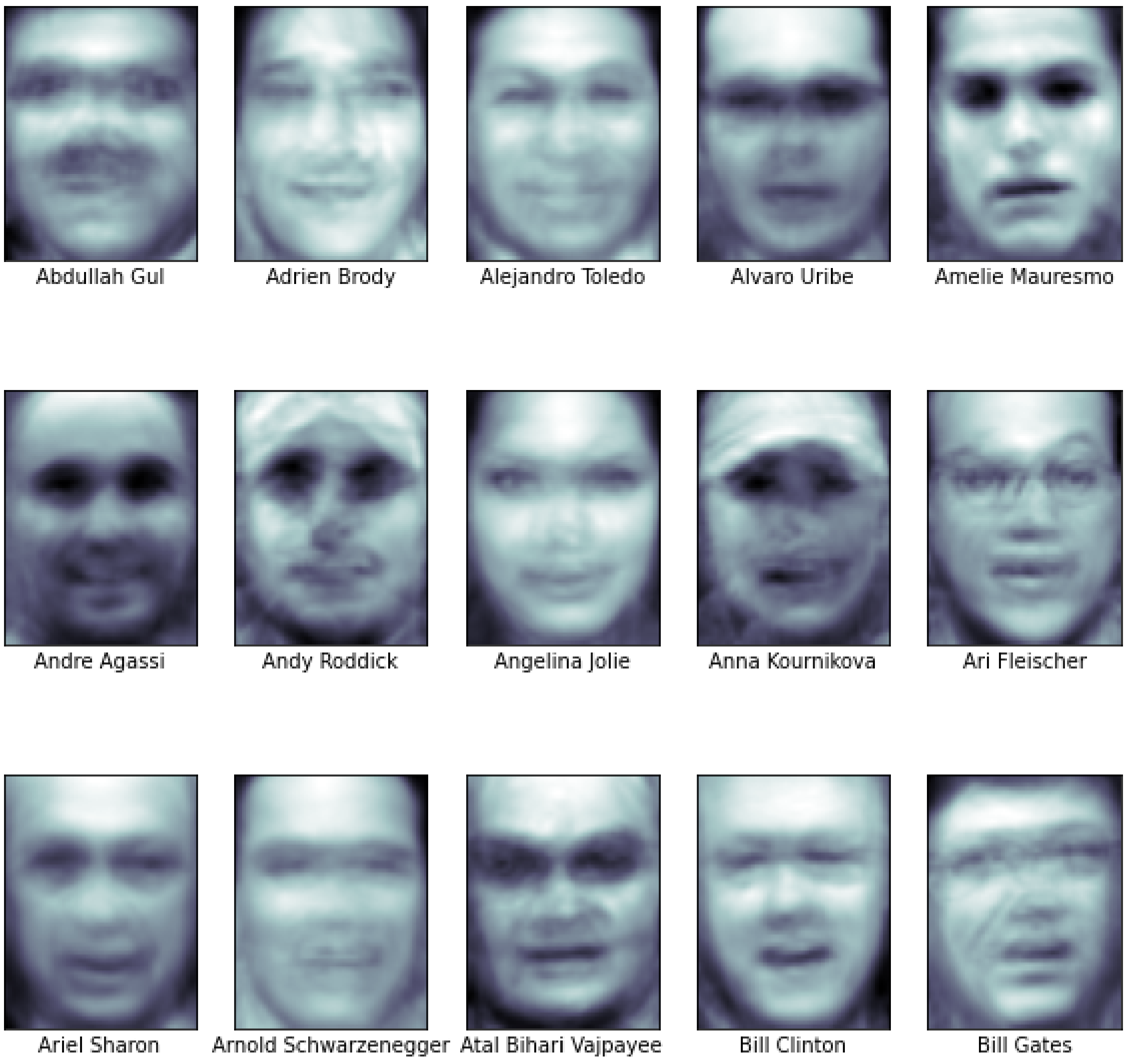
2. Exploratory Visualization

To conduct the experiment and create algorithms, use faces where the number of occurrences in the set is greater than 12. If we assume that the data should be divided into anchors, positive or negative pairs, then for each pair, we should have at least two unique faces. However, in order to prevent duplicate photos, an initial threshold was established which allowed dividing the set into three subsets - training, test, and validation in which there are at least 4 different photos of the same person. The distribution of faces with the division into the number of occurrences in the set is shown below, additionally, it has been checked which of the faces are the most popular.



3. Data analysis example based on dataset

We see that the large number of faces are politicians, especially men. Additionally, the number of photos of George W. Bush and Colin Powell is overestimated. During the process of creating datasets, the number of images for the given people should be normalized. For this reason, the set is divided according to the unique labels, not the presence of a given face, thanks to which a balanced set can be obtained in terms of the distribution of pairs.



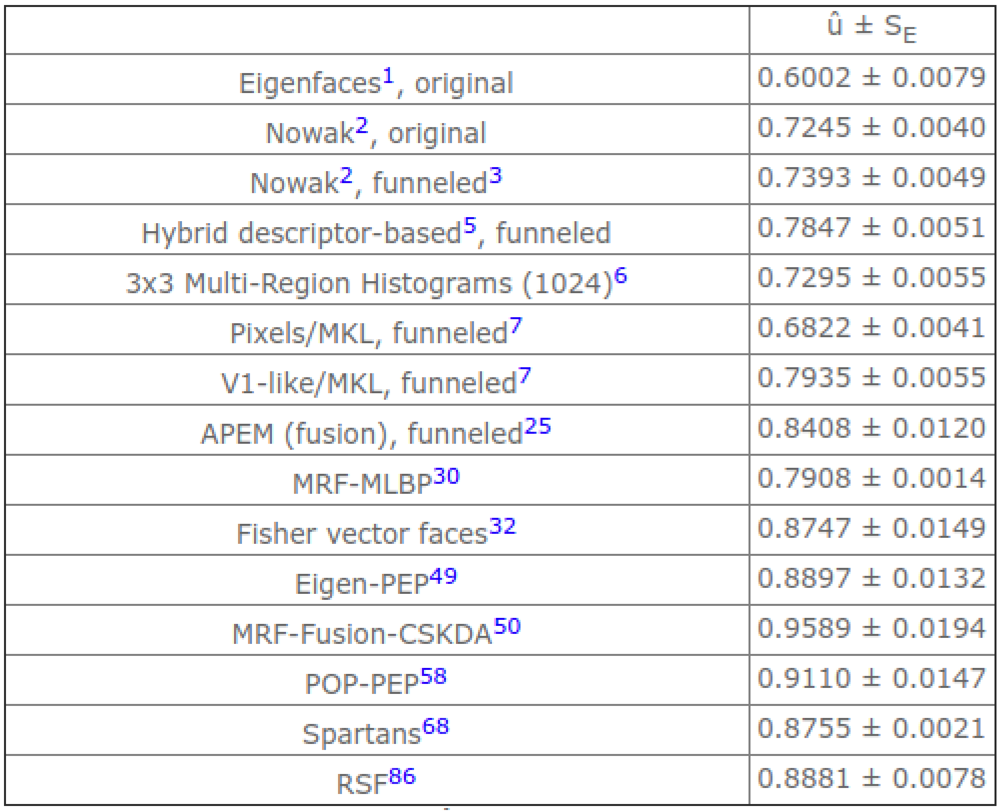
4. Mean average of sample faces

Above we see average per-pixel images of the sample photos in the dataset. You can see that the faces differ in particular with the positioning of the eyes, the nose, and the mouth. Additionally, glasses are visible on some people's faces, which may hinder the process of recognizing them.

3. Algorithms and Techniques

Two algorithms for face re-identification were used in the experiment. The first was used as a baseline model, while the second one is a model whose efficiency has been improved over the base model.

The basic model is to create a PCA to reduce the size of pixels in photos. PCA is a popular method that works well for reducing the dimensionality of data. PCA keeps critical information hidden in data through a linear transformation. In photos, this process can be understood as reducing the number of pixels that do not affect key areas of the image. The method is based on the baseline used and described on the website http://vis-www.cs.umass.edu/lfw/results.html. We can find benchmark models for Image-Restricted results without output-side data. The Table below shows benchmarked models, I will use the first Eigenface to benchmark my algorithm.

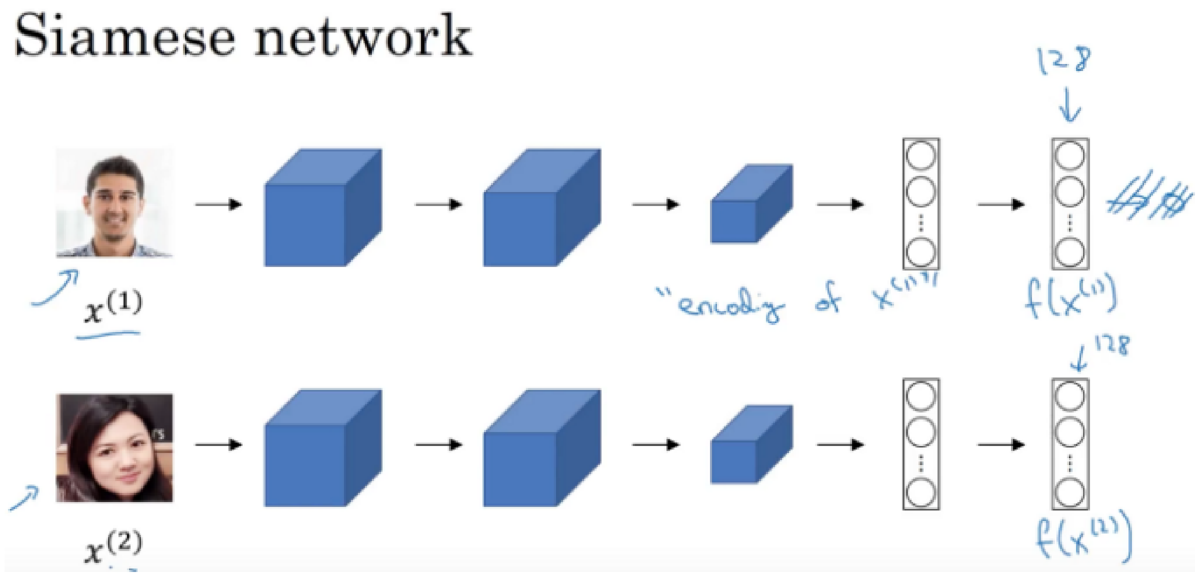


	$\hat{u} \pm S_E$
Eigenfaces[1], original	0.6002 ± 0.0079
Nowak[2], original	0.7245 ± 0.0040
Nowak[2], funneled[3]	0.7393 ± 0.0049
Hybrid descriptor-based[5], funneled	0.7847 ± 0.0051
3x3 Multi-Region Histograms (1024)[6]	0.7295 ± 0.0055
Pixels/MKL, funneled[7]	0.6822 ± 0.0041
V1-like/MKL, funneled[7]	0.7935 ± 0.0055
APEM (fusion), funneled[25]	0.8408 ± 0.0120
MRF-MLBP[30]	0.7908 ± 0.0014
Fisher vector faces[32]	0.8747 ± 0.0149
Eigen-PEP[49]	0.8897 ± 0.0132
MRF-Fusion-CSKDA[50]	0.9589 ± 0.0194
POP-PEP[58]	0.9110 ± 0.0147
Spartans[68]	0.8755 ± 0.0021
RSF[86]	0.8881 ± 0.0078

5. Mean classification accuracy $\hat{u}$ and standard error of the mean S_E

The main solution assumed to use the Siamese Network.[3] The Siamese network is based on the assumption that the network should extract similar features for two or more images of the same person. The model is trained on three images at the same time with triplet loss adaptation. Layers extract features space for images and create a vector that is used in comparison with a given distance metric function e.g. cosine. Below, the siamese network is described.

[3] Chicco, Davide. (2020). Siamese Neural Networks: An Overview. 10.1007/978-1-0716-0826-5_3.



6. Siamese network explanation

The next step of the algorithm is to calculate the threshold at which we consider that the base photo that the user added during the registration process is similar in terms of the distance to the photo that was used during login.

The threshold calculated from the training set will be used to determine whether the face pairs in the training and test sets are the same people or not. This way the accuracy will be calculated and the threshold could be used in the production phase. The pseudo-code in python that was used to calculate the threshold is shown below.

```python
# calculate positive distance pairs
distances_positive = list()
for anchor, positive in anchor_positive_pairs:
    anchor_vector = model(anchor)
    positive_vector = model(positive)
    distance = euclidean_distance(anchor, positive_vector)
    distances_positive.append(distance)

# calucalte negative distance pairs
distances_negative = list()
for anchor, negative in anchor_negative_pairs:
    anchor_vector = model(anchor)
    negative_vector = model(negative)
    distance = euclidean_distance(anchor, negative_vector)
    distances_negative.append(distance)


# optimize threshold
optimize = optimize_threshold(positive=distances_positive, negative=distances_negative,
                              metric=accuracy)
```

7. Optimization threshold snippet

4. Benchmark

For this experiment, the PCA will be used for the set benchmark. Additionally, the threshold will be calculated. The threshold can be interpreted as the best distance of the reduced facial vectors of the same person. The Euclidean metric will be used to compare the reduced vectors. During the experiment, the Hyperopt library was used to optimize the threshold. For optimization, Hyperopt uses a so-called surrogate model that estimates the performance of the expensive objective function on a set of parameters. When a face is detected, the optimization function is specified as

$$threshold = minimize(1 - accuracy(labels, binary)$$

where

$$labels = 1\, if\, pairs(anchor, positive)\, else\, 0$$
$$binary = 1\, if\, distance < threshold\, else\, 0$$

III. Methodology

1. Data Preprocessing

The algorithms that were used in the experiment require the use of images as input in a given format

- For the PCA algorithm: [anchor, positive] or [anchor, negative]
- For Siamese Network: [anchor, positive, negative] because triplet loss was used in training and test phase

8. Sample of preprocessing dataset with anchor, positive and negative examples

The data was collected using the sci-kit-learn library. Then the set was divided into training, test, and validation sets. Moreover, only the training and testing sets have been divided as above. The validation set was created once, to compare the results of the algorithms in format [anchor, positive/negative]. As the data showed a large percentage of photos are the same people, so it was decided to divide them by unique labels, thus avoiding overfitting in the data.

However, this created a problem with the small amount of data as the base number of unique faces in the set exceeding 12 photos per tag was only 127. Therefore, a repetition was used for each tag during the preprocessing. For the training set, it was 8 repetitions, for the test set it was 2 repetitions, while the validation set has only one positive pair and one negative pair for each unique assay. In the case of the training and testing set for the Siamese network, the repetition rate was multiplied by 2. In this way, the algorithms received data in the following form:

- Training set: 2032 examples where half is pairwise [anchor, positive] and the other half is [anchor, negative] or [anchor, positive, negative]
- Test set: 508
- Validation set: 254

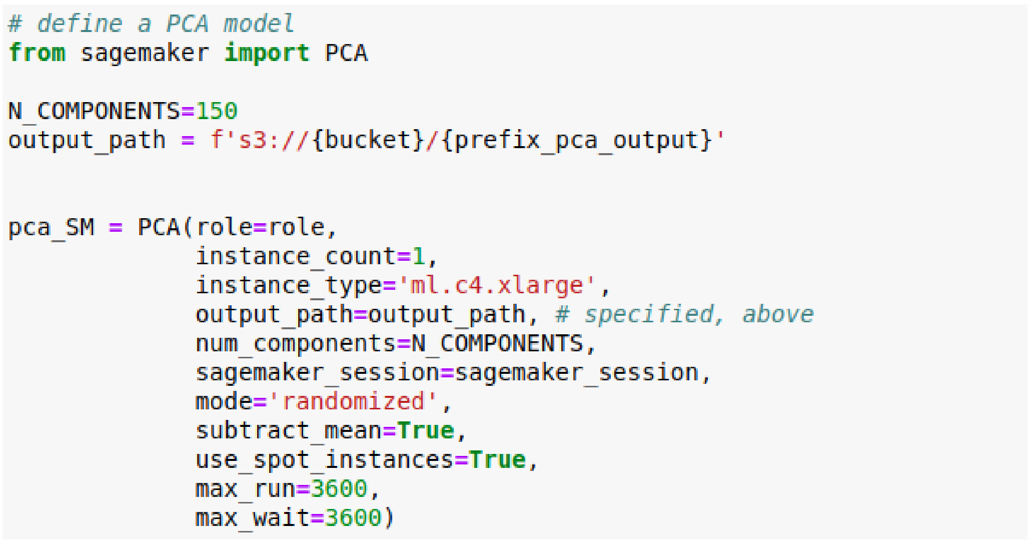
2. Implementation

The implementation of the algorithms for conducting the experiments was divided into three stages, which are described below

1. Creation of a basic model based on PCA
2. Creation of a model based on the Siamese Network
3. Threshold optimization[4]

PCA Implementation

The Amazon SageMaker PCA algorithm is the build-in algorithm that can be used for any type of data. Below, a basic fitting method is described. We can add the number of components for PCA, subtract the mean (which normalizes the data) as parameters. Furthermore, we can add mode, instance type, and spot instance. Spot Instances let us take advantage of unused EC2 capacity in the AWS cloud. Spot Instances are available at up to a 90% discount compared to On-Demand prices.



```
# define a PCA model
from sagemaker import PCA

N_COMPONENTS=150
output_path = f's3://{bucket}/{prefix_pca_output}'


pca_SM = PCA(role=role,
             instance_count=1,
             instance_type='ml.c4.xlarge',
             output_path=output_path, # specified, above
             num_components=N_COMPONENTS,
             sagemaker_session=sagemaker_session,
             mode='randomized',
             subtract_mean=True,
             use_spot_instances=True,
             max_run=3600,
             max_wait=3600)
```

9. PCA training snippet

Siamese Network Implementation

Since Siamese Network is not available as a ready-made algorithm through the SageMaker SDK, it was necessary to create the algorithm in script mode. Sagemaker offers the possibility to use written training scripts. Script mode is a popular method for training custom algorithms with AWS Sagemaker containers. For this stage, the Tensorflow 2 framework was used. The triplet loss function[5] presented in the FaceNet paper[6] was used to learn the network, the equation used to calculate the cost function is presented below

$$L(A,\ P,\ N)\ =\ max(p(a,p)\ -\ p(a,\ n)\ +\ margin,\ 0)$$

[4] http://hyperopt.github.io/hyperopt/

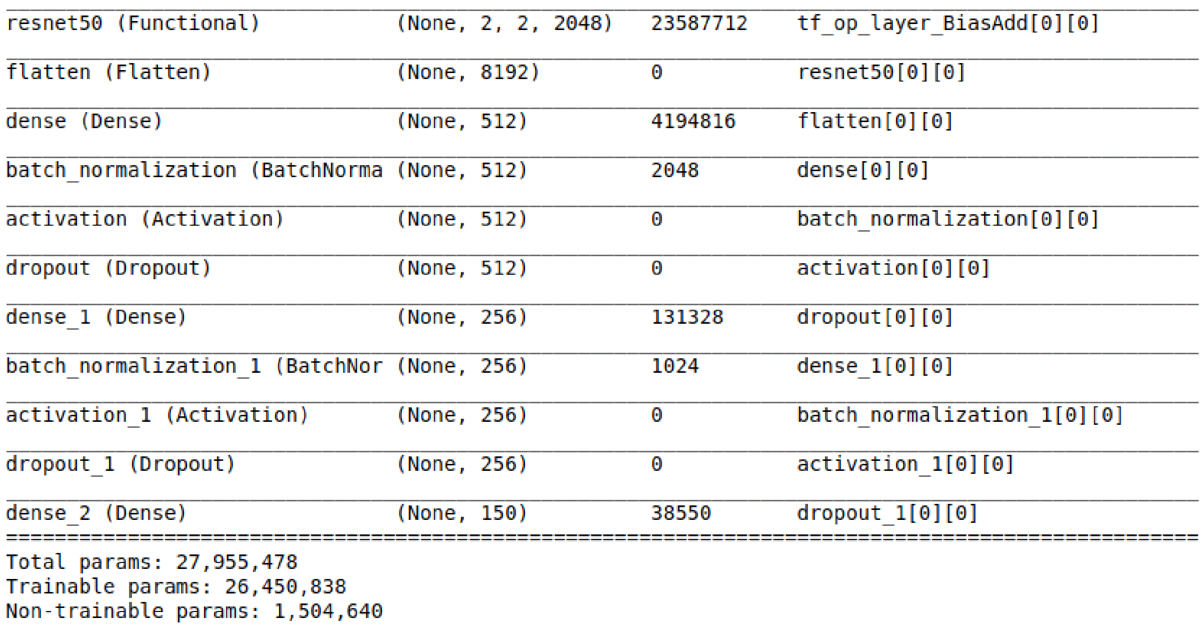
[5] Hermans, Alexander & Beyer, Lucas & Leibe, Bastian. (2017). In Defense of the Triplet Loss for Person Re-Identification.

[6] Schroff, Florian & Kalenichenko, Dmitry & Philbin, James. (2015). FaceNet: A unified embedding for face recognition and clustering. 815-823. 10.1109/CVPR.2015.7298682.

where p function can be described as

$$p(a,\ b)\ =\ ||f(a)\ -\ f(b)||^{2}\ where\ f\ \sim\ neural\ network$$

To generate embedding for each image of the triplet, the pretrained ResNet50[7] model was used. It was connected with three Dense layers. The model summary is described below



```
resnet50 (Functional)           (None, 2, 2, 2048)   23587712    tf_op_layer_BiasAdd[0][0]

flatten (Flatten)               (None, 8192)         0           resnet50[0][0]

dense (Dense)                   (None, 512)          4194816     flatten[0][0]

batch_normalization (BatchNorma (None, 512)          2048        dense[0][0]

activation (Activation)         (None, 512)          0           batch_normalization[0][0]

dropout (Dropout)               (None, 512)          0           activation[0][0]

dense_1 (Dense)                 (None, 256)          131328      dropout[0][0]

batch_normalization_1 (BatchNor (None, 256)          1024        dense_1[0][0]

activation_1 (Activation)       (None, 256)          0           batch_normalization_1[0][0]

dropout_1 (Dropout)             (None, 256)          0           activation_1[0][0]

dense_2 (Dense)                 (None, 150)          38550       dropout_1[0][0]
==================================================================================================
Total params: 27,955,478
Trainable params: 26,450,838
Non-trainable params: 1,504,640
```

10. Siamese Model summary

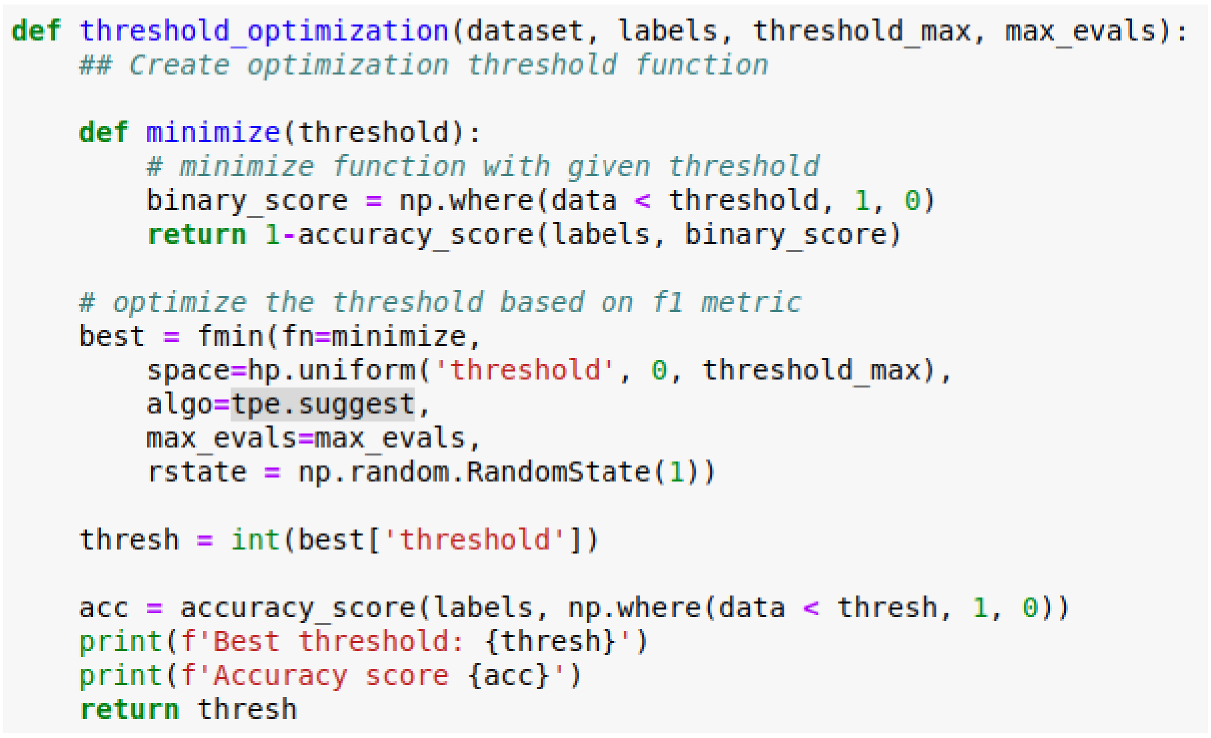
L2 normalization, batch normalization, and Dropout have been added to the model causing a reduction of overfitting. Because the model was created in script mode, where the output was saved in a format compatible with Tensorflow Serving and TensorflowModel class from Sagemaker SDK,. After training, the model was published in endpoint mode.

Threshold optimization

For the threshold estimation, the hyperopt library[8] was used. Hyperopt is a Python library for serial and parallel optimization over awkward search spaces, which may include real-valued, discrete, and conditional dimensions. To compare vectors euclidean distance as used. Moreover, optimization was conducted base on the minimization function described above. As mentioned before, algorithms create embeddings for each image and estimate thresholds based on distance. The threshold obtained in this way will be used to check the metrics for the test and validation sets.

[7] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2015). Deep Residual Learning for Image Recognition. 7.

[8] http://hyperopt.github.io/hyperopt/



```python
def threshold_optimization(dataset, labels, threshold_max, max_evals):
    ## Create optimization threshold function

    def minimize(threshold):
        # minimize function with given threshold
        binary_score = np.where(data < threshold, 1, 0)
        return 1-accuracy_score(labels, binary_score)

    # optimize the threshold based on f1 metric
    best = fmin(fn=minimize,
        space=hp.uniform('threshold', 0, threshold_max),
        algo=tpe.suggest,
        max_evals=max_evals,
        rstate = np.random.RandomState(1))

    thresh = int(best['threshold'])

    acc = accuracy_score(labels, np.where(data < thresh, 1, 0))
    print(f'Best threshold: {thresh}')
    print(f'Accuracy score {acc}')
    return thresh
```

11. Threshold optimization snippet

3. Refinement

The basic problem in the data was the different positions of a face on the photo, which is visible in the images of the mean analysis for the selected labels. In the case of PCA and Siamese Network, we have several parameters that can be optimized. For this purpose, it would be best to use the Hyperparameter tuning job module which finds the best version of the model by running many training jobs with datasets using given algorithm and hyperparameters. Certain parameters should be taken into account during optimization.
For the PCA model:

- Normalization
- Number of components
- Randomization with additional components

For the Siamese Network we have more parameters:

- Backbone network
- Epochs
- Number of Dense layers
- Number of neurons in each Dense layer
- Dropout rate
- Kernel regularization
- Number of components

Of course, an additional parameter that would be able to improve the result is adding an additional portion of data or augmenting the already obtained dataset.

IV. Results

1. Model Evaluation and Validation

This section discusses the model based on the Siamese Network concept. As already mentioned, the data has been divided into three sets. A set of three different photos was used for the training and testing of the final model. These photos can be divided into a base photo, a photo of the same, and a different person. The dataset was created because Triplet Loss was used as the loss function (described above). The premise of the experiment was to create a model that works in an unsupervised manner. This means that after training, we will be able to apply the model to each user who provides a base photo. Another production assumption may be to use multiple photos of the user as base photos to enhance the accuracy of the model. Another advantage of the model is the relation to unsupervised methods such as PCA, the only assumption is to determine the appropriate threshold. That is, determining the threshold unambiguously determines the division of the data into pairs of base and positive or negative images. If the set is clearly divided, that is, there is a difference between the final result of the data distribution or the determined metric that meets the business expectations.

The basic metrics used for classification were used to determine the accuracy of the model. It was assumed that if the distance between the embedding vector of the base face and the embedding vector of the second face of the pair is smaller than the threshold determined on the training set, we assume that the people in the photos are the same person. In this way, it was possible to label the pairs and apply the metrics described above. The table shows the parameters that were used during the experiment. As mentioned before, the best way to optimize the parameters would be to use the HyperParameter Tuning job on AWS. However, in the presented experiment, manual selection of parameters based on the metric was chosen.

	Siamese Network Model Parameters					
Name	Backbone	Components	Batch Size	Threshold	Distance Method	Train Accuracy
V1	Resnet50	250	100	2.9	euclidean	0.78
V2	**Resnet50**	**150**	**256**	**3.11**	**euclidean**	**0.88**
V3	VGG16	250	100	2.4	euclidean	0.7
V4	VGG16	150	256	2.6	euclidean	0.68

12. Siamese model parameters

In the deep machine learning model, two architectures were used that served as a backbone for initial data embedding. ResNet-50 is a convolutional neural network that is 50 layers deep. The TensorFlow library enables pre-trained weights based on the imagenet dataset. Another property of this network is residuals. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers. Typical ResNet models are implemented with double- or triple- layer skips that contain nonlinearities (ReLU) and batch normalization in between.[1][2] Another base architecture that has been tested is VGG16. VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. It was one of the first architectures that used convolutional networks. The next parameters that were checked during the experiment are:

- Components number - defines vector dimension at the output of the model
- Batch size - defines the number of samples that will be propagated through the network.
- Threshold - The threshold below which we assume that the distance between the vectors is the same face
- Distance Method - a method to measuring the distance between vectors

In the case of the models indicated above, a comparison was made for data from training, test, and validation sets. Moreover, data from the validation set was not available at the time of training the network.

	Siamese Network Model Results					
Name	Train Accuracy	Test Accuracy	Validation Accuracy	Train F1	Test F1	Validation F1
V1	0.78	0.70	0.68	0.76	0.66	0.67
V2	**0.88**	**0.73**	**0.72**	**0.89**	**0.73**	**0.73**
V3	0.7	0.67	0.65	0.74	0.65	0.63
V4	0.68	0.63	0.62	0.66	0.61	0.6

13. Siamese model results

The performance of Siamese Networks with larger batch size and lower embedding dimension is slightly worse, in terms of comparing accuracy between sets. This means that neural networks overfit on train dataset. It can be improved by adding more regularization techniques, data, or implementing augmentation techniques.

2. Justification

The comparison of the base model and the model produced during the experiment is presented in the table below

	Model Comparison	
	Validation Accuracy	Validation F1
PCA Model	0.58	0.51
Siamese Network	**0.72**	**0.73**

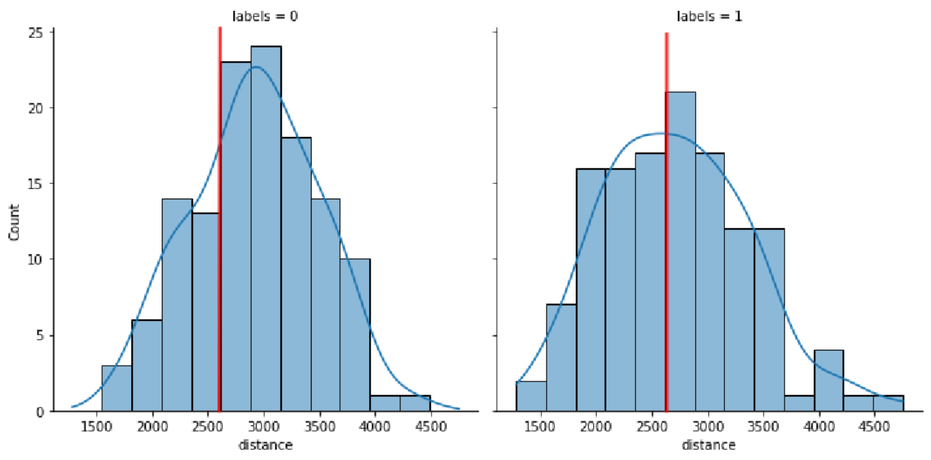
14. Final model comparison table

We can see the results on the validation are differ, which means that our model overfits slightly. The Siamese Network base model achieved 0.72 accuracy for the validation set. This is ~15% better results than the PCA base model. We can assume that more data or a better backbone model could help to stabilize the score similar to the accuracy achieved on train data ~0.88.

V. Conclusion

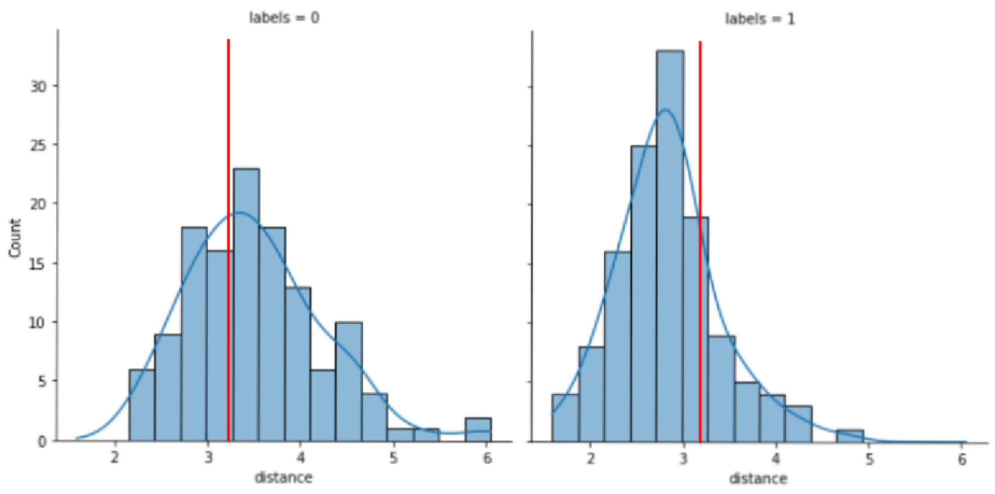
1. Free-Form Visualization

As a result of the experiment, the model threshold was determined, which determined the correct recognition of the face, with the anchor face. It is worth looking at the validation data distributions concerning the threshold which was calculated based on the optimization.



15. Histogram of PCA distances

It can be clearly seen that the distribution of values for negative (label = 0) outside the red line is visibly shifted, which indicates that the PCA model works well for many photos. However, positive image pairs (label = 1) are distributed normally. This means that the result is much more randomized, so the model should not be used in the production phase.



16. Histogram of Siamese Network distances

In the neural network model, the value distributions look much better. We can see that the distribution for positive cases (label = 1) is shifted to the left, which means that a lot of data is properly recognized. However, the distribution of values for negative cases is surprising - it may indicate overfitting of the model, therefore this issue should be corrected during the next experiment.

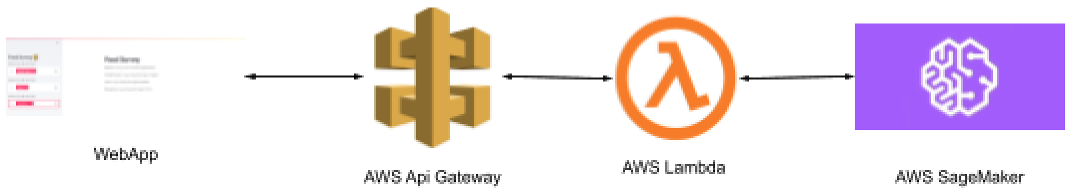
2. Reflection

The most difficult aspect of this project was finding the dataset with proper images. Moreover, the very idea of a threshold-based implementation seemed abstract at the beginning. The final model still needs to be improved and is not ready for implementation, it is possible that the next step could be to use a different form of face re-identification algorithm. To be sure of the algorithm's effectiveness, additional data should also be collected for which the accuracy is above 0.98. This would enable the effective implementation and production use of the system. Ultimately, the implementation of additional mechanisms for checking user photos and protection against attacks should be a key assumption during the production implementation.

3. Improvement

The project involves the development of a model to verify a person using computer vision algorithms. The future project steps could involve the development of an application to verify a person using computer vision algorithms. The first step to improve would be to collect more data and perform new training with additional hyperparameter tuning and augmentation process, at this stage, it is also assumed to add data from other sets in order to normalize the set in terms of ethnic and age diversity.

Next, models to extract faces from the image based on simple computer vision algorithms would be created. Also in this step, tracking face with preventing adversarial attacks It is also assumed to create a web application on which the user will be able to place a photo and re-identify it later based on registration information. For this backend pipeline AWS Lambda, DynamoDB, and SageMaker could be used.



17. Simple architecture scheme for Face Recognition service