Face Recognition with anti spoofing and liveness detection

Mirza Mujtaba Hussain
Email: mirza.1989740@studenti.uniroma1.it

Nagarjun Lakshmipathy
Email:lakshmipathy.1982435@studenti.uniroma1.it

Waddah Alhajar
Email:alhajar.2049298@studenti.uniroma1.it

1. Introduction

Face recognition is a technology that enables the automatic identification of an individual from digital images or videos of their face. This technology has gained significant attention in recent years and has a wide range of applications, including security systems, access control, criminal investigations, and many more.

In this project, our main aim is to create a face recognition system for an attendance system. The attendance system requires an accurate and efficient face recognition system to identify the person, who are present in a particular location.

The face recognition process can be divided into several stages, such as face detection, face feature extraction, and face matching. In this project, we have used a python libraray, dlib's face recognition module for face detection, which provides accurate results and is known for its speed.

Another critical aspect in face recognition systems is the anti- spoofing technique implemented, which ensures that the recognition process is not fooled by fake faces or photographs. In this project, we have trained a classifier that can detect if an image is real or not. We have also added blink detection for liveness detection.

For feature extraction, we have trained Siamese Network using triplet loss. This network takes the face and produces 256 dimensional embeddings which we use later in the face matching phase.

In summary, this project aims to provide an efficient and accurate face recognition system, which can be integrated into an attendance system with confidence. The system is built using state-of-the-art techniques and is robust against face spoofing attacks.

In the coming sections, we will see each of these sub tasks in detail.

2. Data

In this project, we have trained two classifiers, one for training anti-spoofing classifier and another for training the siamese network.

We used the CelebA Spoof dataset for training our anti-spoof classifier. The CelebA Spoof dataset is a large-scale face anti-spoofing dataset that contains images of both real and fake faces. The dataset consists of over 200,000 images, but due to computational limit, we only used 10,000 images from each class (real and fake) to train the classifier.

For training the siamese network, we used "Face Recognition Dataset - Oneshot Learning" from Kaggle which is a database of face photographs designed for the creation of face detection and recognition models. The dataset contains 1680 directories, each representing a celebrity and each directory has 2-50 images for the celebrity.

3. Face Detection

We have used dlib library for face detection.The dlib library provides two functions for face detection:

- **HOG + Linear SVM**
- **MMOD CNN**

For this project we used the The HOG + Linear SVM face detector and it is accessed through the function dlib.get_frontal_face_detector(), which returns a pre-trained HOG + Linear SVM face detector included in the library. This face detector is designed to be fast and efficient, and it uses the Histogram of Oriented Gradients (HOG) descriptor to extract features from an image. It is also combined with an image pyramid and a sliding window detection scheme.

An image pyramid is a collection of multi-scale images, where each level of the pyramid represents the same scene at a different scale. This technique is used in object detection algorithms to handle different object scales in an image. By processing the image at different scales, the object detection algorithm can identify objects at different scales in the same image.

A sliding window detection scheme is used to scan an image at different scales. In this scheme, a window of a fixed size is moved across the image, and a classifier is applied at each window location to determine if there is an object in the current window. The sliding window detection scheme

can be applied at each level of the image pyramid to detect objects of different sizes in the image.

HOG, or Histogram of Oriented Gradients, is a simple yet powerful feature descriptor that is widely used for object detection, including faces. It is robust for object detection because the shape of an object is characterized using the local intensity gradient distribution and edge direction. The process of HOG-based face detection can be divided into three steps:

Color Normalzation: In computer vision image resizing and normalizing the pixels values are very important steps and same goes here

Computing the Gradients:Typically, computing the gradients of an image in computer vision reveals those locations where the pixel gradient intensities change. Thus, this leads to a lot of useful information.

Dividing the image into small connected cells: This allows for the calculation of histograms for each cell.

Computing histograms for each cell: A histogram is a representation of the distribution of a set of values. In this case, the values are the gradient intensity and edge direction in each cell.

Bringing all histograms together to form a feature vector: All the histograms from each cell are combined to form a unique feature vector, or one histogram, that represents the face.

The resulting HOG features are then fed into a machine learning model, such as a support vector machine (SVM), for classification.The main disadvantage of HOG-based face detection is that it does not work well on faces that are viewed from odd angles. It is optimized for straight and front-facing faces and is useful for detecting faces from scanned documents such as driver's licenses and passports, but it is not well suited for real-time video.

4. Anti-spoofing Techniques

Anti-spoofing techniques are used to detect if the person being recognized is real or not. These techniques are important in security systems such as face recognition systems, where it is crucial to ensure that the person being recognized is not using a fake face, such as a mask , video or a photograph. In this project, we are using two methods , one is blink detection for liveness detection and another is a classifier that can detect if the input image is real, or a spoof such as video or image.

Blink detection is a crucial aspect of anti-spoof techniques in face recognition systems. It verifies if the person is a live human or a fake representation, like a photo or video. This step helps in increasing the security and accuracy of the face recognition system.

The facial landmark detector from the dlib library helps in detecting the key facial attributes, including the eyes. Using the eye landmarks, we can calculate the Eye Aspect Ratio (EAR), which is the ratio of the horizontal and vertical measurements of the eye. A decrease in the EAR value indicates that the person is blinking, which is a sign of a live human.

The EAR remains constant when the eye is open but drops suddenly when the eye is blinked.By taking the average of the EAR for both the right and left eye, we can determine if a blink has occurred. A threshold is set to determine if the average EAR value is lower than the threshold, indicating a blink. The threshold value may vary depending on the FPS of the video or webcam, but typically for my project a value between 0.4 to 0.45 worked well.

Next, we trained a **Convolutional Neural Network** that can classify if the input is real or not. For training the network, I trained it in a binary fashion, where it classifies real as real, and all other labels, such as images, videos, posters as fake.

The first layer of the model is a 2D Convolutional layer with 64 filters, each with a size of 3x3. The activation function used is ReLU (rectified linear unit), which is a commonly used activation function in deep learning models. The input shape of the layer is specified as 128x128x3, which means that the input to this layer is a 128x128 image with 3 color channels.

The next two layers are Max Pooling layers, which are used to reduce the spatial dimensions of the input, while retaining important information. The pooling layers have a size of 2x2, which means that the spatial dimensions of the input are reduced by a factor of 2 after each pooling layer.

There are two more Convolutional layers, each with 32 and 64 filters respectively. The activation function used is also ReLU.

After the Convolutional layers, the model has a Flatten layer, which flattens the high-dimensional tensor output from the Convolutional layers into a 1D vector. This is followed by two Dense layers with 64 and 2 neurons respectively, with the activation function ReLU and softmax, respectively. The results obtained from this classifier are in Results section in later part of the report

5. Face Recognition

Siamese networks are neural networks that consist of two or more identical sub-networks. These networks are used for tasks like one-shot learning, face recognition, and verification. The sub-networks in a Siamese network share the same parameters and can be trained end-to-end using backpropagation. One of the most popular applications of Siamese networks is face recognition, where the aim is to recognize a face from a large number of faces.

The use of Triplet Loss in Siamese Networks is particularly important in face recognition. Triplet Loss is used to optimize the embedding of the input image in a feature space, such that the distance between images of the same class is minimized, and the distance between images of different classes is maximized.

Triplet Loss works by training the network on triplets of images. Each triplet consists of an anchor image, a positive image, and a negative image(as seen in fig 1). The anchor image and the positive image belong to the same class, while

Figura 1. Triplet pairs used for training face recognition model

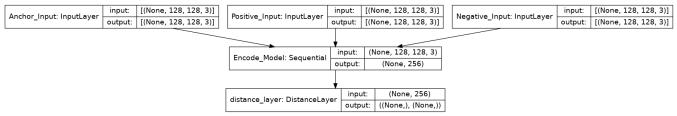

Figura 2. Siamese Network architecture

the negative image belongs to a different class. The aim is to minimize the distance between the anchor image and the positive image, and maximize the distance between the anchor image and the negative image. This way, the network can learn to discriminate between images of the same class and images of different classes. The loss is calculated as the difference between the distances between the anchor-positive and anchor-negative image pairs. The network is trained to minimize the Triplet Loss.Below is the formula of triplet loss, here alpha is the margin.

$$\mathcal{L}(a,p,n) = \max\left(\|\mathbf{f}(a) - \mathbf{f}(p)\|_2^2 - \|\mathbf{f}(a) - \mathbf{f}(n)\|_2^2 + \alpha, 0\right) \tag{1}$$

The Siamese model is build on top of a base encoder.The base encoder used in this architecture is the Inception network from Keras. The Inception network is

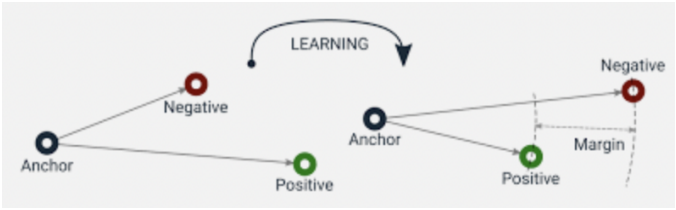

Figura 3. Triplet loss interpretation

a deep convolutional neural network that was designed to address the problem of multi-scale feature learning. The network consists of a series of layers with different types of filters, including 1x1, 3x3, and 5x3 filters. These filters allow the network to learn multi-scale features from the input image.The choice behind using this architecture over other popular architecture like Resnet-50 is because Inception focuses on computational cost, which makes it ideal for real time face recognition.

After training the Siamese Network, we can extract the encoder part from it which takes an image and returns a 256 dimensional encoding which we use further to recognise the faces.

Once the model is fully trained, we can use it extract feature embedding for any face. But to use it for the face recognition pipeline, we need at least one image of a person to get the embedding for the face and save it in a face encoding database. At the time of inference we extract the encoding of the given face and compare it with the encoding present in face encoding database.If the distance is less than a certain threshold,we can classify it as someone in the database otherwise we can return as unknown.

It is important to note that training this model need huge resources, so we used Google Colab to train the models.But unfortunately we were able to train it for only almost 20 epochs max because of computational restrictions on Google Colab.

6. Results

The two commonly used metrics in the field of Biometrics are False Acceptance Rate (FAR) and False Rejection Rate (FRR).

False Acceptance Rate (FAR) is the rate at which the system accepts an impostor as a genuine user. It measures the security of the system, and it is generally desirable to have a low FAR.

False Rejection Rate (FRR) is the rate at which the system rejects a genuine user. It measures the user-friendliness of the system, and it is also generally desirable to have a low FRR.

As the number of false acceptances (FAR) goes down, the number of false rejections (FRR) will go up and vice versa (see the figure below). The point at which the lines intersect also has a name: the Equal Error Rate (EER). This is where the percentage of false acceptances and false rejections is the same.

For our project, we used a set of images and persons that were not used while training the siamese network as a test set. We experimented with a bunch of threshold values and here are the FAR and FRR for each threshold:

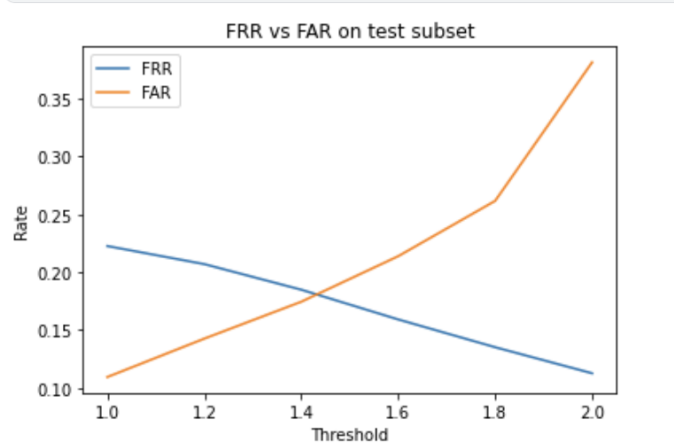

Figura 4. FAR vs FRR for the face recognition classifier on test set

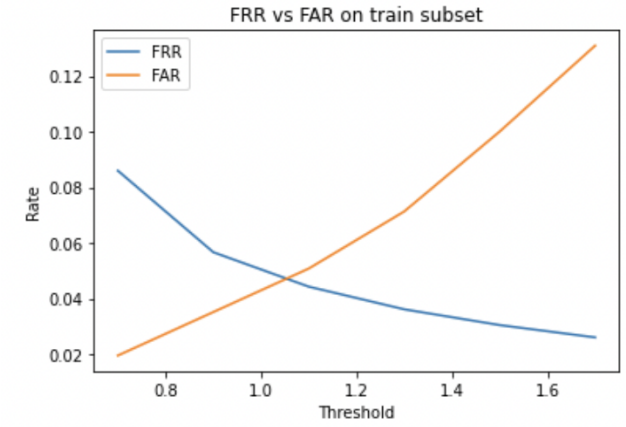

Figura 5. FAR vs FRR for the face recognition classifier on train set

As we can see in the figures above, the FAR and FRR values are not ideal for test set and the EER is around 0.19, this is not surprising given the fact that we were not able to train the network for many epochs as mentioned earlier.

On contrary, when tested on images of people it was already trained on, it was able to perform good with EER of 0.05 as seen from the figure 4.

Next, we calculate the results for the anti-spoof classifier. The FAR and FRR were calculated on a test set and the EER value is around 0.08.

7. Use Case

In this section, we will describe how we put all these modules together to create a facial recognition system for real time use. Our use case was an attendance system which

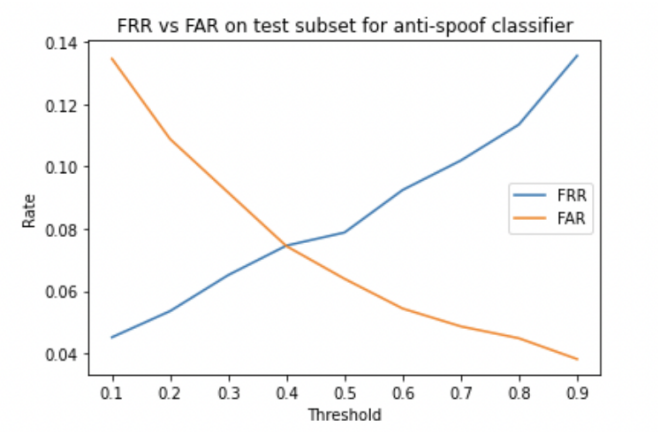

Figura 6. FAR vs FRR for the anti-spoof classifier

can recognize a person and at the same time ensure that there is no spoofing attack.So our main aim is to not mark anyone present unless completely sure,and in order to do that we do some tweaking to the way models works.

To get faster response from the trained models , I converted them to ONNX (Open Neural Network Exchange) format. This is an open-source format for representing deep learning models. It provides a standard representation of models that allows for interoperability between different deep learning frameworks and can lead to faster inference but still the main factor for faster inference is the hardware at hand.

In this setup, we are using OpenCV which is a Open Source Computer Vision Library to get the feed from the webcam. Once we are getting the input from webcam, we use the face detection method discussed above to detect faces.

Let us start by how we avoid spoofing attack. As discussed in the anti-spoofing techniques section, we have used blink detection and a anti-spoof classifier.

Although the anti-spoof classifier showed good results in development phase, its results were very unstable during use on webcam feed. So in order to get stable results, we first wait for 20 predictions made by anti-spoof classifier on the continuous feed from the webcam, and than taking mean of the latest 20 predictions, we use the mean of it to detect if the person is real or its some kind of presentation attack, like presenting video screen. Also based on our goal,we experimented with the values for threshold,and 0.2 seemed to work well.So a frame with prediction anything below 0.2 is considered to be not spoof. One reason for such low threshold is because we were using HD screens of smartphones to do presentation attack, but the model was not trained on such data and thus most of the times with higher threshold, it was not able to detect it as spoof.

In the same time, there is a blink detection system which detects for a blink, and at a frame, where a blink has been detected and also prediction from anti-spoof classifier is less than threshold, we further process this frame for face recognition.

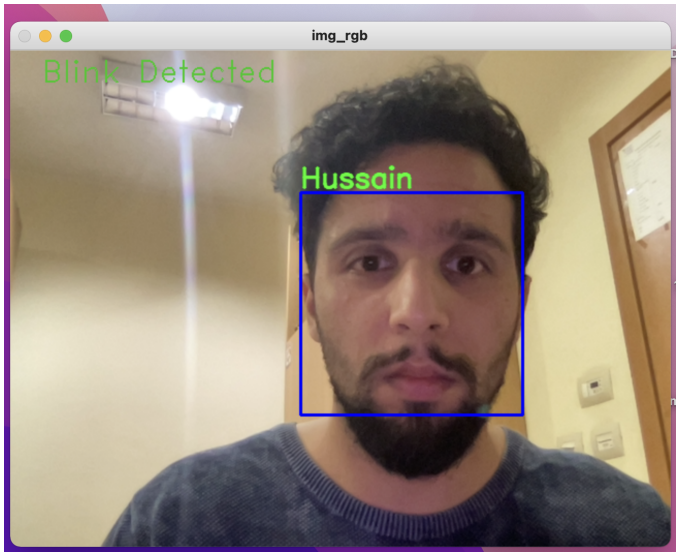

Figura 7. Face recognition system on real person

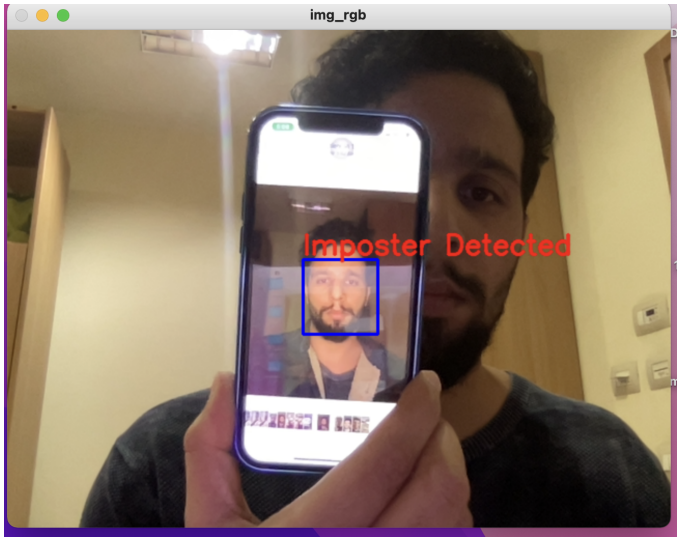

Figura 8. Face recognition system during spoofing attack

For face recognition model, we first fine tuned the Siamese model with our images and got the embeddings for each of our face.Although fine tuning is not necessary usually, but in our case as the model was not trained to the optimal level, it was necessary for demonstration purpose. Next we use extract the encoding from the trained model and compare the embeddings with the reference embeddings of people in database, and when the distance is less than 0.7, we classify it as same person, else it is classified as unknown.

8. Limitations

One of the main limitations of this system is the unstable outputs from the anti-spoof classifier. It is very much effected by the lightning conditions and does not predict accurately for presentation attacks carried through HD screens.This might because of the training data had no such instances so the model perhaps could not differentiate the HD input from the real person.
Other limitation is that it can only work with one face at a time, so the frame must have only face in it.

9. Conclusion

In conclusion, this project focuses on creating an accurate face recognition system for an attendance system. The system uses techniques such as dlib's face recognition module for face detection, Siamese Network for feature extraction, and anti-spoofing techniques such as blink detection and a classifier to detect fake faces. Although the face recognition system works fine, but it would be really beneficial to train it on a more diverse data set and train it for longer time .The results obtained from the implementation of this system show that it is somewhat robust against face spoofing attacks but yet there is lot of room for improvement.

10. References

- Eye Blink detection using Dlib and OpenCV
- Similarity learning using Siamese
- Anti-spoof detection using CNN