PEOPLE’S COMMITTEE OF HO CHI MINH CITY

SAIGON UNIVERSITY

TRẦN MINH KHOA

TRẦN HẢI KIM LONG

AN APPLICATION OF FACE VERIFICATION BASED ON DEEP LEARNING APPROACH

GRADUATION THESIS

MAJOR: INFORMATION TECHNOLOGY

EDUCATION LEVEL: UNIVERSITY

HO CHI MINH CITY, JULY 2022

PEOPLE’S COMMITTEE OF HO CHI MINH CITY

SAIGON UNIVERSITY

TRẦN MINH KHOA

TRẦN HẢI KIM LONG

AN APPLICATION OF FACE VERIFICATION BASED ON DEEP LEARNING APPROACH

GRADUATION THESIS

MAJOR: INFORMATION TECHNOLOGY

EDUCATION LEVEL: UNIVERSITY

INSTRUCTOR: DR. TRINH TAN DAT

HO CHI MINH CITY, JULY 2022

DECLARATION

I hereby declare that this is my own research work, the data and research results stated in the thesis are true, have been authorized to use by the co-authors and have never been published in any other publications.

Authorship

Trần Minh Khoa

Trần Hải Kim Long

ACKNOWLEDGEMENT

We would like to thank the Faculty of Information Technology, Saigon University for creating favorable conditions and environment for students in the process of studying, researching and completing their graduation thesis.

With deep gratitude, I would like to thank my teacher, Dr. Trinh Tan Dat has dedicatedly and enthusiastically guided and oriented scientific research methods for us, and provided a lot of materials and teaching during the study and thesis work. With his guidance, we had good orientations in the implementation and implementation of the requirements in the process of making the thesis. Without the direction and teachings of the teachers, our thesis would be very difficult to complete. Once again, we sincerely thank you.

We would like to thank the teachers in the Faculty of Information Technology, Saigon University for enthusiastically teaching and imparting valuable knowledge and experiences during our study period at school.

We would like to thank the students of class K18-DCT118C1, who accompanied us throughout the course and gave us many useful suggestions. Thank you to our family and friends for their concern and encouragement to help us have the energy to strive to complete this thesis well.

In the first step into practice, learning about the specialized field in Artificial Intelligence, our knowledge is still limited and there are still many surprises. Therefore, errors in the thesis cannot be avoided. We look forward to receiving your valuable comments and suggestions to further improve the thesis.

Once again, thank you very much.

TABLE OF CONTENTS

LIST OF FIGURES, TABLES

LIST OF FIGURES

Figures of chapter 4:

LIST OF TABLES

LIST OF ABBREVIATIONS

MTCNN	Multi-Task Cascaded Convolution Neural Network
DCNN	Deep Convolutional Neural Network
SVM	Support Vector Machine
ID	Identity Document
API	Application Programming Interface
SDK	Software Development Kit
CNN	Convolutional Neural Network
Conv	Convolution
DNN	Deep Neural Network
SRC	Sparse Representation-based Classification
CASIA	Chinese Academy of Sciences
LFW	Labelled Faces in the Wild
YTF	YouTube Faces

CCTV	Closed Circuit Television
IT	Information Technology
RNN	Recurrent Neural Network
ANN	Artificial Neural Network
NLP	Natural Language Processing
FC	Fully Connected layer
RGB	Red Green Blue
NMS	Non-maximum Suppression
TP	True positive
FP	False positive
TN	True negative
FN	False negative
RF	Random forest
KNN	K-Nearest Neighbor

NB	Naïve Bayes
NBC	Naïve Bayes Classification
CPU	Central Processing Unit
GB	Gigabyte
RAM	Random Access Memory
IDE	Integrated Development Environment
GHz	Gigahertz
ML	Machine Learning
OS	Operating System
MVT	Model View Template

ABSTRACT

Face recognition and verification are critical components of the security and investigation processes. A face recognition system can identify or authenticate a person based on a digital photograph or video frame. Facial recognition is utilized in real-time applications such as attendance systems, unlocking mobile phones, tagging people on social media, payments, advertising, and illness diagnosis, among others. The main criteria for real-time applications that use facial recognition are a high recognition rate and a short training period. FaceNet [34] and Cosine Distance are used in the proposed face recognition model for feature extraction and comparison of face embeddings. The notion of transfer learning is utilized to shorten training time while increasing recognition rate. The 5-point landmarks on face frames are extracted using the Multi-Task Cascaded Convolution Neural Network (MTCNN) [2] model, then aligned and transmitted to FaceNet [34] to extract the embedding, which is then compared using the Cosine Distance.

Keywords: Face Verification; CNN; Cosine Distance; FaceNet; MTCNN;

CONTENT

CHAPTER 1: INTRODUCTION

1.1. Actual demand

Face recognition can be classified into two categories: face classification and face verification. Face classification involves classifying a person’s face into the correct face ID, while face verification involves determining whether two face images are of the same person. We can think of face classification as the closed set problem in which the face ID is known in advance from the model, and face verification is the open-set problem where the model may encounter a new face identity.

Facial recognition has become one of the most controversial technologies of recent times [39]. Tech giants like IBM, Microsoft, Amazon, Google and others have done extensive research on this to help improve many mainstream applications, improve security, help organizations stay disconnected in the event of a pandemic, etc.

For the 4.0-day social life, facial recognition contributes a lot to research projects as well as businesses’ activities such as employee face authentication. In the future, facial recognition will become an essential part of businesses. Facial recognition will become common for password authentication features of mobile phones, bank accounts and activities that require the security of applications or businesses. The facial recognition system will certainly develop further. It will also be applied in many different industries.

In this project, we will implement the concept of face verification using integrated neural network to design a system that allows users to compare images and check if the images are correct same person. We can also see the difference between the original image and the modified image.

1.2. Face verification software

Today's facial recognition applications are large-scale, putting them on personal computers will facilitate in many ways, prefer automatic login, use as biological science for convenient access more, identity verification etc. Some organizations now want to include identity verification on personal computers, the most effective facial recognition software on the market today such as Clarifai [45]. Clarifai is being utilized in diverse sectors, together with hospitality, retail, media, etc. The image, video and textual content popularity answers are constructed at the gadget getting to know platform and are made available through API, tool SDK and on-premise. DeepFace [44] application is also one of the famous applications, developed through a team of researchers at Facebook, DeepFace is a gentle face rating and popularity framework that uses an absolutely massive dataset of classified faces, type to get the correct facial expression, illustrate the desired preferred face for different data sets. According to the researchers, this device has greatly reduced the space left in celebrity fashion to the fullest extent of unrestricted facial popularity and is now ready for child-level precision, people. And there are other famous facial recognition software like DeepVision, FaceFirst, Face++, OpenFaceTracker, Paravision, Rohos Face Logon, Trueface.

1.3. Applications and challenging of the problem

A facial recognition system, also known as face recognition, is a type of application capable of recognizing an image of a person's face based on the characteristics that the database has previously stored. This system is well developed thanks to the integration of artificial intelligence technology. Helps increase recognition and high accuracy. Unlike fingerprints, faces have many similarities, so identifying features is difficult. For example, some people have similar faces. If the system is down, similarly-formed strangers can still access our content. Currently, twins using facial recognition technology can still access normally, even if the other has never installed the information in the database system.

Difficulties can be mentioned as for a person’s face, in childhood facial features can vary from adult face or in old age so facial verification can predict guess wrong or give incorrect results. The changes in the surrounding factors also make the facial features different when validating the face at that time. Current facial recognition solutions are applied in many different industries and fields. Below will be a list of 12 industries and fields of highly effective facial recognition technology application:

- Smart parking system management.
- Timekeeping in the company.
- Management of students, students.
- Automated payment.
- Unlock electronic devices.
- Intelligent advertising system.
- Searching for people who lost their lives, stolen…
- Support investigation.
- Diagnosis of diseases in medicine.
- Verify your identity.
- Control movement, in and out.
- Support for Anti-Theft Devices

In the future, many other industries will apply this technology to improve work efficiency. From there, gradually came to replace people in certain jobs.

1.4. Approaches

1.4.1. Related works research projects

Face recognition performance is approaching that of humans, thanks to the use of big data and deep convolutional neural networks (CNN) [16]. Several groups achieve very high performance on LFW using private large scale training datasets, i.e., 97 percent to 99 percent. While there are numerous open source CNN implementations, no large scale face dataset is publicly available. In the field of face recognition, data is currently more important than algorithms. To address this issue, Learning Face Representation from Scratch (Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z. Li, 2014) [43] of the Center for Biometrics and Security Research & National Laboratory of Pattern Recognition Institute of Automation, Chinese Academy of Sciences (CASIA) proposes a semi-automatical method for collecting face images from the Internet and creating the CASIAWebFace dataset, which contains approximately 10,000 subjects and 500,000 images. They use a database-based 11-layer CNN to learn discriminative representation and achieve state-of-the-art accuracy on LFW and YTF. CASIAWebFace's publication will attract more research groups to this field, accelerating the development of face recognition in the wild.

Recent research has shown that deep learning approaches achieve outstanding results on the face detection task. On the other hand, the advancements created a new problem related to the security of deep convolutional neural network models, revealing potential risks of DCNNs-based applications. Even minor changes in the digital domain can cause the network to be deceived. It was demonstrated at the time that some deep learning-based face detectors are vulnerable to adversarial attacks not only in the digital domain but also in the real world. In Real-world Attack on MTCNN Face Detection System (Edgar Kaziakhmedov, Klim Kireev, Grigorii Melnikov, Mikhail Pautov, Aleksandr Petiushko; 2019) paper [1], they investigate the security of the well-known cascade CNN face detection system - MTCNN and present an easily reproducible and robust attack method. They propose various face attributes that can be printed on a regular white and black printer and attached to either the medical face mask or the face directly. In a real-world scenario, their approach is capable of breaking the MTCNN detector. There are related studies such as Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks[2], Research on face detection technology based on MTCNN[38], Face recognition based on MTCNN and Convolutional Neural Network[40], Eyes localization algorithm based on prior MTCNN face detection [41], MTCNN and FaceNet based access control system for face detection and recognition[42].

1.4.2. Proposed Approach

Face Verification is one of the most difficult problems that researchers in Machine Learning - Deep Learning have had to deal with. This problem can be applied in a variety of fields, particularly those requiring high accuracy and security, such as eKYC in E-Commerce and identity recognition via surveillance cameras (CCTV). This problem will be divided into two parts: face detection and face verification. We propose MTCNN (Multi-task Cascaded Convolutional Networks) [2] and FaceNet [34] approaches in the research process to go deeper into the process of fully understanding and applying a Face Verification article.

CHAPTER 2: THEORETICAL FOUNDATIONS

2.1. Convolutional Neural Network (CNN)

In recent years, the IT industry has seen a surge in demand for a specific skill set known as Deep Learning. Deep Learning is a subset of Machine Learning that includes algorithms inspired by the operation of the human brain or neural networks.

These structures are known as Neural Networks. It teaches the computer to do things that humans do naturally. There are several types of deep learning models, including Artificial Neural Networks (ANN) [31], Autoencoders, Recurrent Neural Networks (RNN) [30], and Reinforcement Learning. Consequently, one model in particular has made significant contributions to computer vision and image analysis, and that is the Convolutional Neural Networks (CNN) [16] or ConvNets.

The CNN is one of the most often used deep neural networks (DNN) [29]. Convolution is a mathematical linear action between matrices. Convolutional layer, non-linearity layer, pooling layer, and fully connected layer are some of the layers of CNN. Pooling and non-linearity layers do not have parameters, whereas convolutional and fully connected layers have. In machine learning issues, the CNN performs admirably. Particularly impressive were the applications that deal with picture data, such as the world's biggest image classification data collection (Image Net), computer vision, and natural language processing (NLP) [32], with the results obtained.

2.1.1. Basic Architecture

- There are two main parts to a CNN architecture:
    - Feature extraction: A convolution tool that separates and identifies the distinct characteristics of an image for study in a process known as feature extraction.
    - Classification: A fully connected layer that uses the output of the convolution process to forecast the class of the image based on the features retrieved in earlier stages.

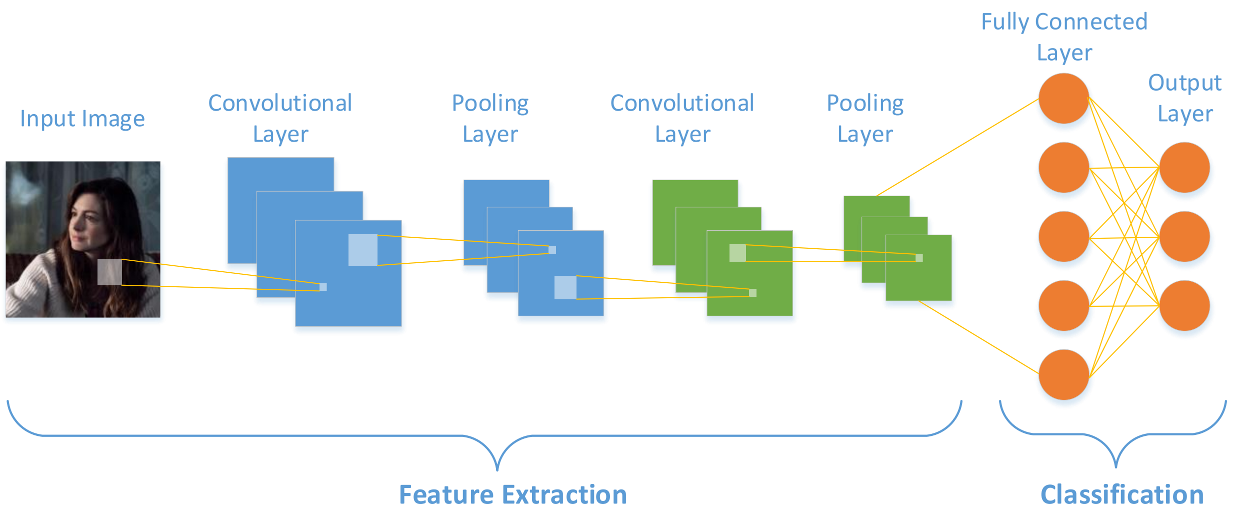


Figure 2. 1: The basic architecture of CNN [26]

2.1.2. Convolution layers

The CNN is made up of three different kinds of layers: convolutional, pooling, and fully linked layers. A CNN architecture will result from the stacking of these layers. There are two other crucial factors, the dropout layer and the activation function, which are described in Sections 2.1.2.4 and 2.1.2.5, in addition to these three layers.

2.1.2.1. Convolutional Layer

This is the first layer used to extract the different features from the input images. Convolution is performed between the input image and a filter of size MxM in this layer. The dot product between the filter and the parts of the input image with respect to the filter size is calculated by sliding the filter over the input image (MxM).

The output is known as the Feature map, and it contains information about the image such as the corners and edges. Later, this feature map is fed to other layers, which learn several other features of the input image.

2.1.2.2. Pooling layer

A Convolutional Layer is usually followed by a Pooling Layer. This layer's primary aim is to decrease the size of the convolved feature map in order to reduce computational costs. This is accomplished by reducing the connections between layers and operating independently on each feature map. There are various types of Pooling operations depending on the method used.

The largest element from the feature map is used in Max Pooling. Average Pooling computes the average of the elements in a predefined Image section size. Sum Pooling computes the total sum of the elements in the predefined section. The Pooling Layer is typically used to connect the Convolutional Layer and the Fully-Connected Layer.

2.1.2.3. Fully connected layer

The Fully Connected (FC) layer, which includes weights and biases as well as neurons, is used to connect neurons from different layers. These layers are typically placed prior to the output layer and constitute the final few layers of a CNN Architecture.

The previous layer's input image is flattened and fed to this layer. The flattened vector is then subjected to a few more FC layers, where mathematical function operations are typically performed. The classification process starts to take price at this point.

2.1.2.4. Dropout

Normally, overfitting in the training dataset can result from all features being connected to the FC layer. When a particular model performs so well on training data that it has a negative effect on the model's performance when applied to new data, this is known as overfitting.

To solve this issue, a dropout layer is used, in which a small number of neurons are removed from the neural network during training, reducing the size of the model. 30% of the nodes in the neural network are randomly removed upon passing a dropout of 0.3.

2.1.2.5. Activation function

Finally, the activation function is a critical parameter in the CNN model. They are used to learn and approximate any type of continuous and complex relationship between network variables. In other words, it determines which model information should be fired forward and which should not at the network's end.

2.2. Multi-task Cascaded Convolutional Networks (MTCNN)

Face detection is an important function of a verification system in general and of a human face verification system in particular. In this chapter, we will present a method for face detection using MTCNN.

Multi-Task Cascaded Convolutional Neural Networks (MTCNN) is a neural network which detects faces and facial landmarks on images. It was published in 2016 by Zhang et al [2].

MTCNN is one of the most popular and accurate face detection tools available today. It has been upgraded from the old Viola-Jones detector [18] to provide a more robust and accurate alternative to the Viola-Jones detector. The assumptions of the Viola-Jones framework frequently fail in practice, but cleverly constructed neural networks can easily perform such tasks. P-Net, R-Net, and O-Net are three neural networks linked in a cascade.

MTCNN is extremely accurate and stable. It detects faces correctly even with varying sizes, lighting, and rotations. It's a little slower than the Viola-Jones detector, but not by much. Color information is also used because CNNs receive RGB images as input.

2.2.1. Architectures of P-Net, R-Net, and O-Net

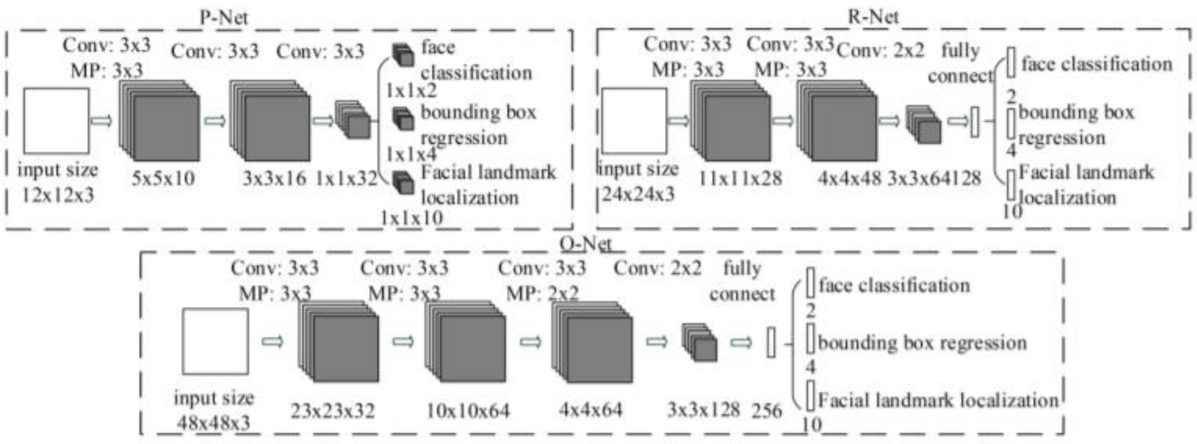


Figure 2. 2: The architecture of P-Net, R-Net, and O-Net [2]

2.2.1.1. P-Net

P-Net is a full convolutional neural network (FCN). Forward propagation produces a 32-dimensional feature vector at each place in the feature map. It's used to determine if any of the 12 x 12 grid cells have a face in them. If a grid cell contains a human face, the human face's Bounding Box is regressed, and the Bounding Box matching to the original image's area is retrieved further. A Non-maximum suppression (NMS) step and all of the other Bounding Boxes keep the Bounding Box with the highest score.

2.2.1.2. R-Net

R-Net (Refine Network) performs the same steps as P-Net. However, the network also uses a method called "padding", which inserts zero-pixels into the missing parts of the bounding box if the bounding box exceeds the boundary of the image. All bounding boxes now is resized to 24x24, treated as a kernel and fed to the R-Net. The result is also the new coordinates of the remaining bounding boxes and is fed to the next network, O-Net.

2.2.1.3. O-Net

Finally, O-Net (Output Network), which also does the same thing as R-Net, resizing to 48x48. However, the output of the network is no longer just the coordinates of the boxes, but returns 3 values including: 4 coordinates of bounding box (out[0]), coordinates of 5 landmark points on the face, including 2 eyes, 1 nose, 2 sides of lips (out[1]) and confidence point of each box (out[2])).

2.2.2. Training phase

Face/non-face classification, bounding box regression, and facial landmark placement are the three core aspects of MTCNN feature training.

2.2.2.1. Face/non-face classification

$$L_i^{det} = -\left(y_i^{det}\log(p_i) + \left(1 - y_i^{det}\right)(1 - \log(p_i))\right), where\ y_i^{det} \in \{0, 1\} \qquad (1)$$

The expression above is a cross-entropy loss function for face classification, where p_i is the face's probability and y_i^{det} is the background's actual tag.

2.2.2.2. Bounding box regression

$$L_i^{box} = \left\| \hat{y}_i^{box} - y_i^{box} \right\|_2^2, where\ y_i^{box} \in \mathbb{R}^4 \tag{2}$$

The regression loss estimated using the Euclidean distance is shown in the equation above. Among these, $\hat{y}$ is predicted through the network, and y is the actual real background coordinates and the quad (upper left x, upper left y, long, wide).

2.2.2.3. Facial landmark location

$$L_i^{landmark} = \left\| \hat{y}_i^{landmark} - y_i^{landmark} \right\|_2^2, where\ y_i^{landmark} \in \mathbb{R}^{10} \tag{3}$$

The Euclidean distance between the projected landmark position and the actual real landmark is computed and reduced, same like in boundary regression. The network predicts $\hat{y}$ and y is the actual real-world landmark coordinates. y belongs to the ten-tuple since there are a total of five points and two coordinates for each point.

2.3. Feature extraction using FaceNet

2.3.1. Architectures of FaceNet

FaceNet [34] was created by Google researchers who used a Deep Convolutional Neural Network (DCNN) [32] to map photos of a person's face into Euclidean spaces (groups of geometrical points), also known as embedding. The amount of similarity and differences in faces determines embedding, thus if the face is similar, the value will be closer, and if the face is different, the value will be further.

In general, feature extraction using the FaceNet model, as shown in Figure 2. 3, involves feeding input images into a deep learning architecture, which is then normalized L2, yielding facial features (embedding) that are trained using Triplet Loss.

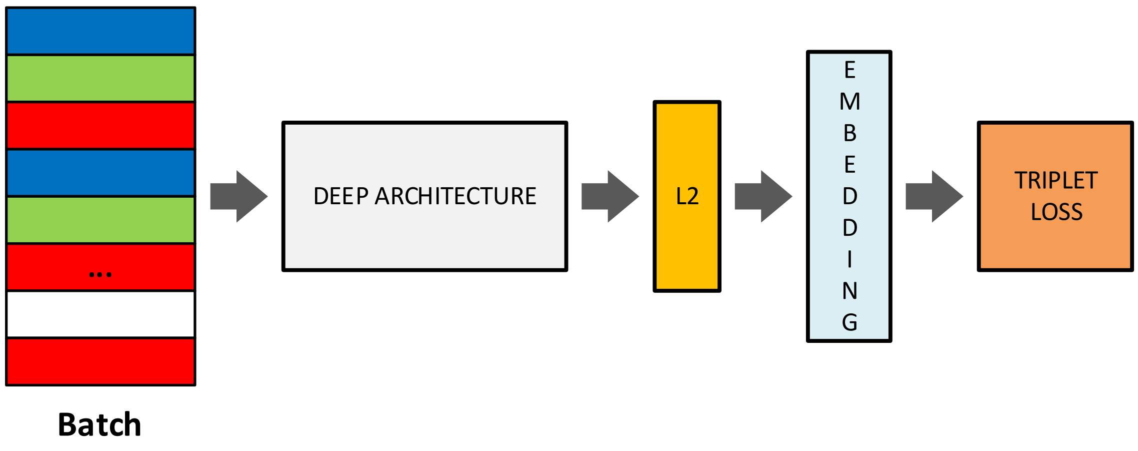


Figure 2. 3: FaceNet's model structure [3]

Embedding Vector: A vector with a fixed dimension (usually smaller in dimension than normal Feature Vectors), learned during training and representing a set of features responsible for classifying objects in the transformed dimension. Embedding is very useful in finding the Nearest Neighbors in a given Cluster, based on the distance-relationship between the embeddings.

Inception V1: A CNN network topology introduced in 2014 by Google, featured in Inception blocks. This block allows the network to be learned in a parallel structure, meaning that with one input, many different Convolution layers can be fed to give different results, which will then be Concatenate into an output. This parallel learning helps the network to learn more details and get more features than the traditional CNN network. In addition, the network also applies 1x1 Convolution blocks to reduce the size of the network, making training faster.

Triplet Loss: Instead of using traditional loss functions, when we only compare the output value of the network with the actual ground truth of the data, Triplet Loss offers a new formula that includes 3 input values including x_i^a (anchor) is the output image of the network, x_i^p (positive) is an image of the same person with the anchor, and negative x_i^n (negative) is an image that is not the same person with the anchor. Figure 2. 3 illustrates this.

$$\left\|f(x_i^a) - f(x_i^p)\right\|_2^2 + \alpha \ < \ \|f(x_i^a) - f(x_i^n)\|_2^2 \tag{4}$$

$$\forall \ (f(x_i^a), f(x_i^p), f(x_i^n)) \in T. \tag{5}$$

Where α is a margin between positive and negative pairs, the minimum required deviation between two ranges of values $f(x_i^a)$ is the embedding of x_i^a. The above formula shows that the desired distance between 2 embeddings are $f(x_i^a)\ and\ f(x_i^p)$ have to be at least α value less than the pair $f(x_i^a)v$à $f(x_i^n)$. What we have to do here is to make the difference between the two sides of the formula as large as possible, in other words, $\left\|f(x_i^a) - f(x_i^p)\right\|_2^2$ must be minimum and $\|f(x_i^a) - f(x_i^n)\|_2^2$ must be maximum. Moreover, in order for the network to "harder" to learn (or in other words, learn more), the selected positive point must be as far as possible from the anchor, and the selected negative point must be as close as possible to the anchor, to make the network "encounter" the worst cases. The general Loss function stated in the paper is the following formula:

$$\sum_i^N \left[\left\|f(x_i^a) - f(x_i^p)\right\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha\right]_+ \tag{6}$$

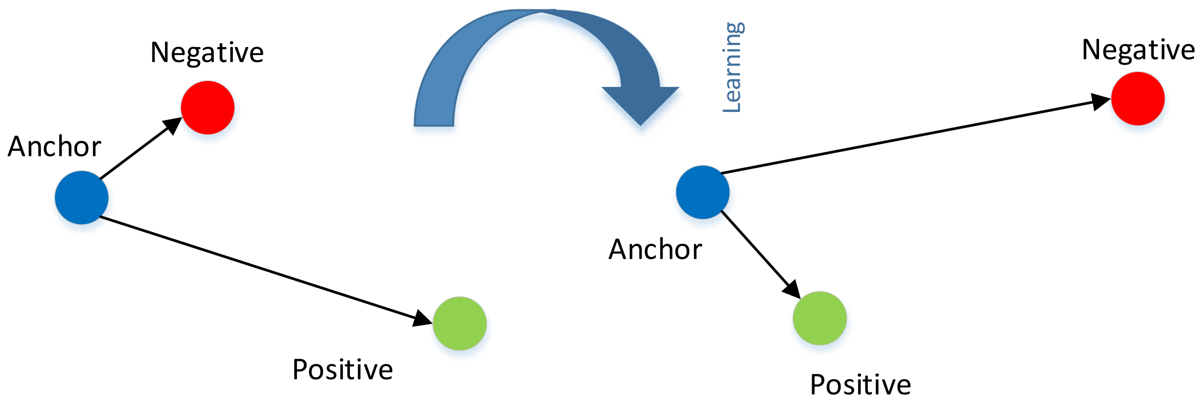


Figure 2. 4: Triplet loss [3]

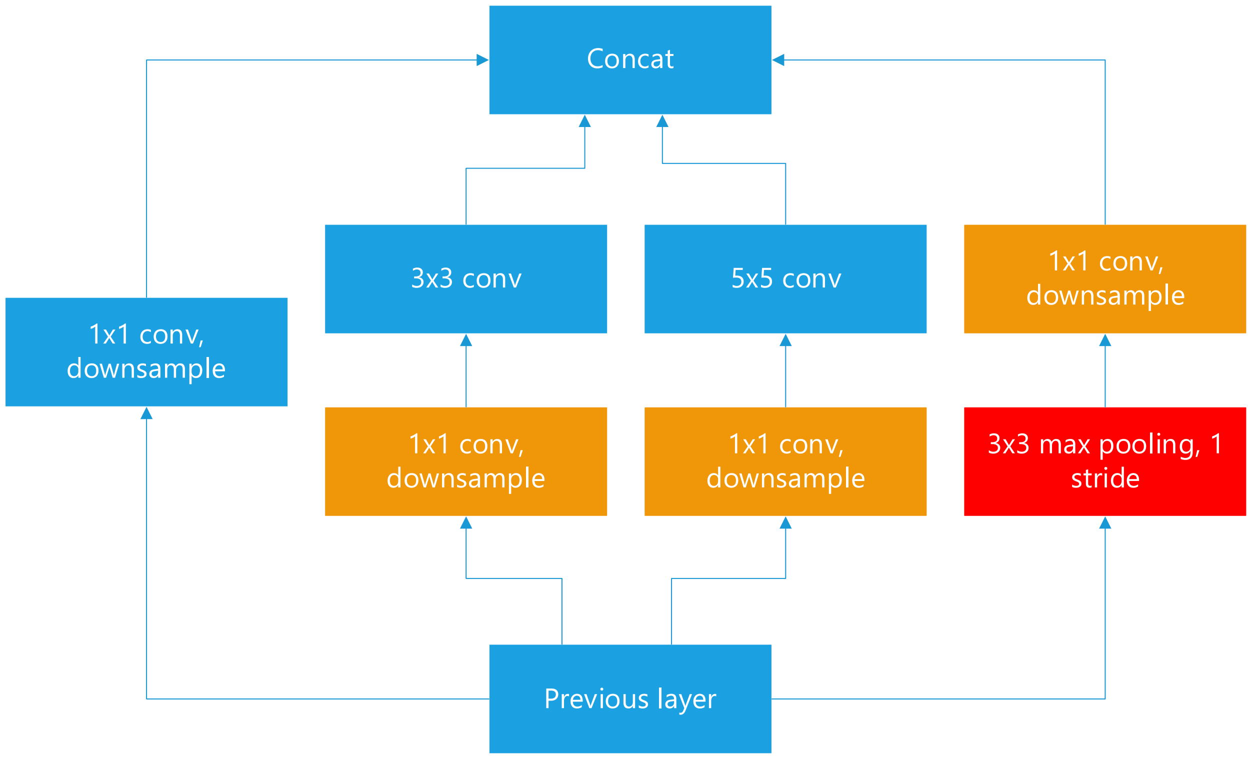


Figure 2. 5: Inception V1 [46]

2.3.2. Training phase

Use a Dataset with many different individuals, each with a certain number of images. Build a DNN network to use as Feature Extractor for the Dataset, the result is a 128-Dimensions embedding.

Training the DNN network so that the embedding result has good recognizability, includes 2 things: using L2 normalization (Euclidean distance) for the output embeddings and re-optimizing the parameters in the network by Triplet Loss.

2.4. Cosine distance

$$\hat{x} = arg \min_{x} \left(1 - \frac{y^T Dx}{\|y\|_2 \|Dx\|_2} \right) + \gamma \|x\|_p \tag{7}$$

where $p \in \{0, 1\}$. Due to the $\|Dx\|$ term in the denominator. The simple formulation of cosine distance based sparse coding (7) is not easy to solve. To simplify the expression, we use the fact that the cosine distance is identical to the square of the Euclidean distance if the norms of both y and Dx are 1, as demonstrated by the simple theorem [4]:

Theorem 1. *If the norms of two vectors are 1, the square of the Euclidean distance is proportional to the cosine distance (with a factor of 2). [5]*

Proof: Assume the original signal is y and the reconstructed signal against dictionary D is Dx with x representing the sparse code. If both the norms of y and Dx are 1, the square of the Euclidean distance between the two vectors is proportional to the cosine distance:

$$\begin{aligned} \|y - Dx\|_2^2 &= y^T y + (Dx)^T (Dx) - 2y^T Dx \\ &= 1 + 1 - 2\frac{y^T Dx}{1} = 2 - 2\frac{y^T Dx}{\|y\|_2 \cdot \|Dx\|_2} \\ &= 2 \cdot \big(1 - cosine\ similarity(y, Dx)\big) \end{aligned} \tag{8}$$

$$= 2 \cdot cosine\ distance(y, Dx)$$

Based on the analysis, we formulate an approximate cosine distance based sparse coding objective function by normalizing the original signal y by its $L2-norm$ and adding a term to the Euclidean based sparse coding objective function that forces the norm of the reconstructed signal Dx to be 1. This may be written as follows:

$$\min_{x} \|\hat{y} - Dx\|_2^2 + \alpha|1 - \|Dx\|_2^2| + \gamma\|x\|_1,$$

$$s.t. 0 < \alpha < 1, \tag{9}$$

$$\|\hat{y}\|^2 = 1,$$

where $\hat{y}$ is y normalized by its $l_2 - norm$ and α and γ are the hyper-parameters to balance the term forcing $\|Dx\|_2^2$ to be close to 1 and l_1 regularizer, respectively.

When the size of the signal itself contains crucial information, normalizing the signal y may degrade classification accuracy in the SRC framework [6]. (9), on the other hand, demands y to be normalized, which necessitates the loss of such information. As a result, we expand Theorem 1 to establish another equivalence between Euclidean and cosine distances without the need to normalize y, as follows:

Theorem 2. *If the norms of two vectors are both n, then the square of the Euclidean distance is proportional to their cosine distance (with a factor of* $2n^2$*).[5]*

Proof: The proof follows the proof of Theorem 1 with $\|y\|^2 = \|Dx\|^2 = n^2$.

We can design an objective function for approximate cosine distance based sparse coding without normalizing y by restricting the norm of the reconstructed signal Dx to be near to the norm of y. The formulation can be rewritten as follows:

$$\min_{x} \|y - Dx\|_2^2 + \alpha \left| \|y\|_2^2 - \|Dx\|_2^2 \right| + \gamma \|x\|_1,$$

$$s.t. 0 < \alpha < 1, \tag{10}$$

2.5. Related Machine Learning Models

2.5.1. Support Vector Machine (SVM)

When dealing with normalised face embedding inputs, it is usual practice to employ Linear SVM [13] for classification. SVM is a method that may be used for classification as well as regression. In classification, SVM takes data points or face embeddings as inputs and generates a hyperplane that separates independent features with the greatest margins possible using a boundary decision. The boundary decision in the case of a Linear SVM is a line. The following formula can be used to determine the forecast for a new input:

$$f(x) = b_0 + \sum \left(a_i * (x, x_i) \right) \tag{11}$$

Where, the input face embedding vector is denoted by x, x_i is the support vector, the coefficients b_0 and a_i are for each input generated during training.

The SVM model may then be used to identify various pictures on the validation set, and the percentage of accuracy can be calculated.

Performance measurement: The formulas in Eqs. (12), (13), (14) and (15) are used to calculate performance measurement characteristics such as sensitivity, specificity, accuracy and f1-score. The comparison is carried out by generating a confusion matrix, which calculates the total number of true positives, true negatives, false positives, and false negatives.

$$Sensitivity = \frac{TP}{TP + FN} \tag{12}$$

$$Specificity = \frac{TN}{TN + FP} \tag{13}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{14}$$

$$f1 - score = \frac{2 * Sensitivity * Specificity}{Sensitivity + Specificity} \tag{15}$$

where,

True positive (TP) = Correctly identified.

False positive (FP) = Incorrectly identified.

True negative (TN) = Correctly rejected.

False negative (FN) = Incorrectly rejected.

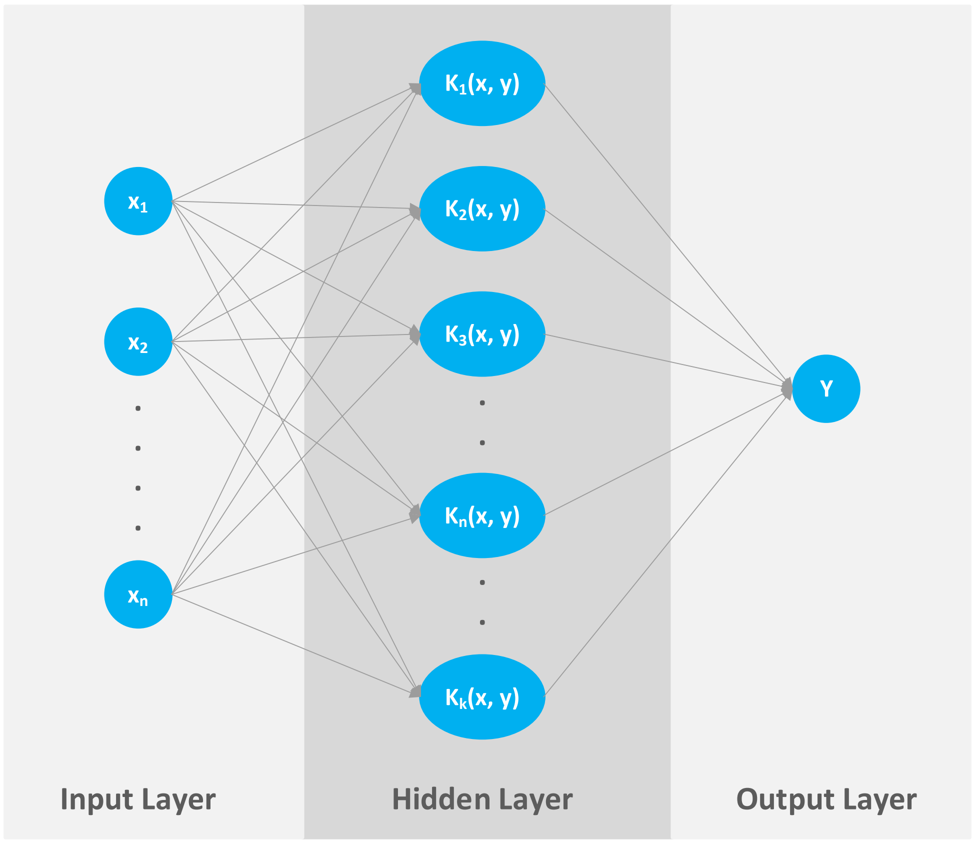


Figure 2. 6: Architecture of the SVM Classifier [47]

2.5.2. Random forest (RF)

A tree-based classifier is called the Random Forest Classifier [35]. It is made up of a set of tree classifiers, each of which is constructed using a random vector sampled independently from the input vector, and each tree casts a unit vote for the most common class in order to classify an input vector [19]. To create a tree, the random forest classifier utilized in this work uses randomly selected characteristics or a mixture of features at each node. For each feature/feature combination chosen, bagging, a method for generating a training dataset by randomly selecting with replacement N samples, where N is the size of the original training set [20], was employed. The most popular voted class from all the tree predictors in the forest is used to classify any pixels [19]. A decision tree's design necessitated the selection of an attribute selection measure as well as a pruning procedure. There are several techniques to choose attributes for decision tree induction, and most of them assign a quality measure to the attribute directly. The Information Gain Ratio criteria [22] and the Gini Index [21] are the most often utilized attribute selection metrics in decision tree induction. The Gini Index is used by the random forest classifier as an attribute selection measure, which quantifies the impurity of an attribute in relation to the classes. Choosing one instance (pixel) at random from a given training set T and declaring that it belongs to some class C_i, the Gini index is denoted as:

$$\sum\sum_{j\neq i}\left(\frac{f(C_i,T)}{|T|}\right)\left(\frac{f\left(C_j,T\right)}{|T|}\right) \tag{16}$$

Where $\frac{f(C_i,T)}{|T|}$ is the probability that selected instance belongs to class C_i.

2.5.3. K-Nearest Neighbor (KNN)

K-Nearest Neighbor [36] is a simple Machine Learning algorithm that uses Supervised Learning method. The K-NN algorithm assumes similarity between the new case/data and existing cases and places the new case in the category that is most similar to the existing categories. The K-NN algorithm stores all available data and uses similarity to classify new data points. This means that when new data is generated, it can be quickly classified into a well-suited category using the K- NN algorithm. The K-NN algorithm can be used for both regression and classification, but we will only use it for classification in our research.

The K-NN rule classifies each unlabeled example in the training set based on the majority label among its k-nearest neighbors. As a result, its performance is highly dependent on the distance metric used to identify nearest neighbors. Most K-NN classifiers utilize a simple Euclidean metric to quantify the dissimilarities between instances expressed as vector inputs in the absence of prior knowledge. The formula for Euclidean distance is as follows.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2} \qquad (17)$$

Where we define an example as a vector x = (a_1, a_2, a_3,…, a_n), n is the dimensionality of the vector input, namely, the number of an example's attribute. *x* is the coordinate of the first point and *y* is the coordinate of the second point, the smaller $d(x,y)$ is the two examples are more similar [38].

A test examples class label is determined by the majority vote of its k nearest neighbors.

$$y(d_i) = arg\max_k \textstyle\sum_{x_j \in kNN} y(x_j, c_k) \qquad (18)$$

Where d_i is a test example, x_j is one of its *k* nearest neighbors in the training set, $y(x_j, c_k)$ indicates that whether x_j belongs to class c_k. Equation (18) means that the prediction will be the class having most members in the *k* nearest neighbors.

2.5.4. Naïve Bayes (NB)

Here we will use another fairly easy to understand and highly accurate probability classification algorithm that also belongs to the group of Supervised Machine Learning Algorithms that is called Naïve Bayes Classification (NBC) [37]. This is an algorithm based on Bayes theorem of probability theory to make judgments as well as classify data based on observed and statistical data. NBC is one of the most widely applied algorithms in the field of Machine Learning used to make the most accurate predictions on a collected data set. Bayes theorem provides a way of calculating the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$, and $P(x|c)$. Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. Class conditional independence is the name given to this assumption.

$$\mathrm{P}(\boldsymbol{c} \mid \mathbf{x}) = \frac{\mathrm{P}(\boldsymbol{c})\mathrm{P}(\mathbf{x} \mid \boldsymbol{c})}{\mathrm{P}(\mathbf{x})} \tag{19}$$

$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c) \tag{20}$$

$P(x|c)$ denotes the posterior probability of class (target) given predictor, P(c) the prior probability of class, $P(x|c)$ the likelihood of predictor given class, and P(x) the prior probability of predictor. (20) is an implementation equation.

CHAPTER 3: PROPOSED APPROACH

In chapter 2, we presented the background knowledge about MTCNN, FaceNet and Cosine Distance. In the following, we will apply that knowledge to solve the human face verification problem. As mentioned earlier, our approach to solving the identification problem is based on feature matching. Using MTCNN, the algorithm detects faces in the frames. When having the output of MTCNN, the detected face image will be rotated until the both eye location become horizontal. FaceNet is used to construct embedding points for faces after face detection. Following the creation of the embedding points, the system compares the output and verifies it to the training dataset. If the matching satisfies the threshold, it displays the person's name and the proportion of the person's face that matches the person's face recorded in the database. If the result does not meet the threshold, it indicates that the individual is unknown. In addition, we feed the embeddings to related machine learning models such as SVM, RF, KNN, and NB for comparation between each model. The flowchart is demonstrated in figure 3. 1.

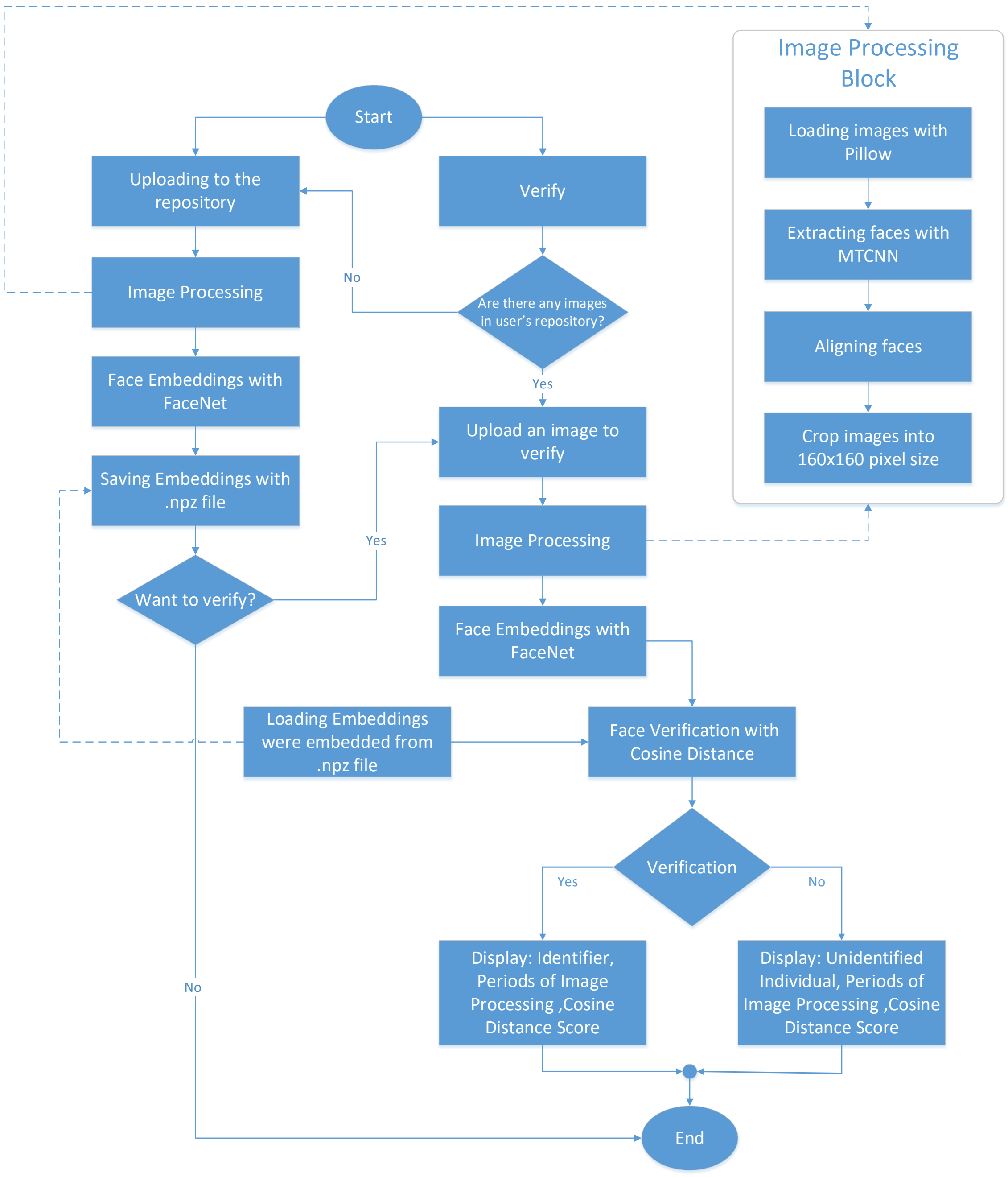


Image Processing Block
Loading images with Pillow
Extracting faces with MTCNN
Aligning faces
Crop images into 160x160 pixel size
Start
Uploading to the repository
Verify
Image Processing
No
Are there any images in user's repository?
Yes
Face Embeddings with FaceNet
Upload an image to verify
Saving Embeddings with .npz file
Image Processing
Yes
Want to verify?
Face Embeddings with FaceNet
Loading Embeddings were embedded from .npz file
Face Verification with Cosine Distance
Verification
Yes
No
Display: Identifier, Periods of Image Processing ,Cosine Distance Score
Display: Unidentified Individual, Periods of Image Processing ,Cosine Distance Score
No
End

Figure 3. 1: Verification process

3.1. Face detection using MTCNN

An image pyramid is created from an input picture by rescaling it into several scales using bi-linear interpolation. This step ensures that the scale is invariant. The MTCNN has three cascaded steps that follow the scaling step:

3.1.1. Step 1: P-Net

First of all, a photo will usually have more than one person (a face). In addition, the faces will often be of different sizes. We need a method to be able to recognize all those faces, of different sizes. MTCNN gives us a solution, using Image Resize, to create a series of copies from the original image with different sizes, from large to small, forming an Image Pyramid.

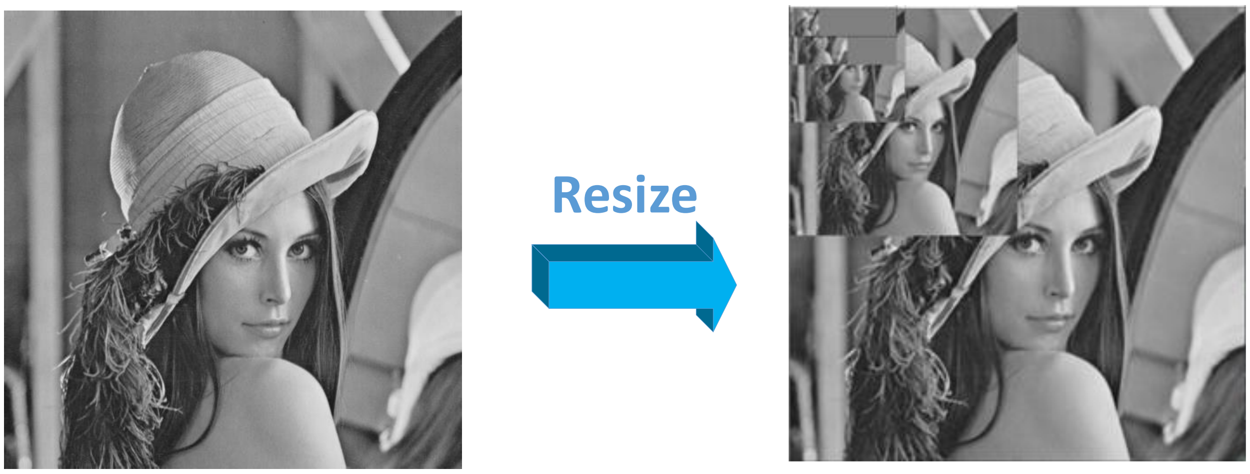


Figure 3. 2: Sampling process [2]

For each sampling version of the original image, we use a Window 12x12 pixel to go through the entire image, detecting faces. Since the copies of the original image are of different sizes, the network can easily recognize faces of different sizes, even though using only one window with a fixed size. After that, we will put the Windows that have been cut out and transmitted over the P-Net (Proposal Network). The candidate of the network is a series of bounding boxes located in each window, each bounding box will contain the coordinates of the four corners to determine the location in the kernel containing it (normalized to the range (0,1)) and the confidence score, respectively.

To eliminate bounding boxes on images and kernels, we use two main methods, the first is to set a confident threshold to remove the boxes with low confidence and the second is to use NMS (Non-Maximum Suppression) to remove the boxes that have the same probability that they pass a certain threshold.

After removing the unreasonable boxes, we convert the coordinates of the boxes to the original coordinates of the actual image. Because the coordinates of the box have been normalized to the interval (0,1) corresponding to the kernel , so now it's just a matter of calculating the length and width of the kernel based on the original image, then multiply the normalized coordinates of the box by the size of the kernel and add the coordinates of the corresponding kernel corners. The result of the above process is the coordinates of the corresponding box on the image with the original dimensions. Finally, we resize the boxes to square shape, get the new coordinates of the boxes and feed into the next network, R-Net.

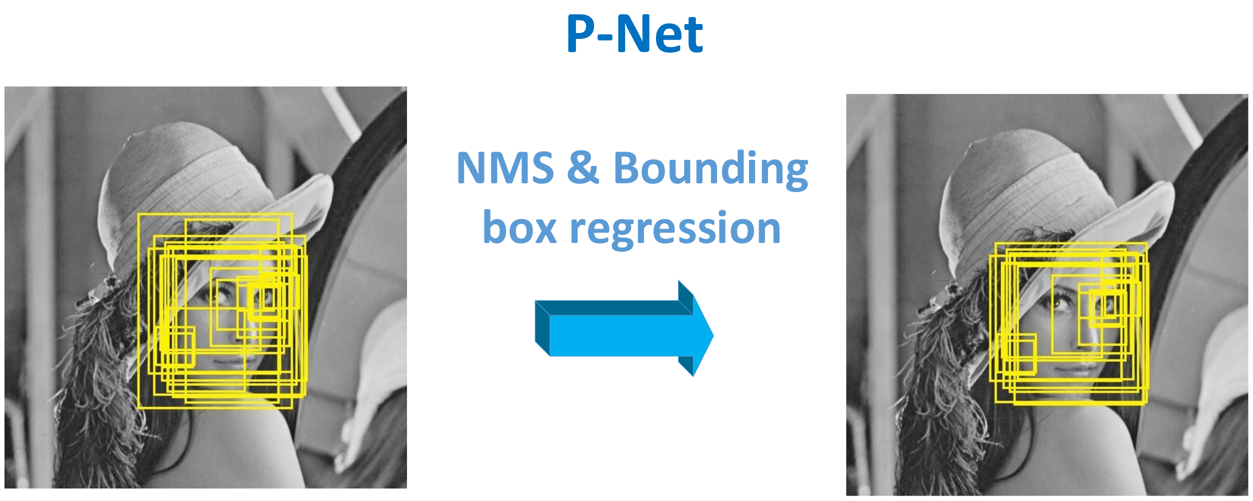


Figure 3. 3: Stage 1 of MTCNN [2]

3.1.2. Step 2: R-Net

R-Net (Refine Network) performs the same steps as P-Net. However, the network also uses a method called “padding”, which inserts zero-pixels into the missing parts of the bounding box if the bounding box exceeds the boundary of the image. All bounding boxes now is resized to 24x24, treated as a kernel and fed to the R-Net. The result is also the new coordinates of the remaining bounding boxes and is fed to the next network, O-Net.

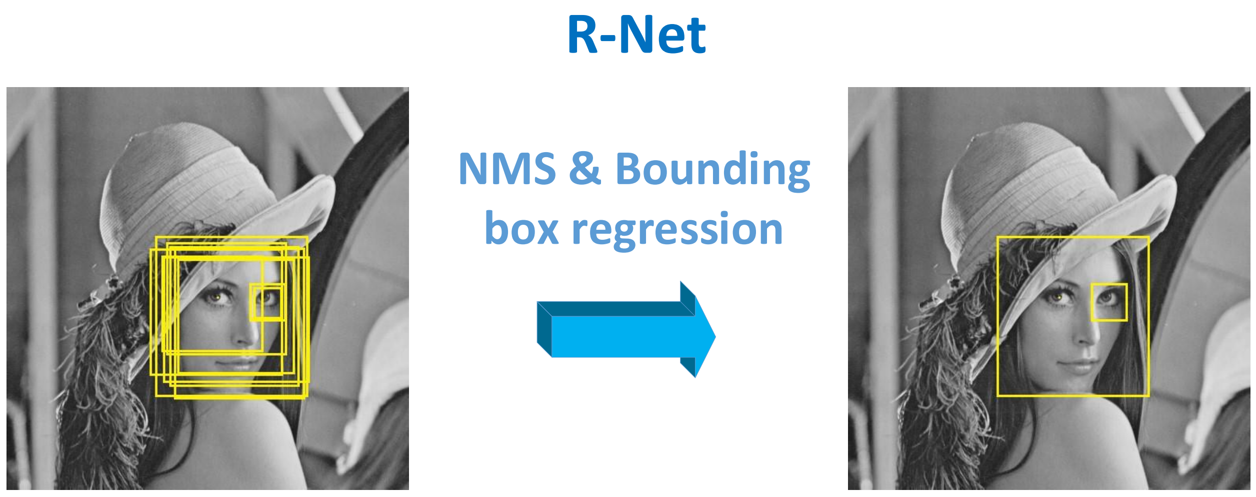


Figure 3. 4: Stage 2 of MTCNN [2]

3.1.3. Step 3: O-Net

Finally, O-Net (Output Network), which also does the same thing as R-Net, resizing to 48x48. However, the output of the network is no longer just the coordinates of the boxes, but returns 3 values including: 4 coordinates of bounding box (out[0]), coordinates of 5 landmark points on the face, including 2 eyes, 1 nose, 2 sides of lips (out[1]) and confidence point of each box (out[2])).

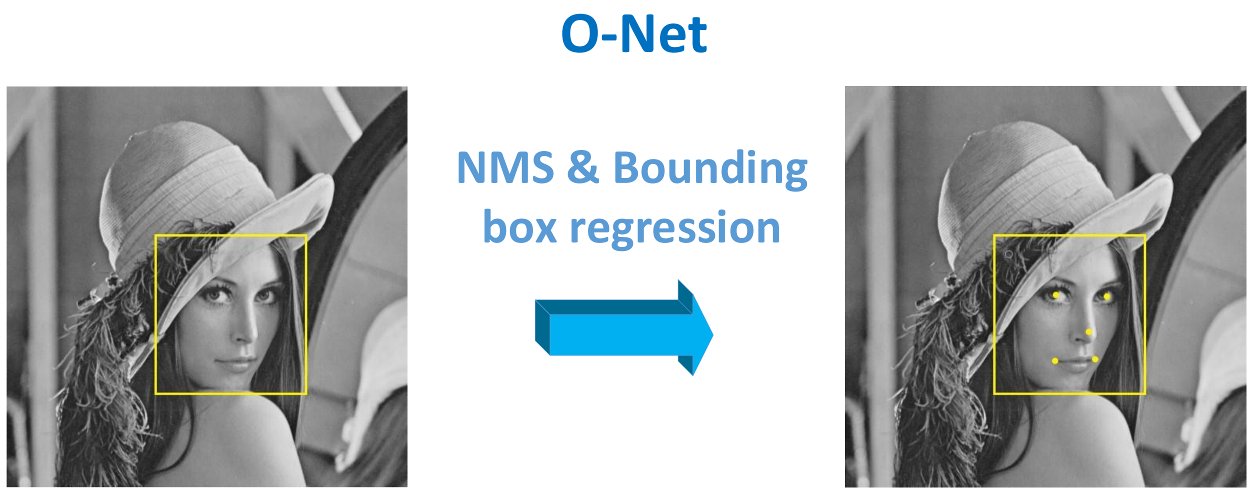


Figure 3. 5: Stage 3 of MTCNN [2]

3.2. Alignment

In order to increase the accuracy, we have one more step to process the MTCNN's output, that is Alignment. It means that when we have the output of MTCNN, we will rotate the detected face image until the both eye location become horizontal.

The coordinates are determined as follows:

- x, y: Top left point of the image is $(0, 0)$.
- Left-eye: (x_1, y_1)
- Right-eye: (x_2, y_2)

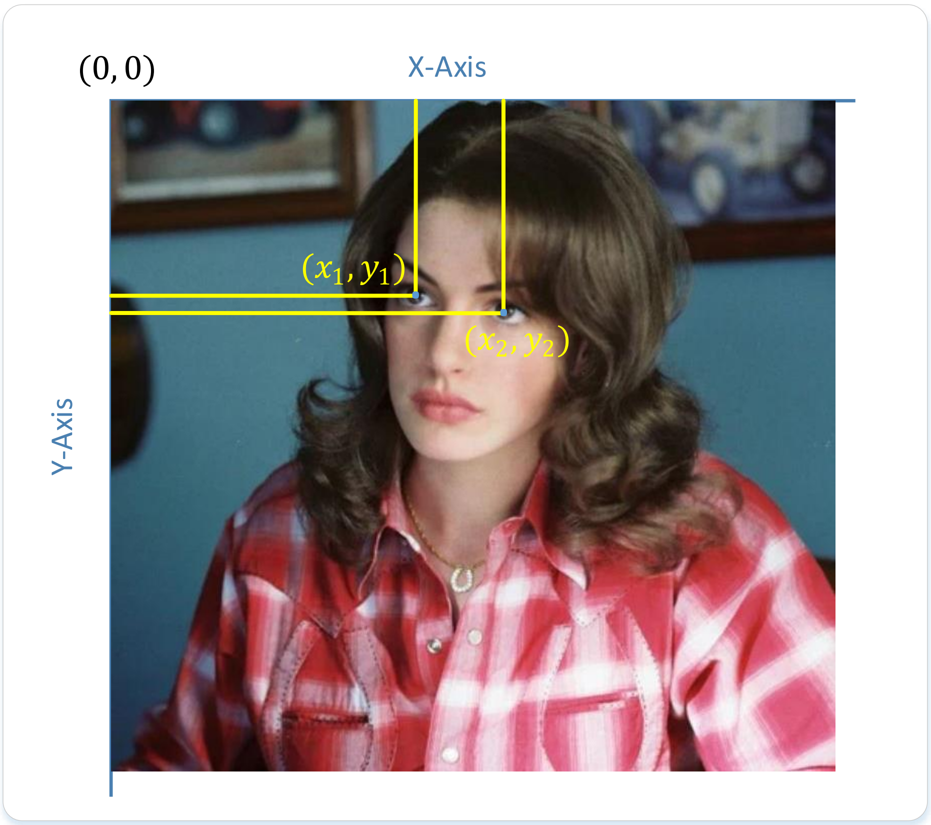


Figure 3. 6: Coordinates of both eyes on the image matrix

We need the angle formed by both eyes. Because the image will be rotated based on this angle. In the image above, the left eye is higher than the right eye. As a result, we will rotate the picture in the inverse direction of the clock. On the other hand, if the right eye was higher than the left, we would rotate the image clockwise.

3.2.1. Trigonometry

With Euclidean Distance, we can calculate the length of the triangle's 3 edges. Demonstrated in (figure 3. 7).

3.2.1.1. Pythagorean theorem

Let’s go through a little bit about the Pythagorean theorem. In mathematics, the Pythagorean theorem, or Pythagoras' theorem, is a fundamental relation in Euclidean geometry among the three sides of a right triangle. It states that the area of the square whose side is the hypotenuse (the side opposite the right angle) is equal to the sum of the areas of the squares on the other two sides. This theorem can be written as an equation relating the lengths of the legs a, b and the hypotenuse c, often called the Pythagorean equation:

$$c^2 = a^2 + b^2 \tag{21}$$

We've previously determined the triangle's edge lengths. As a result, we may also use the cosine rule. In order to calculate the angle from its cosine value, inverse trigonometric functions must be used. Cosine rule demonstrated as follow:

$$\cos\alpha = \frac{b^2 + c^2 - a^2}{2bc} \tag{22}$$

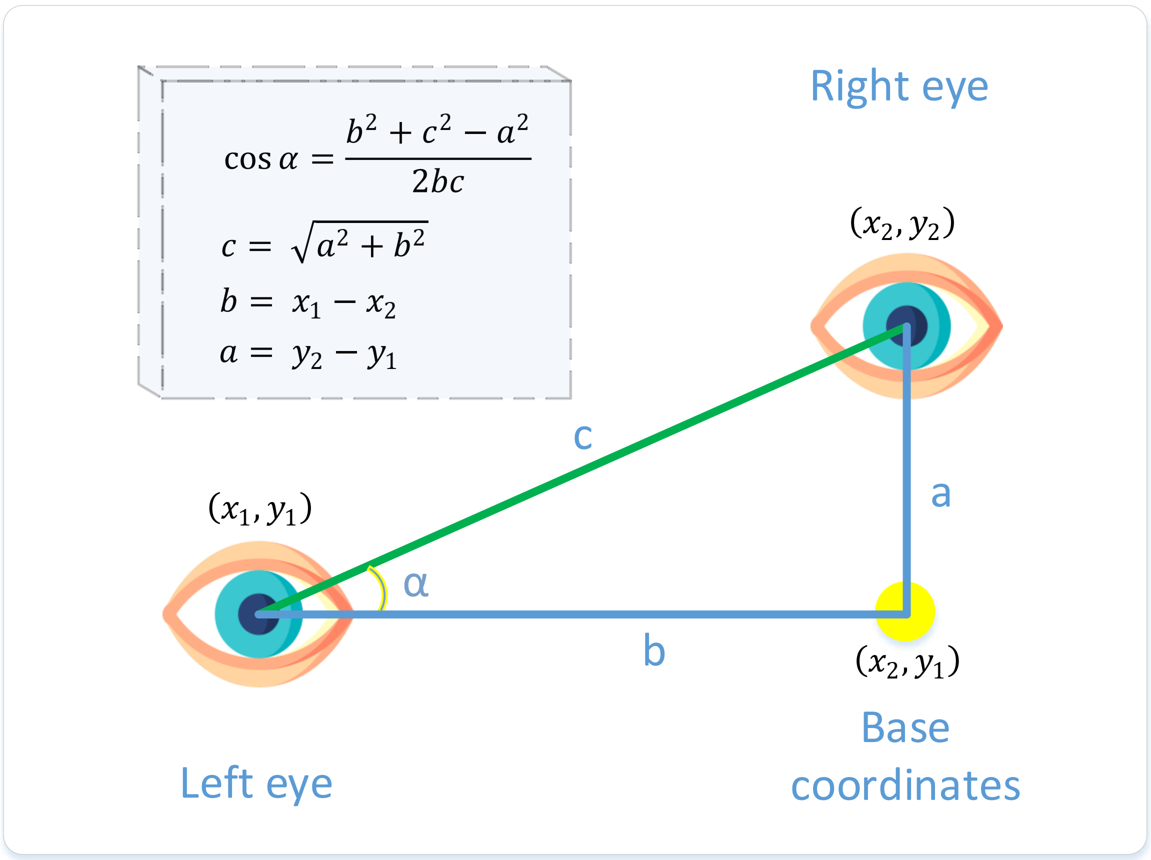


Figure 3. 7: Illustration of Cosine rule

3.3. Feature extraction using FaceNet

FaceNet generates the embedding vector from a face image as its input (a vector of 128 numbers which represent the most important features of a face). As vectors, embeddings can be understood as points in the Cartesian coordinate system. That implies that we can use the embeddings of a face image to plot it in the coordinate system. Calculating an image's embedding and comparing it to known-person images could be one method of identifying a person on an unseen one. If the face embedding is sufficiently close to the embeddings of person A, we can conclude that the image contains the face of person A. FaceNet was train with a large number of face images. We'll assume for the sake of simplicity that we only have a few images from two individuals. If we have thousands of images of various people, the same reasoning still holds true. FaceNet creates random vectors for each image at the start of training, so when the images are plotted, they are distributed randomly. FaceNet learns in the following manner using a method known as triplet loss. FaceNet works similarly to a hash function in that it maps images of the same person to (approximately) the same location in the coordinate system where embedding is the hash code.

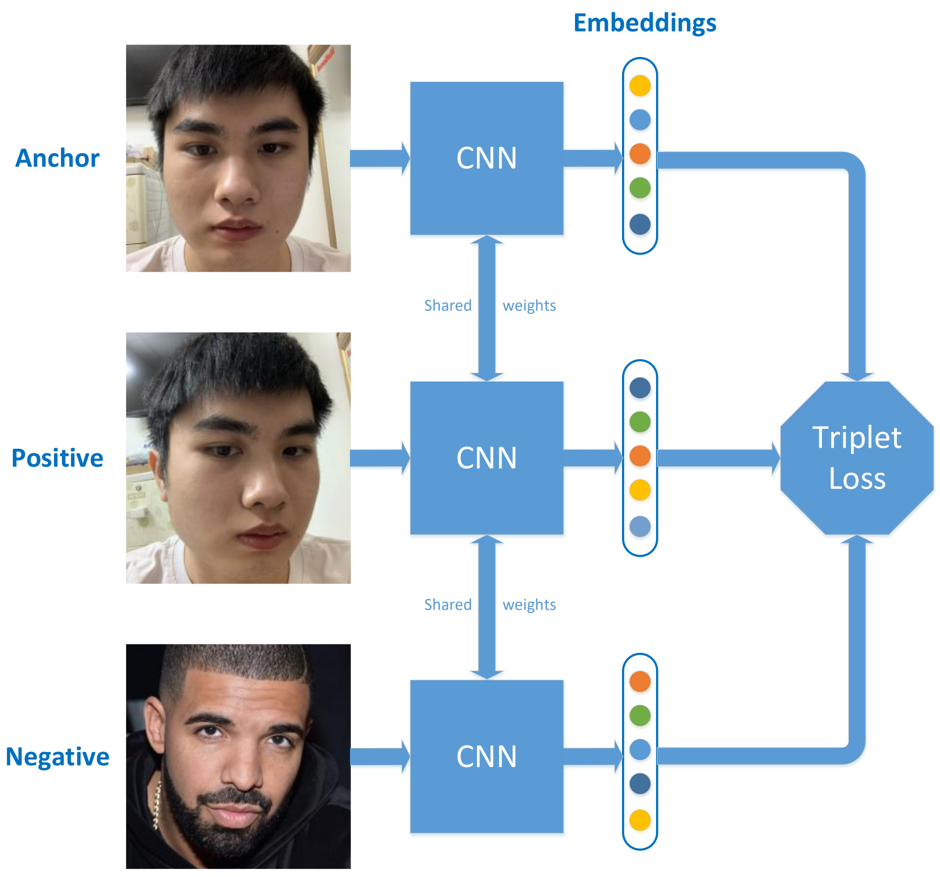


Figure 3. 8: Triplet loss: Image credits

3.4. Related Machine Learning Models

After we have the embeddings of the training and the testing dataset, we feed it into the following models below to verify the results.

3.4.1. Support Vector Machine (SVM)

In classification, SVM takes data points or face embeddings as inputs and builds a hyperplane that uses a boundary decision to separate independent features with the biggest margins feasible. The goal of SVM training is to generate an optimal hyperplane and categorize the data into distinct groups. To avoid error circumstances, this hyperplane is placed as far away from the data as practicable.

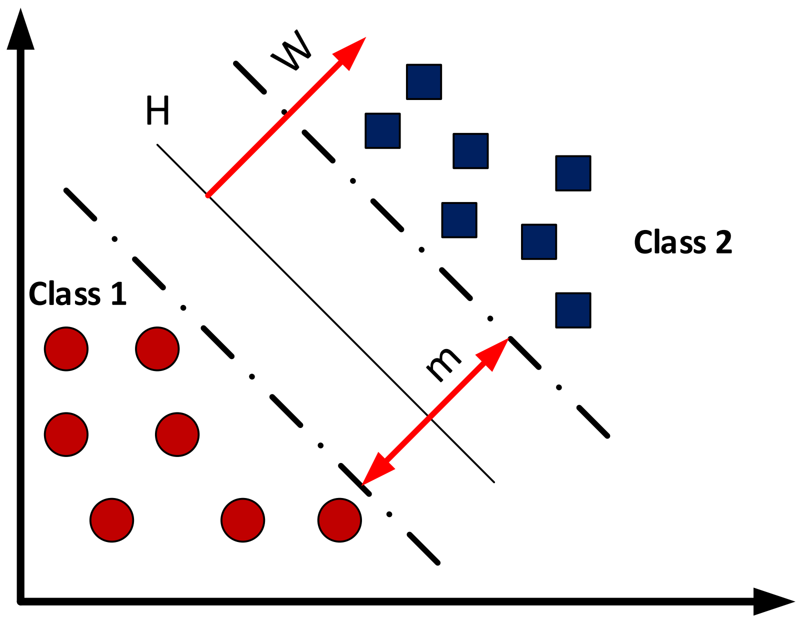


Figure 3. 9: Classification using Support Vector Machines (SVM) between labeled classes

3.4.2. Random Forest (RF)

Many is preferable to one. Simply said, this is the idea underlying the random forest algorithm. That is, several decision trees can yield more accurate predictions than a single decision tree. The random forest approach is a supervised classification algorithm that constructs N slightly distinct trained decision trees and combines them to provide more accurate and consistent predictions.

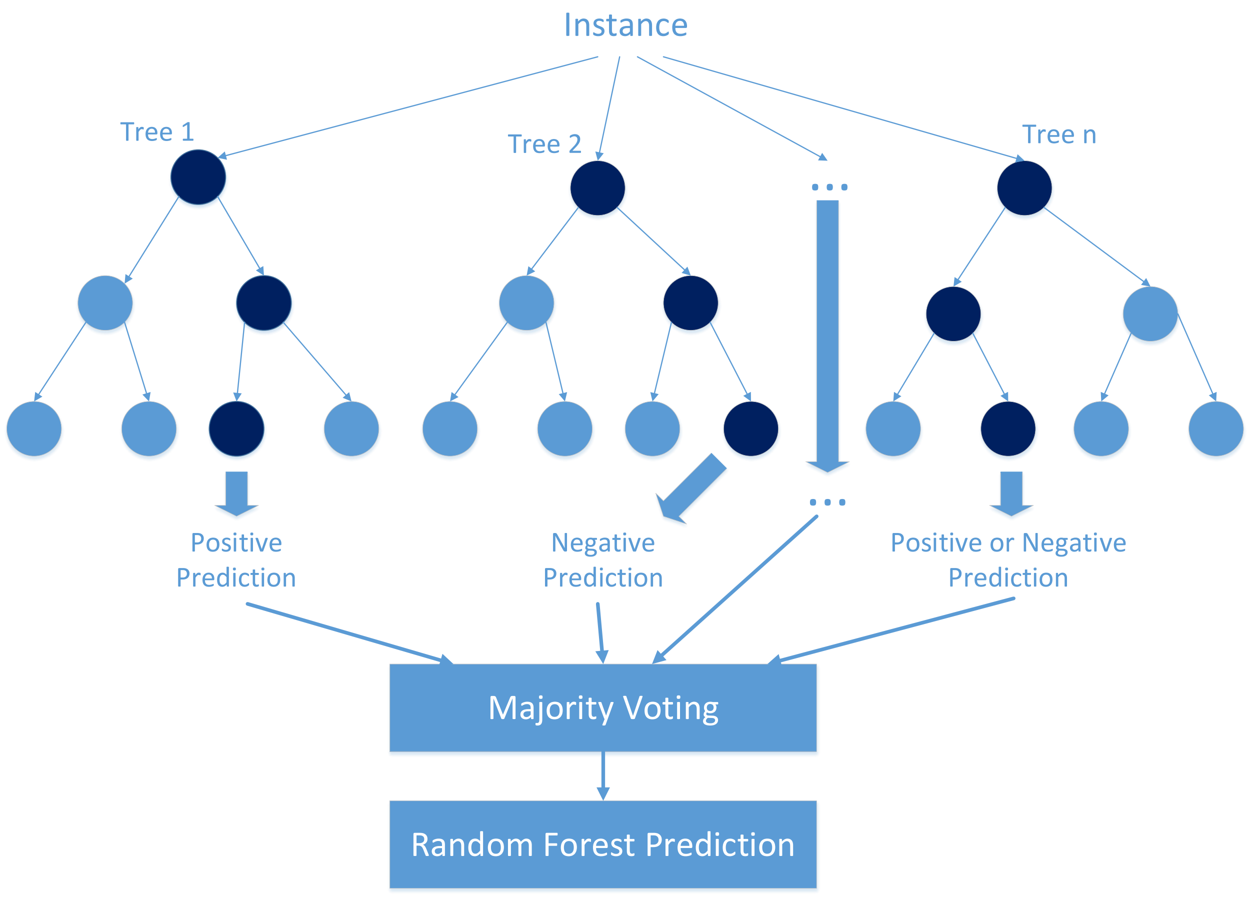


Figure 3. 10: Illustration of Random Forest [28]

3.4.3. K-Nearest Neighbor

The K-NN rule classifies each unlabeled example in the training set based on the majority label among its k-nearest neighbors. As a result, its performance is highly dependent on the distance metric used to identify nearest neighbors. Most K-NN classifiers utilize a simple Euclidean metric to quantify the dissimilarities between instances expressed as vector inputs in the absence of prior knowledge.

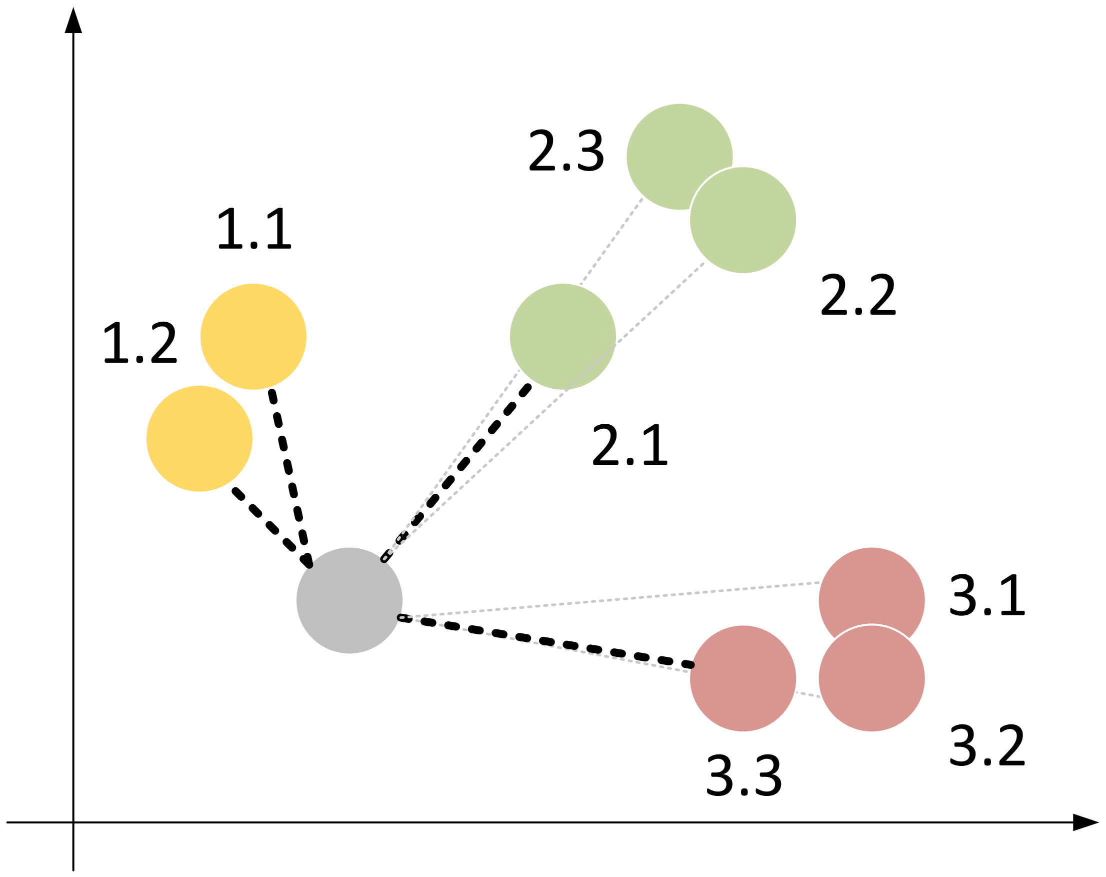


Figure 3. 11: K-Nearest Neighbor with K = 3

As Figure 3. 11, we want to classify the grey point into a class. Here, there are three potential classes – lime green, yellow and orange. Then we start by calculating the distances between the grey point and all other points. As shown in Figure 3. 11, we have determined the closest 4 are 1.1, 1.2, 2.1, 3.3. Then KNN vote on the predicted class labels based on the classes of the k nearest neighbors. The labels were generated using the k = 3 nearest neighbors.

3.4.4. Naïve Bayes (NB)

The Navie Bayes Classifier's objective is to determine conditional probability: for each of K possible outcomes or classes Ck.

$$p(C_k|x_1, x_2, \dots, x_n) \tag{23}$$

Let $x = (x1, x2, \dots, xn)$. Using Bayesian theorem, we can get:

$$p(C_k \mid x) = \frac{p(C_k)p(x \mid C_k)}{p(x)} \propto p(C_k)p(x \mid C_k) = p(C_k, x_1, x_2, \dots, x_n) \quad (24)$$

The joint probability can be written as:

$$\begin{aligned} & p(C_k, x_1, x_2, \dots x_n) \\ = \; & p(x_1 \mid x_2, \dots, x_n, C_k) \cdot p(x_2, \dots, x_n, C_k) \\ = \; & p(x_1 \mid x_2, \dots, x_n, C_k) \cdot p(x_2 \mid x_3, \dots, x_n, C_k) \cdot p(x_3, \dots, x_n, C_k) \\ = \; & p(x_1 \mid x_2, \dots, x_n, C_k) \cdot p(x_2 \mid x_3, \dots, x_n, C_k) \dots \cdot p(x_n \mid C_k) \cdot C_k \end{aligned} \quad (25)$$

Assume that all feature x are mutually independent, we can get:

$$p(x_1 | x_2, \dots, x_n, C_k) = p(x_1 \mid C_k) \quad (26)$$

Therefore, formula can be written as:

$$\begin{aligned} & p(C_k \mid x_1, x_2, \dots, x_n) \\ \propto \; & p(C_k, x_1, x_2, \dots x_n) \\ = \; & p(x_1 \mid C_k) \cdot p(x_2 \mid C_k) \dots \cdot p(x_n \mid C_k) \cdot p(C_k) \\ = \; & p(C_k) \prod_{i=1} p(x_i \mid C_k) \end{aligned} \quad (27)$$

Therefore, this is the final formula for Navie Bayes Classifier.

Prior probability and conditional probability are two parameters that are estimated using the Maximum Likelihood Estimation (MLE) method.

$$p(C_k) = p(y = C_k) = \frac{\sum_{i=1}^{N} I(y_i = C_k)}{N} \quad (28)$$

The prior probability equals the number of certain cases of y occur divided by the total number of records.

$$p(x_1 = a_j \mid y = C_k) = \frac{\sum_{i=1}^{N} I(x_1 i = a_j, y_i = C_k)}{\sum_{i=1}^{N} I(y_i = C_k)} \quad (29)$$

The conditional probability of $\boldsymbol{p\left(x_1 = a_j \mid y = C_k\right)}$ equals the number of cases when $x1$ equals to $a1$ and y equals to $C1$ divided by the number of cases when y equals to $C1$.

Naïve Bayes Classifier uses following formula to make a prediction:

$$\boldsymbol{y = \arg\max p(y = C_k) \prod p(x \mid y = C_k)} \qquad \textbf{(30)}$$

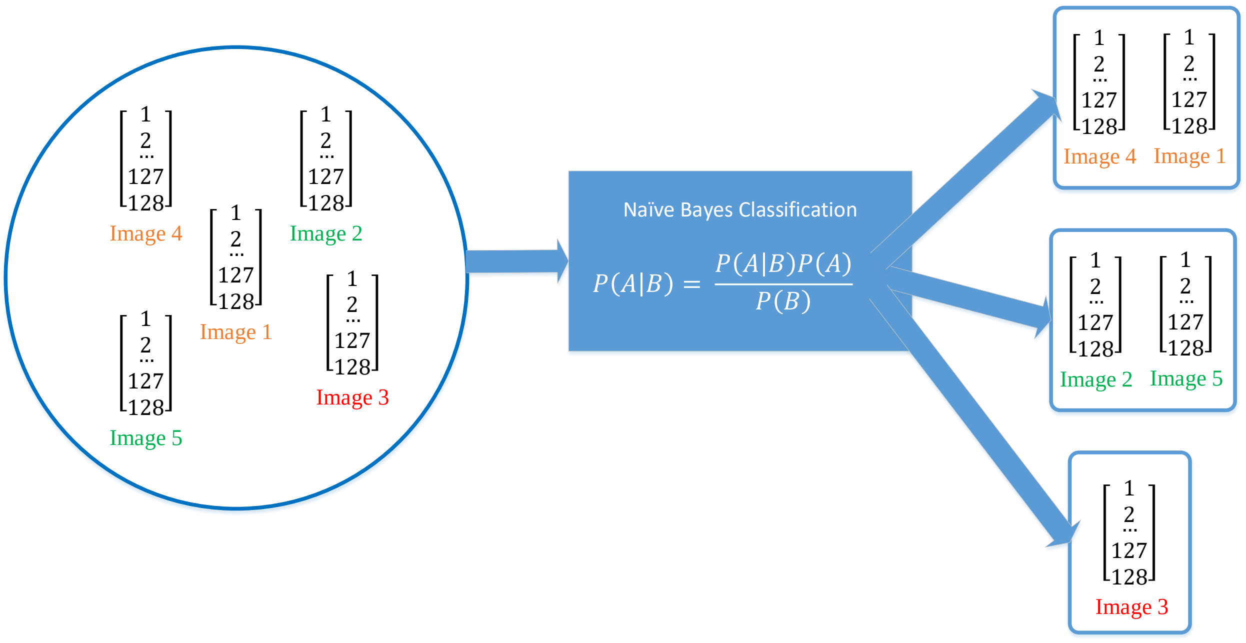


Figure 3. 12: Illustration of Naïve Bayes Classifier

CHAPTER 4: EXPERIMENTAL RESULTS AND PROGRAM SETTINGS

In this chapter, we will present the experimental setup method, including train dataset, application program and obtained results.

4.1. Dataset

The dataset is used for training; in this case, we collect the well-known celebrity faces in Vietnam, China, Korea, American, and England. The dataset contains 35 celebrity faces in this dataset, so we use this dataset because it is easy to collect.

A training dataset and a validation or test dataset are included in the dataset. All of the faces of the thirty-five people mentioned above are provided in a variety of orientations, lights, and sizes in the photo folders. And each individual has a set number of photos of each recipient.

This dataset will serve as the foundation for our comparison, which will be trained solely on the 'train' dataset and will compare face distance in the 'test' dataset. You can apply the same structure to your own photographs to create a comparison.

4.2. Analysis of the Experiments

To carry out the experiment, we will thoroughly examine each specific approach for each problem, as shown below. Face detection will be handled by MTCNN, and feature extraction will be handled by FaceNet. The pre-trained Keras FaceNet model by Hiroki Taniai will be used. It was trained on the MS-Celeb-1M dataset and expects color, whitened pixel values (normalized across all three channels), and input images with 160 x 160 square pixels. An embedded face is output as a 128-elements vector from a 160x160 image. In the simplest terms, we must first detect the face before we can perform face verification. Face detection that we use MTCNN is the process of automatically locating and localizing faces in photographs by drawing a bounding box around their extent. Before verification, we will alignment the output of face detected by MTCNN then we use that result to verification.

That was is our idea to analysis the system workflow and deploy to the application. Based on our idea, there are key steps that we will discuss below.

The first step is to detect the face in each photograph and reduce the dataset to a series of faces only. The image is then converted to RGB. Along the way, this function reports the progress of each loaded photograph as well as the shape of the NumPy array containing the face pixel data. The second that we must align the face to improve accuracy for verification process.

The following step is to make a face embedding. A face embedding is a vector that represents the extracted features of the face by FaceNet. This can then be compared to the vectors created for the other faces. Another vector that is close (by some measure) to another vector that is far (by some measure). A face embedding is a vector that represents the extracted features of the face. This can then be compared to the vectors created for the other faces.

Another vector that is close (by some measures) to the first could be the same person, whereas another vector that is far away (by some measures) could be someone else. We can use the FaceNet model to pre-process a face in order to generate a face embedding that can be stored and used as input to our classifier models, or we can use the FaceNet model to pre-process a face in order to generate a face embedding that can be stored and used as input to our models. Because the FaceNet model is both large and slow to create a face embedding, this latter approach is preferred. As a result, we can pre-compute the face embeddings for all of the faces in our 35 Celebrity Faces Dataset's train and test sets.

The model will then be evaluated. To begin, we must choose a random example from the test set, then obtain the embedding, face pixels, and compare the distance of faces. Then we can use the face integration as input to make a single prediction with the threshold. We can get the probability of the prediction and then get the name of the predicted face and the probability of that prediction.

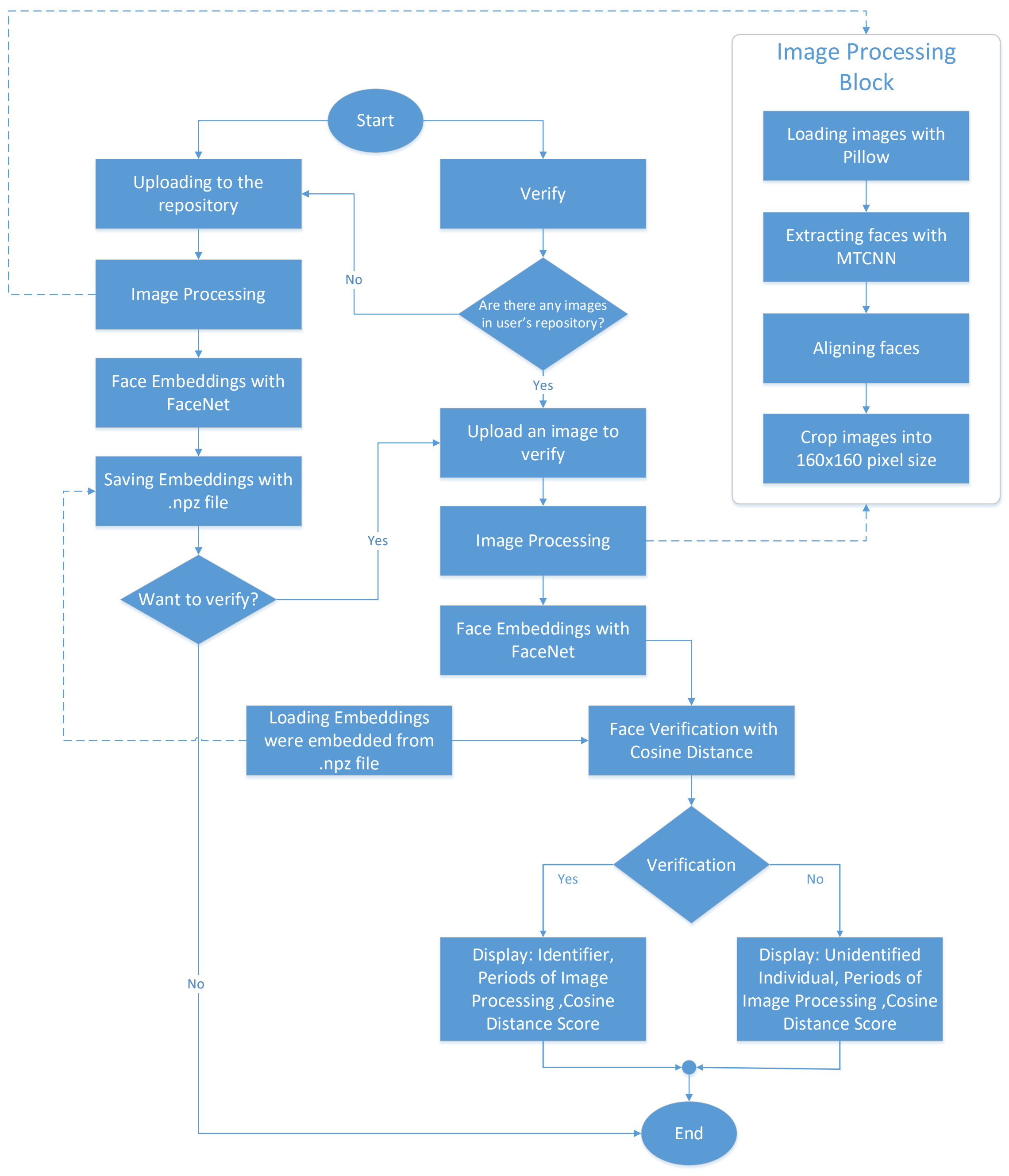


Image Processing Block
Loading images with Pillow
Extracting faces with MTCNN
Aligning faces
Crop images into 160x160 pixel size
Start
Uploading to the repository
Verify
Image Processing
No
Are there any images in user's repository?
Yes
Face Embeddings with FaceNet
Upload an image to verify
Saving Embeddings with .npz file
Yes
Image Processing
Want to verify?
Face Embeddings with FaceNet
Loading Embeddings were embedded from .npz file
Face Verification with Cosine Distance
Verification
Yes
No
Display: Identifier, Periods of Image Processing ,Cosine Distance Score
Display: Unidentified Individual, Periods of Image Processing ,Cosine Distance Score
No
End

Figure 4. 1: Verification process

Based on the preceding analysis, we examine the model as well as the working process of our application in greater detail (Figure 4. 1). We will create an app where each user can verify their face; to use the app, they must first create an account. After successfully creating an account and logging in, the user can perform two functions: upload and verify images. After successfully creating an account and logging in, the user can perform two functions: upload and verify images. The user will upload their own images for the images upload feature, and once confirmed, the feature will update those images in the database. The Image Proccessing Block step will be performed in this feature. The following tasks will be performed in the Image Proccessing Block step: loading the image with Pillow, extracting the face with MTCNN, aligning the face, and cropping images to 160x160 pixels size. Following the Image Proccessing Block step, the face embeddings with FaceNet. The step of face embeddings with FaceNet in this feature is to get embedding all the images that user had upload into the repository and then saved as .npz file.

For the image verification feature, the user will select an image to verify, the application will compare that image with the images that the user has uploaded to their archive and then output validation results, as Figure 4. 1. The user will not be able to verification without uploading the image to the archive. In this feature, there are 2 steps like the upload feature: Image processing block and face embedding with FaceNet. In the step of face embedding with FaceNet in this feature is to get embedding image that the user chooses to compare. When executing at the face verification step with cosine distance, the application will load the .npz file created in the image upload feature then compare the image with cosine distance and output the verification result if the result does not meet the threshold, it indicates that the individual is unknown.

4.3. Software Architecture

This application program is to check whether the distance of the image to be verified meets the set threshold or not? If so, output the image, identifier, and cosine distance score of that user, otherwise output the image, unidentified, and cosine distance score.

4.3.1. Experimental setup

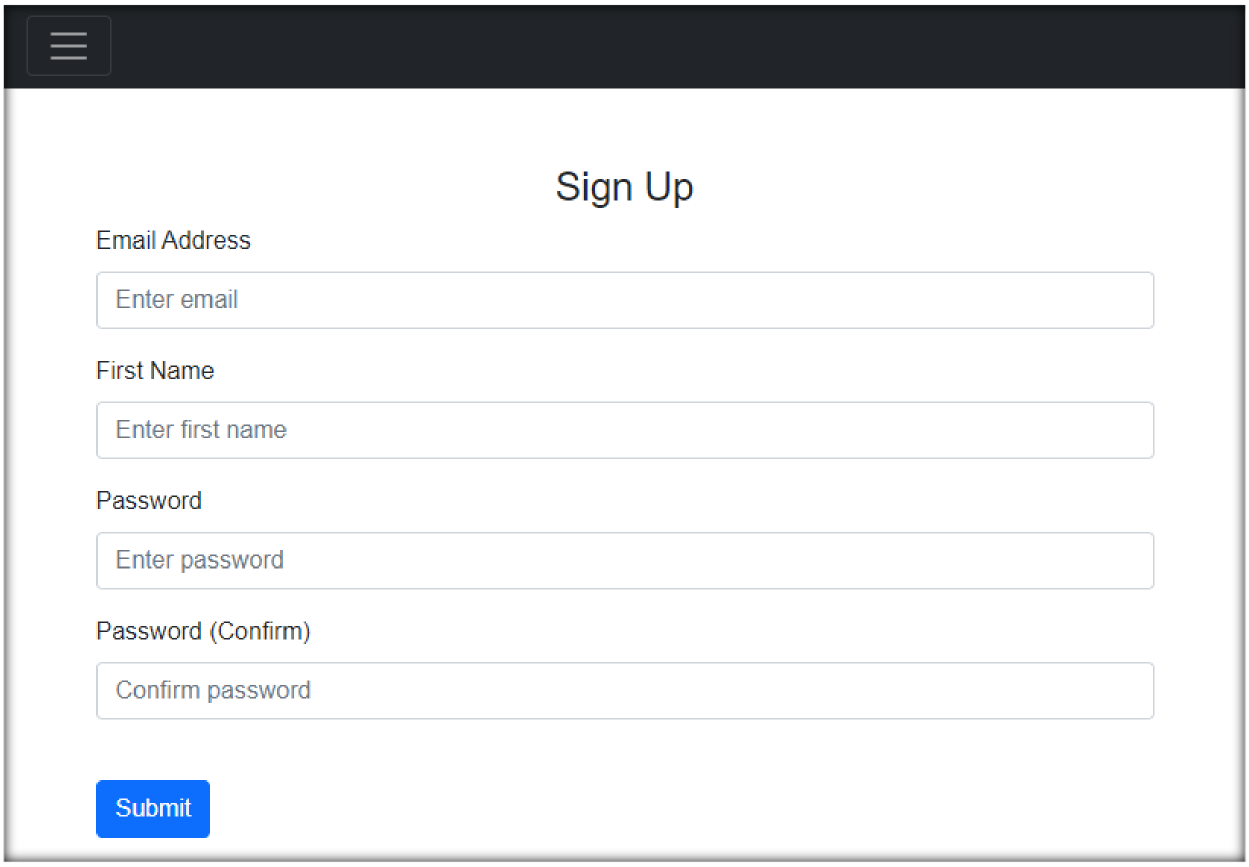
To train and test the program, we use a personal laptop with 16 GB of RAM and an Intel core-i7 with a CPU speed of 2.5 GHz and 64-bit Windows 10 is utilized. The programming language is Python, and the PyCharm IDE is used to run the program. The program is entirely based on deep learning neural networks and is built with the Tensorflow framework. These some Python libraries are required to run the application:

- Tensorflow (2.8.0): TensorFlow is an end-to-end open source machine learning platform. It features a robust, flexible ecosystem of tools, libraries, and community resources that enable academics to push the boundaries of ML and developers to quickly construct and deploy ML-powered applications.
- Numpy (1.18.5): NumPy is the foundational Python library for scientific computing. It is a Python library that includes a multidimensional array object, different derivative objects (such as masked arrays and matrices), and a variety of functions for performing quick array operations, such as mathematical, logical, etc.
- Pillow (7.2.0): Python Imaging Library is a free and open-source extension library for the Python programming language that allows you to access, manipulate, and save a variety of image file types.
- Operating System (OS): provides functions for interacting with the operating system. Python's basic utility modules include OS. This module provides a portable mechanism to access operating system-specific functions.
- OpenCV (4.5.4.58): OpenCV is a programming function library primarily geared towards real-time computer vision. It is primarily utilized to perform all image-related operations.
- MTCNN (0.1.0): MTCNN is a python (pip) library written by Github user ipacz, which implements the paper [2].
- Flask (2.1.1): Flask is a WSGI web application framework that is lightweight. It is intended to be simple to get started with the flexibility to scale up to sophisticated applications. It started as a basic wrapper around Werkzeug and Jinja and has since grown to become one of the most popular Python web application frameworks.

4.3.2. System interface

In general, our application software built completely, we applied the water-fall model to our project to manage. And the architecture we used is MVT (Model-View-Template) combined with Flask framework. In the interface, we used html5, css3 language and boostrap library to build pages like login, registration, main page, image loading page, verification page. In the backend, we used the Python language in Flask to write the processing features. As for the database management system, we used SQLite to store the database.

This is our system interface:


Sign Up

Email Address

Enter email

First Name

Enter first name

Password

Enter password

Password (Confirm)

Confirm password

Figure 4. 2: Sign Up Interface

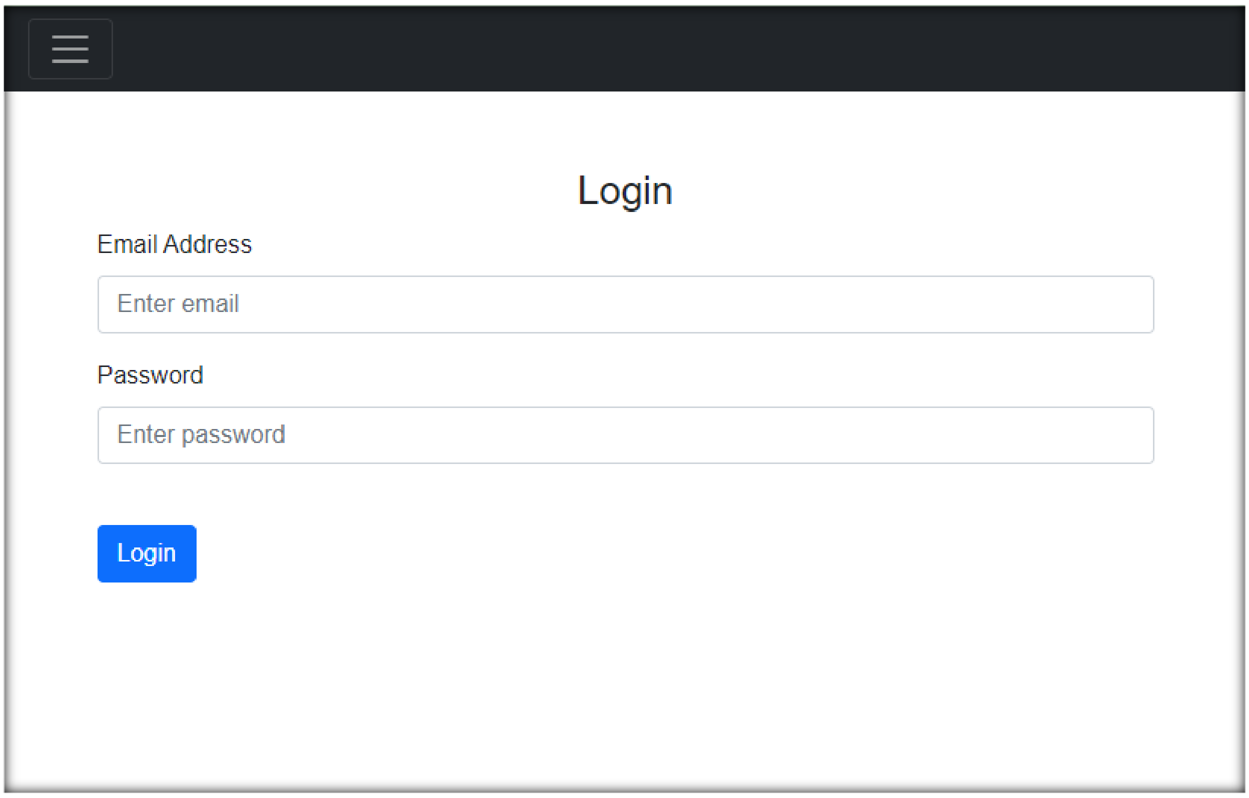


Figure 4. 3: Login Interface

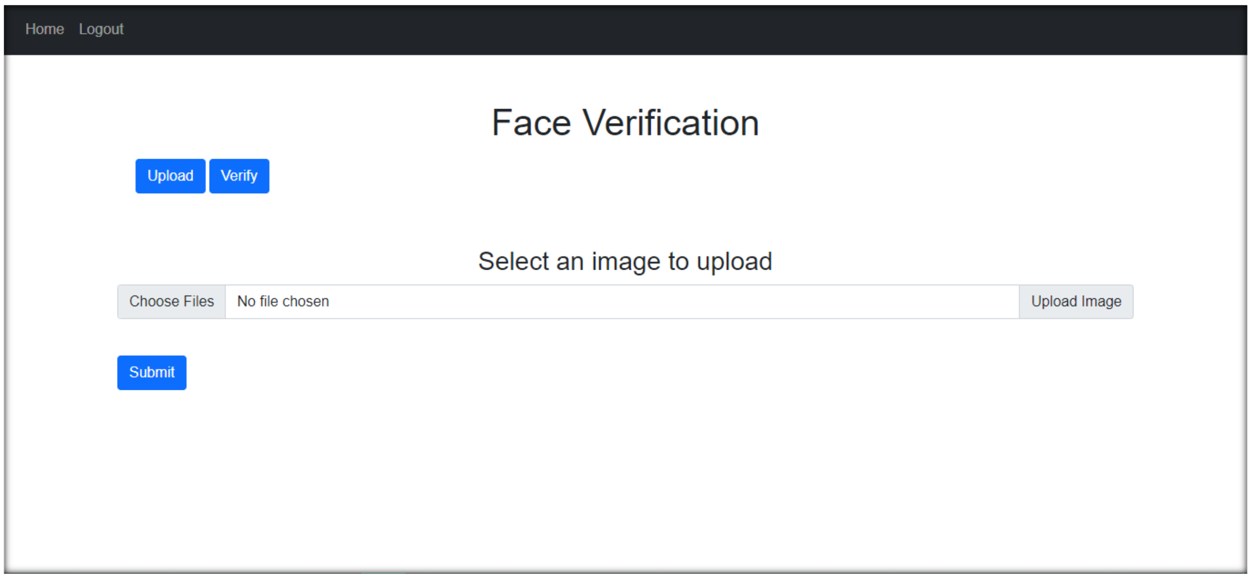


Figure 4. 4: Upload Interface

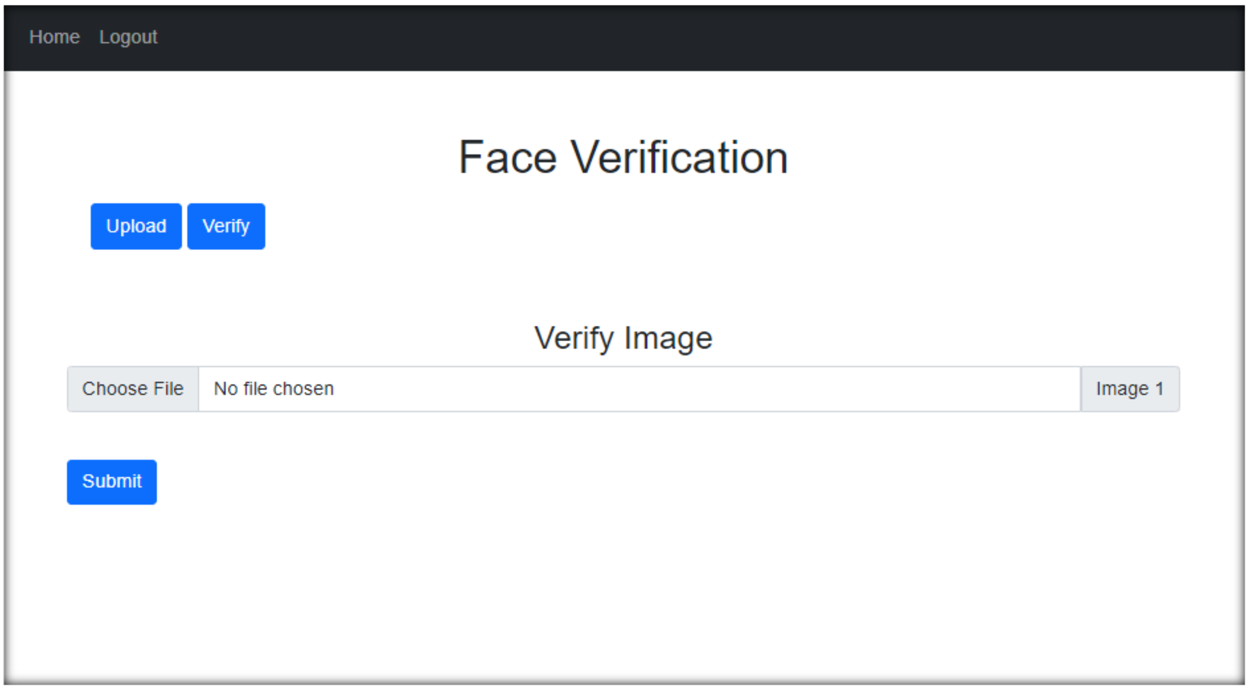


Figure 4. 5: Verification Interface

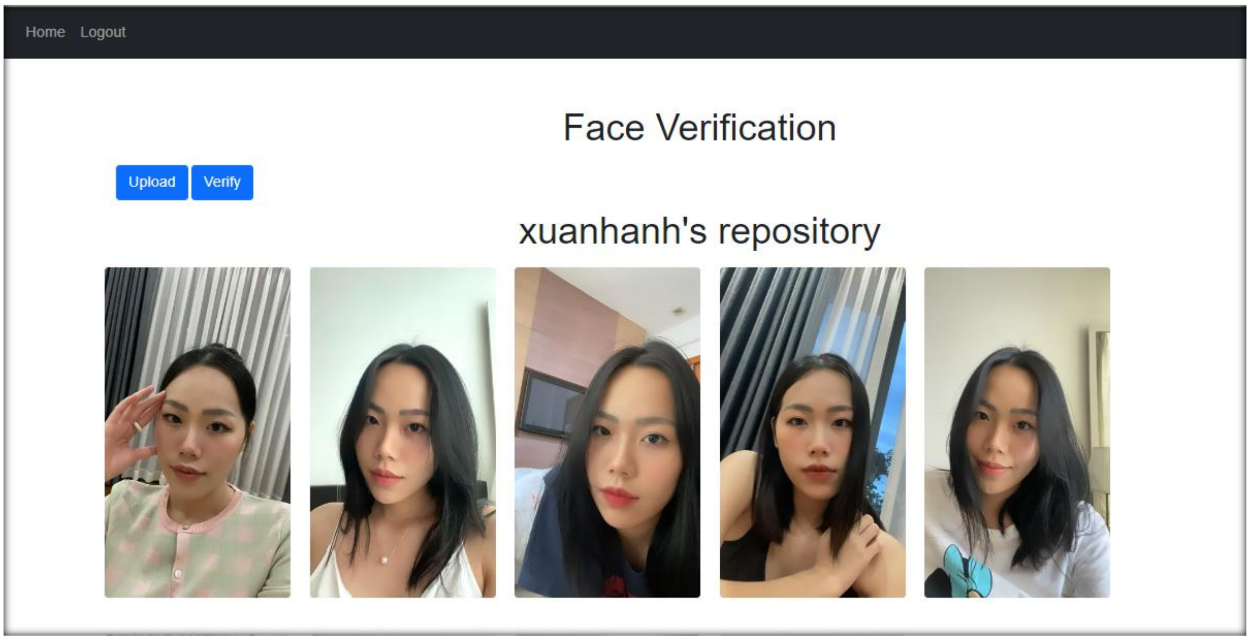
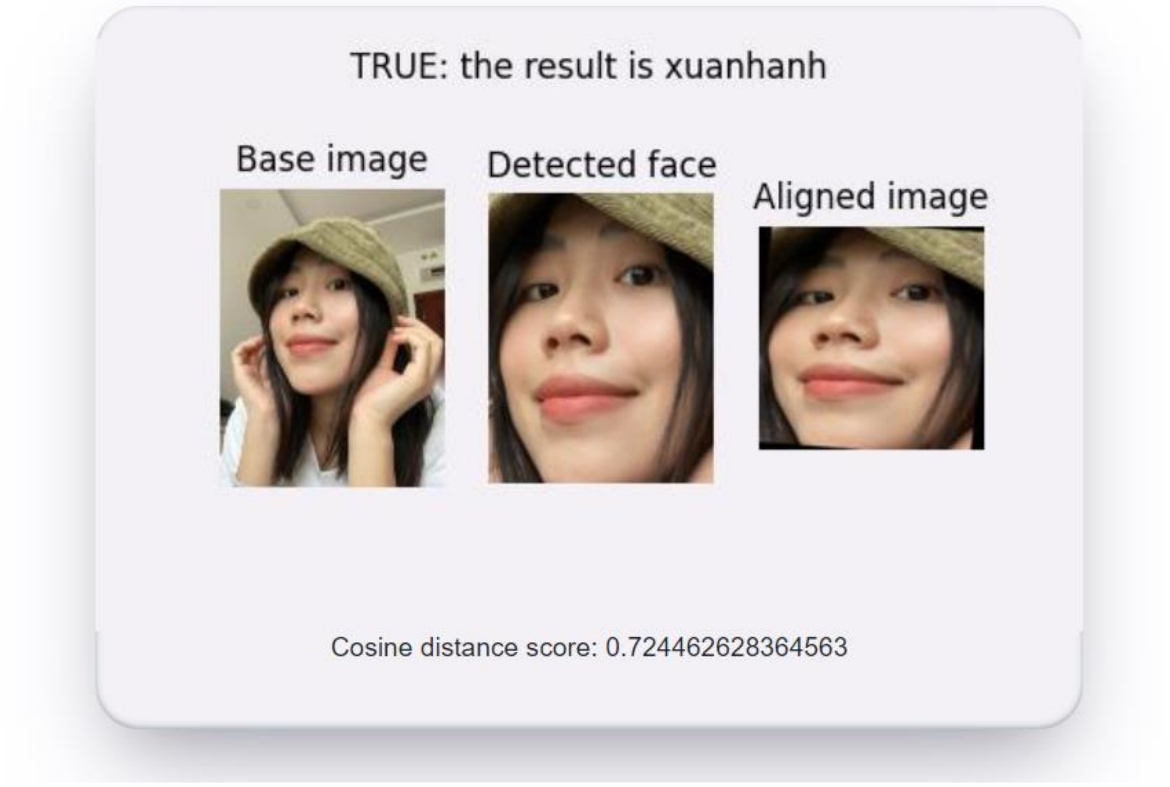


Figure 4. 6: User's repository interface



Cosine distance score: 0.724462628364563

Figure 4. 7: Result after verification

The program includes several basic functions:

- Sign up: This function allows users to register an account to enter the system.
- Log in: This function allows users to log into the system.
- Log out: This function allows users to exit the system.
- Home page: This function allows users to view their own repository.
- Upload: This function allows users to upload photos to their repository.
- Verify: This function allows users to verify their photo by uploading an image and that image will be verified against the photos in their repository.

4.4. Results and comparisons

4.4.1. Results without alignment

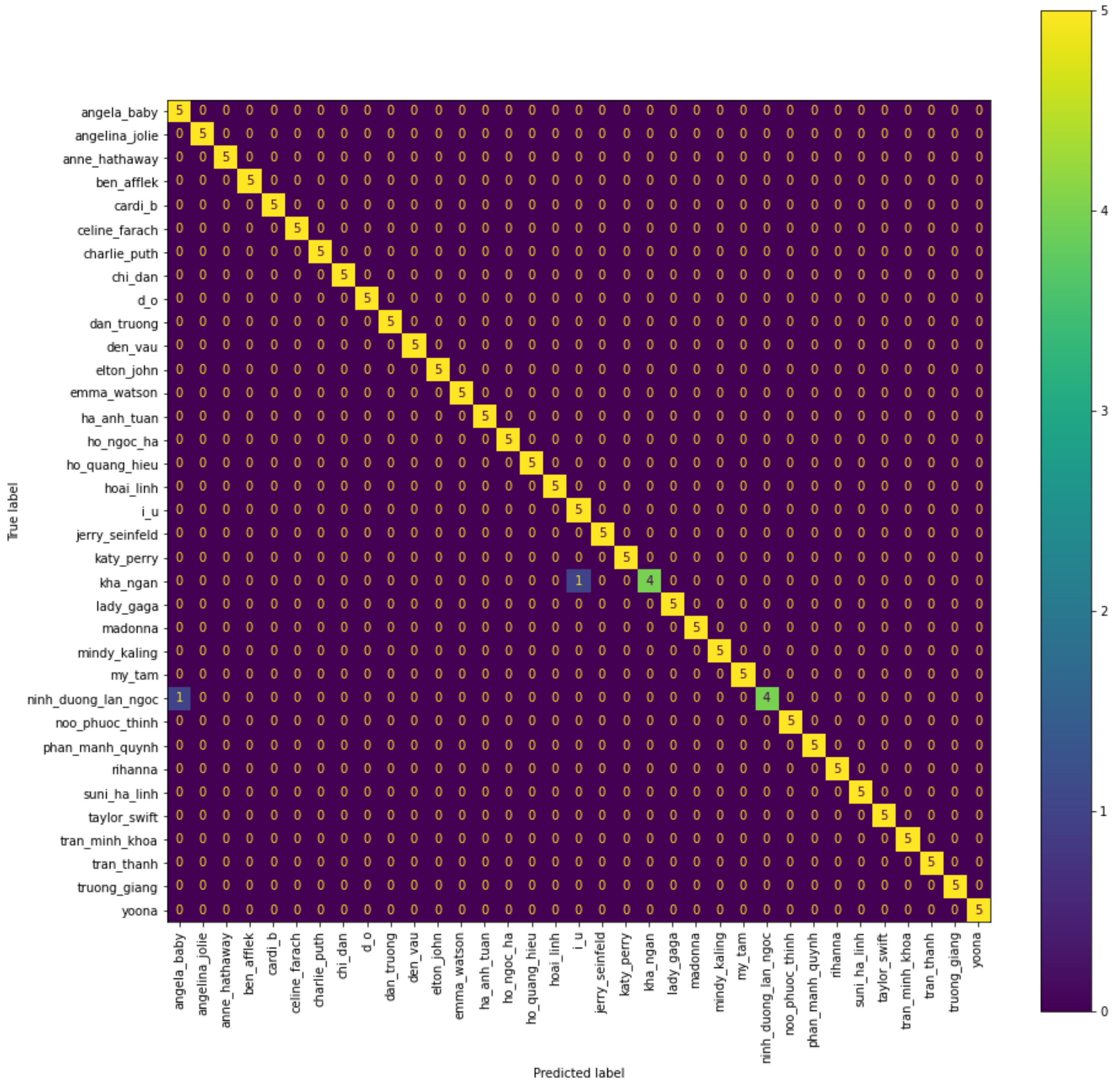


Figure 4. 8: Confusion matrix of KNN without alignment

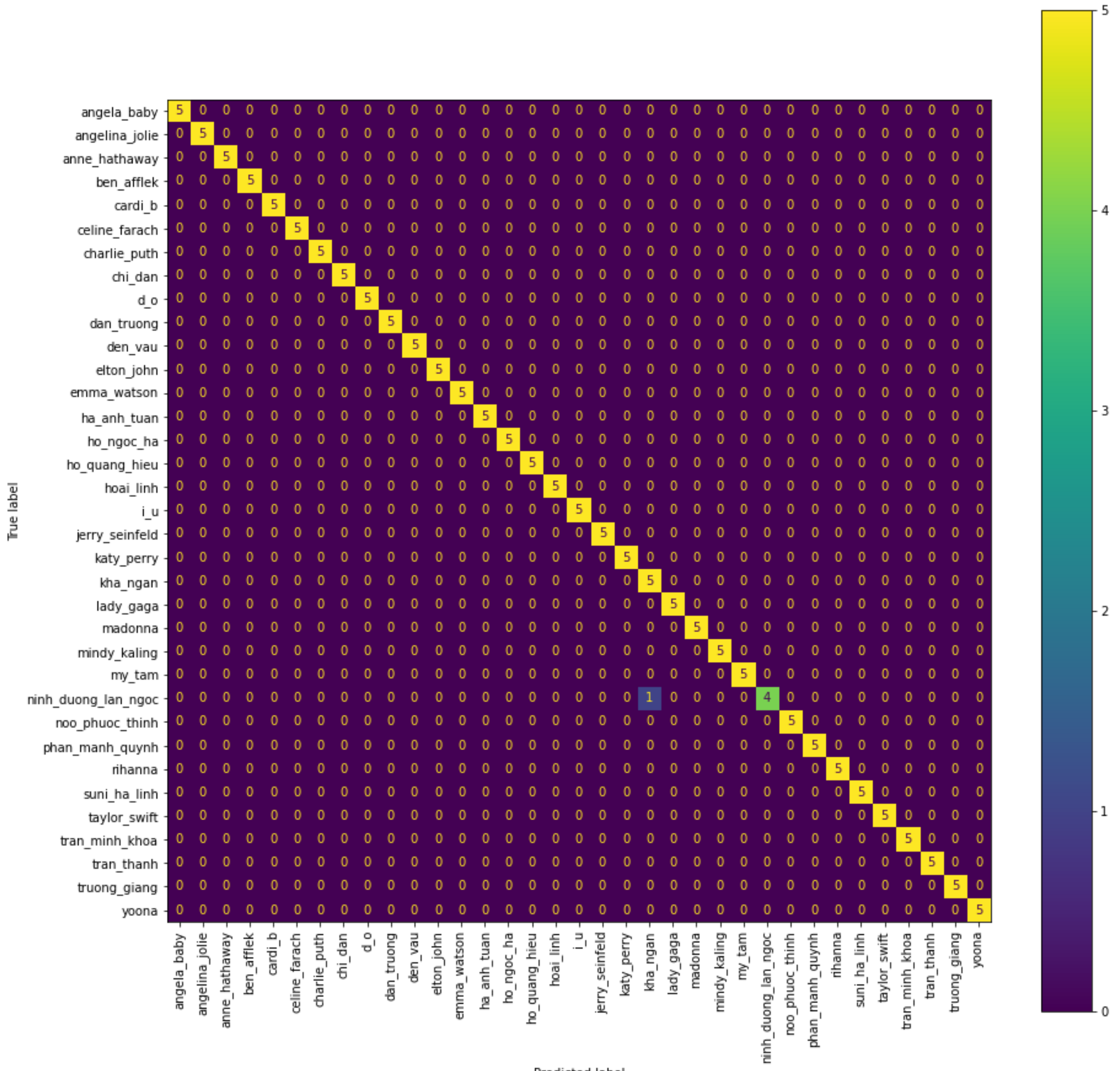


Figure 4. 9: Confusion matrix of NB without alignment

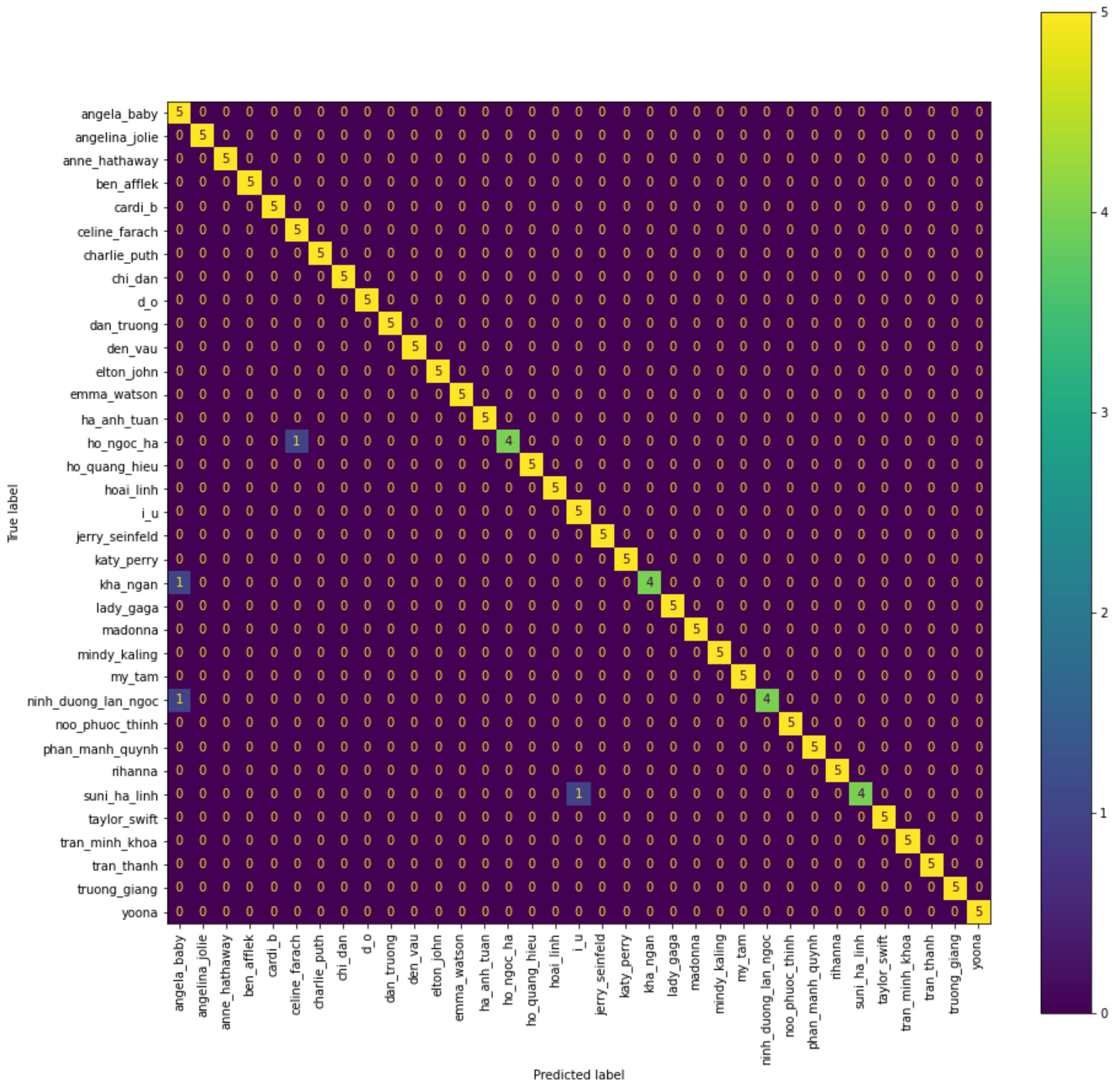


Figure 4. 10: Confusion matrix of RF without alignment

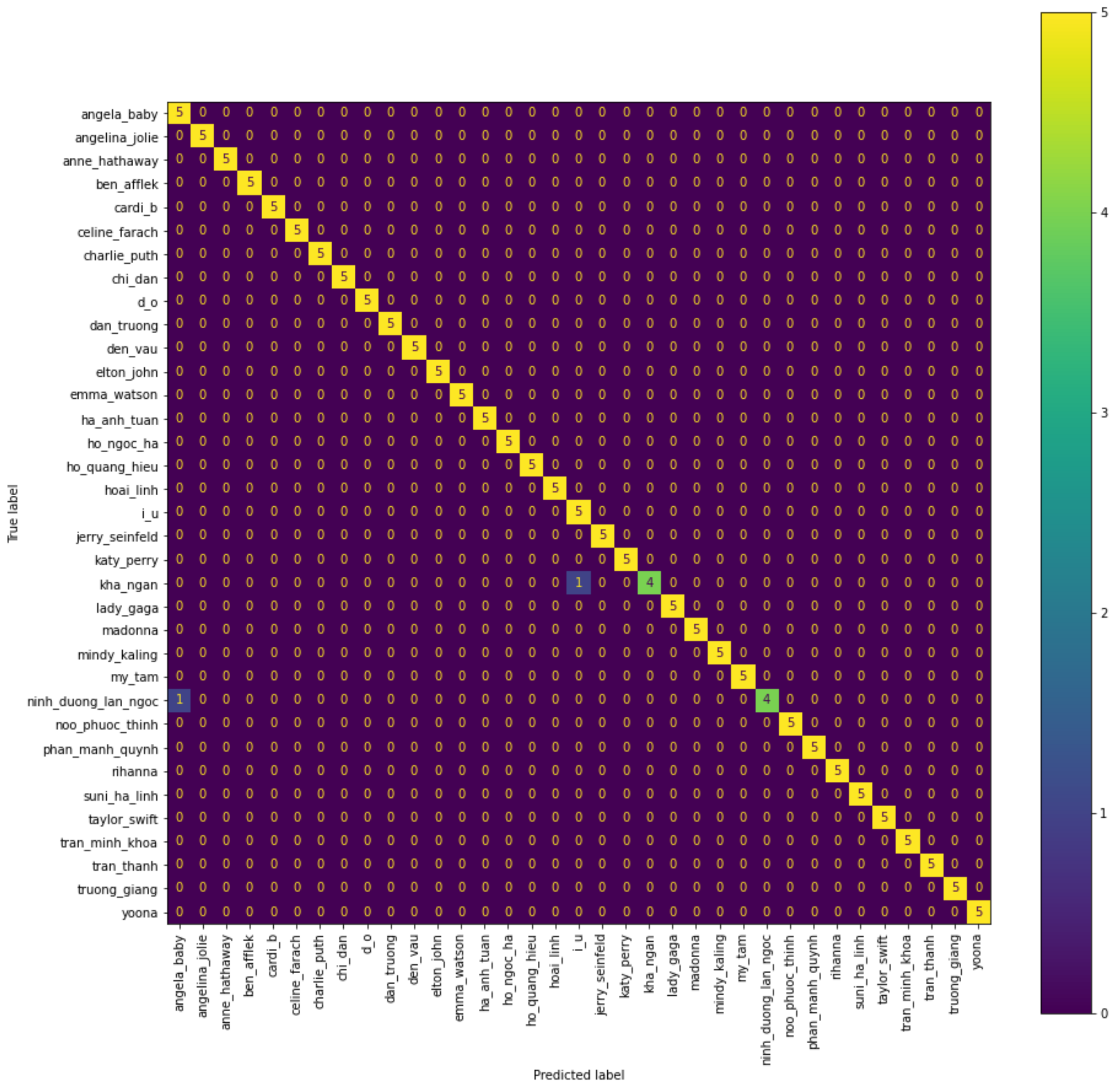


Figure 4. 11: Confusion matrix of SVM without alignment

	KNN	**NB**	**RF**	**SVM**
Training Accuracy (%)	98.998	99.833	100	100
Testing Accuracy (%)	98.857	99.429	97.714	98.857

Table 1: Training and testing results without alignment

From the above results we can see that. If we don't go through the alignment step, then the correct recognition rate will be greatly reduced

4.4.2. Results with alignment

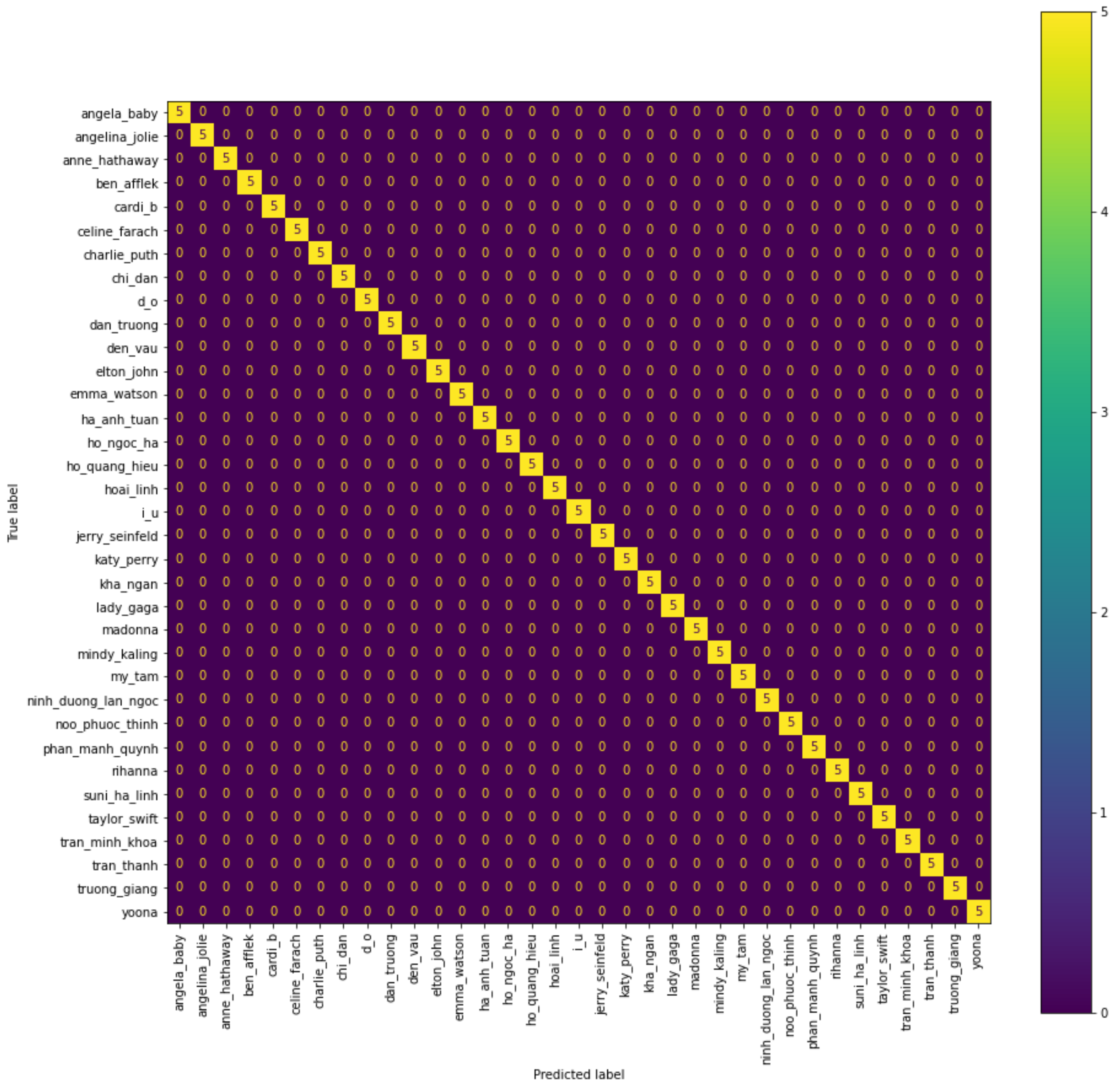


Figure 4. 12: Confusion matrix of KNN with alignment

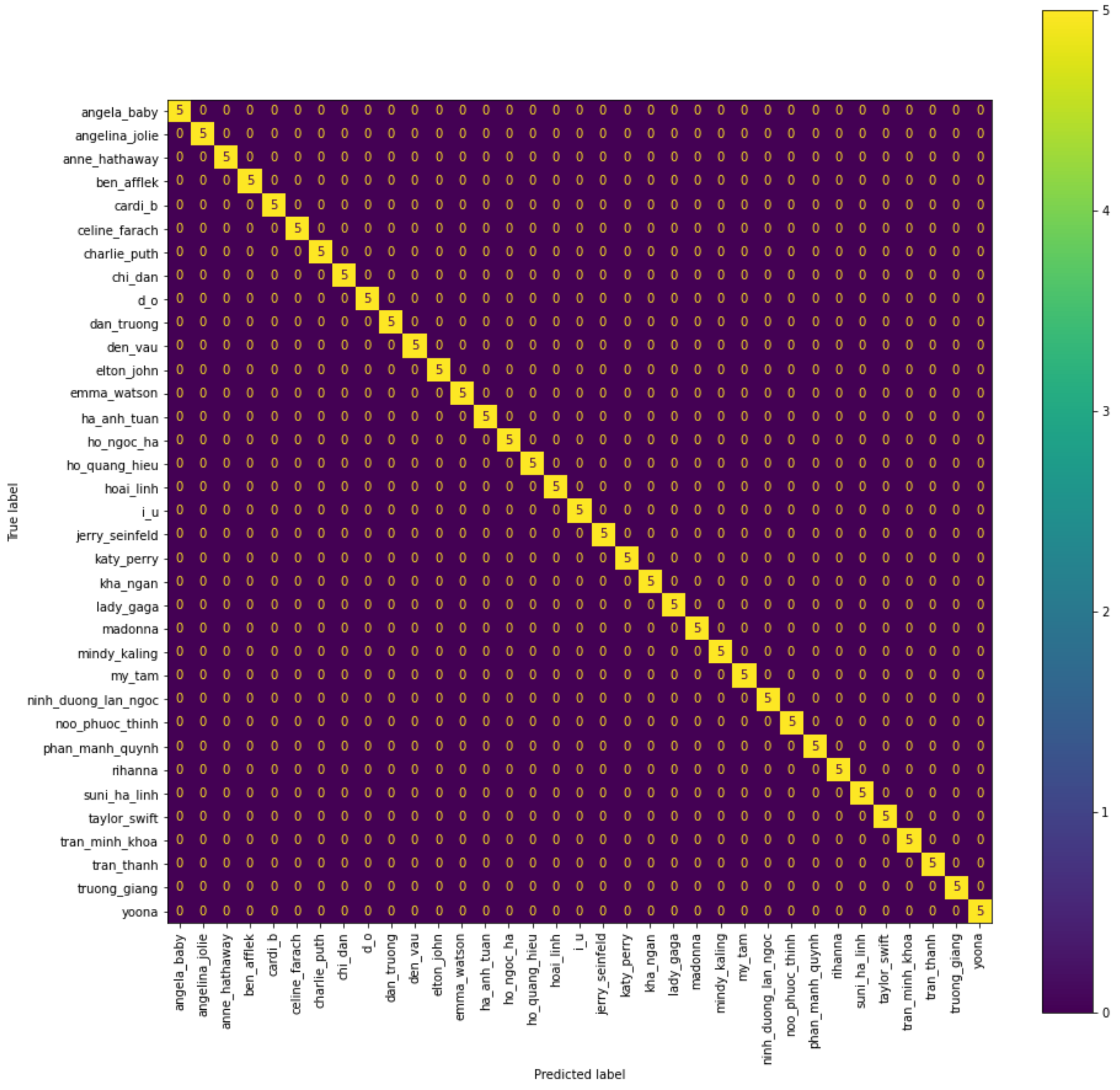


Figure 4. 13: Confusion matrix of NB with alignment

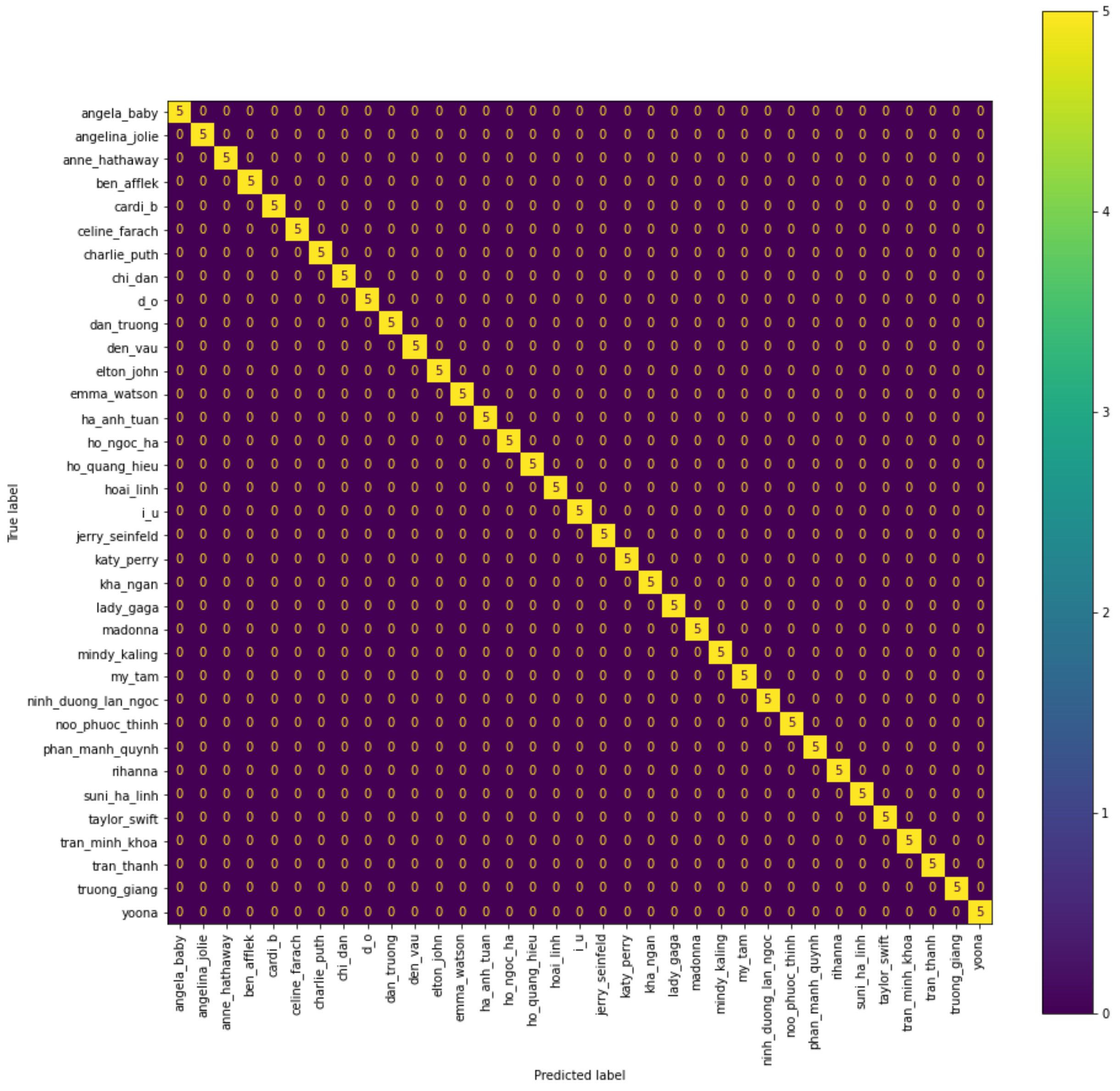


Figure 4. 14: Confusion matrix of RF with alignment

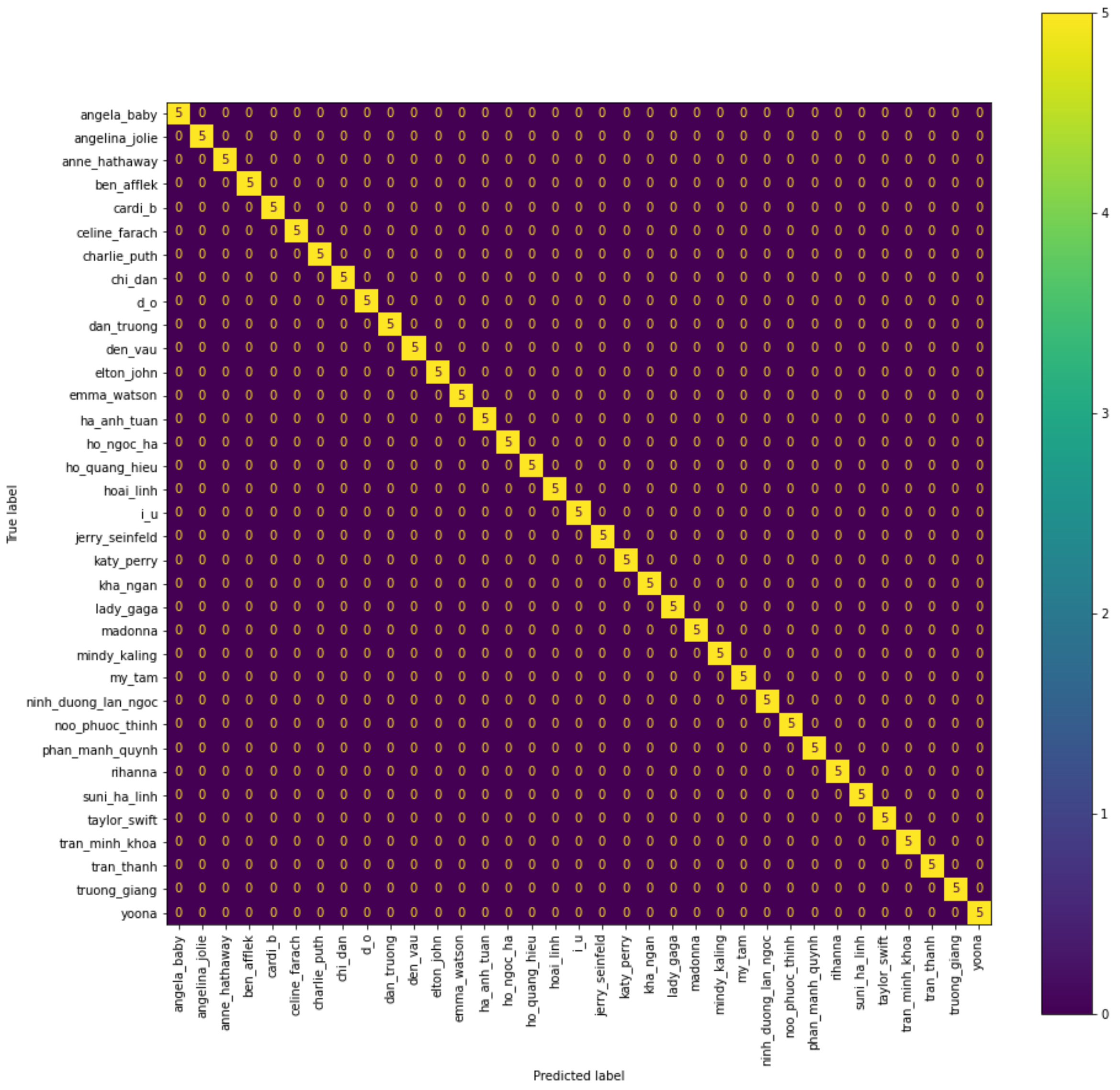


Figure 4. 15: Confusion matrix of SVM with alignment

	KNN	**NB**	**RF**	**SVM**
Training Accuracy (%)	99.833	99.833	100	100
Testing Accuracy (%)	100	100	100	100

Table 2: Training and testing results with alignment

4.5. Conclusions and recommendations

4.5.1. Conclusions

Face verification is a challenging and significant verification method. Among all biometric approaches, the facial verification approach has one significant advantage: it is user-friendly (or non-intrusiveness). We have provided an overview of facial verification technologies in this paper. This research presented a FaceNet-based face verification system based on training five perspective photos exclusively by reproducing those same five viewpoint images over and again. We have discussed concerns such as face alignment, accuracy of training data for face verification, both of which can have an impact on the verifier's performance. We hope that this paper gives readers a better grasp of face verification and contributes to the development of new alternatives and better opportunities in face verification techniques.

4.5.2. Recommendations

We know that the problem of human face verification has been studied since the 1970s, and so far, a lot of research and applications for this problem have been born. However, the problem of human face verification is still difficult, challenged by the diverse and rich transformations on the face. Therefore, there are many different methods to solve this problem. With the solution using FaceNet and MTCNN that we have applied, it has only been studied by experts in the past ten years. With limitations in terms of knowledge, qualifications and implementation time, we end this topic with a small part of the problem of pattern recognition in computer vision as well as computer science in general. With the orientations given below, we hope to be able to further improve human-computer interaction through facial verification system.

This is the result of our experiment, as well as the development of our first real-world application, which we combined with the trained FaceNet model. Our results are not particularly impressive in comparison to the achievements in face recognition in general and face authentication in particular, but in general, we have gained knowledge and research experience in this field. And, without a doubt, in the future, we will have the opportunity to improve, increase efficiency, and create a comprehensive application in order to contribute some achievements in the field of global technology in general, and the Vietnamese technology sector in particular.

REFERENCES

[1] E. Kaziakhmedov, K. Kireev, G. Melnikov, M. Pautov and A. Petiushko, "Real-world Attack on MTCNN Face Detection System," 2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON), 2019, pp. 0422-0427, doi: 10.1109/SIBIRCON48586.2019.8958122.

[2] K. Zhang, Z. Zhang, Z. Li and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," in IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499-1503, Oct. 2016, doi: 10.1109/LSP.2016.2603342.

[3] F. Schroff, D. Kalenichenko and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 815-823, doi: 10.1109/CVPR.2015.7298682.

[4] D. Sun, C. H. Q. Ding, B. Luo, and J. Tang, “Angular Decomposition,” in IJCAI, T. Walsh, Ed., 2011, pp. 1505–1510.

[5] Jonghyun Choi, Hyunjong Cho, Jungsuk Kwac, and Larry S. Davis, “Toward Sparse Coding on Cosine Distance” in 2014 22nd International Conference on Pattern Recognition, pp. 1051-4651, doi: 10.1109/ICPR.2014.757

[6] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T. W. Lee, and T. J. Sejnowski, “Dictionary learning algorithms for sparse representation.” 2003.

[7] T. Guha and R. K. Ward, “Learning Sparse Representations for Human Action Recognition,” IEEE Trans. on PAMI, vol. 34, no. 8, pp. 1576– 1588, 2012.

[8] C. Yuan, W. Hu, G. Tian, S. Yang, and H. Wang, “Multi-task Sparse Learning with Beta Process Prior for Action Recognition,” in CVPR, 2013, pp. 423–429

[9] Anton Razzhigaev, Klim Kireev, Edgar Kaziakhmedov, Nurislam Tursynbek, and Aleksandr Petiushko, “Black-Box Face Recovery from Identity Features” in Computer Vision – ECCV 2020 Workshops.

[10] Rita Goel , Irfan Mehmood , and Hassan Ugail, “A Study of Deep Learning-Based Face Recognition Models for Sibling Identification”, 2021 Jul 27;21(15):5068. doi: 10.3390/s21155068.

[11] Ferry Cahyono, Wirawan Wirawan, and Reza Fuad Rachmadi, “Face Recognition System using Facenet Algorithm for Employee Presence” in 2020 4th International Conference on Vocational Education and Training (ICOVET).

[12] Bin Jiang, Qiang Ren, Fei Dai, Jian Xiong, Jie Yang, and Guan Gui, “Multi-task Cascaded Convolutional Neural Networks for Real-Time Dynamic Face Recognition Method” In book: Communications, Signal Processing, and Systems (pp.59-66).

[13] Anthony Gidudu, Tshilidzi Marwala, and Hulley Greg, “Classification of Images Using Support Vector Machines” October 2007.

[14] Suguna G C, Kavitha H S, Sunita Shirahatti and Sowmya R Bangari, “Face Recognition System for Real Time Applications using SVM Combined with FaceNet and MTCNN”, International Journal of Electrical Engineering and Technology (IJEET), 12(6), 2021, pp. 328-335

[15] G. Anitha, P.Sunitha Devi, J. Vidhya Sri, and D.Priyanka, “ Face recognition based attendance system using mtcnn and facenet” Volume 6, Issue 8, 2020, ISSN No: 0932-4747.

[16] Gurucharan, M. (2020). Basic cnn architecture: Explaining 5 layers of convolutional neural network. URL: https://www.upgrad.com/blog/basic-cnn-architecture.

[17] Ben, X., & Jiang, K. (2021, November). Image Classification of Damage of Houses hit by the hurricane based on Convolution Neural Network. In 2021 International Conference on Big Data, Artificial Intelligence and Risk Management (ICBAR) (pp. 79-82). IEEE.

[18] Viola, P., & Jones, M. (2001, December). Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001 (Vol. 1, pp. I-I). Ieee.

[19] BREIMAN, L., 1999, Random forests—random features. Technical Report 567, Statistics Department, University of California, Berkeley, ftp://ftp.stat.berkeley.edu/ pub/users/ breiman.

[20] BREIMAN, L., 1996, Bagging predictors. Machine Learning, 26, pp. 123–140.

[21] BREIMAN, L., FRIEDMAN, J.H., OLSHEN, R.A. and STONE, C.J., 1984, Classification and Regression Trees (Monterey, CA: Wadsworth).

[22] QUINLAN, J.R., 1993, C4.5: Programs for Machine Learning (San Mateo, CA: Morgan Kaufmann).

[23] Meng Yuan, Seyed Yahya Nikouei, Alem Fitwi, Yu Chen, Yunxi Dong, "Minor Privacy Protection Through Real-time Video Processing at the Edge", Accepted by the 2nd International Workshop on Smart City Communication and Networking at the ICCCN 2020.

[24] Yueying Kao, Ran He, Kaiqi Huang, "Visual Aesthetic Quality Assessment with Multi-taskDeep Learning", April 2016.

[25] John C. Platt, "Probabilities for Support Vector Machines", in book: Advances in Large Margin Classifiers, January 2000.

[26] Wei You, Changqing Shen, Dong Wang, Liang Chen, Xingxing Jiang, and Zhongkui, "An Intelligent Deep Feature Learning Method with Improved Activation Functions for Machine Fault Diagnosis", 27 December 2019, pp. 1975 – 1985, doi: 10.1109/ACCESS.2019.2962734, IEEE.

[27] Steve Lawrence, C. Lee Giles, Ah Chung Tsoi, Andrew D. Back, "Face Recognition: A Convolutional Neural Network Approach", IEEE Transactions on Neural

Networks, Special Issue on Neural Networks and Pattern Recognition, Volume 8, Number 1, pp. 98–113, 1997. Copyright IEEE.

[28] Tuan Minh Le, Thanh Minh Vo, Tan Nhat Pham, And Son Vu Truong Dao, "A Novel Wrapper—Based Feature Selection for Early Diabetes Prediction Enhanced With a Metaheuristic", IEEE Engineering in Medicine and Biology Society Section, Volume 9, pp. 7869 – 7884, 30 December 2020. IEEE.

[29] Cichy, R. M., & Kaiser, D. (2019). "Deep neural networks as scientific models. Trends in cognitive sciences", 23(4), 305-317.

[30] Sherstinsky, A. (2020). "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network". Physica D: Nonlinear Phenomena, 404, 132306.

[31] Livingstone, D. J. (Ed.). (2008). "Artificial neural networks: methods and applications" (pp. 185-202). Totowa, NJ, USA: Humana Press.

[32] Chowdhary, K. (2020). "Natural language processing. Fundamentals of artificial intelligence", 603-649.

[33] Xu, L., Ren, J. S., Liu, C., & Jia, J. (2014). "Deep convolutional neural network for image deconvolution". Advances in neural information processing systems, 27.

[34] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). "Facenet: A unified embedding for face recognition and clustering". In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 815-823).

[35] Liu, Y., Wang, Y., & Zhang, J. (2012, September). "New machine learning algorithm: Random forest". In International Conference on Information Computing and Applications (pp. 246-252). Springer, Berlin, Heidelberg.

[36] Sun, S., & Huang, R. (2010, August). "An adaptive k-nearest neighbor algorithm". In 2010 seventh international conference on fuzzy systems and knowledge discovery (Vol. 1, pp. 91-94). IEEE.

[37] Saritas, M. M., & Yasar, A. (2019). “Performance analysis of ANN and Naive Bayes classification algorithm for data classification”. International Journal of Intelligent Systems and Applications in Engineering, 7(2), 88-91.

[38] S.A. Dudani, “The distance-weighted k-nearest neighbor rule,” IEEE Transactions on Systems, Man and Cybernetics, vol. 6, 1976, pp. 325-327

[39] C. Liu, “The development trend of evaluating face-recognition technology", in 2014 International Conference on Mechatronics and Control (ICMC), Jul. 2014, pp. 1540{1544. doi: 10.1109/ICMC.2014.7231817.

[40] Ku, H., & Dong, W. (2020). “Face recognition based on mtcnn and convolutional neural network”. Frontiers in Signal Processing, 4(1), 37-42.

[41] Chen, X., Luo, X., Liu, X., & Fang, J. (2019, May). “Eyes localization algorithm based on prior MTCNN face detection”. In 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC) (pp. 1763-1767). IEEE.

[42] Wu, C., & Zhang, Y. (2021). “MTCNN and FACENET based access control system for face detection and recognition”. Automatic Control and Computer Sciences, 55(1), 102-112.

[43] Yi, D., Lei, Z., Liao, S., & Li, S. Z. (2014). “Learning face representation from scratch”. arXiv preprint arXiv:1411.7923.

[44] Wang, M., & Deng, W. (2021). “Deep face recognition: A survey”, Neurocomputing , 429, 215-244.

[45] Engel, C., Mangiafico, P., Issavi, J., & Lukas, D. (2019, May). “Computer vision and image recognition in archaeology”. In Proceedings of the Conference on Artificial Intelligence for Data Discovery and Reuse (pp. 1-4).

[46] Zhao, Bo & Feng, Jiashi & Wu, Xiao & Yan, Shuicheng. (2017). “A survey on deep learning-based fine-grained object classification and semantic segmentation”. International Journal of Automation and Computing. 14. 10.1007/s11633-017-1053-3.

[47] Venkatnarayanan, Harini & Bhanumathi, V. (2016). “Automatic cataract classification system”,pp. 0815-0819, 10.1109/ICCSP.2016.7754258.