Masked Face Identification Using Face Recognition

by

Tareq Hossen
18301133
Abbas Uddin
18301198
Niloy Barua
18301087
Chowdhury Azmain Faik
18101224

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

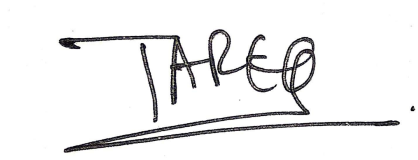
Department of Computer Science and Engineering
Brac University
May 2022

Declaration

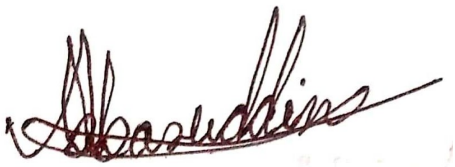
It is hereby declared that

1. The thesis submitted is our own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

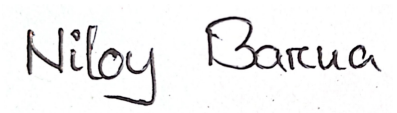
Student's Full Name & Signature:



Tareq Hossen
18301133



Abbas Uddin
18301198



Niloy Barua
18301087

Chowdhury Azmain Faik
18101224

Approval

The thesis titled "Masked Face Identification Using Face Recognition" submitted by

1. Tareq Hossen (18301133)
2. Abbas Uddin(18301198)
3. Niloy Barua(18301087)
4. Chowdhury Azmain Faik(18101224)

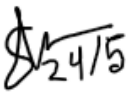
Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 29, 2022.

Examining Committee:

Supervisor:
(Member)

Md. Khalilur Rhaman, PhD
Associate Professor
Department of Computer Science and Engineering
BRAC University

Co-Supervisor:
(Member)



Shaily Roy
Lecturer
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Abstract

This work intends to express one of the several well-known biometric authentications entitled Masked Face Identification models by applying current Face Recognition algorithms and public masked face raw data that predict beneficial use. At the end of 2019, the COVID-19 pandemic has been exotically expanding worldwide, which severely negatively harms people's economies and well-being. Since using facial masks in social environments is now an efficient system to stop the spread of viruses, Nevertheless, appearance identification using facial masks is now a profoundly demanding duty because of the shortage of appropriate facial statistics. Here in our approach, the Deep Learning method will be executed by us to recognize the masked appearance by employing different face portions, some extra-superintendent and some owned-superintendent multi-task training facial appearance spotters, which can compact with different scales of appearance quickly and effectively. Additionally, the features are extracted by us from the masked face's eyes, forehead, and eyebrow areas and merged with characteristics acquired from those methodologies into a combined structure for identifying masked faces. In order to process, we will perform various image processing techniques on our dataset to clean our data for better accuracy. We will train our model from scratch to perform face-mask recognition. The most important part of this project remains the data collection and data cleaning process. Using a data-centric approach, we will systematically enhance our data-set to improve accuracy and prevent overfitting by performing data augmentation and stratified sampling and keeping our model architecture constant. Finally, our proposed systems will be compared by us with multiple unions of genius appearance identification techniques among those advertised by CASIA, LFW, and owned gathered raw data, which are managed from different sources. When wearing a mask, a person's face is hidden by 60–75%. Using only 30–40% of a person's face, we designed a face mask recognition model with an accuracy of 99.84%. Trained on a modified CASIA dataset containing images with and without masks, the model could successfully get the embeddings of 85743 people within a few minutes and perform perfect face recognition with and without masks.

Keywords: Masked Face; Face Recognition; CNN; MTCNN; Deep Learning; ResNet V1;

Dedication

We dedicate the report to our parents. Without their participation, attention, and support, we would not have gotten this far. We owe a debt of gratitude to them. Many thanks to them.

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption. Secondly, to our Dr. Md. Khalilur Rhaman sir and co-supervisor Shaily Roy ma'am for their kind support and advice in our work. They helped us whenever we needed help. And finally to our parents without their throughout sup-port it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

CNN Convolutional Neural Network

ICP Iterative Closest Point

LBP Local Binary Pattern

$MTCNN$ Multi-Task Cascaded Neural Network

PCA Principal Component Analysis

SSD Single-Shot detector.

Chapter 1

Introduction

As a result of rapid progress in modern science and technology, the world is continually expanding and changing. The significant advancement of digital technologies has provided us with a wide range of new alternatives for advancing civilization around the globe. We may use our understanding of fundamental development in technology innovation to increase human wellbeing by reducing human effort and resolving human problems. For example, computer vision, facial recognition, camera networking, and artificial intelligence are all being used in our country to solve various problems and crimes. Here, Face recognition is one of the most challenging things in computer vision and identification. Using facial recognition technology, Images, videos, and real-time videos can be used to identify a person. Face recognition is also a component of biometric security [2]. After the announcement of the COVID-19 virus as a pandemic by WHO, the whole world is going through a COVID-19 pandemic. The COVID–19 infection can be transmitted through contact or touch with infected surfaces. So, the old-style biometric frameworks dependent on pins or fingerprints are now not safe. Facial recognition is safer since touching is not required. Besides, facial identification system is frequently a favorable field in computer vision. Practically everyone wears facemasks at the time of the COVID-19 pandemic. Facemasks conceal the lower part of a person's face, thereby rendering standard face identification technology on face recognition self-pickup cabinets nearly useless. On the other hand, using a facemask creates subsequent issues: Fraudsters use the mask to invade and execute violations without being detected. Again, face authentication gets extremely hard when a facemask conceals a significant portion of the face. General face identification techniques are ineffective when a facemask covers the face, as the entire facial region is not readable. During the process of facial identification, exposing the facial region is crucial as it is utilized for facial normalization, posture correction, and facial familiarization. Facemasks have posed a substantial challenge to existing face recognition systems. As a result of these issues, improving the effectiveness of remaining facial recognition technologies for facemask identification is crucial. We approached the masked face recognition system to handle these problems, aiming to detect the particular face based on the eyes, ears, and the upper frontal head regions. We will handle this job using a deep learning-based method. Besides, we came to know that there are several techniques to complete this job like the Matching approach, Facial Landmarks, Image augmentation, generating masked face using MaskTheFace open-source tool, FaceNet model, Restoration approach, Discard occlusion based approach, ResNet V1, local

constrained dictionary learning method, etc. However, to do facial recognition, we must first detect the face in the image. We can distinguish the individual after detecting their face. It is simple for a human to recognize a face, but it is more difficult for a computer to do so. As a result, we must first train the system to recognize the facial component of pictures. After recognizing the face in the input picture, we will apply the Pre-trained deep learning-based method to extract features from the image, which will be used for further training.

1.1 Problem Statement

At the end of December 2019, there was a breakout of strange pneumonia indicated by fever, dry cough, and weakness that emerged from China, resulting in an intimidating outburst and expansion globally. The disease was named COVID-19 [12] formally. COVID-19 is a viral infection that spreads through inhalation or contact with infected droplets. About fifty-one million people were infected, and more than one million people died worldwide because of COVID-19[13]. WHO and other health organizations strongly suggested that using a face mask can decrease the transmission of COVID-19 by about 65 percent [10]. It was suggested that the people cover their faces in crowded places to reduce the transmission of infection. Because of this outbreak, our regular day-to-day life is also hampered to some extent. The COVID-19 outbreak also raised the question of some technology-based security systems such as face recognition, Biometric fingerprint, Iris scanner, etc. As wearing facemasks has become an essential part of our lives, it has become essential to ensure maximum security in every institution. We know almost every institution is dependent on technology-based security systems. Because of COVID-19 transmission, some of the systems that need direct contacts, such as Fingerprint and Iris Scanner, have become unusable to reduce transmission. However, face recognition remains the only AI-based system that could provide contactless security. However, it is very challenging to recognize faces and detect a person's identity through face recognition when a person is wearing a facemask. Moreover, it is very annoying and also not hygienic enough to touch the mask very often to show the entire face every time for face recognition.

Face recognition is easier when a face is open, and there is no occlusion in the face. Nevertheless, when a face is heavily disguised, like wearing a facemask makes it challenging to detect faces. In institutions, face recognition is used not only to detect faces but also to detect the identity of a person or employee. Some institutions also use face recognition systems to attend to their employees and students. At present, algorithms are unsuccessful on masked faces, which have been used to have success in detecting unmasked faces. One of the significant drawbacks of identifying a face that is not masked is that the entire face to recognize a person would be utilized by the deep learning models. Nevertheless, in a masked face, the nose and mouth remain covered. It breeds the complication of recognizing any individual utilizing only the remaining facial region: eyes and, in some cases, the upper frontal region of the head. Though forehead size also changes with the change in a person's hairstyle. When the initial goal will be recognizing faces when people are wearing caps, glasses, and facemasks in addition to other items that can block portions of the face amidst keeping other parts intact in those images; the basic facial identification algorithm becomes finite[15].

When the features required to foresee the identity of any person accurately gets minimized from the entire face to only the eye and, every so often, the upper frontal region of the head, then it is a very demanding issue for any computer vision model to recognize the identity of masked faces. The SSD FaceNet model and the Inception ResNet-V1 architecture, which is trained on human faces, can be used in solving the problem of recognizing an individual's identity who is the masked person.

1.2 Research Objectives

Our ultimate objective is to develop a face detection system that will determine the identity of humans wearing face masks using the Deep Learning Algorithm and Inception ResNet-V1 Architecture. This system can further help us to identify masked faces, and it will also ensure everyone's safety on the streets by identifying faces. We want to implement more accuracy in one of our research-based paper applications. We want to create a system that will identify the masked faces with fewer parameters and less power consumption with the help of training our dataset and detecting face identity. To be more specific, our objective is to focus on the following ideas to determine masked faces:

1. To gain an intense understanding of face recognition.
2. To deeply understand Deep Learning and its application.
3. To intensely learn about Inception ResNet V1 and other CNN models and their applications.
4. To achieve a broader concept about deep learning.
5. To create a model that can detect a person's identity wearing a facemask.
6. To evaluate the Deep Learning and Inception ResNet V1, whether these are more reliable approaches in face detection or not.
7. To present feasible improvements on the masked and unmasked face recognition model.

Lastly, we want to introduce Transfer Learning differently in our research to ensure more privacy for the data. Transfer learning will be beneficial in the case of data processing between different kinds of models, and it will also ensure data privacy, which is very important in the case of video data or image data set research.

Chapter 2

Literature Review

Nowadays, security and authentication are two standard terms in our daily life. Moreover, we are using various authentication systems like face recognition, Irish scanning, and biometric authentication to ensure robust workplace security. However, the current Covid-19 situation creates a big question about using these systems. WHO and other organizations strongly suggested that people use face masks in public places and avoid direct contact. Consequently, Irish scanning and biometric authentication are now unusable for reducing Covid-19 transmission because these two systems need direct contact. So, some work has been directed in the field of occlusion-based face recognition.

To detect and recognize a human face in real-time and improve the method's robustness, the authors in paper [1] proposes a software framework based on real-time activity detection and recognition. Researchers introduced intelligent control and fail-over mechanisms to detect appearance and track a moving person's face. The detection process is herculean due to frame differences and feature correlation techniques. Additionally, the author developed a scheme that employs the relative positions of people, velocity, and individual trajectories as parameters for identifying targeted stalker groups. This straightforward scheme can be used in place of more complicated models. However, camera calibration errors and a lack of pixel processing make paperless efficient in this paperless world.

In the article [3], a 3D face recognition system is used to restore portions of the face that are covered. By utilizing the ICP Algorithm, the model takes 3D input of an individual's face. After implementing the PCA algorithm and using the restoration technique, the covered portions of the face are identified.

In the article [4], the facial image is segmented into compact localized blocks. Then the support vector machine is then incorporated to identify disguised parts of the face and remove those. Again, the mean-based weight matrix determines the remaining partial face.

Wing at [6] marks out critical points on the disguised parts of the face, then removes those parts of the face, and from those key points, he extracts features. Furthermore, alongside approximating the resemblance of the two facial images as the space between two aligned attributes, a matching mechanism is being implemented to line up extracted information with those in the gallery.

The article [7] depicts calculating crowd flux can be an extremely effective tool for appearance detection. Prudent use of resolution can enhance surveillance's dynamic nature.Providing the exact resolution throughout the surveillance area can occasion-

ally result in a loss of resolution. Thus, using a constant population to determine whether or not a space is crowded is a very noble idea. Different resolutions in the dense zone can help ensure the model's accuracy. Dividing the entire area into local and global coverage zones is an excellent idea. This strategy ensures a minimum resolution in each area segment, with an emphasis on the crowded zone, resulting in a highly productive, cost-effective, and practical system. Another method of determining if a location is crowded is to analyze a data set. Additional surveillance of those problematic areas may make the approach more effective.

The research article [8] depicts an astounding Principal component analysis (PCA) implementation on both masked and unmasked face recognition. According to the authors, they have used the Viola-Jones algorithm to identify facial regions from the image. After that, they extracted the facial features from face regions using PCA. Then they train the model for face recognition. In the test phase, a target face image is given to the system. In the beginning, computation is done for its PCA representation. Hence, possible facial segments are distinguished. Then, the Ghost faces are selected and used for recognition. By differentiating the chosen characteristics from the aimed image, recognition is executed opposite to those chosen from the correlated guided images.

Aqeel Anwar and Arijit Ray Chowdhury, in paper [9], propose a Face Recognition System using Facenet architecture. Facenet generates integrated embedding of the facial images and then differentiates the faces in the embedded space to bring out the verdict. Here, the author used the VGGFace2 dataset to train Facenet. VGGFace2 dataset contains a large-scale face dataset. Some performance metrics have been used to inspect the proficiency of the trained network.

The author [11] has primarily suggested a strategy that is based on person re-identification association for masked face recognition. In addition, for the verification purpose, we have also pioneered a specific face-masked pedestrian dataset using real-world surveillance images. Experimentations outcomes demonstrate that their proposed strategy can faithfully resolve the face-masked pedestrian's individuality with outlying more elevated exactness than the extant disguised appearance identification strategy. Nearly, the suggested approach delivers a viable explanation for the difficulty of individuality recognition of entirely masked appearances.

In the article [14], they applied an illustration editing method highlighted to form masked appearance models with a simple image structure. As a result, massive raw data of 137,016 group, masked appearance pictures are designed and conveniently on the web. MaskedFace-Net is observed as a standard raw data for designing according to machine learning principles associated with the mask-dressing review; prominently, identifying the appearance of using the facial mask or not analyzed facial appearance pictures, the accurate or inaccurate costuming to identify concealed appearances. Then, MaskedFace- Net will be applied to magnify observation monitoring practices for several purposes, like monitoring the honor of regulations related to mask-appearing or forming group stats. Furthermore, the mentioned procedure approached the shaping of MaskedFace-Net, which has been expressed as sanctioning the formation of masked appearance illustrations by employing other masks. Thus, this procedure has been constructed for inquiring about practices and infection manners linked to the COVID-19. In addition, MaskedFace-Net has been constructed to limit the range of COVID-19 by strengthening healthiness instruction. This procedure may be a stand to educate manners and contagion phenomena

in the state of the emerging novel virus becoming a related conveyance type.
According to [15], two main propositions are used in this scenario. One is the discard occlusion-based approach, and another one is the restoration approach. Based on the images in training, the restoration approach reforms the occluded portions of the face images. So far, Face recognition is now the only steadfast AI-based security and authentication system available. Nevertheless, there remains a concern about Facial Recognition as everyone has to wear face masks. Face masks cover the significant areas of a face like the mouth and a nose which leads to failure to identify an individual with a 5% to 50% error rate of some widely used facial recognition algorithms.
In a research article [16], ResNet-50 has been used to determine the identity of the masked face. ResNet-50 is a CNN-based architecture that is fifty layers deep [5]. The author trains the ResNet50 architecture and then identifies the faces with masks developing a deep learning-based model. A transfer learning technique applies a system designed for another face recognition task to their own project. The transfer learning is implemented on ResNet-50 architecture. After applying transfer learning, the parameters are fine-tuned on the dataset of faces without masks. Then, the model is being run on the faces, which are masked and finely tweaked the system depending on the results. The goal was to identify a masked individual, followed by training the data on facial images that are not masked. Furthermore, as an alternative approach, the occluded part of the face on the masked face was cropped as it was pointed out that the model was challenged because of the occlusion on the facial region where there was a reduction in the number of attributes on the faces as well as masks which represents a non-uniform feature mapping. In the discard occlusion-based approach, the occluded parts are being declined altogether to skip a poor reconstruction procedure. After that, the feature extraction and classification process use existing parts of the face.
The article [17] used a method that combines Local Binary Pattern (LBP) characteristics and deep learning models into a united skeleton to detect the masked appearance. Their applied method uses deep models and decoctions of face points connected with LBP characteristics extracted from eyebrows and eyes for detecting a face. A practical operation is directed to verify the applied strategy. Their evaluation outcomes have illustrated that the applied approach is notably exceeded the rare case of performance face identification strategy, together with InsightFace and Dlib on the advertised Essex metadata and their self-assembled raw-data COMASK20, with has an efficiency of 87% f1-score on the owned gathered raw data from institutions COMASK20, and 98% f1-score on the declared Essex raw data. Thus, it designated the strength and appropriateness of the suggested strategy. Furthermore, they have engineered to optimize the recommended model for forthcoming works, including lessening parameters and potential waste for efficiently operating the model on movable and transportable tools that will be expanded to check the classroom at instructional systems.
The article [18] introduced a contactless distribution cabinet, incorporating a masked face recognition algorithm. The delivery cupboard is an essential carrier self-pickup tool as Covid-19 transmits through human touch. The proposed device works founded on a concentration mechanism. It is incorporated in this research to refine the masked face image recognition rate. RMFRD and SMFRD are also used in this system as experimental databases. Firstly, a provincial restrained dictionary

learning technique is implemented to differentiate the masked faces. After that, an enlarged twist methodology is employed to lessen the resolution contraction. After that, an attention mechanism is applied to extract multiple features by following the critical face image features like eyes and eyebrows. Different algorithm approaches are also compared through both the RMFRD and SMFRD databases which depict that the proposed algorithm of this study has superior accuracy in facial recognition. Proposals for supervision-based masked face identification with face difficulties in selecting appropriate detecting patterns and a lack of appropriate data sets. Existing models are insufficiently capable due to their higher percentage of false positives and lack of accuracy. To address these issues in our proposal, we will attempt to use these parameters in our model. The above deliberation shows that researchers used different CNN-based methods and other mechanisms for detecting facial identification. Moreover, there are significant differences between those methods and mechanisms. Nevertheless, we have to select the idiosyncratic and most acceptable approach from the above articles. For instance, Inception ResNet-V1 architecture is more advanced than FaceNet architecture in terms of masked face detection. Furthermore, OpenCV and Dlib is more robust and triumphant analytical technique and broadly used algorithm, and ResNet50 architecture has better time and memory performance than other CNN models like VGGNet19 and DenseNet121.

Chapter 3

Background Study

3.1 Face Verification vs Face Recognition

3.1.1 Verification - Is this the same person?

Face verification is quite simple. By taking FaceID, for example, which will be used to unlock phone or at the airport when scanning passport and verifying if it is really that person. It works in 2 steps:

- Taking image or the ID of a person as input
- Verify if the output is the same as the claimed person.

Figure 3.1: Face Verification

Figure 3.1 defines it is a 1:1 problem. It is expected to have a high accuracy of the face verification system nearly 99% - so that it can be further used into the face recognition system.

3.1.2 Face Recognition - Who is this person?

Face recognition is much harder than face verification. It is used mainly for attendance system in offices, or when Facebook automatically tags friend. The process is as such:

- Having a database of K persons.
- Taking the image of a person.
- Give the ID as output if the image is any of the K persons or if it is not.

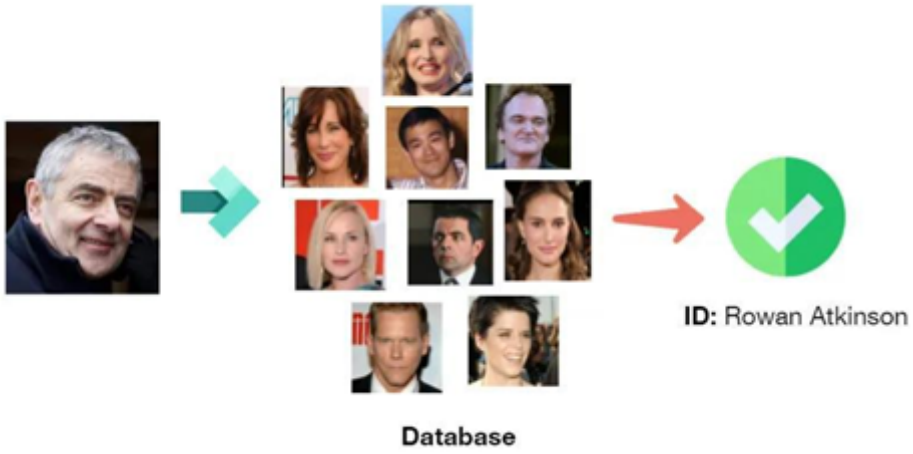


Figure 3.2: Face Recognition

Figure 3.2 introduced Face recognition, which is a 1:K problem where K is the number of persons in database.

3.2 The One Shot Learning Dilemma

The problem with face recognition is that a particular person is recognized from only one image. Since there will be only one picture of each individuals, the system should be able to recognize the person again.
If there is 100 individuals, then one simple solution would be to take that one image and feed it into a CNN and output a Softmax unit with 101 outputs (100 outputs are for the individuals and the one left is to indicate none). However, this would not work well for two reasons:

- It will not be possible to train a robust neural network.
- If another individuals were added, the output will needed to be increased and retrain the Conv-Net.

Instead, the neural network should learn a similarity function.

3.3 Similarity Function

The similarity function takes as input two images and output the degree of difference between the two images - *d(img1,img2)*

- For two images of the same person, *d(img1,img2)* should be small.
- For two images of different person, *d(img1,img2)* should be big.

So this is the way to address the face verification problem:

- If $d(img1,img2) \leq \tau$, it is predicted as same.
- If $d(img1,img2) > \tau$, it is predicted as different, where τ is a threshold.

Given a new image, the function d is used to compare against the images in the database. If the image pairs are different, then it would be a large number, and if they are the same, then it would be a small enough number which will be less than the τ threshold.

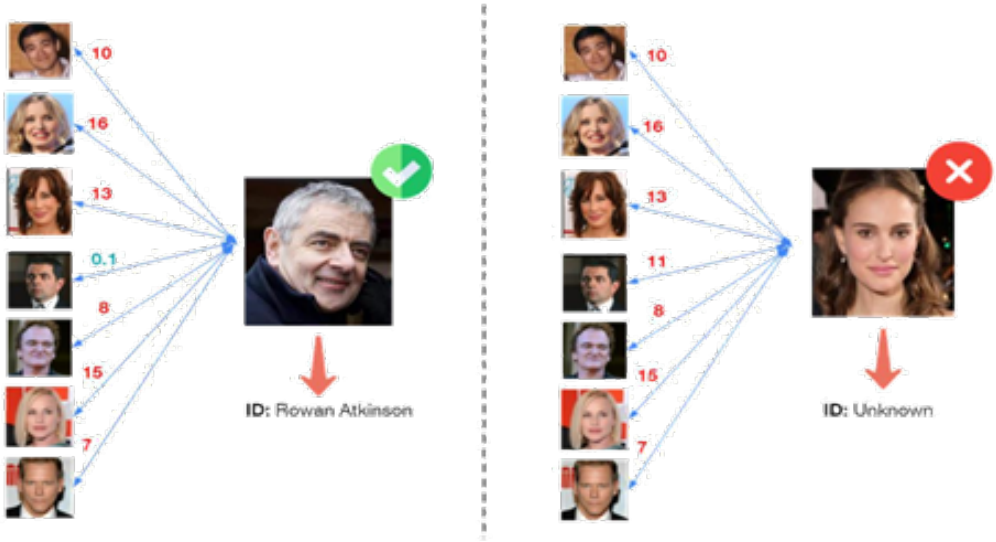


Figure 3.3: Similarity Function process

In Figure 3.3, if someone is not in the database, when pairwise comparison will be done and the function d will be computed, then a very large numbers for all the pairs will be generated as shown. The d function solves the one shot learning problem whereby if someone new joins the team then adding that new person's image will only be needed to the database and it would work just fine.

3.4 Face Detection with SSD

The job of detecting numerous things inside an image and creating a bounding box around each of those objects is known as "object detection." Initially, researchers developed R-CNN for object detection, localization, and classification. The output is a bounding box surrounding the object detected with the classification result of

the object. With time, we improved the R -CNN network and came up with Fast R-CNN and Faster R-CNN. However, one major drawback of the network was that the inference time was too long for real-time object detection. New architectures, such as YOLO and the ones described below, are better suited for real-time object detection. There are several methods for face detection:

- SSD
- MTCNN
- Dlib
- OpenCV

Figure 3.4: Crop with different margin

The goal of this research is to use a face detection algorithm to detect faces and crop it with margin 20 or 40 as shown in Figure 3.4.
While MTCNN is more widely used, SSD performs faster inference, but has low accuracy. SSD makes use of layers with a lower resolution in order to identify things on a greater scale. It does this by removing the need for the regional proposal network to be used, which speeds up the process.There are three primary components that make up SSD's architecture:

- Base network - VGG-16
- Extra feature layers
- Prediction layers

3.5 VGG-16

The ImageNet dataset, which contains over 14 million photos and 1000 different categories, is used to train it. The fact that it consists of 16 different layers is where the term VGG-16 originates from. Convolutional layers, Max Pooling layers, Activation layers, and Fully Connected (fc) layers are some of the layers that are included.

Figure 3.5 represents VGG-16 architecture. The total number of layers is 21, yet there are only 16 weight layers despite the fact that there are 13 convolutional layers,

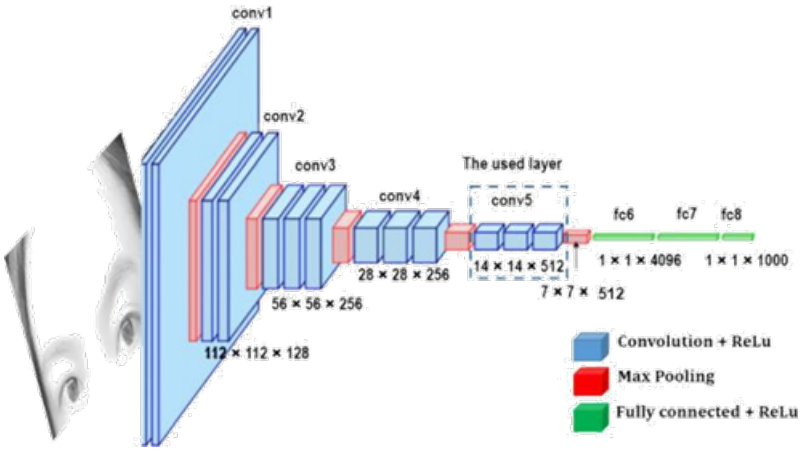


Figure 3.5: VGG-16 architecture

5 Max Pooling layers, and 3 Dense layers. In this study, the VGG-16 was chosen as the base network, and the only feature mappings (FMs) that were taken into consideration were those at the very last convolutional layer, which is also referred to as channels. This layer is being utilized as a feature extractor, and its output will be used for the quantization in the stage that comes after this one.

3.6 FaceNet

FaceNet is a deep neural network used for extracting features from an image of a person's face. It was proposed in 2015 by three Google Researchers, Florian Schroff, Dmitry Kalenichenko, and James Philbin, in the paper titled FaceNet: A Unified Embedding for Face Recognition and Clustering. It achieved state-of-the-art results in the many benchmark face recognition dataset such as Labeled Faces in the Wild (LFW) and Youtube Face Database. FaceNet is able to learn a mapping from face pictures to a compact Euclidean Space, in which the distances directly correlate to a scale of face similarities. Then it uses the Triplet Loss function to train this architecture. FaceNet is capable of achieving state-of-the-art performance (record 99.63 percent accuracy on LFW, 95.12 percent on Youtube Faces DB) while only requiring 128 bytes per face to store each face. FaceNet seeks an image embedding f(x) into feature space R^d (where d is typically 128) such that the squared L2 distance between all face photos of the same identity is modest, while the range between a set of face images from different identities is significant.

Figure 3.6: FaceNet Architecture Block Diagram

3.7 Triplet Loss Function

Unlike other losses, which urge all faces of the same identity to converge on a single point in R^d, the triplet loss also seeks to impose a buffer between each pair of faces from one person (anchor and positive) and all other people's faces. Other identities are discriminated against by this margin.

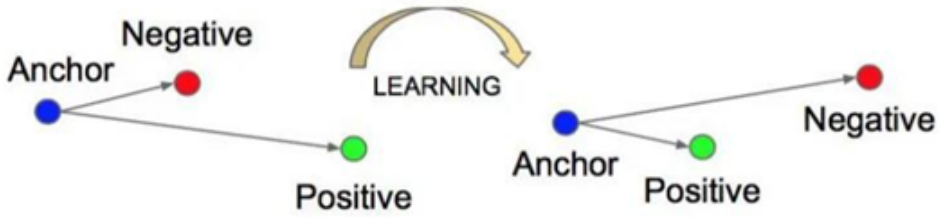


Figure 3.7: Triplet loss training

Here in Figure 3.7 in order to train a model to classify, triplet loss to minimize the distance between images of the same person can be used and also the distance between images of different people can be maximized. In FaceNet, all the images are processed and encoded into a 128-dimensional vector which contains the coordinates. Using the Triplet Loss function, the images of the same person and images of different people only with the distance can be identified. If the distance between two images is low, then they are images of the same person. On the other hand, if the distance is high, then the images belong to different people.

3.8 Inception Network

When designing a layer for a convNet, the type of filters should be picked: 1x1, 3x3 or 5x5 or even the type of pooling. To get rid of this conundrum, the inception layer allows us to implement them all. For example, our image will be of the same dimension but the target in the image may be of different sizes, i.e, a person may stand far from the camera or one may be close to it. Having different kernel size allow us to extract features of different size. By understanding the Naive version of the Inception model, different types of kernel on an input is applied and the output is concatenated. The idea is instead of selecting the filter sizes, using them all and concatenate their output and let the NN learn whichever combination of filter sizes it wants. However, the problem with this method is the computational cost. Deep Convolutional Networks are computationally expensive. Also, very deep networks are susceptible to over fitting - it is hard to pass gradient updates through the entire network.

3.9 Network in Network

The computational cost of the 5x5 filters of the 28x28x192 input volume is 120M multiplies to perform. It is important to remember that this is only for the 5x5 filter and computation is needed for the other 2 filters and pooling layer. A solution to this is to implement a 1x1 convolution before the 5x5 filter that will output the same 28x28x32 volume but will reduce the number of multiplies by one-tenth.

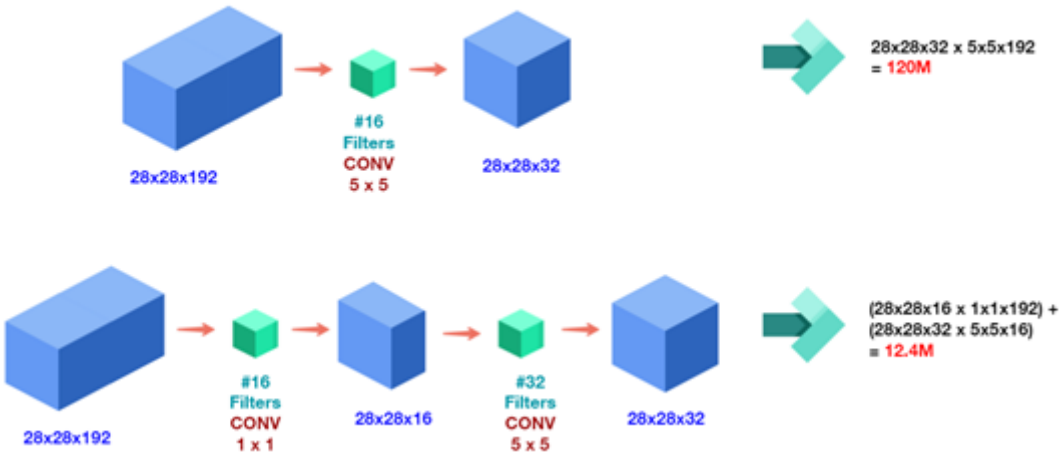


Figure 3.8: 1x1 convolution, Network in network

Here in Figure 3.8, A 1x1 convolution also called a Network in network will take the element-wise product between the 192 numbers (example above) in the input and the 192 numbers in the filter and apply a relu activation function and output a single number. A number of filters will be generated. So the output will be HxWx#filters. To reduce the height and width of an input pooling can be done. However, to reduce the number of channels of an input (192), a 1x1x#channels filter can be used with the numbers of filters equal to the number of channels of output needed. In the example above in the middle section, the channel is needed to be 16, so 16 filters will be used. A bottleneck should be created by shrinking the number of channels from 192 to 16 and then again increased it to 32. This allows to diminish dramatically the computational cost which is now about 12.4 M multiplies. The 1x1 convolution is an important building block in the inception network which allows to go deeper into the network by maintaining the computational cost and learning more features.

3.10 Inception with Dimension Reduction

To reduce the computational cost the architecture should be modified and 1x1 convolution should be added to it. As shown above, the 1x1 filters will allow to have fewer weights therefore fewer calculations and therefore faster inference.

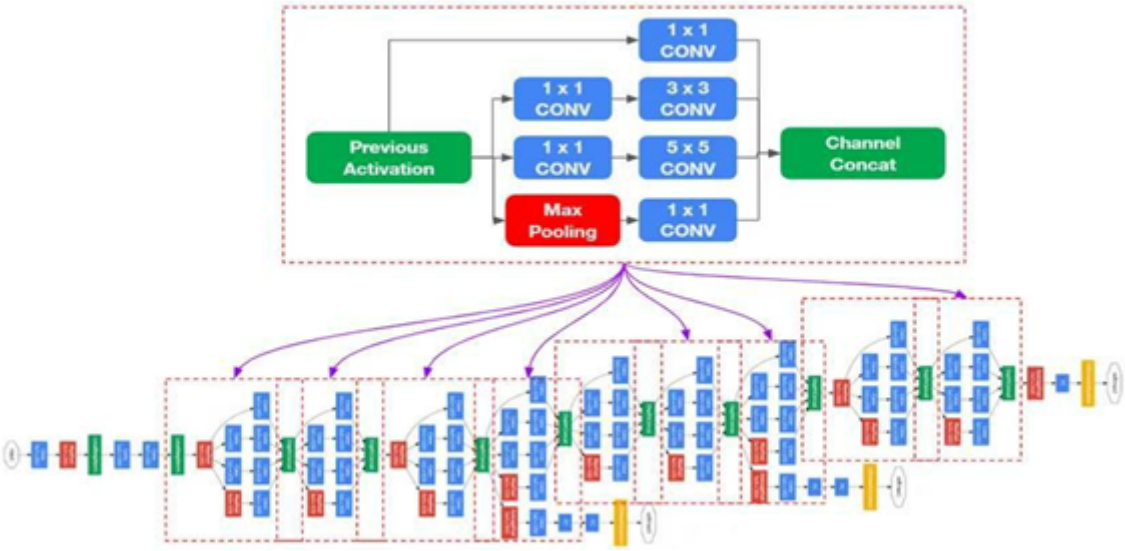


Figure 3.9: Inception network with some additional max pooling

In Figure 3.9 there are 9 of the inception block concatenate to each other with some additional max pooling to change the dimension. It should be noted that the last layer is a fully- connected layer followed by a softmax layer to make predictions but there is also two side branches coming from the hidden layers trying to make predictions with a softmax output. This helps to ensure that the features computed in the hidden layers are also good to make accurate predictions and this helps the network from over fitting.

3.11 Inception-Resnet V1 Network

A hybrid inception module was suggested, inspired by the ResNet's performance. Inception ResNet is divided into two versions: v1 and v2. The computational cost of Inception-ResNet v1 is equivalent to that of Inception v3, while the computational cost of Inception-ResNet v2 is similar to that of Inception v4. Below are some features of the Inception ResNet architecture:

- It is necessary for both the input and the output after convolution to have the same dimensions for residual addition to work. Therefore, in order to make the depth sizes consistent, 1x1 convolutions is applied after the first convolutions (Depth is increased after convolution).
- The pooling mechanism that took place inside the major inception modules was changed to focus on the residual connections instead.
- If the number of filters surpassed 1000 in a network that had residual units further in the design, the network would "die". As a result, in order to make the system more stable, the scientists scaled the residual activations by a number that was somewhere between 0.1 and 0.3.

It was found that Inception-ResNet models were able to achieve higher accuracies at a lower epoch. This network will be used to train and test the Face Recognition with Masks model.

3.12 Object Localization and Object Detection

3.12.1 How dows it work?

To perform object localization whereby if a person's face detection in a picture is needed, it should be known where in the picture is the face located. There might be other objects in the picture, for example a car, this also be needed into consideration. Now the CNN softmax output will not just contain the object class label but also the parameters below:

$$y = \begin{bmatrix} p \\ c_1 \\ c_2 \\ x \\ y \\ w \\ h \end{bmatrix} \tag{3.1}$$

here,

- p: "0" if no object and "1" if object
- c1: "1" if face, "0" otherwise
- c2: "1" if car, "0" otherwise
- x: x-position - center point of object
- y: y-position - center point of object
- w: width of bounding box
- h: height of bounding box

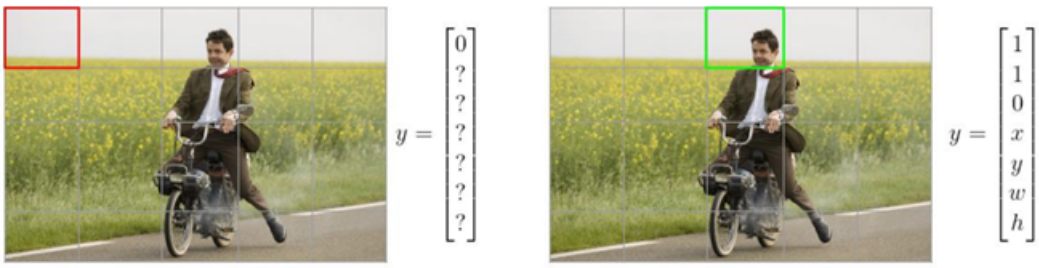


Figure 3.10: Divide pictures into a 5x5 grid cells and predict value of y predict in each individual cell

Here in Figure 3.10, For the first grid cell which does not contain an object, p = 0 is the first parameter in y value and for the rest, "?" is used to represent them. For the third grid cell, an object and a face is detected, so out p = 1 and c1 = 1, and the x,y,w and h represent values for the bounding box. During training, network output will be similar vectors. SSD does not employ a pre-defined area proposal network. Instead, it uses modest convolution filters to calculate both the location

and class scores simultaneously. In order to produce predictions, SSD first extracts the feature maps from each cell, and the n applies 3 x 3 convolution filters on each of those maps.

3.12.2 How to know the bounding box is correct?

IoU(Intersection over Union) is used to measure the overlap between two bounding boxes.

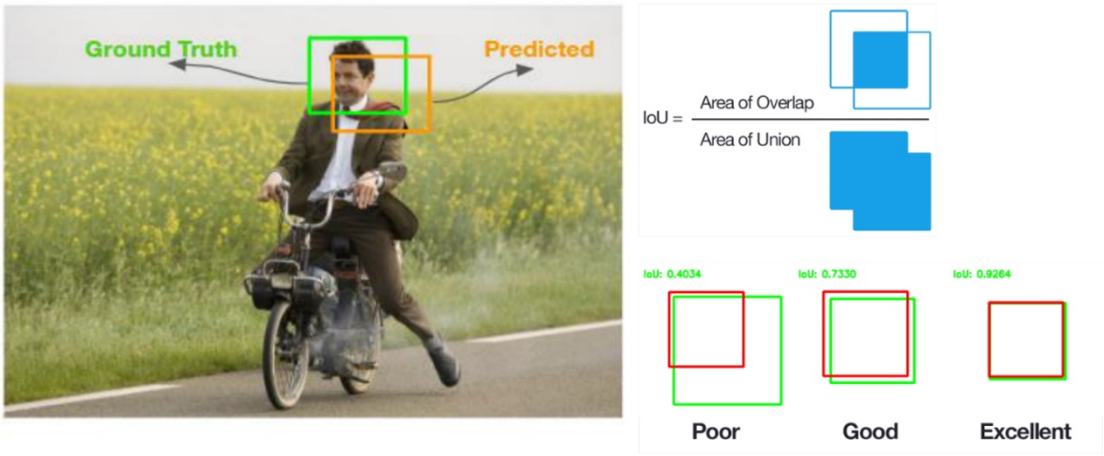


Figure 3.11: Intersection over Union

Here in Figure 3.11, green border illustrates the Ground Truth and orange border illustrates the prediction. Normally if the IoU is greater than or equal to 0.5 it is deemed to be a correct prediction. But to be more stringent the threshold can be increased where 1 is the maximum value.

3.12.3 Anchor Boxes

Problem 1: It is not possible for one object to be strictly within one grid cell. And when it is not, how to determine which cell to associate to the object.
Problem 2: Each of the grid cells can detect only one object. But there might be one grid cell containing more than one object. How to handle multiple center points?

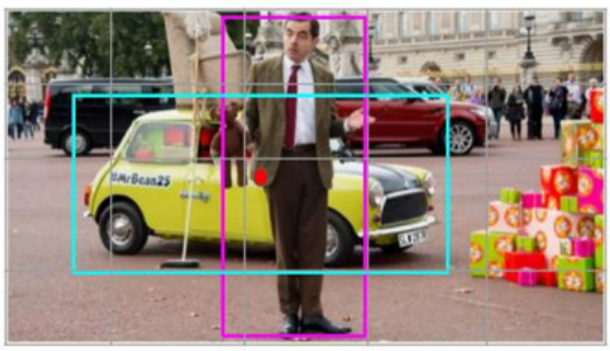


Figure 3.12: Anchor boxes for prediction

In Figure 3.12, both objects have their counterpoint in the same cell. So setting a tall anchor box that can be used to predict a standing person and a wide anchor box can be used to predict a car. These anchor boxes are used in each of the grid cell and output one vector y for every anchor box. The solution for problem 1 is to associate the cell which contains the center point of the bounding box of the object. Also, the solution of problem 2 is, a bigger grid - 19x19 - instead of a 5x5 can be used which reduces this problem. Also, to predefined anchor boxes and associate prediction with the anchor boxes is needed to be done. Previously, each object was assigned to a grid cell that contained that object's midpoint. Now, each object is assigned to a grid cell which contains that object's midpoint and anchor box for the grid cell with the highest IoU (similar shape).

3.13 Non-Maximal Suppression (NMS)

SSD contains 8732 default boxes. During the process of inference, there are 8732 boxes allocated to each class (because the output is a confidence score for each box). The vast majority of these checkboxes are negative, and even among the positive ones, there is a high chance that several of the boxes will overlap. Non-Maximal Suppression, abbreviated as NMS, is a technique that eliminates overlapping box instances per class.

Figure 3.13: Output of Non-Max Suppression

In Figure 3.13, it illustrates the NMS technique. It works such as:

- Arrange the checkmarks in descending order of confidence.
- Select the category that has the highest level of confidence.
- Get rid of all the other projected boxes if their Jaccard overlap is more than the NMS threshold (0.45 here).
- Continue with step 3 until all of the boxes have been covered.

In short, the network is quite sensitive to the default boxes, and it is critical to choose the default boxes in accordance with the dataset on which it will be employed. Because early layers, which are responsible for detecting tiny things and have smaller receptive fields, are too shallow, SSD does not perform well with items that are on the smaller side.

Chapter 4

Methodology & Explanation

Before we explain our proposed methodology, we want to explain about dataset. These following datasets from online source and an amount of raw data are used by us:

1. CASIA-Webface Dataset to train our model: This dataset contains 494,414 images of 10,575 people. This dataset does not provide any bounding boxes for faces or any other annotations.

2. LFW Dataset for evaluation of Face Recognition without Masks: This dataset contains more than 13,233 images of faces collected from the web of 5,749 people. Each face has been labeled with the name of the person pictured. 1,680 of the people pictured have two or more distinct photos in the data set.

3. 1% of the MS-Celeb-1M dataset for evaluation of Face Recognition with Masks: This dataset has 8,456,240 images of 99,892 celebrities.

4. Some amount of data of our friends that were collected form different social handle.

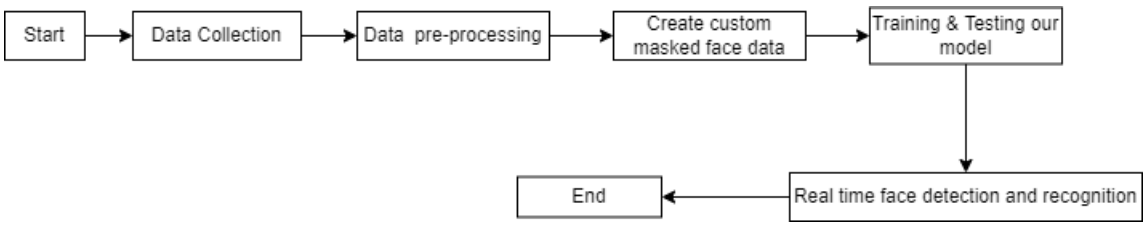


Figure 4.1: Work Plan: A complete overview of our face detection and recognition process

4.1 Data Pre-processing

Our data pre-processing contains data cropping, data cleaning, data augmentation and data embedding.

4.1.1 Data Cropping

We have used our trained facemask detection algorithm to crop the pictures. The bounding box is adjusted to crop the images with the help of SSD FaceNet.

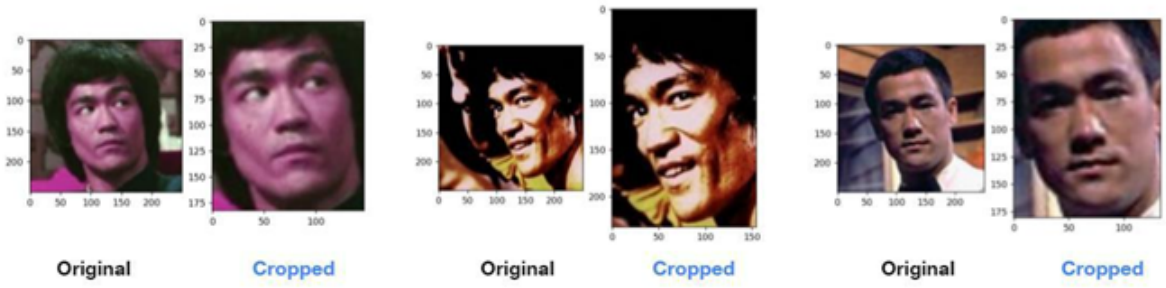


Figure 4.2: Cropped Image from the original one

Figure 4.2 is representation of the cropped images. To perform it, the .pb file is loaded with a default margin of 40 and GPU-ratio of 0.1 . Then, we have restored the model to get the nodes and get the input shape which is used to resize the images. In the inference function used is sess.run to get the detection boxes, the detection scores. Then we have decoded the bounding boxes and do the non-max suppression. In order to draw the bounding box the (x_{min},y_{min}) and (x_{max},y_{max}) coordinates are needed. Our bounding boxes are unit coordinates in the range [0,1]. We had to make them to real sizes. Then, the width of the bounding box is calculated using (x_{max} - x_{min}) and height is calculated using (y_{max} - y_{min}). Our original image is 250x250 so the cropped face image should be over 100x100. This enables to detect faces which is well aligned (the main person in the image). Two thresholds is used for the width and height. In an if condition it is checked if the width of our bbox is more than the width_threshold and the height is more than the height_threshold then the height of the cropped image is bbox[1] : bbox[1] + bbox[3] and width is bbox[0] : bbox[0] + bbox[2]. Then the file is saved and the images are being displayed.

4.1.2 Data Cleaning

After cropping the pictures, now performing the data cleaning process is needed to remove the mislabeled images.

Figure 4.3: Finding the mislabeled images

While checking the folders and it has been seen that folder 057 got one mislabeled image as shown in Figure 4.3. This signifies that the CASIA dataset is not cleaned and there may be other instances of mislabeled images. To check each of the 10,575 folders individually we need an algorithm that will do this work. We can set a process of removing mislabeled images using the distance function d described in Siamese Network.

Siamese Network

A Siamese Neural Network is a kind of neural network that is created by applying two similar convolutional neural networks to two distinct inputs and then analyzing the results of both networks.

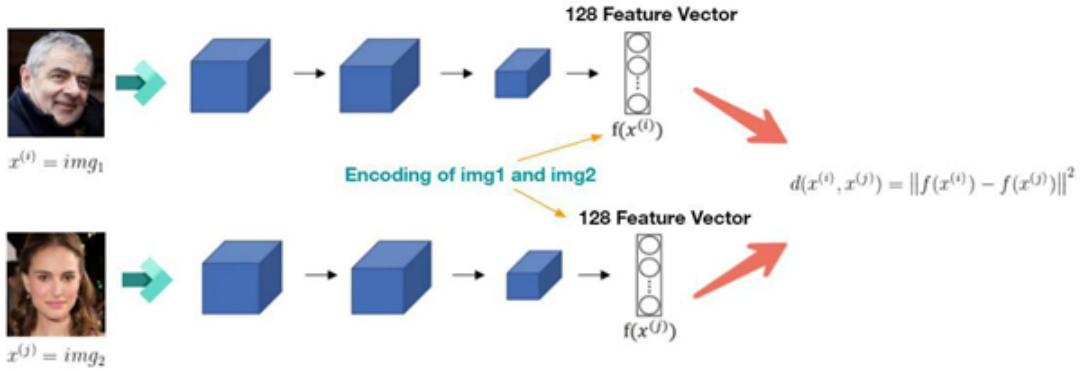


Figure 4.4: Encoding between two images and calculating distinct two inputs

In Figure 4.4, after running a series of convolutions, pooling, and fully connected layers on an image of a human, we are left with a feature vector that has 128 individual values. These 128 numbers are represented by $f(x^{(i)})$ and are called the encoding or embedding of the image where $img_{(i)} = x^{(i)}$. Then, a face recognition system is built to have a second picture feed into that same CNN and compare their two 128 feature vectors. Then we need to define the function d which computes the norm of the difference between the two encodings:

$$d(x(i), x(j)) = ||f(x^{(i)}) - f(x^{(j)})||^2 \tag{4.1}$$

In short,

- Our neural network defines a 128-dimensional encoding of an image.
- The parameters are acquired in such a way that if two photographs are the same, then there should be little difference in their encodings.
- For two different images, distance should be large.

While changing the parameters of the different layers of the NN, it ends up with different encodings. But we want to learn a specific set of parameters such that the above two conditions are met.
To remove the mislabeled images, the distance function d is used which is shown in Figure 4.5.

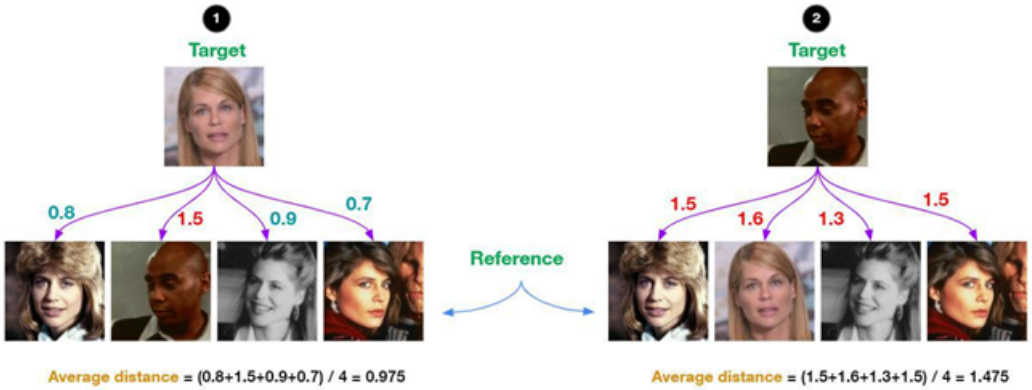


Figure 4.5: Calculating the average distances between the target image and the reference image to find the mislabeled images

1. In a subfolder in the main directory, we have selected images one by one as the target image and the other images become the reference images.
2. The average distance between the target image and the reference image is calculated.
3. The average distance shows that, when a correct image is selected as the target image, is not much as compared to when the mislabeled image is selected as the target image. Also, there might have been more than one mislabeled image in a folder. That is the reason why we make each image the target image and calculate the average distance. Note: The distance between a target image and itself is zero.
4. The average distances to a threshold is compared.
5. The target image(mislabeled image) is removed when its average distance exceeds the threshold.

We have also used the pre-trained weights of Inception Resnet V1 trained on the VGGFace dataset and have an accuracy of 0.9965 on the LFW dataset. We start by restoring the .pb file and creating a function *img_removal_by_embed* to do the following processes:

1. Collecting all folders from the root directory.
2. Initializing our model by restoring from pb.
3. Setting the method to calculate the distance d by using default tf_graph and configuring GPU settings that help to initialize the TensorFlow global variables.
4. Process each folder and create subfolders to move the mislabeled images.
5. Calculate the embedding.
6. Calculate the average distance by using the embedding.

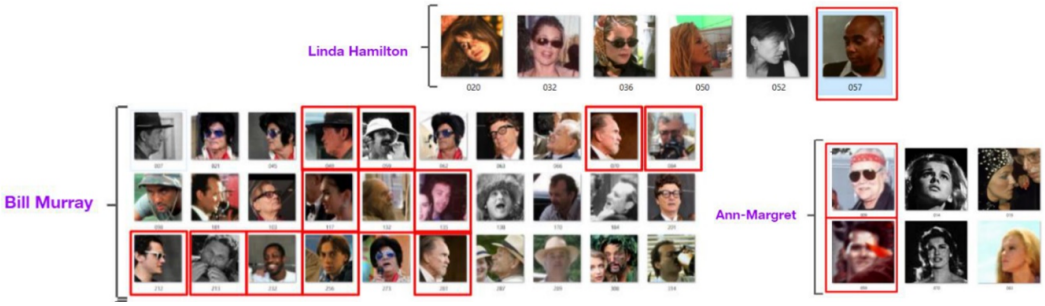


Figure 4.6: Finding mislabeled images in the dataset

7. Remove the mislabeled images if the average distance is greater than the threshold (1.25).

We got 3981 folders after running and checking the folders which had 20079 wrong images in total as shown in Figure 4.6. In the folders, we see that our algorithm correctly identified the wrong person in Linda Hamilton's folder, and in Bill Murray's folder, we had more than one mislabeled image. However, the algorithm also removed the images of the correct label. Mainly because the images were blurred or fuzzy, or the subject had sunglasses in them or there were pictures when the person was too young or too old. Nevertheless, data cleaning will now allow our NN to train on a more accurate dataset to make better predictions.

4.1.3 Data Embedding

After finishing the data cleaning, now performing data embedding is needed. Using a vector of 128 numbers, our AI will try to compress all the necessary features of a face into this space. The process of transforming high-dimensional input, such as photographs, into such low-dimensional forms (embedding) is what will allow us to give each face an ID. As such, the embedding of faces with similar characteristics is comparable - a person can have only one ID.

How to know the correct embedding dimension?

FaceNet experimented with different embedding dimensionalities and 128 remains the best performing one. In this work our embedding length will be a constant 128 feature vector.

Dimensions	Values
64	86.8% ± 1.7
128	87.9% ± 1.9
256	86.7% ± 1.9
512	85.6% ± 2.0

Table 4.1: Different accuracy on different embedding lengths

4.1.4 Data Augmentation

With a first initial training, the accuracy of the model was moderate. One possible way to increase the accuracy would be to have more data. Instead of finding new images, we have created these images using Data Augmentation techniques. We have used five data augmentation techniques namely:

1. Random Crop
2. Random Noise
3. Random Rotation
4. Random Horizontal Flip
5. Random Brightness Augmentation

Instead of using TensorFlow's Data Augmentation API, we have created the scripts and generated the images using OpenCV packages and NumPy.

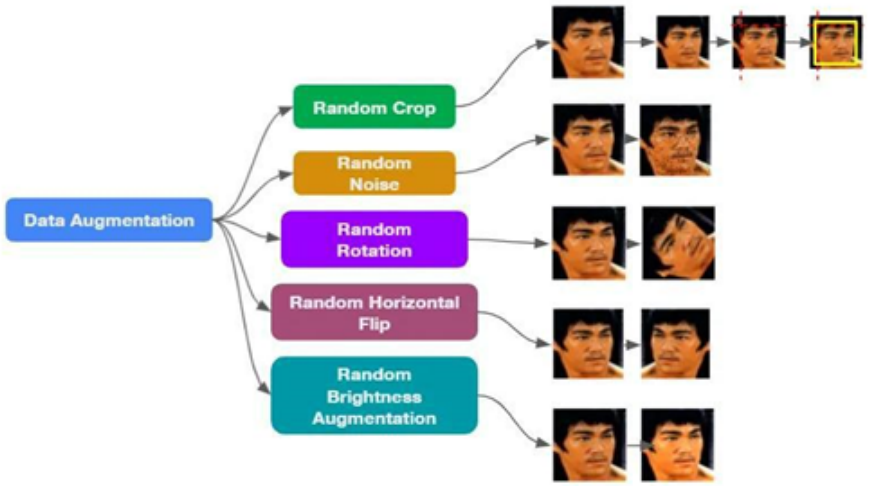


Figure 4.7: Data Augmentation Process

Figure 4.7 refers to our data augmentation techniques which are following:

Random Crop

We want to create a script to crop randomly our images to become 150x150. We have created a frame of this size in a small 10x10 square on the top left. Then, we generated random points and used this as the first x-y values of the frame. Then, we position the frame and crop the image.

Random Noise

For the random noise, a mask is created which is a NumPy array of the same size as the image with only one channel. A uniformly-distributed array of random numbers is created. If the number of pixels exceeds a threshold (240), it is set to 255, otherwise, it is set to 0. If the mask pixel value is 255, we use cv2.bitwise_and() operation else we pass.

Random Rotation

The angle range is set from -60 to 60 degrees. Then, we define our center point of rotation and use cv2.warpAffine to get the result.

Random Horizontal Flip

We don't want to flip our image vertically as it is not expected to see someone upside down when doing inference. So only Flip_type = 1 is used for the horizontal flip.

Random Brightness Augmentation

The mean brightness is calculated and set to a 30% variation range: 0.3xnp.mean (img). Then, we find a number in the range as the new brightness and normalize the image by dividing by the average value. Here, np.clip() is used to apply the brightness. Since our data has now been doubled, we divide the batch size by 2 in order to have the same number of data in one batch as before. We also reverse the paths of the sub-folders in the directories of the CASIA mask and without mask such that in half a batch, so that we have the original images and in the other half we have the data augmented images. Then, the model is re-trained.

4.1.5 Custom Face Mask Dataset

Our end goal is to be able to recognize faces with masks. The CASIA dataset already has half a million pictures of faces and By using the Inception Resnet V1 model we can create a face recognition model. What we want to do now is have the same CASIA dataset with the same folders and same pictures but with the persons wearing a mask. But we don't have such a dataset. So creating one is needed. We want to show our AI the picture of a person without a mask, then a picture of the same person with a mask and tell him that it is the same person.

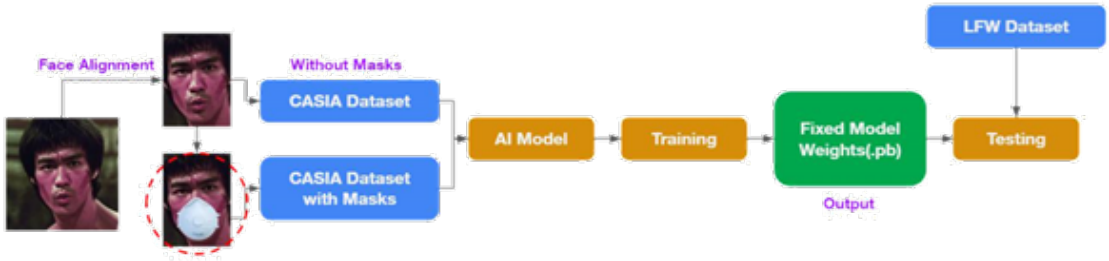


Figure 4.8: Creating masked face dataset using CASIA dataset to train and test through LFW dataset

In order to achieve the process in Figure 4.8, our mask should be in png format. PNG format has 4 channels. The fourth channel is used to describe transparency. We need to use the Dlib library which is pre-trained to recognize 68 landmark points that cover the jaw, chin, eyebrows, nose, eyes, and lips of a face.

Figure 4.9 shows the landmarks numbers 48 to 68 for the mouth. According to the figure, we start by creating a function detect_mouth which will be used to read the face landmarks from 48 to 68 and calculate the coordinates. In another

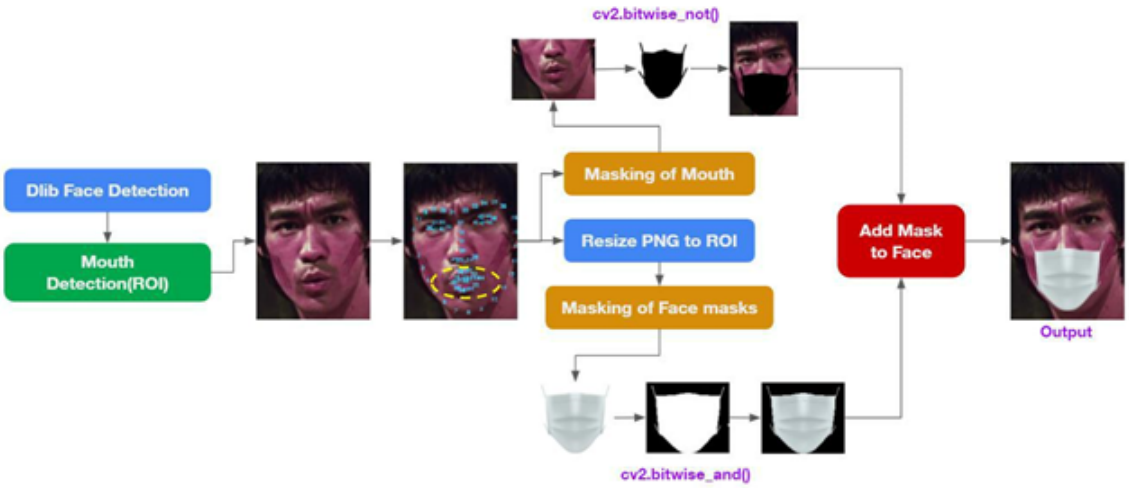


Figure 4.9: Adding mask to face by using Dlib

mask_wearing function, we first process the folders and create directories for each folder in the main folder. Then we randomly select a PNG mask image from the folder. Then, the image is read with cv2.IMREAD_UNCHANGED to make sure the image has 4 channels. We resize it based on our mouth detection coordinates found before. We use cv2.threshold to make the values of the image of the mask 0(black) or 255(white), i.e, we have the image of a white mask with black background. We use cv2.bitwise_and to create the mask of the face mask. We declare the coordinates of our Region of Interest (ROI) from the mouth detection values. We create an inverted mask with cv2.bitwise_not of the face mask and then use cv2.bitwise_and to mask the face mask onto the person's face. Then, two images are added: face mask and face.

Figure 4.10 shows that, the face masks have successfully been added to the images. Although there is some side face images, we cannot really modify the mask to fit in each image correspondingly. So it is a satisfying result.

4.2 Face Recognition using Facenet

FaceNet is used to implement our Face Recognition with mask model. We have already done Face Detection to perform Face Alignment whereby we cropped the images with a margin value of 40. We will not build an Anti-Spoofing or Face Liveness Detection for now. We will train our model for the first time and then use various Data Augmentation techniques to improve the training accuracy. As FaceNet is able to learn a mapping from face pictures to a compact Euclidean Space, in which the distances directly correlate to a scale of face similarities. Then it uses the Triplet Loss function to train this architecture. FaceNet is capable of achieving state-of-the-art performance while only requiring 128 bytes per face to store each face.

Figure 4.10: Custom masked face dataset

4.2.1 Triplet Loss Function

During the process of Triplet Loss, we are going to be looking at three separate pictures at the same time: an anchor, a positive image (which is an image that is comparable to the anchor image), and a negative image (one who is different from the anchor image). We want there to be as little space as possible between the anchor and the positive picture, and as much space as possible between the anchor and the negative image. For a robust face recognition, we want the following :

$$||f(A) - f(P)||^2 \leq ||f(A) - f(N)||^2 \tag{4.2}$$

where,

$$d(A, P) = ||f(A) - f(P)||^2$$

and,

$$d(A, N) = ||f(A) - f(N)||^2$$

The equation can also be written as,

$$||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 \leq 0 \tag{4.3}$$

Here, The anchor is denoted by the letter A, positive as P and negative as N.

To make sure the neural network does not output zero for all the encodings, it does not make all of the encodings equivalent. We had to modify the above equation such that the difference between d(A,P) and d(A,N) should be 0 – α where α is called a margin. Finally, the equation becomes:

$$||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha \leq 0 \tag{4.4}$$

For example, if we have d(A,N) = 0.50 and d(A,P) = 0.49, then the two values are too close to each other and it is not good enough. We want d(A,N) to be much bigger than d(A,P) like 0.7 instead of 0.50. To achieve this gap of 0.2, we have introduced a margin which helps push d(A,N) up or push d(A,P) down to achieve better results.

To define our loss function on a single triplet we need 3 images: A, P and N:

$$L(A, P, N) = max(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha, 0) \tag{4.5}$$

We take the max of the loss because as long as,

$$||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha \leq 0, \, the \, loss = 0$$

However if,

$$||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha > 0,$$

then, the loss,

$$||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha$$

We will have a positive loss. To define our cost function :

$$J = \sum_{i=1}^{m} = L(A^i, P^i, N^i)$$

We need at least more than 1 picture of a person as we need a pair of A and P in order to train our NN.
Figure 4.11 illustrates,

- One of the anchor images is chosen at random (orange border).
- We randomly select an additional photograph of the same individual who serves as the positive anchor photograph (green border).
- As the negative anchor image, we will use a picture of a different individual who will be chosen at random (red border).
- We put our model through its training and make adjustments to the parameters such that the positive picture is located as near to the anchor as possible and the negative image is located as far away as possible.
- We repeat that process above so that all images of the same person are close to each other and further from the others.

One of the problems when we choose A,P and N randomly is that the condition $||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha \leq 0$ is easily satisfied and the NN will not learn much from it.
To do this, we choose triplets that are challenging to practice on. In order to satisfy this condition: $d(A, P) + \alpha \leq d(A, N)$, *we want* $d(A, P) \approx d(A, N)$.

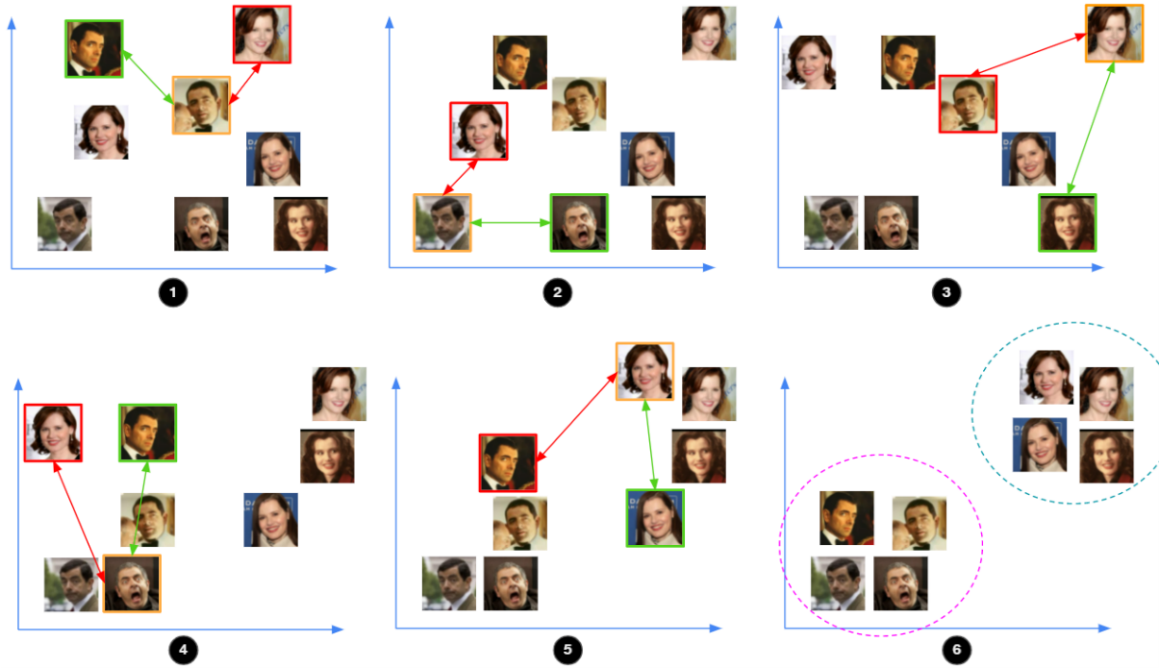


Figure 4.11: The anchor (in orange) pulls images of the same person closer and pushes images of a different person further away

Now the NN will try hard to push $d(A, N)$ and push $d(A, P)$ up so that there is at least a margin between the two components. Thus, it is important to understand that it is only by choosing hard triplets that our gradient descent will really do some learning of the similarity and differences in the images.
After receiving training, our artificial intelligence will be able to recognize a person on an unviewed picture by computing the picture's embedding and then using this embedding to compute the distances to photos of known persons. When the face embedding of this picture is sufficiently similar to the embedding of person A (Rowan Atkinson), we assert that this image includes the face of person A. [Figure 4.12] In our dataset, we have more than 2000 individuals and more than 20,000 images (10 pictures of each individual), then we take these 20,000 pictures and generate triplets of (A, P, N) and then train our learning algorithm by using gradient descent to minimize the cost function defined above. This will have the effect of back-propagating to all of the parameters in the NN in order to learn an encoding such that $d(x(i), x(j))$ is small when comparing photographs of the same person to images of other people and large when comparing images of different people.

Face Verification with Binary Classification

In addition to the Triplet Loss Function, we also have the option of using the Siamese Network and asking it to calculate the 128 dimensional embedding. This information can be sent to a logistic regression unit, to use to create predictions.

- Same person: $\hat{y} = 1$
- Different person: $\hat{y} = 0$

The output $\hat{y}$ will be a sigmoid function applied to the difference between the two sets of encodings. The formula below computes the element-wise difference in absolute

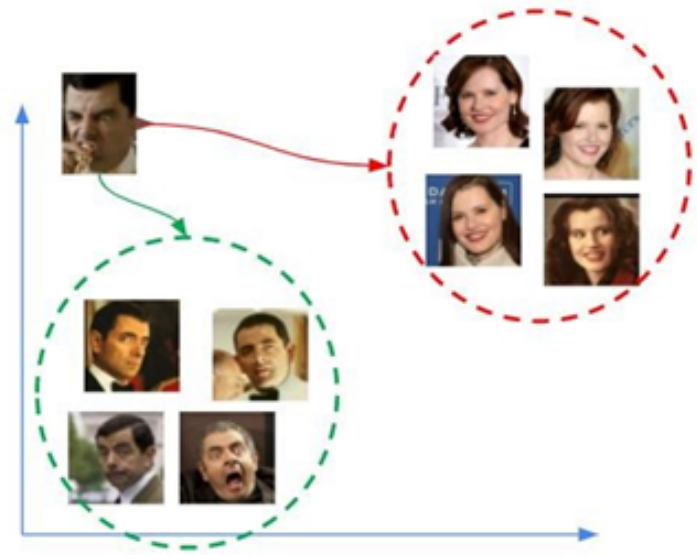


Figure 4.12: Face Recognition from dataset

values between the two encodings:

$$\hat{y} = \sigma(\sum_{k=1}^{128} w_i \left| f(x)_k^i - f(x)_k^j \right| + b)$$

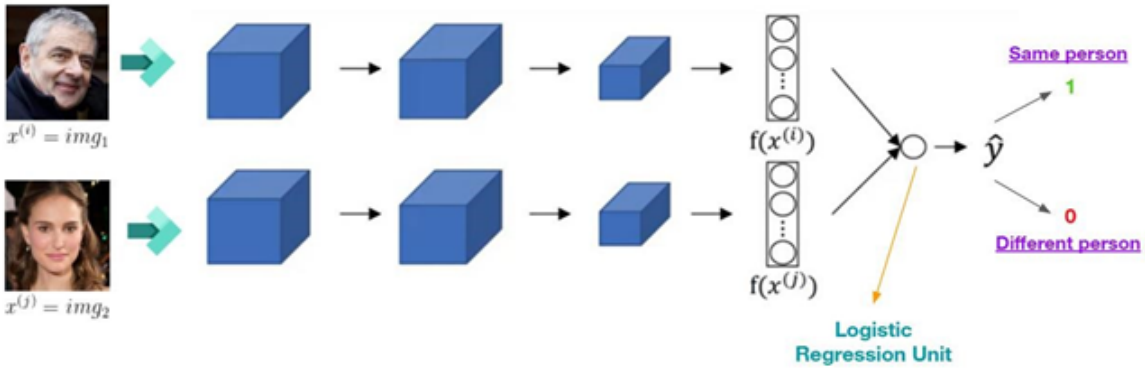


Figure 4.13: Face verification with Binary classification

In Figure 4.13, we have created a training set of pairs of images where target label = 1 of same person and target label = 0 of different person. As FaceNet uses Zeiler & Fergus architecture and GoogLeNet style Inception model as its underlying architecture. Since we will be using the Inception ResNet V1 Network, we start by ResNet Network.

4.2.2 Resnet Network

In Figure 4.14, a 56-layer CNN gives more error rate on both training and testing datasets than a 20-layer CNN architecture. This happens due to the phenomenon

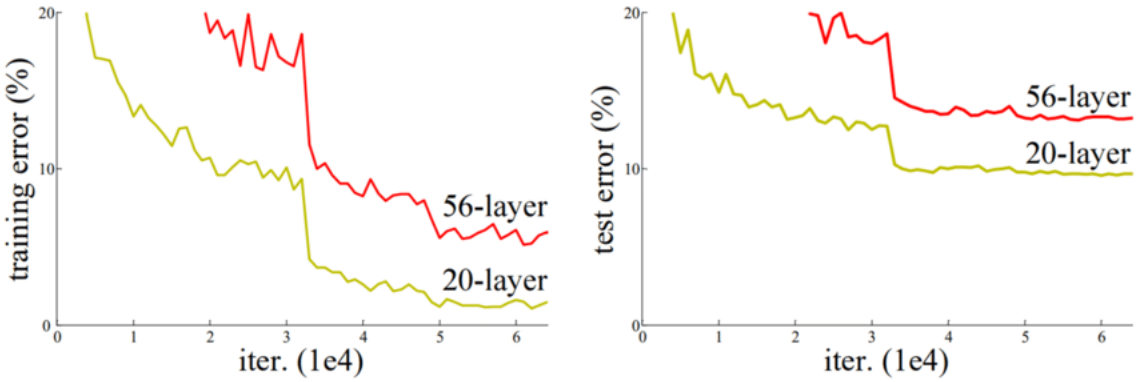


Figure 4.14: Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks

of the Vanishing gradient during back-propagation whereby the gradient becomes 0. The deeper network has higher training error, and thus test error.

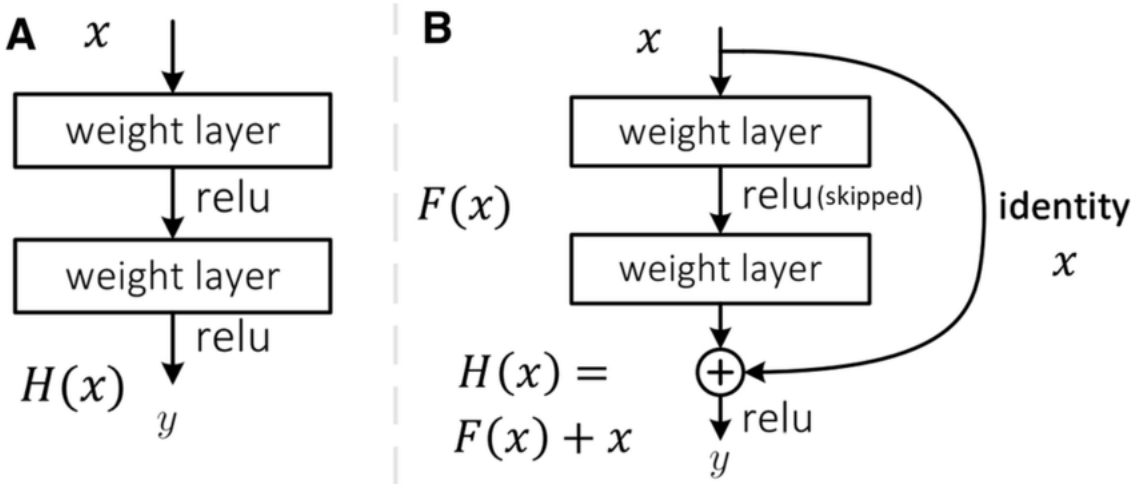


Figure 4.15: Relu function

In order to solve the vanishing gradient problem (due to the use of L1 and l2 regularize), the architecture introduced the skip connections which skips training from a few layers and connects directly to the output [Figure 4.15]. This will avoid that problem as gradient can back-propagate via the skip connection. Even if the main path is zero, performance will not degrade as information will flow through skip connection during forward propagation.
Figure 4.16 show how a simple network compares to a residual network in terms of accuracy. It's worth noting that the accuracy of a 34-layer simple network reaches saturation sooner than ResNet's. We propose this simple ResNet architecture of four ResNet blocks and four max pooling. We flatten our layer and feed it into a FC of 128 units to represent our embedding.

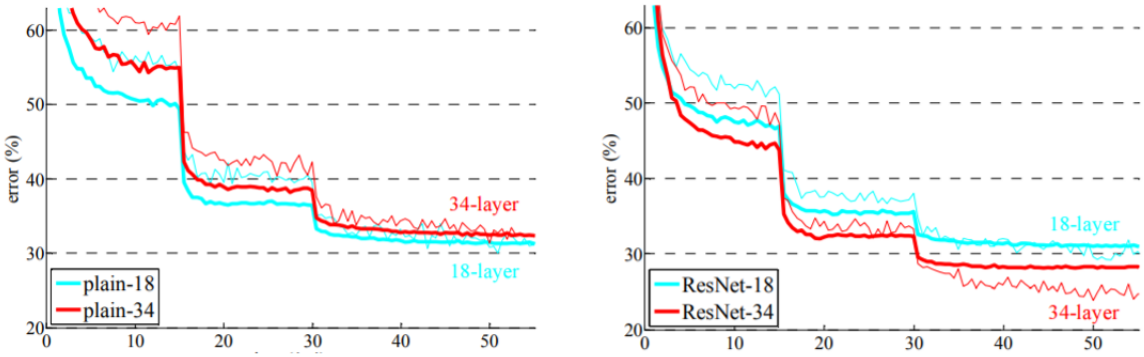


Figure 4.16: Accuracy comparison between a simple network and residual network

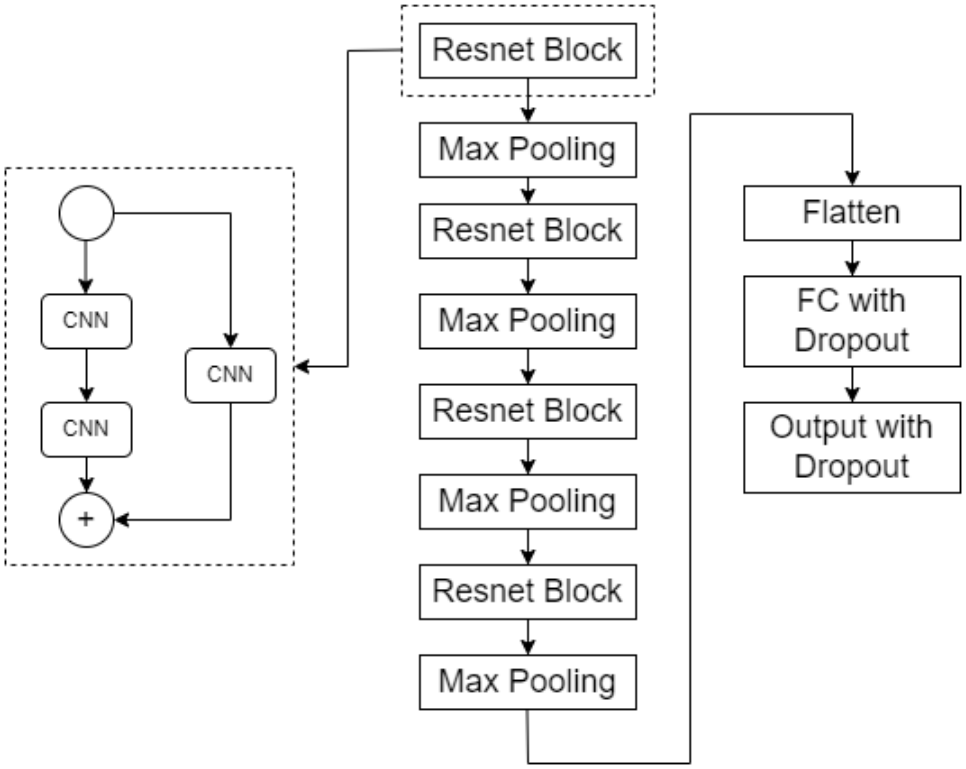


Figure 4.17: Simple ResNet architecture of four ResNet blocks and four max pooling

To perform Figure 4.17, we start by building our Resnet Block which will be duplicated 4 times in the whole architecture. In our function resnet_block, we define a kernel size of 3 x 3, 32 filters, with padding = "same", a l2 regularizer and a relu activation function. Next, we define a function simple_resnet where we will design the whole architecture. We implement the first Resnet block and the max pooling layer, where kernel size was 3, number of filters was 16, max poolsize was 2x2 and the number od stride was 2. Then, we got the shape of first pool. Then, we duplicate the above method by increasing the number of filters (16, 32, 48, 64) by keeping their same size of kernel, max pool and strides we go deeper. We then flatten our layer to get the flatten shape. Then, we feed into into a fully connected layer with dropout and units = 128 which represent the encoding.We use the TensorFlow implementation of the Inception ResNet V1 architecture proposed by David Sand-

berg. He proposed two models, one trained on the CASIA-Webface dataset and the other on the VGGFace2 dataset. They achieved an accuracy of 99.05% and 99.65% respectively. We will not use the pre-trained model as we will train our model from scratch but we will definitely use the metrics as a benchmark for our model.

4.3 Embedding

During the training phase, we don't provide our NN any explicit instructions for what the numbers in the vector should represent; rather, our sole requirement is that the embedding vectors of faces that are similar have comparable values (i.e. close to each other). It is up to our NN to find out a way to represent faces using vectors in such a way that the vectors representing different persons do not resemble one another, but the vectors representing the same person do. In order for this to be the case, our NN has to be able to recognize the distinguishing characteristics of a person's face that set it apart from other people's faces. During the training process, our NN is using a wide variety of various combinations of these characteristics in order to determine which ones perform the best. The way that our neural network represents features in a picture is different from the way that humans do (distance, size, etc.) and it is better this way as this enables it to do a far better job than us humans.

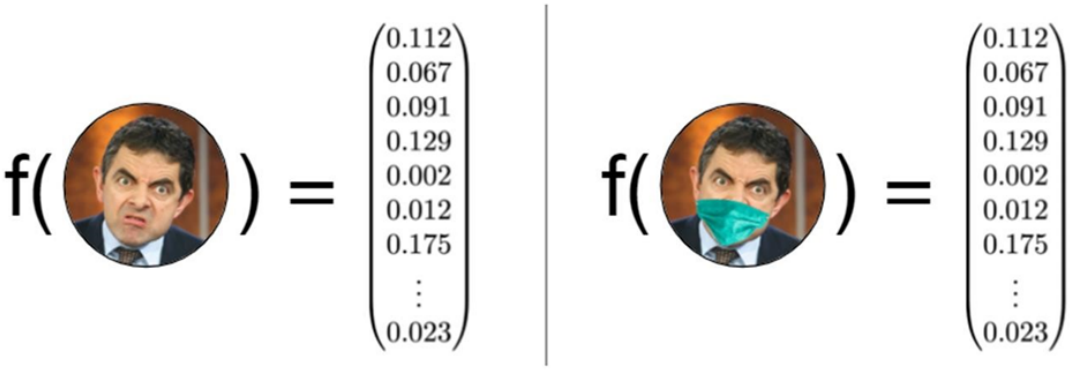


Figure 4.18: A function that, as its input, receives a picture and, as its output, generates a face embedding (a summary of the face)

The process of training a convolutional neural network to generate face embedding calls for a significant amount of data as well as the processing capacity of a computer. It took us roughly 46 hours of consistent practice before we could achieve decent accuracy. However, after it has been trained, the network will be able to provide measures for any face, including ones that it has never seen before. Therefore, we only need to do this step once.

4.4 Real-time Face Recognition

Our objective is to recognize a person who is behind the mask. We start by reading the image from a camera input.

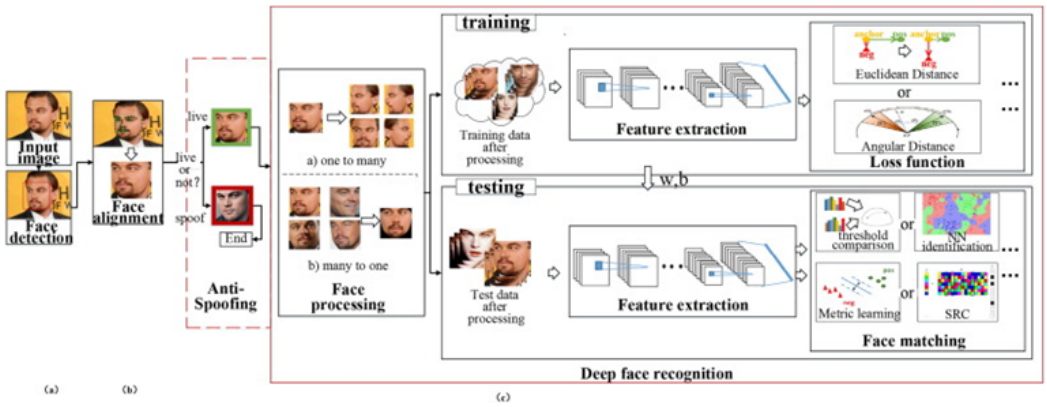


Figure 4.19: Face Recognition using Deep Learning

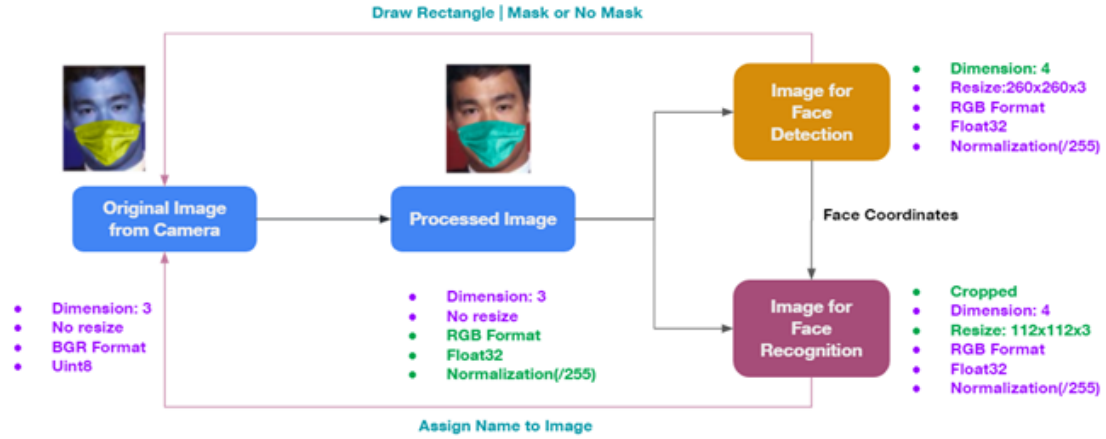


Figure 4.20: Real-time face recognition process

Figure 4.20 depicts that, we need to process the image from BGR to RGB. Using our pre-trained face mask SSD model we had, we perform face detection and crop the face. Then, we send this image to our face recognition model for face matching. If a face is detected, our face mask detection model will draw a rectangle showing if the person is wearing a mask or not. We use the face coordinates from the face detection model to crop the image, and our face recognition model will perform the embedding and assign a name to the image with the least distance. The process is divided into three phases:

1.Real-Time streaming

In a function called stream, we have a while loop which will stream video directly from our webcam.

2.Adding Face Mask Detection

We upload the face mask model from our pb file and create two variables named as margin and i2dclass. Here margin = 40 and id2class = 0: 'Mask', 1: 'NoMask'. Then, we initialize our face detection model. After that, we need to convert our image from BGR to RGB and do normalization. After normalization, we resize our image and increase the dimension by 1. At the end, we perform the inference and

check if we detect a mask or not. Here draw a green rectangle if we detect one(mask) and if not a red rectangle. We display the information just above the rectangle.

3.Adding Face Recognition

For the third phase, we have the Microsoft Celebrity dataset to test our model, which contains 85,742 persons' faces. We also insert our raw images in the database and check if the model can recognize us among all these people. We restore our face recognition model from our pb_model.pb file and initialize our face recognition model. Then, we read the images from the database. We then use a batch data of 32 to read our images, normalize the data and get the embeddings. After using a batch data of 32 to read our images, we normalize the data and get the embeddings. Then, we set our Euclidean distance equation to calculate the distance. After the face detection step, if a face is detected we get the face coordinates to crop the image, resize and change the dimension to 4. After getting the embeddings, the distance is calculated and then we get the index of the image of the smaller distance. If the distance of this particular index is also smaller than our threshold (0.8), we conclude we have a match. We also get the name of our image file which is initially an empty variable and display it on top of the rectangle. After executing the program and it took 307.093s (about 5 min) to get the embeddings of 85743 people and the test is successful.

Chapter 5

Training Phase

5.1 First Training (Evaluation: No Mask Dataset)

Now we train our model on the CASIA dataset with no masks and the custom CASIA dataset we made with masks. We have to evaluate the performance of our model on the LFW dataset, which contains images with no masks. The hyper-parameters which we need to tune are as follows:

- model_shape : [None, 112, 112, 3] or [None, 160, 160, 3].
- infer_method : inception_resnet_v1 or inception_resnet_v1_reduction.
- loss_method : cross_entropy or Arcface.
- opti_method : adam.
- learning_rate : 0.0001 or 0.0005.
- embed_length : 128 or 256.
- epochs : 40 or above.
- GPU_ratio : [0.1, 1].
- batch_size : 32 or 48 or 96 or 128.
- ratio : [0.1, 1.0].

5.2 Second Training with Data Augmentation

5.2.1 Evaluation: No Mask Dataset

After performing data augmentation on our pictures, we train the model and fine tune the hyper parameters as before. We use the LFW as our validation set when training the model. Below are the parameters to be tuned:

- model_shape : [None, 112, 112, 3] or [None, 160, 160, 3]
- infer_method : inception_resnet_v1

- loss_method : cross_entropy or Arcface
- opti_method : adam
- learning_rate : 0.0002 or 0.0005
- embed_length : 128 or 256
- epochs : 40 or 60 or 100
- GPU_ratio : None
- batch_size : 32 or 96 or 196
- ratio : 0.1 or 0.4
- process_dict : 'rdm_flip': True, 'rdm_br': True, 'rdm_crop': True, 'rdm_angle': True, 'rdm_noise': True

5.2.2 Evaluation: Mask Dataset

Whatever we have been doing till now is training our images on the CASIA dataset with masks and without masks and then testing them on our without mask LFW dataset. The accuracy we got before was for face recognition without masks. We propose another method for the evaluation of faces with masks. The steps are as follows:

1. Be using novel datasets for FaceNet training that has never been used before.
2. Choose 1000 distinct class photos (1000 individuals) for our face database ((reference data: ref data): No Mask Folder).
3. In these 1000 class images make them wear masks - these images will be used as test images (target data: tar_data): Mask Folder
4. We calculate the embeddings of both images (masks and without masks) and use them to do face matching.

In Fig 5.1, here, these images contain 3-dimensional embedding. Calculating the Euclidean distance using the d function as explained before for the target image (image with masks) and the images in our No Mask folder. If the indexes of both images match and the distance is below a threshold (0.3), we conclude after having a correct match.

5.3 Third Training (Evaluation: Mask Dataset with Stratified Sampling)

In the previous training, we introduce a sampling bias. For example, our first class has 100 images in our CASIA dataset and the second class has 10 images, then when uses a ratio of 0.4. We take 40 from the first folder and only 4 from the second folder. This disparity in the number of images in the folders creates this sampling bias. A better method ensures the test set represents all of the different classes

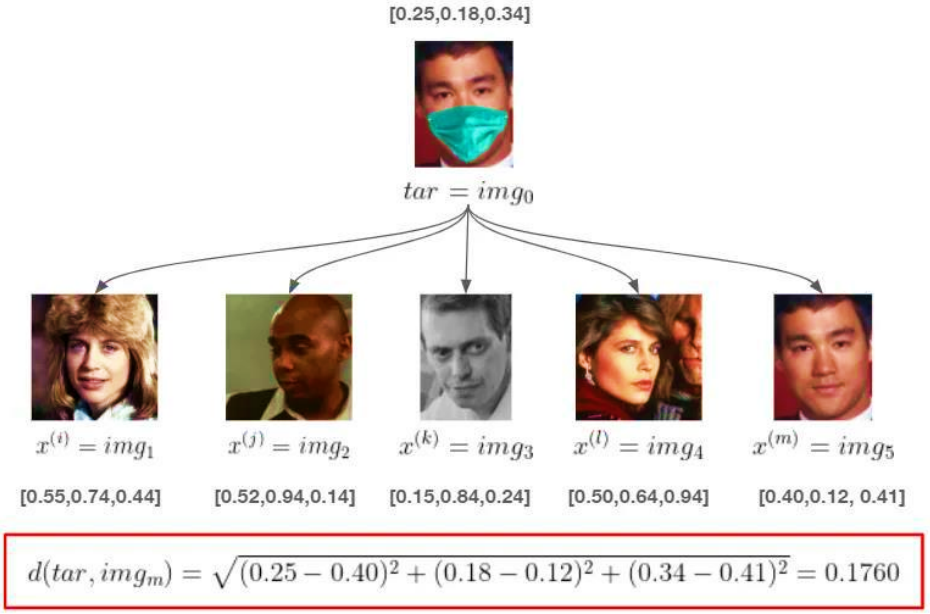


Figure 5.1: Calculating Euclidean distance between mask and no mask, to get the best match

in the dataset. As a result, stratified sampling is used. The classes are separated into homogenous sections named strata, and the appropriate numbers of photos are selected from each stratum to ensure the test dataset is illustrative of the whole dataset. We begin by exploring how many classes have fewer than 10 images and how many have more than 100 images. We get 195 for the first condition and 859 for the second condition. While it is challenging to remove classes with fewer than 10 images, we cannot remove both sets of classes as we lose about 1000 classes. At this stage, we randomly select a constant x (2 or 5 or 15) number of images from all the different classes. By doing so, the numbers of images are reduced and training process still ongoing, also gain the training time. More data augmentation are performed to increase our data (since we are selecting only (2 or 5 or 15) images from all the classes) for each image we have selected. Then, we create a copy of four images on which we do data augmentation. For example, from a folder of 10 images, we select only 5 images, then we see that we have reduced by 50% the number of images on which the AI will train. However, we perform data augmentation on each of the 5 images such that the 5 images will multiply by 4 to become 20 images
Fig 5.2 follow these principles

- Original image without mask
- Original image without mask with Data augmentation
- Original image with random mask with Data Augmentation
- Original image with random mask with Data Augmentation

We change our get_4D_data function to accommodate the changes describe above. Creating a variable aug_times and assign it the value of 4 since each image are augmented with 4 pictures. Here, we have a dictionary p_dict_1 where we input the

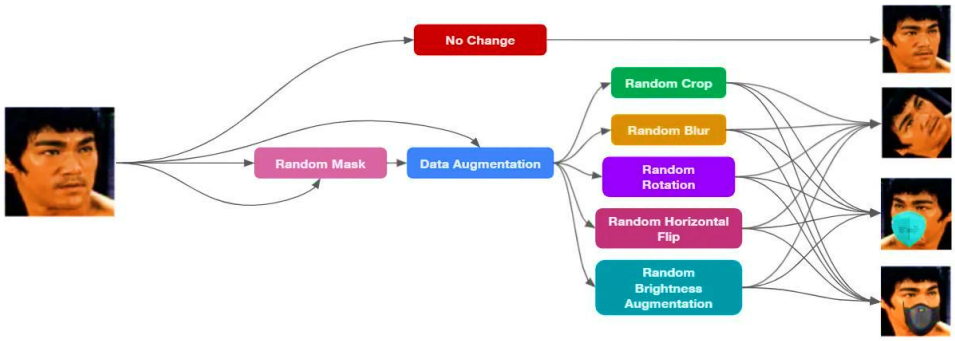


Figure 5.2: 4 times augmentation

type of data augmentation which has been done. The variables are enumerated by us to apply those data augmentations. After creating another variable.select_num, that will input the number of images whichever we want to select from each class. We check either it is an integer and greater than 1. If so, we reset our training paths and labels. paths is a list because we use append to collect images from each folder. We shuffle that list and use np.min() to find the minimum between select_num and len(paths), i.e., we take all pictures in the latter case. Finally, the list are transformed to a NumPy array and shuffled it.

Chapter 6

Result and Discussion

6.1 Result on First Training (Evaluation: No Mask Dataset)

Using only 20% of the dataset for testing purposes, we get low values for the training and testing accuracy and some weird-looking graphs as expect. Then, we increase the ratio (ratio of images of the whole dataset)to 0.01 but reduce the batch size to 32 as our GPU run out of memory and the results start to be promising, with the test accuracy at nearly 72.9%. However, we see our training accuracy at 100%, which is ideal, but we suspect over fitting of data. In order to avoid this, we need to feed the model more data, so the ratio is increased to 0.1. The training accuracy is increased slightly, but the testing accuracy is increased by nearly 20%.

Epoch	Accuracy
100	52.31%
25	100%
40	99.993%

Table 6.1: Training set accuracy

Epoch	Accuracy
64	65.21%
16	72.9%
3	86.93%

Table 6.2: Testing set accuracy

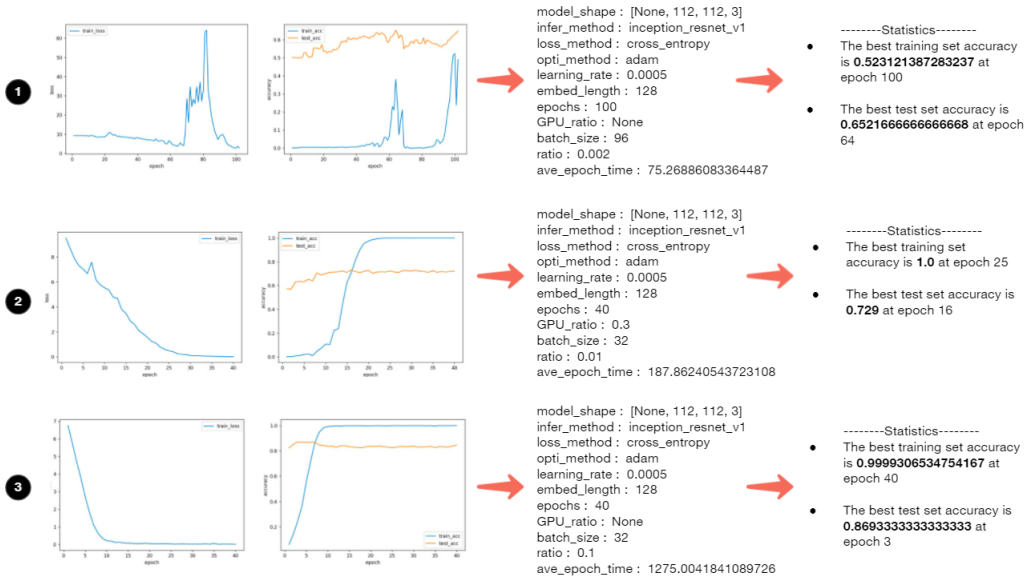


Figure 6.1: Result on first training for no mask dataset

Figure 6.1 illustrates the testing accuracy never exceeded the training accuracy. With only a testing accuracy of 86.93%, the model would work, but it is not a reliable one. So, we need to have an accuracy of nearing 98% or 99% in order to be used in an industry setting. However, our GPU (GTX 1050 Ti-4GB memory) ran out of memory, so we have to stop the training for the time being till we find another way of training the model.

6.2 Result on second Training with Data Augmentation

6.2.1 Evaluation: No mask Dataset

At this stage, setting the ratio to 0.4 and the batch size to 196, the accuracy is increased from 86.93% from our last training to a whopping 92.07%, which is an increase of nearly 6%. Using the same settings, we only change the learning rate to 0.0002, and the testing accuracy went up a little to 0.9618.

Epoch	Accuracy
28	79.41%
5	79.57%
50	79.98%
83	79.98%

Table 6.3: Training set accuracy

Epoch	Accuracy
21	96.17%
2	95.97%
24	96.18%
57	96.18%

Table 6.4: Testing set accuracy

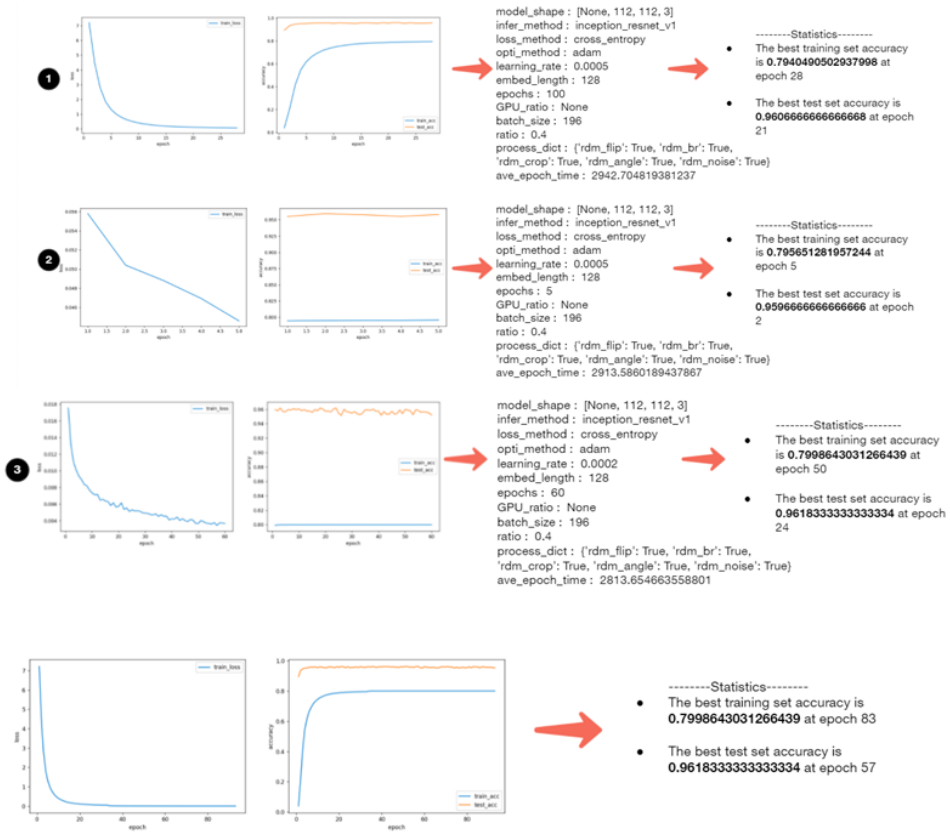


Figure 6.2: Result on second training with data augmentation for no mask dataset

In Figure 6.2, we clearly see a great improvement in our testing accuracy, but we need to do more tests to ensure the robustness of the model.

6.2.2 Evaluation: Mask Dataset

When evaluating the pre-trained weights of the model 20180408-102900 on our mask dataset, we get only an accuracy of 16.3% at a threshold of 0.7 and nearly thrice that at a threshold of 0.8. The same is seen for the model 20180402-114759, with slightly better accuracy. This clearly shows that we have not use the model for face recognition with masks. Our model before data augmentation has a shockingly low accuracy on both thresholds. However, since we reach only an accuracy of 0.8693 of the LFW dataset, we can conclude the model was not that robust.After data augmentation, our model accuracy increase sharply on both the LFW dataset and the masks dataset, with a maximum accuracy of 99.98% at a threshold of 0.8. Our model surpassed the accuracy of the pre-trained weights of the original Inception

ResNet V1 model and has been optimized to perform better at recognizing faces with masks. While the accuracy values may seem promising, we suspect we may

					Threshold	
Model Name	Architecture	Training Dataset	LFW Accuracy	Testing Dataset	0.7	0.8
2018408-102900	Inception ResNet V1	CASIA	0.9905	With & Without Masks	0.16321	0.45212
2018402-114759	Inception ResNet V1	VGGFace2	0.9965	With & Without Masks	0.33528	0.60711
Our_model_before_aug	Inception ResNet V1	CASIA+CASIA with Masks+Cropped	0.8693	With & Without Masks	0.06161	0.20396
Our_model_after_aug	Inception ResNet v1	CASIA+CASIA with Masks+Cropped+Data Augmentation	0.9618	With & Without Masks	0.99870	0.99979

Table 6.5: Accuracy before and after Data Augmentation

still be over fitting the data and this is due to the unbalanced CASIA dataset which we have. As we will explain below, we think we should do a fairer sampling of our data for training.

6.3 Result on third Training (Evaluation: Mask Dataset with Stratified Sampling)

We start training the model with a large epoch number of 100. We reduced our batch size to 96 to avoid our GPU running out of memory. We have a select_num = 2 for faster training and we keep all the other parameters unchanged. After 33 hours of training, we managed to get a decent accuracy of 0.9693. Now we can be sure we are not over fitting as much when solving the bias sampling issue.

Epoch	Accuracy
100	75.74%
97	94.19%

Table 6.6: Training set accuracy

Epoch	Accuracy
93	95.51%
89	96.93%

Table 6.7: Testing set accuracy

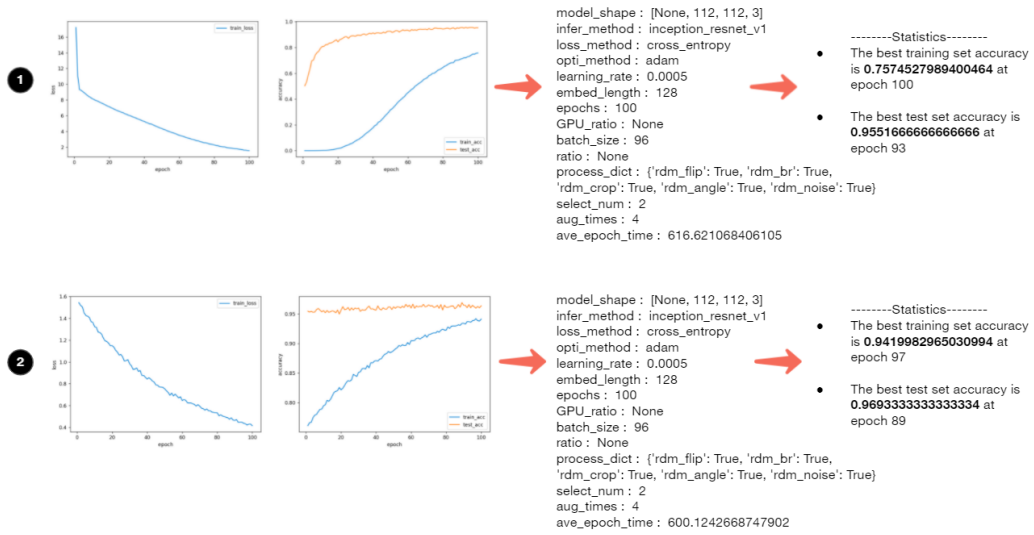


Figure 6.3: Result on third training for mask dataset with Stratified Sampling

Table 6.8 is the test results for our whole training process till the beginning.

					Threshold	
Model Name	Architecture	Training Dataset	LFW Accuracy	Testing Dataset	0.7	0.8
2018408-102900	Inception ResNet V1	CASIA	0.9905	With & Without Masks	0.16321	0.45212
2018402-114759	Inception ResNet V1	VGGFace2	0.9965	With & Without Masks	0.33528	0.60711
Our_model_before_aug	Inception ResNet V1	CASIA+CASIA with Masks+Cropped	0.8693	With & Without Masks	0.06161	0.20396
Our_model_after_aug	Inception ResNet V1	CASIA+CASIA with Masks+Cropped+Data Augmentation	0.9618	With & Without Masks	0.99870	0.99979
Our_model_after_data_aug : select_num=2	Inception ResNet V1	CASIA+CASIA with Masks +Cropped + Data Augmentation +Strata	0.9693	With & Without Masks	0.99233	0.99849
Our_model_after_data_aug : select_num=5	Inception ResNet V1	CASIA+CASIA with Masks +Cropped + Data Augmentation +Strata	0.9661	With & Without Masks	0.99470	0.99860
Our_model_after_data_aug : select_num=15	Inception ResNet V1	CASIA+CASIA with Masks +Cropped + Data Augmentation +Strata	0.969	With & Without Masks	0.99416	0.99892

Table 6.8: Testing accuracy from the beginning

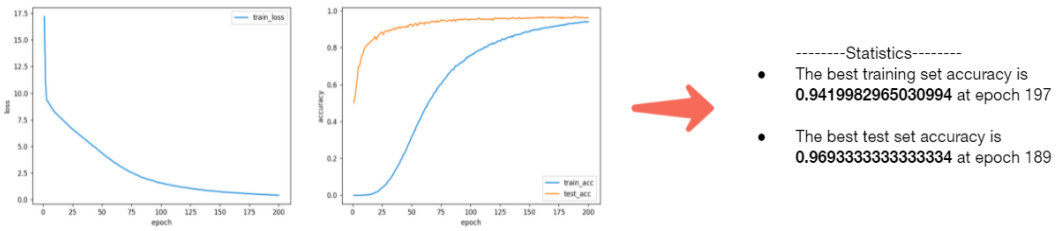


Figure 6.4: Higher Accuracy with Stratified Sampling after data augmentation

Here in Figure 6.4, we observe that the difference when selecting 2 or 5 images from each class and 15 images from each class is very small (0.9947 compared to 0.9941). However, the average time of one epoch was 65 min with a GPU of GTX 1050 Ti. We see that when we selected only 2 images from each folder and performed data augmentation, then our accuracy was nearly the same and our average time for one epoch was only around 10 min. We can now achieve high accuracy even when our selected number is small because we have solved the data imbalance problem.

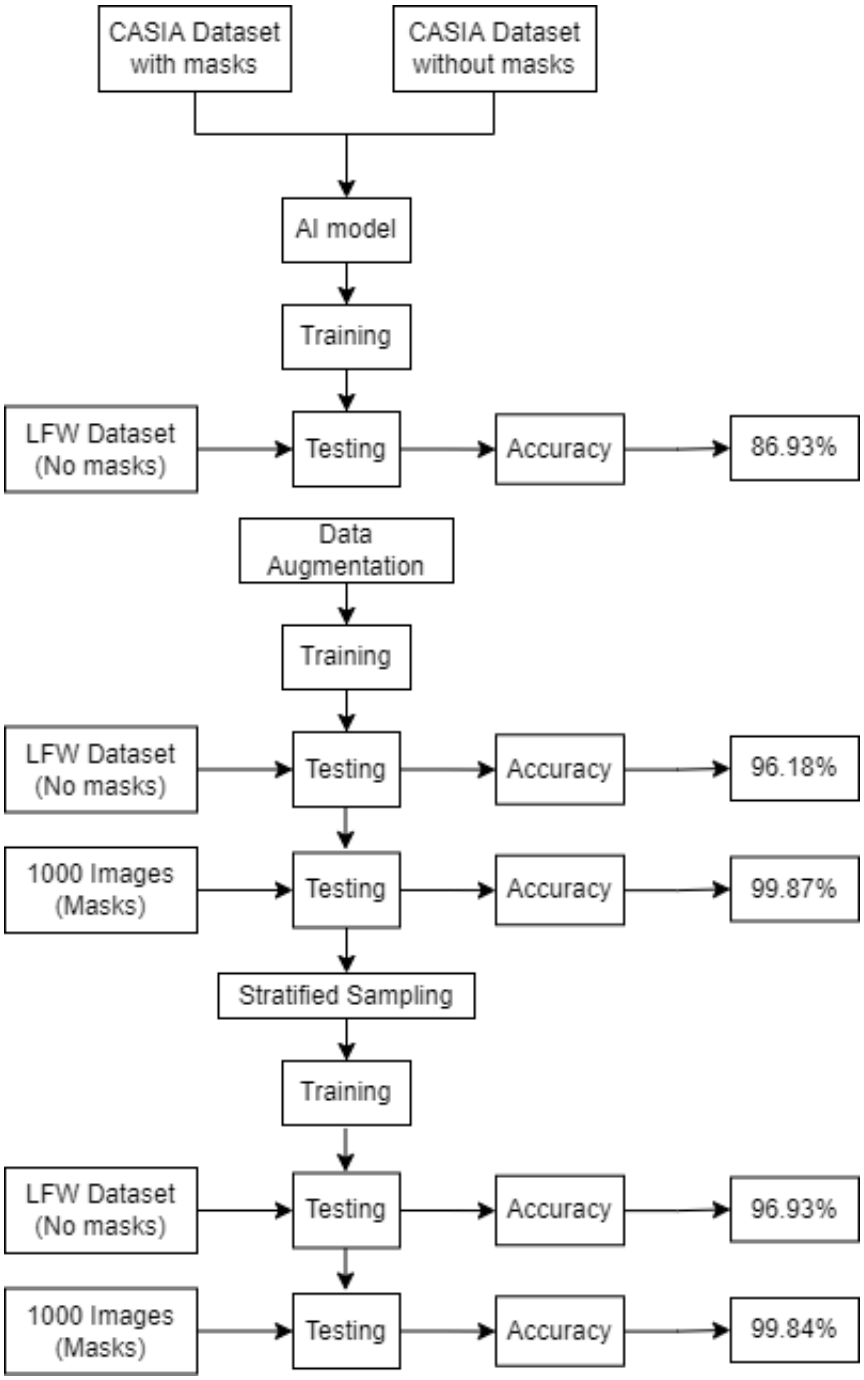


Figure 6.5: Accuracy schema

Figure 6.5 explains the final schema which resumes our whole training and testing accuracies.

Chapter 7

Conclusion

This paper started with a simple face detection algorithm and incorporated SSD to create a masked face identification model. Inception ResNet V1 architecture was implemented to train that model from scratch and test the accuracy. To avoid overfitting, data augmentation techniques were performed, and it achieved 96.18% accuracy from 86.93%. Due to the imbalance of data, a stratified sampling was implemented which takes more than 3 hours of data coupled with data augmentation so as not to decrease the training dataset. Achieving an accuracy of 99.84% after performing data augmentation and stratified sampling, the model is ready to be deployed. The Data-Centric approach - holding the model fix and iteratively improving the quality of the data - seemed to be fruitful in the end. By constantly enhancing the dataset with data augmentation and a more appropriate sampling technique, it was possible to build a robust, scalable model. About 80% of the time has been spent on data preparation. Injecting more and more data into that model without a proper cleaning process, sampling technique, or data augmentation process would never have created a good performing AI model. The Data-centric AI model was the key to building this masked face identification model, and more data processing will improve the prediction accuracy.

Bibliography

[1] W. Niu, J. Long, D. Han, and Y.-F. Wang, "Human activity detection and recognition for video surveillance," in *2004 IEEE international conference on multimedia and expo (ICME)(IEEE Cat. No. 04TH8763)*, IEEE, vol. 1, 2004, pp. 719–722.

[2] A. V. Ponkia and J. Chaudhari, "Face recognition using pca algorithm," *Inventi Rapid: Image & Video Processing Journal*, vol. 4, no. 1, pp. 519–524, 2012.

[3] P. Bagchi, D. Bhattacharjee, and M. Nasipuri, "Robust 3d face recognition in presence of pose and partial occlusions or missing parts," *arXiv preprint arXiv:1408.3709*, 2014.

[4] G. N. Priya and R. Wahida Banu, "Occlusion invariant face recognition using mean based weight matrix and support vector machine," *Sadhana*, vol. 39, no. 2, pp. 303–315, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[6] R. Weng, J. Lu, and Y.-P. Tan, "Robust point set matching for partial face recognition," *IEEE transactions on image processing*, vol. 25, no. 3, pp. 1163–1176, 2016.

[7] N. Bisagno, N. Conci, and B. Rinner, "Dynamic camera network reconfiguration for crowd surveillance," in *Proceedings of the 12th International Conference on Distributed Smart Cameras*, 2018, pp. 1–6.

[8] M. S. Ejaz, M. R. Islam, M. Sifatullah, and A. Sarker, "Implementation of principal component analysis on masked and non-masked face recognition," in *2019 1st international conference on advances in science, engineering and robotics technology (ICASERT)*, IEEE, 2019, pp. 1–5.

[9] A. Anwar and A. Raychowdhury, "Masked face recognition for secure authentication," *arXiv preprint arXiv:2008.11104*, 2020.

[10] D. K. Chu, E. A. Akl, S. Duda, K. Solo, S. Yaacoub, H. J. Schünemann, A. El-harakeh, A. Bognanni, T. Lotfi, M. Loeb, *et al.*, "Physical distancing, face masks, and eye protection to prevent person-to-person transmission of sars-cov-2 and covid-19: A systematic review and meta-analysis," *The lancet*, vol. 395, no. 10242, pp. 1973–1987, 2020.

[11] Q. Hong, Z. Wang, Z. He, N. Wang, X. Tian, and T. Lu, "Masked face recognition with identification association," in *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, 2020, pp. 731–735.

[12] L. Schnirring, *Global covid-19 total tops 51 million as us hospital cases rise*, https://www.printfriendly.com/p/g/GRyFMb, 2020.

[13] Y.-C. Wu, C.-S. Chen, and Y.-J. Chan, "The outbreak of covid-19: An overview," *Journal of the Chinese medical association*, vol. 83, no. 3, p. 217, 2020.

[14] A. Cabani, K. Hammoudi, H. Benhabiles, and M. Melkemi, "Maskedface-net–a dataset of correctly/incorrectly masked face images in the context of covid-19," *Smart Health*, vol. 19, p. 100 144, 2021.

[15] W. Hariri, "Efficient masked face recognition method during the covid-19 pandemic," 2021.

[16] B. Mandal, A. Okeukwu, and Y. Theis, "Masked face recognition using resnet-50," *arXiv preprint arXiv:2104.08997*, 2021.

[17] H. N. Vu, M. H. Nguyen, and C. Pham, "Masked face recognition with convolutional neural networks and local binary patterns," *Applied Intelligence*, pp. 1–16, 2021.

[18] G. Wu, "Masked face recognition algorithm for a contactless distribution cabinet," *Mathematical problems in engineering*, vol. 2021, 2021.