Towards An Ethnic-Bias Free Approach For Facial Recognition

By

A.T.N Kalinga

15000664

This dissertation is submitted to the University of Colombo School of Computing
In partial fulfillment of the requirements for the
Degree of Bachelor of Science Honours in Computer Science

University of Colombo School of Computing
35, Reid Avenue, Colombo 07,
Sri Lanka
July 2020

Declaration

I, A.T.N Kalinga 2015/CS/066 hereby certify that this dissertation entitled "Towards An Ethnic-Bias Free Approach For Facial Recognition" is entirely my own work and it has never been submitted nor is currently been submitted for any other degree.

Date	Student's Signature

I, Dr. G.D.S.P Wimalaratne, certify that I supervised this dissertation entitled "Towards An Ethnic-Bias Free Approach For Facial Recognition" conducted by A.T.N Kalinga in partial fulfillment of the requirements for the degree of Bachelor of Science Honours in Computer Science.

Date	Supervisor's Signature

Abstract

In this study, we investigate the issue of ethnicity bias on the existing facial recognition approaches and provide a novel way to consider the ethnicity at the model training stage. This study aims to explore the ethnicity bias issue on the current state of the art facial recognition algorithm Facenet and to provide a novel image selection method to minimize the effect from the ethnicity bias without making changes to the neural network architecture.

Building upon the works of Facenet, modifications were done to the image selection process which triggers the training stage of the model. Since the input data to the model for training is fed using batch by batch, the batch creation operation is modified so that each batch represents the total ethnicity distribution of the training dataset. Then the trained model is evaluated thoroughly using classification tasks, clustering attempts by comparing with the base model. From the classification tasks, the focus directed towards the metrics such as accuracy, false acceptance rate, area under the curve (AUC). In the process to create batches that have the same ethnic distribution, we have presented an ethnicity classifier that can classify a given image of a person to its relevant ethnicity.

The model trained with the proposed method has shown significant improvement when considering the AUC metric which reduces the chance to misclassify a person with underrepresented ethnicity. Thus reducing the false acceptance rate which increases the reliability of the proposed model as well as minority representation. The clustering evaluation tasks revealed that the model trained with the proposed approach can form near-perfect clusters while the base model struggles to do so.

Keywords: Facial Recognition, Ethnicity-bias, Ethnicity Classification

Preface

In this dissertation novel method is proposed to explore the ethnic-bias present in the current state of the art facial recognition algorithm Facent and optimizations to be carried out without changing the neural network architecture of the model to handle the ethnic-bias issue. Design in Chapter 3 and implementation of Chapter 4 relies on the works of the Facenet [1] and building upon the current framework to reduce the effect on accuracy and validity which occurred from the ethnic-bias issue. The results from different ethnicity classification models, sampling methods have been used to justify the design decisions which are described in the Chapter 3. The evaluation of the proposed model is carried out by myself from recreating a training environment somewhat similar to the original approach and using the calculated metrics to analyse the results. For the training and classification tasks of the models, relevant benchmark datasets are used which are described in the Chapter 3 Design section by citing the authors to recognize their works. Moreover the results and analysis from the results described in the evaluation chapter and conclusion chapter are my own work.

Acknowledgement

I would like to express my sincere respect and gratitude to my research supervisor, Dr. G.D.S.P Wimalaratne, Senior lecturer of University of Colombo School of Computing. I would also like to extend my respect and gratitude to Prof. N D Kodikara Senior Professor of UCSC and Dr. L N C De Silva, Senior lecturer of University of Colombo School of Computing for providing valuable feedback on the research proposal and interim evaluation to improve the results. I would like to express my gratitude for acknowledge the assistance provided by Dr. H. E. M. H. B.Ekanayake, senior lecturer of University of Colombo School of Computing as the final year research project coordinator.I would also like to thank all of my batch mates for giving comments, motivation and their warm friendship. It is a great pleasure for me to acknowledge all the teachers, mentors and people for the immense support they have given. Finally to my parents and family for their immeasurable sacrifices and love they have given throughout this beautiful journey.

Contents

Declaration i

Abstract ii

Preface iii

Acknowledgement iv

Contents vii

List of Figures ix

List of Tables x

Acronyms xi

1 Introduction 1

1.1 Background 1
1.2 Research Problem and Research Questions 2
1.3 Justification for the research 2
1.4 Scope and Delimitation 2
1.5 Methodology 3
1.6 Outline of the Dissertation 3
1.7 Definitions 4
1.8 Conclusion 4

2 Literature Review 5

2.1 Introduction 5
2.2 Facial Recognition 5
2.3 Ethnicity Bias 7
2.4 Ethnicity Classification 8
2.5 Datasets 9

3 Design **10**
3.1 Introduction . . . 10
3.2 Facial Recognition . . . 10
3.2.1 The existing approach . . . 10
3.2.2 Proposed approach . . . 12
3.3 Ethnicity Detection . . . 16
3.3.1 Previous approaches from literature . . . 16
3.3.2 Proposed Model . . . 16
3.4 Datasets . . . 20
3.5 Conclusion for the Chapter . . . 20

4 Implementation **21**
4.1 Ethnicity detection . . . 21
4.1.1 Explore the UTKFace dataset . . . 21
4.1.2 Preprocessing the dataset . . . 23
4.1.3 Model Training . . . 26
4.2 Facial recognition . . . 28
4.2.1 Building data table containing ethnicities for a dataset . . . 28
4.2.2 Implementation of the proposed design . . . 31
4.3 Conclusion for the Chapter . . . 32

5 Results and Evaluation **33**
5.1 Introduction . . . 33
5.2 Ethnicity Classification . . . 34
5.2.1 Classification from randomly selected images from test split 34
5.3 Facial Recognition . . . 36
5.3.1 Classification using randomly selected images from LFW dataset 37
5.3.2 Classification using total LFW dataset . . . 38
5.3.3 Area Under the Curve (AUC) . . . 39
5.3.4 Clustering of the output embeddings from model . . . 40

6 Conclusion **43**
6.1 Introduction . . . 43
6.2 Conclusions about research questions . . . 43
6.3 Conclusions about research problem . . . 44
6.4 Limitations . . . 45
6.5 Implications for further research . . . 45

References **46**

Appendices **49**

A MTCNN face alignment on CASIA dataset **50**

List of Figures

3.1 High level diagram of training process . . . 11
3.2 Triplet Selection . . . 11
3.3 Triplet Selection on a mini batch . . . 12
3.4 Process Flow of Proposed Approach . . . 13
3.5 Lookup table . . . 13
3.6 Ethnicity Taxonomy [2] . . . 14
3.7 Sampling methods taxonomy . . . 14
3.8 Overview of the UTKFace dataset [3] . . . 17
3.9 Samples of ethnicities in UTKFace dataset [3] . . . 18
3.10 VGG16 Model Architecture [4] . . . 19

4.1 Exploring the UTKFace dataset . . . 21
4.2 UTKFace naming convention . . . 22
4.3 PreProcessing Implementation . . . 23
4.4 PreProcessing Implementation . . . 24
4.5 PreProcessing Implementation . . . 25
4.6 Model Training . . . 26
4.7 VGG Model Structure Implementation . . . 27
4.8 Usage of Adam optimizer . . . 27
4.9 Usage of EarlyStopping, histogram generation . . . 27
4.10 Building data-table Pt.1 . . . 28
4.11 Building data-table Pt.2 . . . 29
4.12 Implemented data table . . . 30
4.13 Implemented data table . . . 30
4.14 Proposed approach implementation Pt.1 . . . 31
4.15 Proposed approach implementation Pt.2 . . . 32

5.1 Paths of the evaluation . . . 33
5.2 Convergence point in training . . . 34
5.3 Validation Accuracy of Ethnicity Classification . . . 35
5.4 t-SNE clustering- Base Model . . . 40

5.5 t-SNE clustering- Improved Model 40
5.6 DBSCAN clustering 41
5.7 KMeans clustering 41
5.8 Self Organizing Map 42

A.1 MTCNN face alignment pt1 51
A.2 MTCNN face alignment pt1 52

List of Tables

3.1 Comparison of Sampling methods 15
3.2 Comparison of Ethnicity Classifier 16
3.3 Dataset Statistics 20

4.1 Naming Convention description 22

5.1 Summarized Evaluation 36
5.2 AUC analysis 39

Acronyms

SVC	C-Support Vector Classification
RAM	Random Access Memory
HDD	Hard Disk Drive
LR	Learning Rate
t-SNE	t-distributed stochastic neighbor embedding
DBSCAN	Density-based spatial clustering of applications with noise
SOM	Self Organizing Maps
ROC	Receiver Operating Characteristic
AUC	Area Under Cover
MTCNN	MultiTask Cascaded Convolutional Networks
VGG	Very Deep Convolutional Networks for Large-Scale Visual Recognition
ADAM	A Method for Stochastic Optimization
FAR	False Acceptance Rate

Chapter 1

Introduction

1.1 Background

Facial recognition is the process of identifying or verifying a person by comparing and analyzing patterns based on the person's contours. While Identification focuses on identifying a previously seen face, verification compares faces and verify that the face is previously seen. Facial recognition can be mainly divided into four categories, but for this research, the focus will be on the approaches that use deep neural networks [1],[5],[6] and addresses the issue of facial recognition using minimal input data.

The main challenges faced when designing a facial recognition model are how well it performs, what is the minimum amount of images of faces need for the model to function properly, scalability, how the model performs on the issue of outliers. So usually when designing a model, the impact of the above issues are directly considered and fused on to the architecture. Yet according to the literature [5] above considerations are not applied to the features like ethnicity and gender of the person. Thus it makes this issue a largely unexplored phenomenon in the facial recognition context.

Practically, when a neural network is designed to address an issue on a specific dataset there can be some hidden layers which handle the problems of the dataset. (Eg: If the images are too bright, add a mask to normalize it or discard it). But when the same model trained on a different dataset the previously added mask becomes obsolete and can affect the accuracy of the model. Thus the need to update the structure of the neural network to tailor into the dataset.

So the motivation for this research is to find an approach to address ethnicity-bias that exists in the facial recognition algorithms that use triplet loss without making changes to the internal structure of the neural network.

1.2 Research Problem and Research Questions

- Explore the ethnic-bias issue on the facial recognition model Facenet which was proposed by Schroff et.al [1]
- How to address the ethnic-bias issue without changing the structure of the neural network

In the literature [6] it is shown that by only extracting the attributes on the face that represents the ethnicity, gender and by training a classifier on the extracted attributes, a model can be produced that can predict the ethnicity of a person with 85% when given a photo. Thus pointing out that the gender and ethnicity of a face plays a key role in a facial recognition algorithm.

Although many approaches [5],[6],[7] have taken in the literature to address the issue of this ethnic-bias for the facial recognition algorithms that use neural network approaches there is a lacks of the approaches and optimizations that are targeted at the facial recognition algorithms that use transfer learning as their training process which requires minimal data per class.

Also, the approach by Schroff et.al [1] is considered as the state of the art facial recognition algorithm that requires minimal dataset, it's training process does not specifically take ethnicity attributes into account.

1.3 Justification for the research

The main goal of this research project is to address the issue of ethnic-bias which resides on facial recognition algorithms. Since facial recognition is considered as a vast research area, the main focus is to find the solution for ethnic-bias issue on facial recognition algorithms which are designed to work with a minimal amount of data per class.

1.4 Scope and Delimitation

The proposed approach the ethnicity-bias is focused on the facial recognition algorithms that use triplet loss [8] as the loss calculation function. Hence the scope is limited to the algorithms which use triplet loss function.

The proposed approach solely focused on the optimization of the dataset chunking process. And maintain an equal ethnic distribution of the elements in the original dataset and newly created data chunks. The reason behind the data chunking

is because the lack of the hardware memory to keep the whole dataset on the memory, but if the whole dataset can fully fit into the memory of the computer then the proposed optimizations are not needed.

Facial recognition is mainly divided into two categories.

- Verification
- Identification

Verification focuses on the re-recognizing a person over and over, while identification focuses on matching the features of a previously seen face into a new one. Hence the identification problem (Eg: Unlocking a mobile phone via the facial features) solely focuses on the same person over and over again, optimizations that consider the ethnicity is not needed.

1.5 Methodology

The goal is to provide a better way of selecting the data chunks via considering the ethnicity distribution of the original dataset. By using the approach, the facial recognition algorithms which use triplet loss training to measure the loss of the training process can find a hard negative face to the selected positive face. Thus in return will provide better accuracy when used with people who are from less represented ethnicities.

As for the research methodology, since the ultimate goal of this research is to reduce the ethnic-bias on currently existing facial recognition algorithms thus improving the total accuracy of the algorithm constructive research methodology [9] will be used. In addition, since this research topic is evaluated by multiple approaches, constructive research methodology followed by mixed-method [10] research methodology will be used. Mixed-method is used because it allows the researcher to take more than one approach to address the research questions. Also, results from the multiple approaches can be used to validate each other's results

1.6 Outline of the Dissertation

First Chapter of the dissertation will outline the background of the research area as well as research questions, aims, methodologies and scope of the research. The second chapter focuses on the relative works carried out in the literature as well as the similar works and the approaches that is used as solutions. Chapter 3 focuses on the Design element of the proposed solution as well as the justification for the

decisions taken in the design process. Chapter 4 covers the detailed implementation aspect of the proposed design And Chapter 5 focuses on results and analysis. Chapter 5 is used to comment on the observations noticed from the experiments carried out as well as the reasoning behind the observations. Finally, Chapter 6 will provide discussion and conclusion about positive and negative outcomes of the results, as well as limitation and future avenues for the research.

1.7 Definitions

AUC: Considered as a performance measurement for classification problems at various threshold settings. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.

FAR: Is abbrivation for False Accept Rate which is the rate of face pairs that are different and yet have a distance that is below a certain threshold (which could be classified as positive).

1.8 Conclusion

This chapter laid the foundations for the dissertation. It introduced the research problem, research questions, research aim and objectives. Then the research was justified, definitions used in this dissertation were presented, the methodology was briefly described and justified, the dissertation was outlined, and the scope and limitations were given. Based on these foundations, the dissertation can proceed with a detailed description of the research and its contributions.

Chapter 2

Literature Review

2.1 Introduction

Since the research questions go along with multiple fields, the related works divided into four segments. One covering the advances and related works in the field of Facial Recognition. Secondly, the ethnic-bias issue presented in the current state of the art facial recognition algorithms is explored. Following, related works of the ethnicity classification mainly because of the proposed approach ventures into the field of ethnicity classification. Lastly, the datasets used for training, evaluation and benchmarking are reviewed.

2.2 Facial Recognition

Facial recognition considered to be a very broad field that joins the fields of computer vision, image processing, and machine learning. Due to this vast scope of the problem, there exist many approaches that address different contexts hence a general taxonomy [11] was needed to summarize the field of facial recognition by the context they are used on. Facial recognition was categorized into four main categories, from which the appearance-based approach mainly focuses on the variations of the face, scale, pose and expression changes. Model-based facial recognition algorithms provide solutions that are derived from analyzing the geometrical characteristics of the face while on the other hand hand-crafted based facial recognition focuses on computationally efficient yet manual adjusted approaches. Learning-based approaches provide solutions by modeling and learning relationships from the input data which in fact it uses machine learning concepts. In the time of writing current state-of-the-art facial recognition algorithms are using neural networks more specifically Convolutional Neural Networks as their backbone.

One-shot learning [12] is an object categorization problem that is found in the

computer vision field. This approach focuses on to learn information about an object using a very minimal amount of sample data. Bayesian one-shot learning algorithm identifies the foreground and background of an image and in the training process after each iteration, the features known from the previous iteration are passed as a parameter. Which in fact can be used to segment and estimate the ratio of the background and foreground of a new image.

Siamese neural networks [13] are built on top of the theories of one-shot learning which tries to use the knowledge learned in a different context and map it to a problem with smaller training data. Siamese network is a type of neural network architecture that contains the two more identical subnetworks which are largely used to find the distance between two inputs thus can be used to find the similarity and dissimilarity between two inputs. In the literature, this approach is used to classify the handwritten characters of the Omniglot dataset. Omniglot dataset [14] is a collection of 50 alphabets from around the world which designed to develop human-like algorithms by providing a minimal amount of data for each character.

Another concept that is used in the facial recognition context is deep metric learning using a triplet network [8]. It tries to reduce the eucleadian distance between similar objects and at the same time increases the distance between different objects. Anchor, positive, negative are the three main pointers used in this algorithm and triplet loss tries to reduce the distance between the anchor and positive and increase the distance between the anchor and negative. In the facial recognition context, the anchor is the face of a person and positive is another image of the same person, negative is a very different face than the anchor.

FaceNet [1] is a facial recognition which transforms the features of a human face into 128 dimensions Euclidean space. The model generated in this approach is trained with triplet loss [8] for different classes of faces to capture the similarities and differences between them. Once the 128-dimensional embedding returned by the FaceNet model can be used to clusters faces. Once the model is trained, It uses two identical FaceNet models and uses two images inputs to check the similarities between two outputs by calculating the Euclidean distance between the two returned embeddings. And then the above-calculated difference is used to identify the person. And if Euclidean difference is lower FaceNet successfully recognizes the face. The training process of the model using the above approach requires a considerable amount of data to accurately identify the features of a human face. Hence as a performance enhancement, the creators have split the training dataset by randomly named them as mini-batches. For the triplet selection process, the positives and negatives which reside inside mini-batch are randomly considered.

To optimize the current high computational complexity triplet selection process,

there are some approaches mentioned in the literature. Incremental group-wise training [15] which map the original image to compact binary codes via a deep convolutional neural network proved to reduce the total training time of a model.

2.3 Ethnicity Bias

A study has been carried out to evaluate the bias present in the automated facial analysis algorithms and datasets which are commercially available [5]. They approached the issues with two frontiers, one which found the bias present in the datasets and others with finding the ethnic and gender bias with facial recognition algorithms. In the first approach, a Fitzpatrick skin type classifier is used to classify the two facial analysis benchmark datasets which namely IJB-A and Adience. Through the classification process, it revealed that respectively 79.6% and 86.2% are light-skinned subjects. From building upon the observations a facial analysis dataset has been created which is balanced by gender and skin type. Which then later used to evaluate the gender and ethnicity bias on commercially available facial recognition algorithms. The evaluation was carried out on three main service providers who are namely Face++, Microsoft and IBM. From the analysis of the results, it is observed that darker-skinned females are the most misclassified group and with error rates up to 34.7% while the maximum error rate for lighter-skinned males was 0.8%.

The literature presents another approach which was used to measure the gender and ethnicity bias on deep models for facial recognition [6]. The objective was to measure what extent gender and ethnicity information are present in the feature vectors generated by VGGFace [16] and Resnet50 facial recognition models. The research focused on two approaches, fixed classification and retrained classification. In fixed classification, the last layer was stripped from the above facial recognition models and added a fully connected layer in between to classify the attributes. Then iteratively suppressed the most relevant and common features from the output and carry out attribute classification and identity verification. The goal was to test whether a high performance can be achieved in the verification while suppressing the features related to gender or ethnicity. In the retrained classification, the suppressed feature vector is used and the attribute classification was carried out on it by after retraining the attribute classification layer using the remaining features.

Correction of the bias that occurs when the selection of the triplets [7] is explored in the literature to a certain extent. The objective of this research was to provide an efficient solution to triplet selection that grows with the size of the dataset and how to minimize the slow convergence speed when random triplets

were selected. Proposed a new triplet loss function named adapted triplet loss which in fact tries to reduce the bias by adaptively correcting the distribution shift on the selected triplets.

2.4 Ethnicity Classification

Xiaoguang et.al [17] proposed a methodology for gender and ethnic classification of a human face using 3D images. To reduce the complexity of the problem, authors only used binary classification for the ethnicity classification problem by limiting the number of classes to Asian and Non-Asian. From a 3D image, authors have extracted the range and intensity features and after a process of 3D normalization used SVM (Support Vector Machine) to predict the gender and ethnicity. The evaluation was done using UND and MSU datasets and recorded 2.0% average and standard deviation of the error rates using 10-fold.

Mohammad et.al [18] has proposed a methodology to detect ethnicities by using learning-based classifiers. By extracting local features from Local Binary Pattern (LBP) and Histogram of Oriented Gradient (HOG) authors fused them and fed into four classifiers namely support vector machine (SVM), Multi-Layer Perceptron (MLP), Linear Discriminant Analysis (LDA), and Quadratic Discriminant Analysis (QDA). The process is evaluated using a FERET dataset and recorded test performance accuracy of 98.5% using SVM. Yet the authors have pointed out that in some instances when classifying images of a certain ethnicity, accuracy became as low as 28%.

Works of Simonyan et.al [19] have presented the model VGG16 for large scale image recognition. Authors have investigated the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting and found that depth is beneficial for the classification accuracy. Hence they've created a neural network architecture which model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.

2.5 Datasets

UTKFace [3] dataset is a large-scale face dataset that contains the identities of people with different ethnic attributes. It contains images of a long age span (range from 0 to 116 years old). The dataset consists of over 20,000 faces with annotations of age, gender, and ethnicity. The images cover large variations in pose, facial expression, illumination, occlusion, resolution. Since UTKFace is considered as one of the largest datasets of ethnicity annotated images, it can be utilized for benchmarking tasks of the neural network models which addresses the ethnicities.

Labeled Faces in the Wild [20] is a benchmark dataset for face verification. The dataset contains more than 13,000 images that are gathered from the web and each image is labeled with the name of the person pictured. From the 13,000 images, more than 1680 people (identities) have more than two images. Since the face detection done by using the Viola-Jones face detector, the authors have identified it as a drawback because of the error rate of the face detector. LFW dataset has four variations, which contain aligned and funneled images. The variation provided by Huang et.al [21] is currently considered as the state of the art benchmark dataset for face verification.

CASIA-WebFace [22] is a large scale dataset including about 10,000 subjects and 500,000 images of faces. It is considered as the second largest face dataset and the largest publicly available face dataset. The faces are extracted from a web crawler program and some images were labeled using the IMDB (International Movie Data Base) web page semi-automatically.

MTCNN [23] is a neural network approach that is used for facial recognition. The network uses a cascade structure with three networks which used to find the bounding boxes of a face when given an image. Firstly the image is rescaled into different sizes and the image is passed on to the subsequent neural networks. The first model proposes the candidate facial regions while the second model filters the bounding boxes. Then the third model is used to propose facial landmarks of an image. The main usage of the MTCNN approach is to act as a library for face alignments when given a normal image of a person.

Chapter 3

Design

3.1 Introduction

In this chapter, It is addressed how the optimization of dataset chunk selection is carried out by generating the ethnicity distribution of the training dataset. And whether chunking the dataset according to the ethnicity distribution of the dataset can improve the triplet selection and thus improving the total accuracy of the model. The design of the ethnicity classifier which is used to generate the ethnicity distribution of the training stage and a structural review of the datasets used for the train and evaluation stages of the proposed approach is presented in this section.

3.2 Facial Recognition

3.2.1 The existing approach

According to the literature, the Facenet approach [1] chunks its input data by randomly selecting the elements from the dataset which are called mini-batches. And then those mini-batches are fed to the neural network which extracts and adjusts the weights according to the facial features. And then outputs a feature vector of 128 dimensions that contains the values for major aspects of a face.

The above-selected mini-batches in Figure 3.1 are trained using the triplet loss function which is used to minimize the distance between baseline input to the positive input and maximize the distance between baseline input to the negative input.

When the triplet (3.2) loss training used in the context of facial recognition the anchor becomes the face of a person and the positive becomes another image of the same person. The negative will the furthest image/face that exists in the dataset for the selected anchor.

Figure 3.1: High level diagram of training process

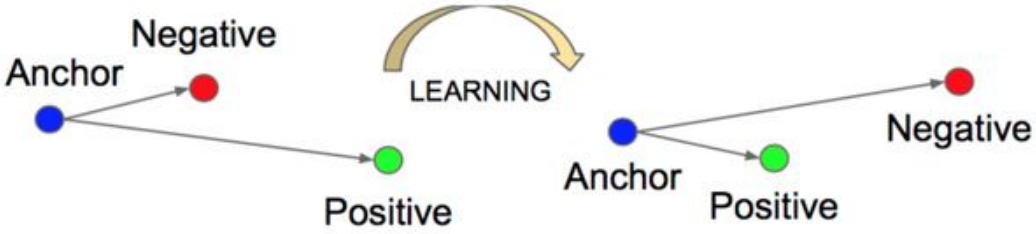

Figure 2: Triplet Loss Training

Figure 3.2: Triplet Selection

The approach by Facenet [1] uses triplet loss training for the calculation of the loss function which considers the mini-batch to adjust the weights of the neural network. Since the usage of mini-batches to calculate loss, the hard negative element is chosen from the local mini-batch, although there exists a true hard negative element on the dataset but assigned to another mini-batch.

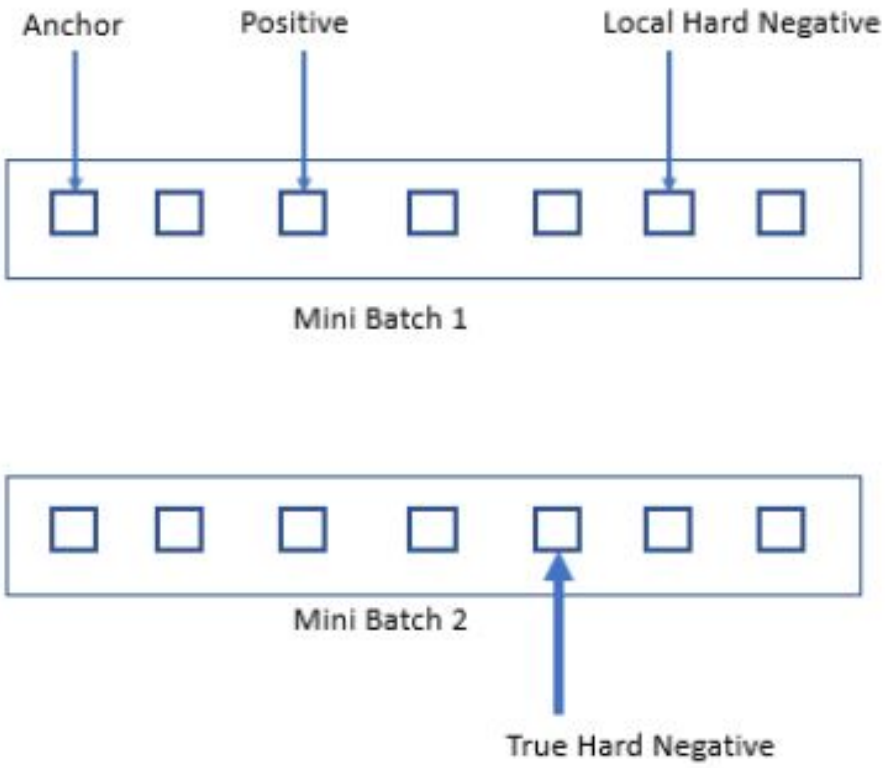

Figure 3.3: Triplet Selection on a mini batch

3.2.2 Proposed approach

To address the above issue of selecting the true hard negative element following approach is proposed. The proposed approach consists of three main stages.

1. Align the faces of the dataset as a pre-processing step
2. Label the training dataset by considering the ethnicity
3. Create a lookup table contains each face's ethnicity and confidence level of the ethnicity
4. Select dataset chunks by considering the ethnic distribution on the training dataset

3.2.2.1 Align the faces of the dataset

As a preprocessing step the images are subjected to alignment and to cropping stage to extract only the portion of the image that contain the facial area. To accomplish this task as described in the chapter 2, MTCNN approach with pre-trained weights are used.

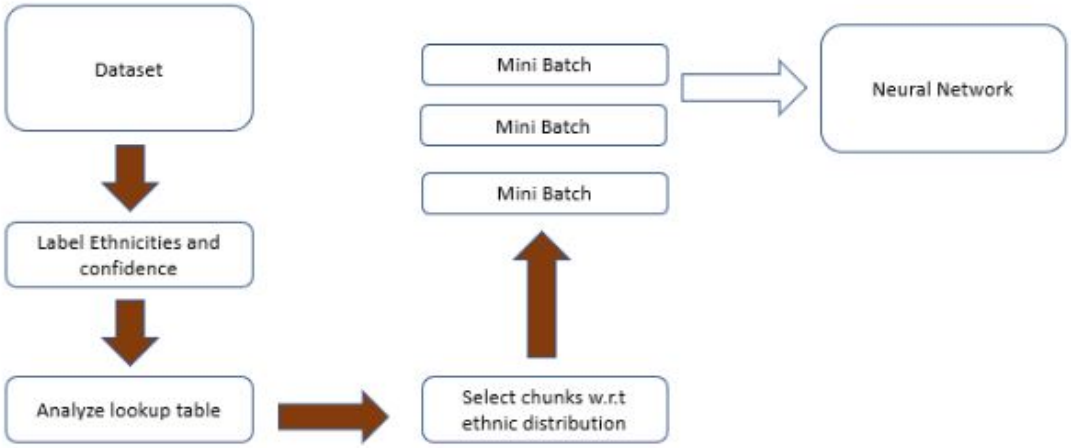

Figure 3.4: Process Flow of Proposed Approach

3.2.2.2 Label the training dataset by considering the ethnicity

To label the faces of the dataset, the approach of Alejandro et.al [6] will be used. Although this approach records 85% on LFW dataset, using it is justifiable because the triplet loss training does not use punishment for false triplets. It just records the loss as 0 and moves on to the next triplet.

3.2.2.3 Create a lookup table contains each face's ethnicity and confidence level of the ethnicity

Class Name	Unique ID	Ethnicity	Confidence
Person1	Person1_01	Caucasian	0.85
Person1	Person1_02	Caucasian	0.64
Person2	Person2_01	Indian	0.78
Person3	Person3_01	Asian	0.66

Figure 3.5: Lookup table

In this step, the labels generated by the previous step and their confidence percentages will be stored on a lookup table. In the lookup table, there is a record for each and every image on the dataset.

The ethnicity taxonomy that was proposed by Gonzales et.al [2] in the literature will be used for the labeling process.

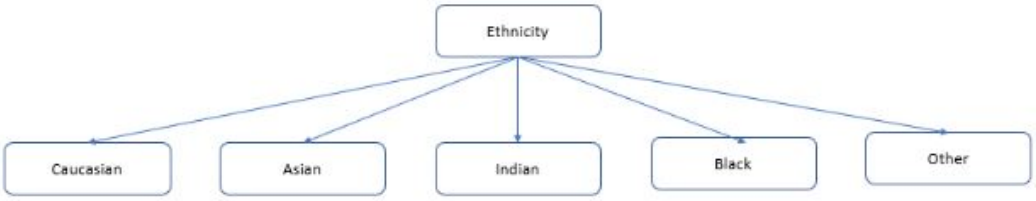

Figure 3.6: Ethnicity Taxonomy [2]

3.2.2.4 Selection of Mini-batches

In the literature, multiple methods have been mentioned as previously stated in chapter 2 to select a sample from a dataset. Sampling can mainly be divided into probabilistic and non-probabilistic sampling methods. From those, non-probabilistic sampling is not considered as a sampling method because the selection method of non-probabilistic methods relies on the judgment of the researcher rather than the distribution of the dataset.

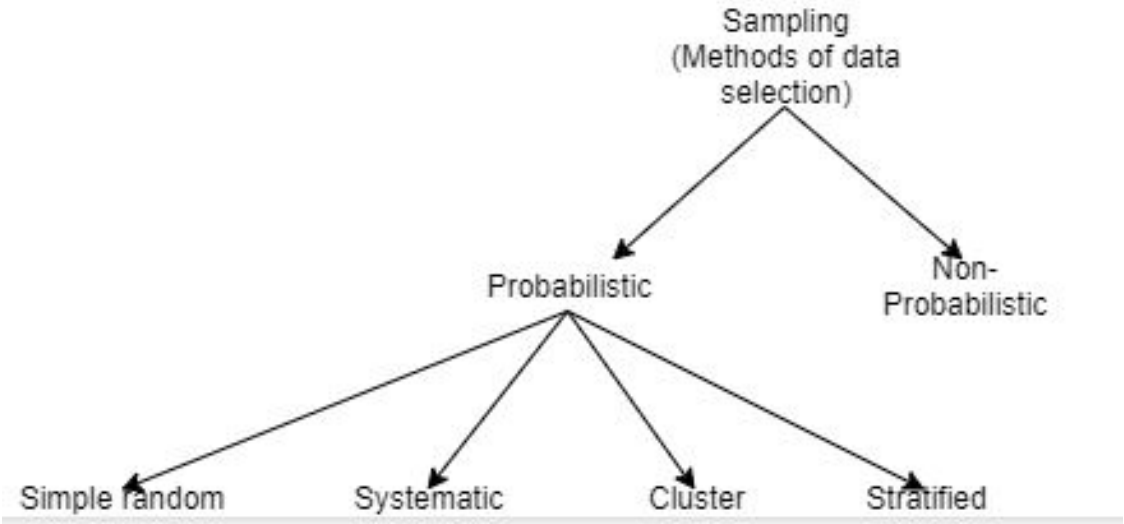

Figure 3.7: Sampling methods taxonomy

Different probability sampling methods are considered to select the dataset chunks for the training process such that dataset chunks represent the ethnicity distribution. Systematic sampling was discarded because of its tendency to select data points that are close to each other which outcomes as not random sampling. Although the cluster sampling method can be used to cluster the dataset by ethnicity and then select the data chunks is feasible, it was discarded because of its complexity and the tendency to error when sampling. To select the data chunks according to the ethnicity distribution of the dataset, the stratified sampling method

Table 3.1: Comparison of Sampling methods

	Pros	**Cons**
Simple random	Represents the whole population Unbiased random selection	Needs a complete list of the members on the population (dataset)
Systematic sampling	Spreads the sample more evenly over the population	In some instances, the sampling will not be random
Cluster sampling	Can apply Multi-Stage Sampling Smaller search space	Sampling errors
Stratified sampling	Reduced potential for human bias in the selection Represents the distribution of the population	Infeasible to make the subgroup sample sizes proportional

was used over the simple random sampling method mainly because of its characteristic of representation of the distribution of the population.

By considering the distribution of ethnicities in the lookup table (figure 3.5), and ethnicity distribution for the whole dataset can be derived. Which then used to create the dataset chunks that contain the same ethnicity distribution as the initial dataset by using the stratified sampling method.

3.3 Ethnicity Detection

3.3.1 Previous approaches from literature

As stated in the chapter 2, there are many attempts in the literature to classify or predict the ethnicity of a person in an image. Yet most of the approaches are limited in scope to detect a focused number of ethnicities [24]. Hence a need of an ethnicity classifier that represents the mainly recognized ethnicities [2] mentioned in figure 3.6 arised to label the ethnicities in the training dataset. Table 3.1 covers the comparison of the different approaches in the literature to classify images considering the ethnicity.

Table 3.2: Comparison of Ethnicity Classifier

Approach	**Accuracy**	**Training Dataset**	**Number of Ethnicities considered**
Xiaoguang et.al [17]	3.2% (Intensity)	UND [25] MSU[26]	2 (Asian, Non Asian)
Saeed et.al [18]	28% (Accuracy- Middle eastern)	FERET [27]	4
Proposed approach	Discussed in Section 5	UTKFace [3]	5 (Caucasian, Black, Asian, Indian, Other)

This section covers the ethnicity classifier which is used to label the images mentioned in section 3.2.2.1

3.3.2 Proposed Model

A neural network that takes images as an input and returns the label (ethnicity) of the person present in the image. In the following sections design information about the dataset used to train and test the model, pre-processing steps carried out, the architecture of the model and training constraints used will be described.

3.3.2.1 Dataset

UTKFace dataset is a large-scale face dataset that contains the identities of people with different ethnic attributes. It contains images of long age span (range from 0 to 116 years old). The dataset consists of over 20,000 faces with annotations of age,

gender, and ethnicity. The images cover large variation in pose, facial expression, illumination, occlusion, resolution, etc (figure 3.8).

Figure 3.8: Overview of the UTKFace dataset [3]

3.3.2.2 Pre-Processing

UTKFace dataset is annotated with 5 classes of ethnicities. Namely white, black, asian, indian, and other (like hispanic, latino, middle eastern). Since the goal of this stage is to classify the faces into the above mentioned 5 ethnicities, the dataset was divided into 5 classes considering the ethnicity of the image.

A 70/30 train/test split was used to divide the dataset for training and validation tasks. Although the dataset represented major 5 ethnicities, when divided into five classes a higher class-imbalance was observed with hispanic identities versus other identities. To fight the class-imbalance issue, a normalization step is carried out to resample the training dataset.

Although the original dataset contains the full images of a person (figure 3.8), the authors [3] used dlib [28] a machine learning toolkit to align and crop the images and extract only the area which contains the faces (figure 3.9). Hence there was no need for image alignment and cropping.

3.3.2.3 Neural Network Model architecture

For the neural network model, VGG16 [19] a successor to AlexNet 3.10). Since the ethnicity classification is done only for 5 classes the final layer has been modified to output a 1x5 feature vector which then can be used to make predictions.

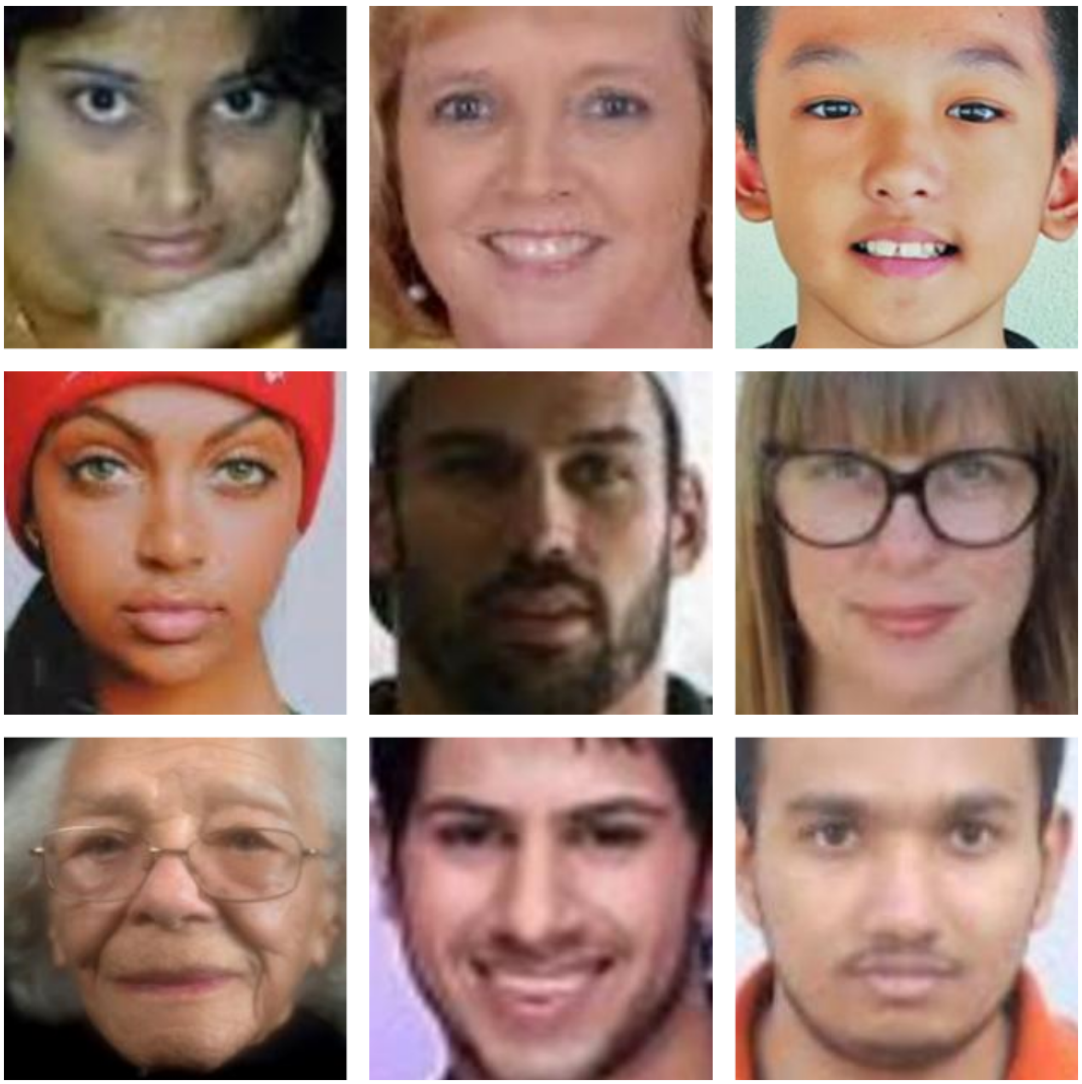

Figure 3.9: Samples of ethnicities in UTKFace dataset [3]

The above mentioned model is then used to train with the preprocessed dataset in section 3.3.2.2 while using ADAM [29] as its optimizer.

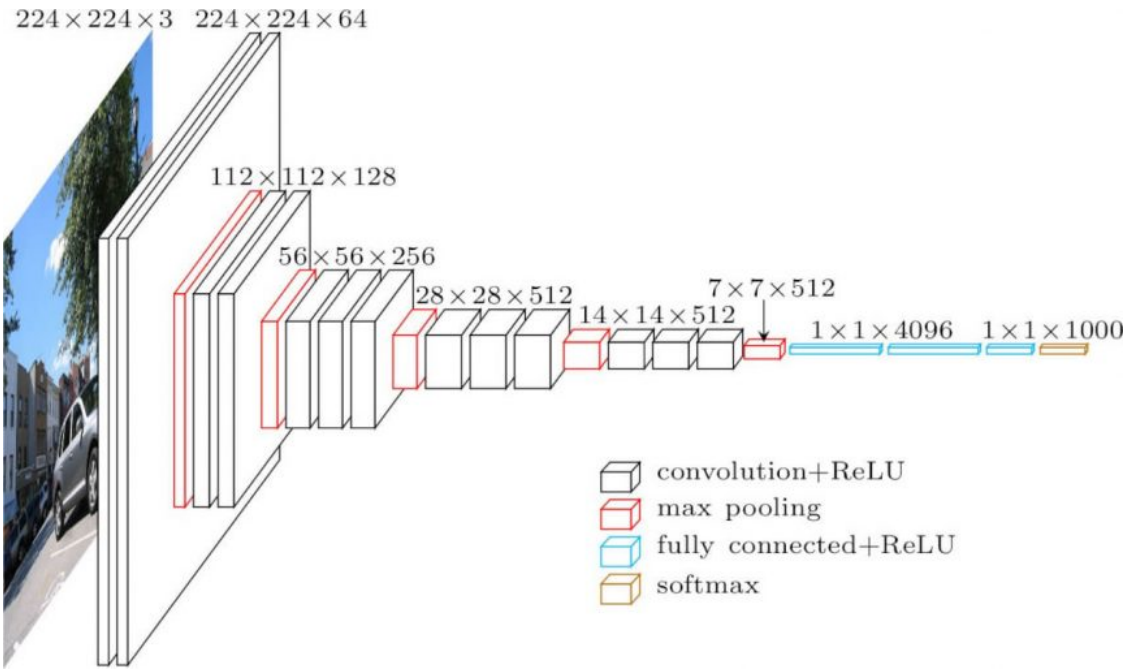

Figure 3.10: VGG16 Model Architecture [4]

3.4 Datasets

As mentioned in the chapter 2 the following datasets are being used for the above proposed design architecture.

Table 3.3: Dataset Statistics

	Number of images	**Number of unique identities**	**Usage of the dataset**	**Remarks**
UTKFace	20,000+ images	Not specified. Images with ages of 0 to 116 are included	Used to test and train the ethnicity classifier	
CASIA	494,414 images	10,575 identities	Used to train the proposed modified approach for facial recognition	Considered as the largest publicly available face dataset
LFW	13,000+ images	1680+ identities	Used to verify the trained facenet model and used to classification tasks which carried out to evaluate the trained model	Considered as the benchmarking dataset for facial recognition approaches

3.5 Conclusion for the Chapter

In this chapter the existing facial recognition approach and the current data selection method for training the model is described. Building upon that, a proposed methodology is introduced which contains the alignment of faces in the dataset, labeling the faces on the dataset and building a lookup table for each face. Different sampling methods from the literature is analysed to select the best sampling method to build data chunks for neural network model training. Building upon the works from the literature, an ethnicity classifier is proposed to label the data chunk which was generated from the previous stages. A model based on VGG16 [19] neural network architecture and UTKFace [3] a dataset which contains a variety of ethnicities is used in this stage.

Chapter 4

Implementation

The implementation process can be divided into two main segments that cover

1. Ethnicity detection
2. Facial recognition

4.1 Ethnicity detection

4.1.1 Explore the UTKFace dataset

In this step (figure 4.1), the dataset is exploration is done to find out the folder structure, the image formats and the characteristics and the content of the images.

```
[ ]  !tar -xvzf /content/utkface.tar.gz -C /content/raw
```

```
[ ]  import os

     path = '/content/raw'

     files = folders = 0

     for _, dirnames, filenames in os.walk(path):
       # ^ this idiom means "we won't be using this value"
         files += len(filenames)
         folders += len(dirnames)

     print("{:,} files, {:,} folders".format(files, folders))
```

```
23,708 files, 1 folders
```

Figure 4.1: Exploring the UTKFace dataset

Since all 23708 images are on the same folder it is infeasible to train a model

that can classify the images into 5 classes. Hence the dataset needs to go through a preprocessing stage.

Also figure 4.2 shows the naming convention used to annotate the images on the dataset. The naming convention format used is as below. And the Table 4.1.1 provides the descriptive information of the naming convention.

[age]_[gender]_[race]_[date&time].jpg

Table 4.1: Naming Convention description

[age]	Integer value 1-116 indicating the age
[gender]	0 (male) or 1 (female)
[race]	Integer value in range of 0-4 Denoting White, Black, Asian, Indian, and Others (like Hispanic, Latino, Middle Eastern)
[date&time]	the date and time when the image was collected to UTKFace in 'yyyymmddHHMMSSFFF' format

Figure 4.2: UTKFace naming convention

4.1.2 Preprocessing the dataset

```
[ ]  image_counter = [0,0,0,0,0]
```

```
[ ]  def get_race(img_name):
         try:
             race = int(img_name.split("_")[2])
             #print(race)
         except:
             f.write(img_name + ' \n')
             race = -1

         return race

     def createImageName(race):
         count = image_counter[race]

         temp_x = len(str(count))
         file_name = str(race)

         if(temp_x == 1):
             file_name = file_name + "0000" + str(count) + ".jpg"
         elif(temp_x == 2):
             file_name = file_name + "000" + str(count)+ ".jpg"
         elif(temp_x == 3):
             file_name = file_name + "00" + str(count)+ ".jpg"
         elif(temp_x == 4):
             file_name = file_name + "0" + str(count)+ ".jpg"
         else:
             file_name = file_name + str(count)+ ".jpg"
         return file_name
```

Figure 4.3: PreProcessing Implementation

The variable "image_counter" is an initialization variable that is used to store the total number of images per class in the whole dataset. That variable is used mainly to generate the next image name by keeping a record of the previous image name. Figures 4.3 and 4.4 describes the process of iterating the whole dataset and by extracting the ethnicity from the image annotations and then it is used to move images to different classes.

Figure 4.5 contains the code snippet that is used to split the dataset into training and test segments. 70 percent training and 30 percent test split is used to divide the whole dataset into two segments. The data selection is done randomly to keep the diversity among the dataset, because the initial dataset is sorted by the age, hence there is a chance that the train split set can only contain the images of the

```
[ ] #getting the all files
    mypath = "/content/raw/UTKFace"
    onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f))]
    for image in onlyfiles:
        race = get_race(image)
        if(race > -1):
            filename = createImageName(race)

            shutil.copy2('/content/raw/UTKFace/'+image, '/content/filtered/'+str(race)+"/"+filename) # complete target filename given

            image_counter[race] = image_counter[race] + 1

    f.close()  # you can omit in most cases as the destructor will call it
    print("COMPLETE")
```

```
COMPLETE
```

```
[ ] import os

    path = '/content/filtered'

    files = folders = 0

    for _, dirnames, filenames in os.walk(path):
      # ^ this idiom means "we won't be using this value"
        files += len(filenames)
        folders += len(dirnames)

    print("{:,} files, {:,} folders".format(files, folders))
```

```
23,705 files, 5 folders
```

Figure 4.4: PreProcessing Implementation

young individuals and at the same time test set can only contain the images of the old individuals.

```
[ ]  import math

     base_path = '/content/filtered/'
     count = 0
     for i in range(0,5):
       cur_path = base_path + str(i)
       dest_path_base = '/content/splited'

       onlyfiles = [f for f in listdir(cur_path) if isfile(join(cur_path, f))]

       len_of_files = len(onlyfiles)

       #70/30 split
       train_value = math.floor(len_of_files * 0.7)
       test_value = math.floor(len_of_files * 0.3)

       onlyfiles = np.array(onlyfiles)

       training = onlyfiles[:train_value]
       test = onlyfiles[train_value:]
       # print(len(onlyfiles), len(training), len(test), len(training) + len(test))

       for image in training:
         im_path = cur_path + "/" + image
         dest_path = dest_path_base + "/train/"+ str(i) + "/" + image
         count = count +1
         shutil.copy2(im_path, dest_path)

       for image in test:
         im_path = cur_path + "/" + image
         dest_path = dest_path_base + "/test/"+ str(i) + "/" + image
         count = count +1
         shutil.copy2(im_path, dest_path)
     print(count)
```

Figure 4.5: PreProcessing Implementation

4.1.3 Model Training

```
[ ] import keras,os
    from keras.models import Sequential
    from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
    from keras.preprocessing.image import ImageDataGenerator
    import numpy as np
```

```
Using TensorFlow backend.
The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the %tensorflow_version 1.x magic: more info.
```

```
[ ] train_data_dir = '/content/splited/train'
    validation_data_dir = '/content/splited/test'
```

```
[ ] trdata = ImageDataGenerator()
    traindata = trdata.flow_from_directory(directory=train_data_dir,target_size=(224,224))
    tsdata = ImageDataGenerator()
    testdata = tsdata.flow_from_directory(directory=validation_data_dir, target_size=(224,224))
```

```
Found 16591 images belonging to 5 classes.
Found 7114 images belonging to 5 classes.
```

Figure 4.6: Model Training

Figure 4.6 proveds the code snippet used to process the train and test data segments which were splitted from the previous stage is resized into 224 by 224 (224x224) dimension so that it can be used to feed into the VGG16 model.

Figure 4.7 provides the implementation of the VGG16 model which is mentioned in the chapter 2. The input for this model is selected as 224x224 dimension images because the images in the training dataset are of the dimension above. As the final output layer a dense layer of 5 units have been added because the ethnicity classification's aim is to predict the the ethnicity of main five ethnicities.

Figure 4.8 shows the usage of the Adam [29] optimizer for the compilation of the above implemented keras model. With a fine tuning process, it is found that the learning rate of 0.01 is the feasible value to reach to convergence.

Figure 4.9 focuses on the visualization of the evaluation results from the test dataset and more features to increase the reliability of the training process. Features like checkpoints are used to continue the training process if some interruption occurred while running the training script. The function EarlyStopping from keras library is used to track the increase of the accuracy and stop the training process if there are not any improvements to the accuracy metric in 20 epochs (iterations).

```
#VGG16 Model
model = Sequential()
model.add(Conv2D(input_shape=(224,224,3),filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(Conv2D(filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(Flatten())
model.add(Dense(units=4096,activation="relu"))
model.add(Dense(units=4096,activation="relu"))
model.add(Dense(units=5, activation="softmax"))
```

Figure 4.7: VGG Model Structure Implementation

```
from keras.optimizers import Adam
# opt = Adam(lr=0.001)
opt = Adam(lr=0.01)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy, metrics=['accuracy'])
```

Figure 4.8: Usage of Adam optimizer

```
from keras.callbacks import ModelCheckpoint, EarlyStopping

checkpoint = ModelCheckpoint("vgg16_1.h5", monitor='val_acc', verbose=1, save_best_only=True, save_weights_only=False, mod
early = EarlyStopping(monitor='val_acc', min_delta=0, patience=20, verbose=1, mode='auto')
hist = model.fit_generator(steps_per_epoch=100,generator=traindata, validation_data= testdata, validation_steps=10,epochs=
```

Figure 4.9: Usage of EarlyStopping, histogram generation

4.2 Facial recognition

4.2.1 Building data table containing ethnicities for a dataset

Figure 4.10 shows the logic that was used to build a data table that is created to store the ethnicities for each class (identity). When given a dataset following function lists all the classes and the paths of the images that reside inside of them. An array of size four is used to store the predictions because as mentioned in chapter 3 the selected approach for predicting the ethnicities supports mainly four races. Namely Asian, Caucasian, African, Hispanic.

```
def preprocess_new(directory):
    #list all the folders in the directories
    subfolders = [f.name for f in os.scandir(directory) if f.is_dir() ]

    people_count = len(subfolders)
    i = 0
    print("Total Classes" + str(people_count))

    #loading the model
    means = np.load('/content/drive/My Drive/casia_pre/means_ethnic.npy')
    model = create_face_network(nb_class=4, hidden_dim=512, shape=(224, 224, 3))
    model.load_weights('/content/drive/My Drive/casia_pre/weights_ethnic.hdf5')

    print("Total Classes", len(subfolders))

    for personClass in subfolders:
        #finding the classname (name of the person/identity)
        class_path = directory + "/" + personClass
        images_per_class = [f for f in listdir(class_path) if isfile(join(class_path, f))]
        image_count_per_class = len(images_per_class)

        # legend for the prediction {0: 'Asian', 1: 'Caucasian', 2: "African", 3: "Hispanic"}
        prediction_array = np.array([0, 0, 0, 0])
```

Figure 4.10: Building data-table Pt.1

Figure 4.11 which is a continuation of Figure 1, is used to predict the ethnicity of a given image and store it. Here, a maximum of 10 images from a class randomly selected to use in the predictions. The reason behind this limitation is to handle the performance issues when this method is faced with a class that consists of more than 10 images. For every class, the model is used to predict the ethnicity of each and every selected image and then the mean value of the predictions is used to classify the ethnicity of the given identity. Then the record is inserted to the data table.

```
#select 10 images per class
#if there are more, select randomly
if(image_count_per_class > 10):
  images_per_class = random.sample(images_per_class , 10)

for image in images_per_class:
    #convert image into numpy array
    im_path = class_path + "/" + image
    im = cv2.imread(im_path, cv2.IMREAD_COLOR)
    im = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    im = cv2.cvtColor(im, cv2.COLOR_GRAY2RGB)
    im = cv2.resize(im, (224, 224))
    im = np.float64(im)
    im /= 255.0
    im = im - means

    #get predictions
    pred = model.predict(np.array([im]))[0]
    prediction_array = prediction_array + np.array(pred)

#getting the average and get the max
prediction_array = prediction_array / float(image_count_per_class)
result = np.argmax(prediction_array)
temp_prsn = Person(personClass, int(image_count_per_class), int(result), class_path)

#add to the database
insert_record(temp_prsn)
```

Figure 4.11: Building data-table Pt.2

Figure 4.12 shows the structure of the data table that is created in this stage. It consists of details of classname, number of images per class, ethnicity which was predicted from above Figure 2 and the relative file path for the classname for ease of access. Figure 4 displays the top records from the above-generated data table

Class Name	Unique ID	Ethnicity	Confidence
Person1	Person1_01	Caucasian	0.85
Person1	Person1_02	Caucasian	0.64
Person2	Person2_01	Indian	0.78
Person3	Person3_01	Asian	0.66

Figure 4.12: Implemented data table

	id	classname	no_of_imgs	race	path	count
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1165660	82	1	/content/casia...	0
2	2	0477127	157	0	/content/casia...	0
3	3	1679452	19	3	/content/casia...	0
4	4	4483476	53	1	/content/casia...	0
5	5	2997484	34	2	/content/casia...	0
6	6	0001642	79	0	/content/casia...	0
7	7	1577190	157	2	/content/casia...	0
8	8	5431622	18	2	/content/casia...	0
9	9	0396924	44	1	/content/casia...	0
10	10	0666398	30	2	/content/casia...	0
11	11	3347140	36	2	/content/casia...	0
12	12	0504897	28	0	/content/casia...	0

Figure 4.13: Implemented data table

4.2.2 Implementation of the proposed design

Figure 4.14 covers the implementation aspect of the proposed data selection process for training the model. At the initialization step, a quota is allocated for each ethnicity considering the ethnicity distribution of the dataset which is being used to train. For each data chunk selected, the size of the chunk is estimated from the initial parameters namely people_per_batch which represents the number of people for a given batch, images_per_person which represents the number of images for a person. Thus both require the average values of image count in the training dataset.

```
def sample_people_moded(dataset, people_per_batch, images_per_person, ethnic_distribution):
    nrof_images = people_per_batch * images_per_person

    #Calculating images needed
    asian_count = math.floor(ethnic_distribution[0] * nrof_images * 0.01)
    caucasian_count = math.floor(ethnic_distribution[1] * nrof_images * 0.01)
    african_count = math.floor(ethnic_distribution[2] * nrof_images * 0.01)
    hispanic_count = math.floor(ethnic_distribution[3] * nrof_images * 0.01)

    #updating the new "needed image count"
    temp_x = asian_count + caucasian_count + african_count + hispanic_count
    diff = nrof_images - temp_x
    asian_count = asian_count + diff

    image_paths = []
    num_per_class = []
```

Figure 4.14: Proposed approach implementation Pt.1

Figure 4.15 reveals the logic of selection of images considering the quota for the ethnicities which they are allocated. Here a random variable is used to choose which image from which ethnicity at each iteration of the loop. The idea behind that is to maintain a diversity of identities on the selected data chunk.

```
while len(image_paths)<nrof_images:
    random_number = random.randint(0,3)

    if(random_number == 0 and asian_count > 0):
        random_element = random.randint(0, len(asian_records) - 1)
        picked = asian_records[random_element]
        person_classname = picked[1]
        nrof_images_in_class = picked[2]
        #path = picked[4]
        path = '/content/casia_dataset/extracted/CASIA-WebFace/' + person_classname

        # creating array for the number of images per class  [0 1 2 3]
        image_indices = np.arange(nrof_images_in_class)

        #shuffle
        np.random.shuffle(image_indices)

        try:
          onlyfiles = [f for f in listdir(path) if isfile(join(path, f))]

          nrof_images_from_class = min(nrof_images_in_class, images_per_person, nrof_images-len(image_paths))

          #getting only the min amount of images from index array
          idx = image_indices[0:nrof_images_from_class]

          image_paths_for_class = []
          for j in idx:
              img_path = path + "/" + onlyfiles[j]
              image_paths_for_class.append(img_path)

          image_paths += image_paths_for_class

          asian_count = asian_count - len(idx)

          num_per_class.append(nrof_images_from_class)
        except:
          print("Error in "+ path)
```

Figure 4.15: Proposed approach implementation Pt.2

4.3 Conclusion for the Chapter

In this chapter implementation aspects of the proposed design is discussed. On the first section, implementation of the ethnicity detection classifier is presented. Initially, a script was ran to explore the UTKFace [3] dataset and to pre-process the data such that they are categorized correctly in to their respective ethnicity labels. Then using the preprocessed data, an neural network based on VGG16 [19] architecture is trained and Adam [29] is used as an activation function for this task. Second section of this chapter focuses on the implementation aspects of the proposed facial recognition model. Starting from the step of building a data table containing the ethnicities for a training dataset, various decisions were taken into pre-process the dataset. Using the built data table, and selected sampling method from previous chapter, the proposed approach was implemented.

Chapter 5

Results and Evaluation

5.1 Introduction

The evaluation process can be mainly divided into two paths. One evaluating the ethnicity classification and another one evaluating the improvements to the facial recognition when the proposed method have applied. Figure 5.1 is used to summarize the evaluation process.

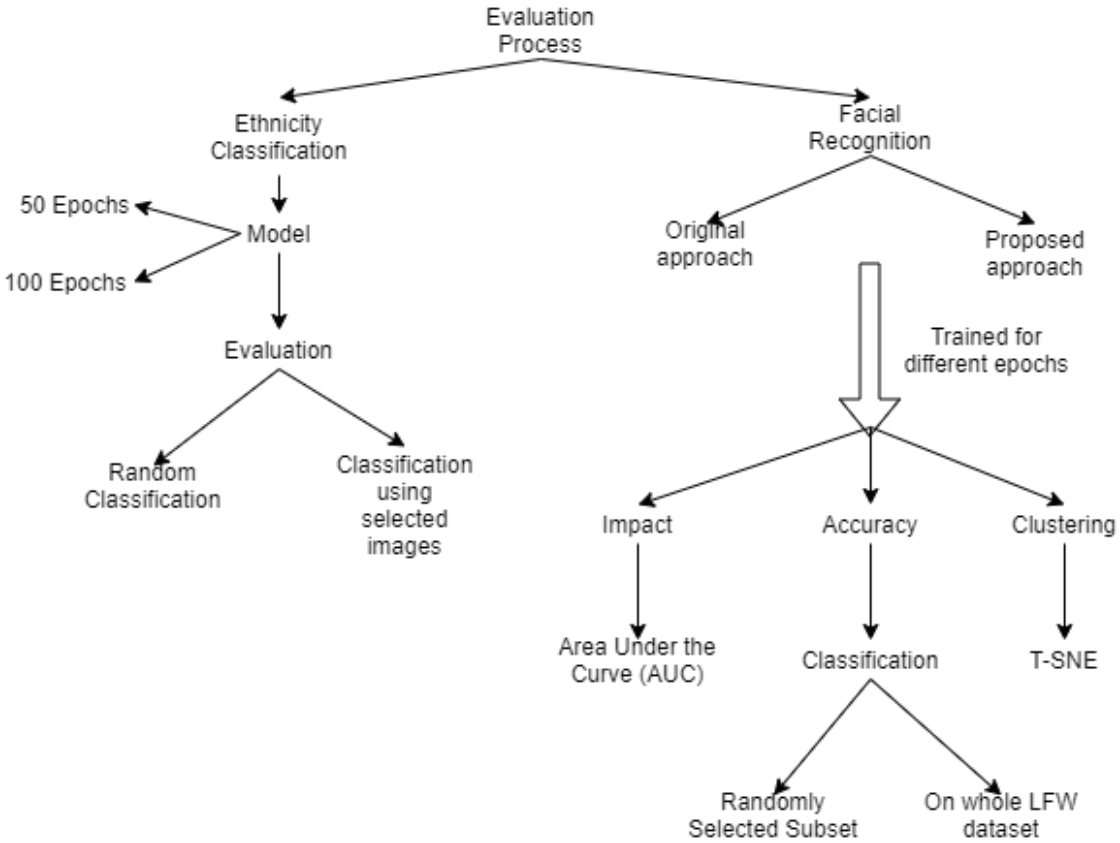

Figure 5.1: Paths of the evaluation

5.2 Ethnicity Classification

As mentioned in the chapter 3 the model will be trained for 50 epochs and 100 epoches while considering the early termination. That means if there is no change in accuracy for a certain number of epochs observed then the training process is terminated. Although the model was trained on 50 or 100, the early termination function trigger before reaching the 50 epoch mark. The convergence epoch number was between 27-41 epochs and since there was no improvement of the accuracy is discovered, the weights from the highest accuracy is used to output the final model.

The learning rate for the model is selected by going through a tuning process, which tries different learning rates and selects the best rate that maximizes the accuracy and reduces the number of epochs that needs to reach convergence. Different values between 0.001 to 0.1 were used and the model seems to provide much better results with a lower number of epochs when the learning rate of 0.01 is used. Hence the learning rate 0.01 with the Adam optimizer is used to compile and train the ethnicity classifier model.

Then the evaluation of the trained model is done by performing a classification task. Classification is also done on two datasets. One is the test data split from the UTKFace dataset which was used in the training process of the mode. The second classification task is by using the LFW dataset.

5.2.1 Classification from randomly selected images from test split

The accuracy of the trained model sits between 0.48438 and 0.51562 from the validation split using the UTKFace dataset. Figure [x] and figure [x] emphasizes the metrics like the converging epoch number, loss, validation loss, validation accuracy.

```
Epoch 22/100
100/100 [==============================] - 52s 523ms/step - loss: 9.2626 -
acc: 0.4253 - val_loss: 7.8072 - val_acc: 0.5156

Epoch 00022: val_acc improved from 0.48438 to 0.51562, saving model to
vgg16_1.h5
```

Figure 5.2: Convergence point in training

The low accuracy on the test data split is mainly due to the class imbalance issue present in the UTKFace dataset. From the five classes (main ethnicities) one has a total of 10,078 images while every other remaining four ethnicities have 13,627 images. Hense from the normalization stages, a certain portion of the dataset is

discarded to prevent the model overfitting. The lack of training data played a part in the accuracy although the classification tasks carried out using the lfw dataset produced reliable accurate predictions.

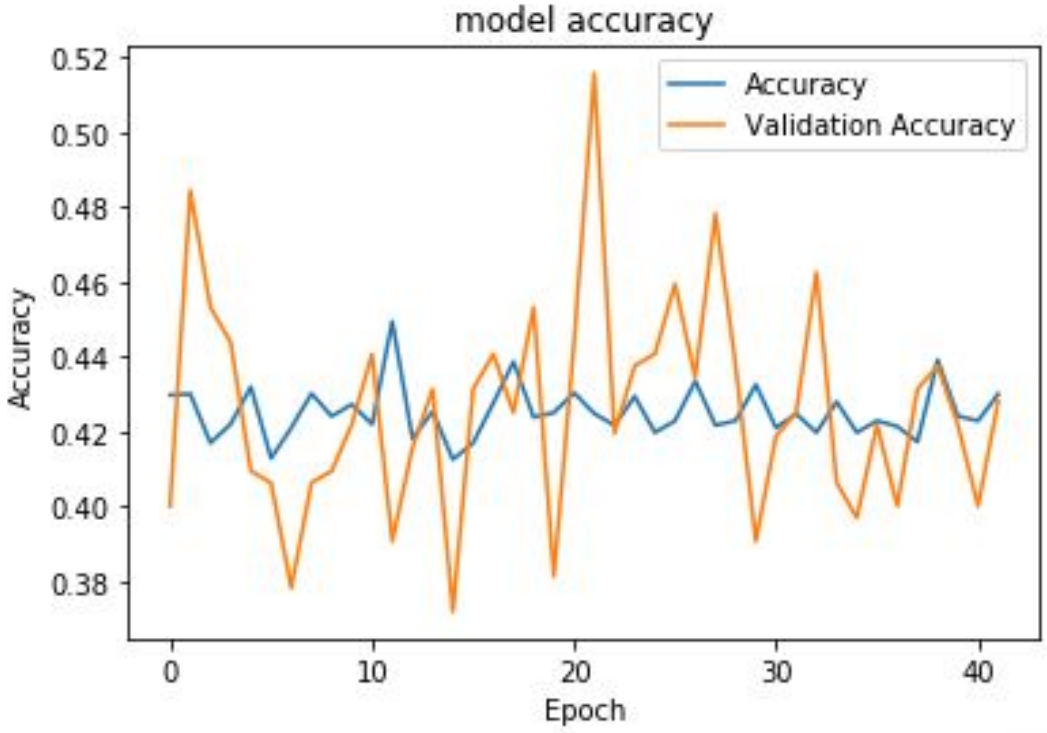

Figure 5.3: Validation Accuracy of Ethnicity Classification

5.3 Facial Recognition

Since the improvement of the accuracy and reliability of currently available Facenet architecture is a goal of this research, two models have been trained with different epochs (iterations). One model trained with the base Facenet model and the other one trained by using the modified triplet selection approach. Then the two models are subjected to evaluation methods which cover all aspects of the model.

In this section, we describe the results gathered from the evaluation process which contains the accuracy, Area Under the Curve (AUC), classification accuracy and Clustering of the embeddings from the model. Table 5.3 represents a summary of some of the above mentioned evaluation methods which will be discussed in the following chapters.

Table 5.1: Summarized Evaluation

Epoch No	Type	Accuracy	Validation Rate	Area Under Curve (AUC)
30	Base	0.66733+-0.01546	0.02267+-0.00786 @ FAR=0.00167	0.744
30	Modified	0.65583+-0.01401	0.03333+-0.01520 @ FAR=0.00100	0.659
50	Base	0.68433+-0.01236	0.05400+-0.01562 @ FAR=0.00100	0.718
50	Modified	0.66767+-0.02285	0.04067+-0.01569 @ FAR=0.00133	0.735
75	Base	0.72533+-0.01933	0.07300+-0.01676 @ FAR=0.00133	0.708
75	Modified	0.69433+-0.01767	0.04033+-0.00849 @ FAR=0.00100	0.766

5.3.1 Classification using randomly selected images from LFW dataset

The two trained models have been used to carry out a classification task using the LFW dataset. The images selected for this task are primarily selected randomly. Number of identities to classify is given as an input to the script and then extracted the following metrics namely, accuracy, validation rate, area under the curve (AUC). The figures obtained from the different approaches and different epoch numbers are provided in table 5.3.

When the accuracy metric considered the modified approach is lagging behind the original approach by a margin of 1.1% to 3.1%.

But when the metric validation rate considered FAR (False Acceptance Rate) is reduced in the modified approaches in 30 and 75 epochs. The publication of the Facenet [1] defined the FAR as False Accept Rate which is the rate of face pairs that are different and yet have a distance that is below a certain threshold (which could be classified as positive). Hence the modified approach when considered the FAR value, is improved in 2 out of 3 experiments carried out.

The reason behind shortcoming in accuracy in the modified approach is mainly because the data selection method which the Schroff et.al [1] used for the evaluation. That approach selected identities randomly rather than utilizing the whole LFW dataset.

To address this issue, an experiment is carried out to classify all identities in the LFW dataset.

5.3.2 Classification using total LFW dataset

The classification is carried out using the LFW dataset and the identities which have more than 10 images per class selected to the task. From this criteria, the total number of classes which was eligible for the task was 2402. Firstly one image from each class is selected and the embedding (feature vector of 128 elements) is generated for that image and stored using their classname. Then the embedding array for the 2402 classes were fed into a SVC (Support Vector Classifier) in Keras library and compiled and trained a model to carry out classification.

The evaluation of this model was carried out for the models trained on 30,50 and 75 epochs. In the evaluation of the model trained on 50 epochs, the optimized approach recorded 3.08% increase of accuracy. Although the total accuracy was between 20

Hence another classification was carried out by extracting the above embeddings which are used to create the SVC model. Then by iterating each class, calculated an average embedding using the rest of the images for that specific class. Then used the averaged embedding to find the closest embedding from the initially extracted embedding array. The difference is calculated using euclidean distance.

Above mentioned experiment carried out for the base and modified models which are trained on the 30 epochs, and the model which trained modified method recorded a 3% increase than it's baseline counterpart.

From analysis, the reason for behind the low accuracy values are as follows. The original Facenet approach [1] was trained on 200 million faces of 8 million unique identities. Yet the models which are used for this research have been only trained with 494,414 images of 10,575 identities. Hence the decrease in the accuracy level is expected because the model trained with CASIA have not seen much facial features like the originally published facenet model. Since the 200 million image dataset is a private repository, if there was a way to access it, then an increase of the accuracy levels is to be expected.

5.3.3 Area Under the Curve (AUC)

From the works of chapter 5.3.1 and the table 5.3 an analysis has been carried out to evaluate the AUC metric. Since facial recognition is a multi class classification problem, normal methods of evaluation are inefficient, hence the usage of the ROC (Receiver Operating Characteristics) curve is used in evaluating the multi class classifications. In ROC-AUC is considered as a performance measurement for classification problems at various threshold settings. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s. In the case of facial recognition, higher the AUC, model is better at recognizing the same person the model has previously seen before.

The table 5.2 is derived using the table 5.3 such that it contains only the AUC values for better analysis.

Table 5.2: AUC analysis

Epoch No	Type	Area Under Curve (AUC)	Increase in AUC from counterpart
30	Base	0.744	-
30	Modified	0.659	-0.085
50	Base	0.718	-
50	Modified	0.735	**0.017**
75	Base	0.708	-
75	Modified	0.766	**0.058**

It is observed that when the number of epochs increases, the increase of AUC considering it's counterpart value, increases gradually in the modified model. Which provides the ideology that the when the ethnicity is considered to train the facial recognition model, the tendency to make errors or the tendency to make the model confused with new data is gradually reduced.

5.3.4 Clustering of the output embeddings from model

The outputs of the model, which are called embeddings are used to cluster to find whether they form visible clusters. Meaning, if different images of the same person were to be fed into the model, the embeddings should be closer together and form a cluster if sufficient data points per class is present.

The clustering algorithm used for this task is t-SNE [30] (t-distributed stochastic neighbor embedding) which is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. Hence the embeddings returned from the model is of 128 dimensions, that embedding array is fed into the t-SNE algorithm. Considering the visualization clearity, a randomly selected 10 identities is used from the LFW dataset (the selected identities had a minimum number of 10 images per identity) .Figure 5.4 represents the clustering carried out using the base model which is trained on 75 epochs and the Figure 5.5 represents the counterpart of the modified model.

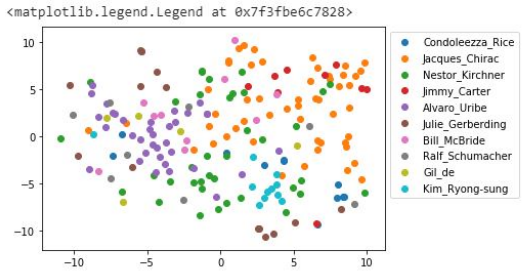

Figure 5.4: t-SNE clustering- Base Model

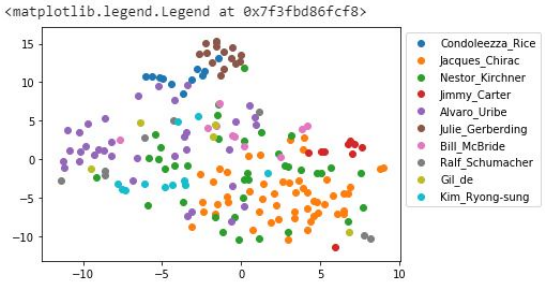

Figure 5.5: t-SNE clustering- Improved Model

By analysing the visualization it is observable that the distinction between the cluster of 'Julie_Gerberding' in the 5.4 versus 5.5 as well as the clusters of'Condoleezza_Rice', 'Alvaro_Uribe', 'Jacques_Chirac'. By analysing t-SNE clustering it is observable that the model trained with considering the ethnicity distribution can form perfect clusters but it's counterpart is struggling to do so.

The reasoning behind the selection of the clustering algorithm as t-SNE from different clustering algorithms for the above task is as belows. The main contenders for the clustering algorithm for this task were DBSCAN [31], K-Means [32], SOM [33] and t-SNE. Four experiments were conducted to select the best candidate for the above clustering of the face embeddings.

DBSCAN is an clustering algorithm which aims to guess the number of classes (clusters) when a data array is given. It groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). Embeddings of 50 identities were fed into the DBSCAN algorithm with an eps value of 0.4 and the result was a graph with 9 clusters Figure 5.6. Hence it was discarded.

K-Means is an clustering algorithm aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. Above same experiment was carried out with 50 identities and a max iteration of 500. Yet by analysing the Figure 5.7, K-Means was unable to provide a better visualization of the embeddings.

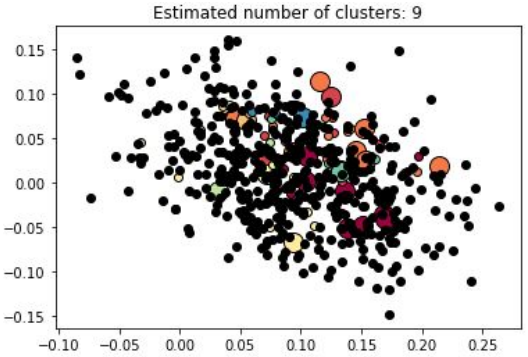

Figure 5.6: DBSCAN clustering

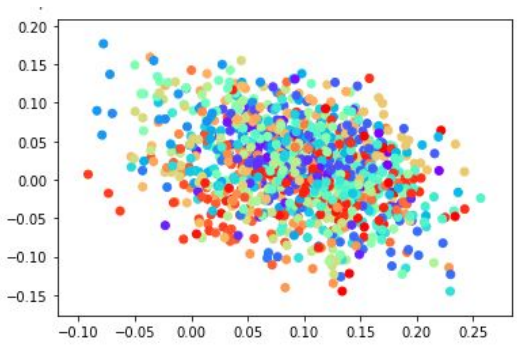

Figure 5.7: KMeans clustering

SOM is a visualization technique that helps to understand high dimensional data by reducing the dimension of data to map. SOM also represents the clustering concept by grouping similar data together. A network of 20x20 is built for the 50 classes and trained it with a learning rate of 0.01 for 1000 epochs. Yet by analysing the figure 5.8, SOM was unable to provide meaningful information regarding the 50 classes it classified.

Considering the above clustering methods, t-SNE was selected as the more informative visualization method to display higher dimension data which is returned from the model.

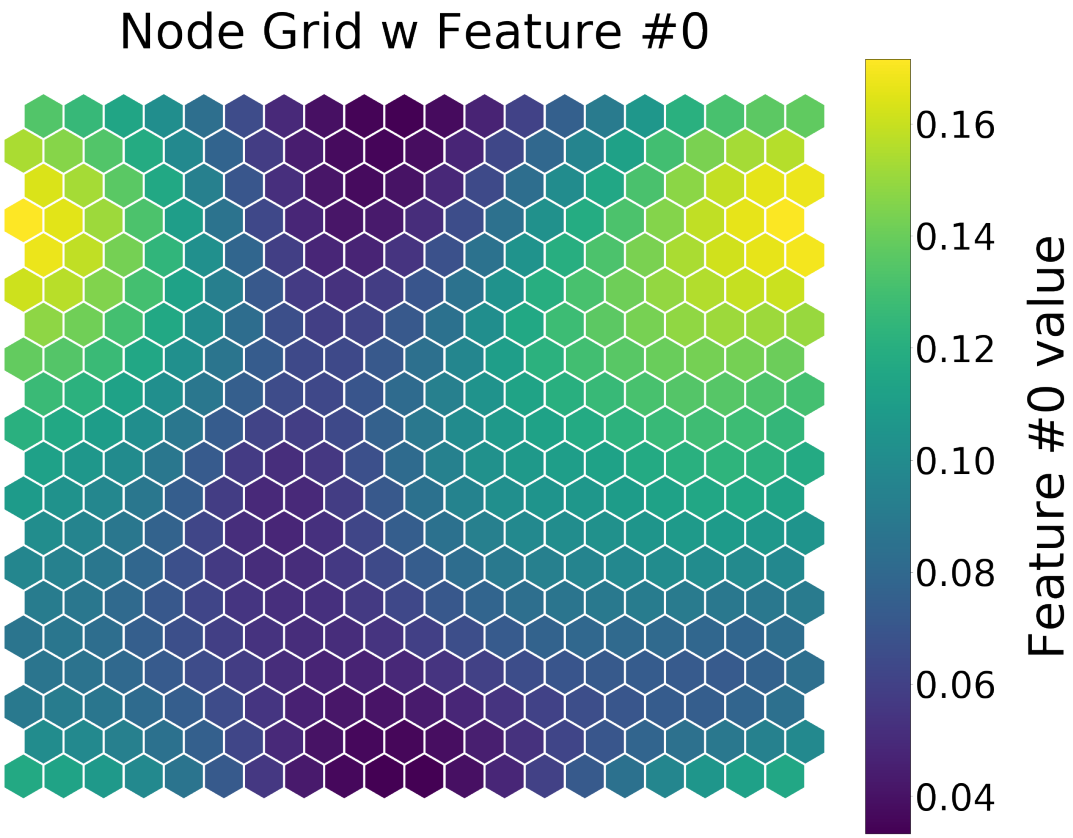

Figure 5.8: Self Organizing Map

Chapter 6

Conclusion

6.1 Introduction

This chapter gives an analysis of research aims and objectives, research problem, limitation of the work, and future works of the research

6.2 Conclusions about research questions

Aims of this research was to explore the ethnic-bias issue on the state of the art facial recognition algorithm Facenet by Schroff et.al [1] and provide a novel approach to minimize the effect without changing the neural network architecture.

By analysing the results from the chapter 5, results show that when the classification tasks carried out, in some instances the proposed approach in this research provides better accuracy than the base facenet model. By following the proposed approach the AUC metric was increased when the epoch number is increased thus reducing the false acceptance rate of the model. So that producing an environment where the less represented ethnicities are properly recognized when the facenet algorithm is used.

Also from the the comparative results which are gathered from the t-SNE clustering, it is shown that the proposed methodology has a higher chance to re-identify a person when that same person is seen before. This conclusion is arrived from the fact that the model trained with the proposed method, tends to form near perfect clusters from the data points from each identity while the base approach struggles to do so.

Hence by factoring all above mentioned facts, of the improved performance of the model when the modified approach of using ethnicities in the training stage leads to the conclusion that there is an ethnic-bias issue present in the Facenet architecture. And by considering the ethnicities at the training stage, the reliability

and the accuracy of the outputs are improved without changing the inner workings of the neural network.

6.3 Conclusions about research problem

With this novel approach, we were able to provide a methodology that can be used with facial recognition algorithms which uses triplet loss as the loss function, to reduce the error and misclassification when used with the less represented ethnicities.

In the process of the development of the proposed method, we've created an neural network model which is based on the VGG16 neural network architecture that can classify an image of a human into main five categories by considering the ethnicity. Apart from addressing the research questions, as a result of the research we were able to create an ethnicity classifier which can classify and images into following categories caucasian, black, asian, indian and other.

6.4 Limitations

Limitations of the hardware resources was a major setback while conducting the research. Mainly the slower read and write speeds of the HDDs (hard disk drives) which took considerable time in the preprocessing stages. As well as the ethnicity labeling stage, the dataset extraction time was higher because of the slower read write speeds of the disk. Lack of computational power was also an major issue when training a facial recognition model with a considerably large datasets. Although the models were trained on the google colab platform, because of the limitations of the given platform, the continuous run time recycling of the environments was a hassle.

The limitations of the datasets was also an major issue and a limitation while conducting the research. Mainly the dataset was used to train the facial recognition model was the largest publicly available dataset. Yet there were privately owned datasets (From Google) of faces which have more than 40 times the number of images (200 million images) of the CASIA dataset. As well as the lack of the labelled datasets for the ethnicity classification was also an limitation faced while conducting the research.

6.5 Implications for further research

Rather than chunking the dataset and using the batches to train the model, it will be interesting to observe the effects of using the whole dataset at once by storing it in the ram and compare it with the outcome of this research. (Given that adequate amount of RAM available)

Conduct the above mentioned optimization methods with the privately owned Face dataset by Google which has more than 200 million images with 8 million unique identities and compare the results.

Use transfer learning to re-use the weights of a similar problem and port this model into the above model and compare the results.

References

[1] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015.

[2] E. Gonzalez-Sosa, J. Fiérrez, R. Vera-Rodríguez, and F. Alonso-Fernandez, "Facial soft biometrics for recognition in the wild: Recent works, annotation, and cots evaluation," *IEEE Transactions on Information Forensics and Security*, vol. 13, pp. 2001–2014, 2018.

[3] S. Y. Zhang, Zhifei and H. Qi, "Age progression/regression by conditional adversarial autoencoder," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017.

[4] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 1943–1955, 2016.

[5] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *FAT*, 2018.

[6] A. Acien, A. Morales, R. Vera-Rodríguez, I. Bartolome, and J. Fiérrez, "Measuring the gender and ethnicity bias in deep models for face recognition," in *CIARP*, 2018.

[7] B. Yu, T. Liu, M. Gong, C. Ding, and D. Tao, "Correcting the triplet selection bias for triplet loss," in *ECCV*, 2018.

[8] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," 12 2014.

[9] W. L. Kuechler and V. K. Vaishnavi, "On theory development in design science research: anatomy of a research project," *European Journal of Information Systems*, vol. 17, pp. 489–504, 2008.

[10] K. N. Mccusker and S. Gunaydin, "Research using qualitative, quantitative or mixed methods and choice based on the research," *Perfusion*, vol. 30, pp. 537 – 542, 2015.

[11] A. Sepas-Moghaddam, F. Pereira, and P. Correia, "Face recognition: A novel multi-level taxonomy based survey," 01 2019.

[12] Li Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 594–611, April 2006.

[13] G. R. Koch, "Siamese neural networks for one-shot image recognition," 2015.

[14] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[15] B. Zhuang, G. Lin, C. Shen, and I. D. Reid, "Fast training of triplet-based deep binary embedding networks," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5955–5964, 2016.

[16] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *BMVC*, 2015.

[17] X. Lu, H. Chen, and A. K. Jain, "Multimodal facial gender and ethnicity identification," in *Advances in Biometrics* (D. Zhang and A. K. Jain, eds.), (Berlin, Heidelberg), pp. 554–561, Springer Berlin Heidelberg, 2005.

[18] A. S. Mohammad and J. A. Al-Ani, "Towards ethnicity detection using learning based classifiers," in *2017 9th Computer Science and Electronic Engineering (CEEC)*, pp. 219–224, Sep. 2017.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.

[20] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," Tech. Rep. 07-49, University of Massachusetts, Amherst, October 2007.

[21] G. B. Huang, M. Mattar, H. Lee, and E. Learned-Miller, "Learning to align from scratch," in *NIPS*, 2012.

[22] D. Yi, Z. Lei, S. Liao, and S. Li, "Learning face representation from scratch," 11 2014.

[23] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, 04 2016.

[24] X. Lu, H. Chen, and A. K. Jain, "Multimodal facial gender and ethnicity identification," in *Advances in Biometrics* (D. Zhang and A. K. Jain, eds.), (Berlin, Heidelberg), pp. 554–561, Springer Berlin Heidelberg, 2005.

[25] K. I. Chang, K. W. Bowyer, and P. J. Flynn, "Multi-modal 2d and 3d biometrics for face recognition," in *Proceedings of the IEEE International Workshop on Analysis and Modeling of Faces and Gestures*, AMFG '03, (USA), p. 187, IEEE Computer Society, 2003.

[26] D. Wen, H. Han, and A. Jain, " Face Spoof Detection with Image Distortion Analysis," *IEEE Trans. Information Forensic and Security*, vol. 10, pp. 746–761, April 2015.

[27] P. J. Phillips, A. Martin, C. L. Wilson, and M. Przybocki, "An introduction evaluating biometric systems," *Computer*, vol. 33, pp. 56–63, Feb 2000.

[28] D. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 07 2009.

[29] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[30] L. van der Maaten and G. Hinton, "Viualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 11 2008.

[31] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," vol. 96, pp. 226–231, 01 1996.

[32] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, (USA), p. 1027–1035, Society for Industrial and Applied Mathematics, 2007.

[33] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, pp. 1464–1480, Sep. 1990.

Appendices

Appendix A

MTCNN face alignment on CASIA dataset

Below mentioned code set figure A.1 and figure A.2 describes the function which MTCNN is used to align faces from the CASIA dataset so that it can be used to train the facial recognition model.

Below is the parameters that were fed into the above mentioned function.

```
my_args = {
    "input_dir" : "/content/casia_dataset/extracted/CASIA-WebFace",
    "output_dir" : "/content/casia_dataset/aligned",
    "image_size" : 182,
    "margin" : 44,
    "random_order" : True,
    "gpu_memory_fraction" : 1.0,
    "detect_multiple_faces" : False,
}
```

```python
def main(args):
    sleep(random.random())
    output_dir = os.path.expanduser(args["output_dir"])
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    # Store some git revision info in a text file in the log directory
    # src_path,_ = os.path.split(os.path.realpath(__file__))
    # facenet.store_revision_info(src_path, output_dir, ' '.join(sys.argv))
    dataset = facenet.get_dataset(args["input_dir"])

    print('Creating networks and loading parameters')

    with tf.Graph().as_default():
        gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=args["gpu_memory_fraction"])
        sess = tf.Session(config=tf.ConfigProto(gpu_options=gpu_options, log_device_placement=False))
        with sess.as_default():
            pnet, rnet, onet = align.detect_face.create_mtcnn(sess, None)

    minsize = 20 # minimum size of face
    threshold = [ 0.6, 0.7, 0.7 ]  # three steps's threshold
    factor = 0.709 # scale factor

    # Add a random key to the filename to allow alignment using multiple processes
    random_key = np.random.randint(0, high=99999)
    bounding_boxes_filename = os.path.join(output_dir, 'bounding_boxes_%05d.txt' % random_key)

    with open(bounding_boxes_filename, "w") as text_file:
        nrof_images_total = 0
        nrof_successfully_aligned = 0
        if args["random_order"]:
            random.shuffle(dataset)

        for cls in dataset:
            output_class_dir = os.path.join(output_dir, cls.name)
            # print(cls)
            if not os.path.exists(output_class_dir):
                os.makedirs(output_class_dir)
                if args["random_order"]:
                    random.shuffle(cls.image_paths)
            for image_path in cls.image_paths:
                nrof_images_total += 1
                filename = os.path.splitext(os.path.split(image_path)[1])[0]
                output_filename = os.path.join(output_class_dir, filename+'.png')
                # print(image_path)
                # print(output_filename)
                if not os.path.exists(output_filename):
                    try:
                        img = misc.imread(image_path)
                    except (IOError, ValueError, IndexError) as e:
                        errorMessage = '{}: {}'.format(image_path, e)
                        #print(errorMessage)
                    else:
                        if img.ndim<2:
                            #print('Unable to align "%s"' % image_path)
                            text_file.write('%s\n' % (output_filename))
                            continue
                        if img.ndim == 2:
                            img = facenet.to_rgb(img)
                        img = img[:,:,0:3]
```

Figure A.1: MTCNN face alignment pt1

```
                    bounding_boxes, _ = align.detect_face.detect_face(img, minsize, pnet, rnet, onet, threshold, facto
                    nrof_faces = bounding_boxes.shape[0]
                    if nrof_faces>0:
                        det = bounding_boxes[:,0:4]
                        det_arr = []
                        img_size = np.asarray(img.shape)[0:2]
                        if nrof_faces>1:
                            if args["detect_multiple_faces"]:
                                for i in range(nrof_faces):
                                    det_arr.append(np.squeeze(det[i]))
                            else:
                                bounding_box_size = (det[:,2]-det[:,0])*(det[:,3]-det[:,1])
                                img_center = img_size / 2
                                offsets = np.vstack([ (det[:,0]+det[:,2])/2-img_center[1], (det[:,1]+det[:,3])/2-img_c
                                offset_dist_squared = np.sum(np.power(offsets,2.0),0)
                                index = np.argmax(bounding_box_size-offset_dist_squared*2.0) # some extra weight on th
                                det_arr.append(det[index,:])
                        else:
                            det_arr.append(np.squeeze(det))

                        for i, det in enumerate(det_arr):
                            det = np.squeeze(det)
                            bb = np.zeros(4, dtype=np.int32)
                            bb[0] = np.maximum(det[0]-args["margin"]/2, 0)
                            bb[1] = np.maximum(det[1]-args["margin"]/2, 0)
                            bb[2] = np.minimum(det[2]+args["margin"]/2, img_size[1])
                            bb[3] = np.minimum(det[3]+args["margin"]/2, img_size[0])
                            cropped = img[bb[1]:bb[3],bb[0]:bb[2],:]
                            scaled = misc.imresize(cropped, (args["image_size"], args["image_size"]), interp='bilinear
                            nrof_successfully_aligned += 1
                            filename_base, file_extension = os.path.splitext(output_filename)
                            if args["detect_multiple_faces"]:
                                output_filename_n = "{}_{}{}".format(filename_base, i, file_extension)
                            else:
                                output_filename_n = "{}{}".format(filename_base, file_extension)
                            misc.imsave(output_filename_n, scaled)
                            text_file.write('%s %d %d %d %d\n' % (output_filename_n, bb[0], bb[1], bb[2], bb[3]))
                    else:
                        print('Unable to align "%s"' % image_path)
                        text_file.write('%s\n' % (output_filename))

print('Total number of images: %d' % nrof_images_total)
print('Number of successfully aligned images: %d' % nrof_successfully_aligned)
```

Figure A.2: MTCNN face alignment pt1