What is my name?

by

Kashyap Nirmal
202011031

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
in
INFORMATION AND COMMUNICATION TECHNOLOGY
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

June, 2022

Declaration

I hereby declare that

i) the thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,

ii) due acknowledgment has been made in the text to all the reference material used.

[illegible]

Kashyap Nirmal

Certificate

This is to certify that the thesis work entitled "WHAT IS MY NAME?" has been carried out by **KASHYAP NIRMAL (202011031)** for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my/our supervision.

[illegible]

Prof. Manish K. Gupta
Thesis Supervisor

Acknowledgments

I want to express my special thanks of gratitude to my thesis supervisor Prof. Manish K. Gupta, for providing me with the opportunity to work under his supervision for this M.Tech thesis. He gave me the chance to work on this beautiful thesis topic of **What is my name?** He provided me with continuous guidance and support throughout my thesis journey. He helped me push through my limits. Thus, he helped me discover my unknown potential. Thus, he helped me bring out the best in me. And allowing me to grow more in every aspect.

I want to express my thanks to my friend Darshan for being there with me all the time. Without him, I would have almost been lost. I would also like to thank Akash, Darshil, Kishan, Dhyanil and Mahir for their help and support. I also wish to thank all my peers for their assistance in the data collection phase and their moral support.

I want to express my gratitude to all the responders for filling the data through the Google Form. The sample photographs shown in this thesis are used with the consent of the responder. Thus, I would like to thank all these responders for their valuable input.

I also wish to thank the institute and the Help desk team for providing me with the necessary GPU resources. I am grateful to the Help desk team for their constant help and support regarding the resources.

I wish to thank my parents and family for always believing in me and supporting and motivating me. Thank you for all the blessing you have endowed me.

Kashyap Nirmal

Contents

Abstract

The thesis work introduces a new concept of predicting the first name from the facial image. Several studies show a relationship between first names and facial features. The prediction problem here is treated as an image classification problem. Here we have made a comparative analysis of various Ensemble learning techniques and a Transfer learning method. Dataset Name100 consists of 100 popular names of the U.S is used. The previous study was done on U.S. face-name dataset. We have created an Indian face-name dataset. The Indian dataset right now contains 5 Indian names. Thus we have used these two datasets for our experiments. The dataset mentioned here has no additional labelling cost apart from the name tags freely available on the internet. And we have also circulated a Google Form to collect the Indian data samples shown in this thesis.

The prediction accuracy of this system is very low. Despite such a system's low accuracy, the prediction was correct at more excellent rates than random chances. Though this system is imperfect for use in the real world today, it still has its applications in security and biometrics. And it also has the application of Face Name association. We have observed that misclassification happens amongst names of the same gender. While in the case of gender-neutral names, it is highly non-deterministic. Thus, a lot of research work needs to be done in this area.

Keywords : Image Classification, Facial Image, First Name.

List of Tables

List of Figures

CHAPTER 1

Introduction

What's in a name? That which we call a rose
By any other name would smell as sweet.

- Shakespeare [17].

In this chapter, we briefly discuss the thesis objective and the motivation. We note and describe the potential challenges for this thesis work. We describes the potential Application of the First Name prediction. We also mention our contributions. And after that, we also say the organization of this thesis.

1.1 Thesis Objectives

This thesis work aims at predicting the first name of a person from the facial image. The problem task here is assumed to be an image classification problem. The Classification problem means identifying a category from the available classes. The system predicts the class label which accommodates the new training sample. This task is supervised learning as the class labels are predefined. Thus, the final output label will also be from the same set of these class names used in the dataset.

1.2 Challenges

The patterns found in previous studies had the underlying dataset of the U.S. population. One significant challenges for this problem is the availability of the data. We had a keen desire to carry out similar research on Indian faces.

We as a researcher need cultural knowledge of the people. We are now talking about specific instances of Indian First names. In some cultures or localities, particular names are treated as females in one locality, while the same name can be treated as male names too in another locality. For instance, Sonal can be treated majorly as female, but the name Sonal is also used as a male name in some localities. Similarly, Muskan is a female name majorly, but there are few males with the first name Muskan. So, we need to be careful and take only those photographs which are of the dominant gender for that particular first name. People may be biased and fill out their full names even when asked to fill in only the first name. So in some cases, we may have to manually filter out these data entries during the data collecting phase. So, we at least need to understand the name breakdown in these particular cases.

One of the major challenges here is name selection. The names for which we would want to consider this research. For the Name100 [3], they had taken top 100 names from SSA [19]. In the scenario of Indian names, there is no such availability of these top 100 names or something like that. To the best of our knowledge, there is no provision for an Indian first name list too.

After selecting the first name, the next and the most significant challenge would be to search for photographs of individuals with these first names. So here, we also need to take the consent of the responder to use the data for our educational use. Some people may not want to share their images for academic research here. Here certain people may be unaware of machine learning / deep learning. So these people are not confident in giving their data for this kind of studies.

So to overcome the problem of name selection, we tried selecting five common names from our networks through brainstorming. To collect the photographs for this dataset, we created a Google Form. To give the respondents confidence and assurance, we had a disclaimer about the data used for purely academic purposes. We also asked for the responder's consent to make the collected data public if needed.

This form was circulated through our networks through various means and platforms like LinkedIn, Twitter, Gmail, Whatsapp etc. Even mouth to mouth interactions was there among the people in the institute. While collecting the dataset, many people were not comfortable sharing their photographs. We have selected five classes. And now, searching for this set of people with these specific names is a very challenging task.

1.3 Applications

Every individual has a faceprint [6]. It's an electronically stored portrayal of a person's face, which is as unique as a fingerprint. So it may have applications for security purposes and biometrics.

One of the potential applications of this work is name association. Suppose we have a group photo. So we can predict the names of the persons present in the image using our model. And we can associate the predicted first name to the face in the group photo. This application of Name association comes with given constraints. The trained model is a supervised learning model and a classifier. So, as a result, the classifier only outputs the first names from the training set.

Name association is already a part of the present-day social media platforms like LinkedIn, Instagram, Facebook etc. And it is required to associate/tag the social media handle of the persons present in the group photo. As a privacy setting provided to social media users, some users may have turned off the tagging feature, so others can not tag these individuals in their social media posts.

One of the ways of Name association can be using **Face Matching**. But then, a critical difference between these two systems lies in the methodologies. The face matching uses the previously available photographs. In contrast, our system uses Name prediction. Our system is assumed to be an Image Classification problem, thus, outputs are restrained to the training classes. And in the face matching system, there have to be previous photographs of the person and the name, and then only it can associate the name.

1.4 Thesis Motivation

Suppose a person meets a new face. They have several questions in their mind.

- Who are they? (i.e., First Name).
- How old are they? (i.e., Age).
- Where are they based? (i.e., Ethnicity in a more general way).
- What is their gender?
- What is their occupation?
- Even one can think of the emotions of the other person.

Also, consider a similar scenario for the image of a new face. So consider that above-mentioned are the features we can extract from a facial image.

The human brain can predict some of these attributes through a glance. It can predict gender in almost all cases. It can also predict the age group. Predicting the exact age can be a bit difficult, though. Through some other facial features, it can also predict certain races. Not from the face, but at times may be through the clothing, some occupations can also be predicted. It can also predict happiness or sadness, or other emotions on a face. So, anticipating the first name can be a new and nearly impossible task from the aforementioned features.

Parents expecting a baby spend a significant amount of time deciding on the child's name. Most of the time, it may be assumed that the choice of the name is nearly random from a vast pool of names [3]. Does there exist any relationship between the person's first name and the facial image? Mostly if people are posed with the question, they would deny it. But several studies indicate the existence of a relationship between first name and face.

The focus here is on first names and not last name because the former illustrates more freedom and variety when selecting. At the same time, the latter tends to be determined more by lineage or, in some cases, based on occupation. Rather than arbitrariness, clear patterns have been observed between the owner and the name. Even though very few first names are gender-neutral like Jamie, first names are generally gender-specific. Even the education of the parents and their race does influence the choices of the first names. Names do carry the information about the age because trends in naming keep on varying throughout decades (Krammer et al., 2015, p. 2) [12].

1.5 Contributions

The thesis contributions includes the following :

- In this thesis work, we have created an Indian face-name dataset. We have used a Google Form to collect the dataset. This Google Form was circulated through various social media platforms available.
- We have made a comparative analysis of the Transfer learning and Ensemble learning techniques. We derived the analysis results based on the Confusion Matrices and prediction accuracy. Here we have used two different datasets as well.
- We also developed a Flask demo to showcase Name Association based on the Name prediction method instead of the traditional Face Matching.

1.6 Thesis Organization

This thesis is split into following 5 chapters:

The second chapter is Related work. It includes the existing studies done in this area. The third chapter is Experimental setup and Implementation details. It showcases the experiments conducted throughout the thesis. It also provides information regarding the datasets used for this thesis work. The fourth chapter is Results and Analysis. It summarizes the results and analysis based on the comparisons undertaken in the experiments. It also showcases the visualization of the results. The fifth chapter is Conclusion and Future Work. It concludes this thesis, along with the potential future works.

CHAPTER 2
Related Work

Shouldn't you be asking for my name first?

- Kip Fulbeck [8].

According to [11] (as cited in [3]) the achievements of face detection and recognition in computer vision have been found dating back to approximately four decades.

This work poses as well as thus starts to solve a unique topic in facial processing. Is it possible to infer a person's name from through a single facial image? While that too, without additional sample images of such face. Expecting a high level of accuracy from this work is impractical. However, even the identical twins get their unique names.

This flawed system could have a variety of usages, for instance, in terms of security and biometrics. In the security aspect, finding fake IDs from databases can be solved using this system. And for biometrics, we can infer the ethnicity, sex, age by taking a guess of probable names from a face [3].

A variety of aspects influences the name selection. The gender of the individual can control the name selection. Even the age, race, social culture, economic culture, the popularity of names, names of near and dear also influence name selection. Thus, even within an ethnicity, the occurrence of particular first names varies. There is a significant age difference in easily distinguishable name pairs. The name pairs which exhibit the same popularity trends seem indistinguishable [3].

The visual characteristics that distinguish any given set of people differ. First names and numerous face aspects contain a relationship. According to [16] (as cited in [3]), aspects such as skin colour, male-ness, facial feature size, age, and potentially other unnamed traits correlate with first names. Now, for instance, the genders of "David" and "Mary" vary. At the same time, the names "David" and "Ethan" are distinguished primarily in age because "Ethan" is a newer name [3].

Danny has a boyish appearance and a permanent smile in someone's imagination. Zoe has big eyes, wild hair, as well as a mildly amused demeanour. According to studies, the concept that persons with the same name have the same typical "look" can be genuine.

The researchers offered an unfamiliar face that had five choices of names. And the individuals selected the correct name for around thirty-five per cent of the cases. In contrast, the probability is only twenty per cent [21]. The researchers never claimed that this can be carried out by anyone anywhere without having any cultural familiarity.

Individuals having the same name are prone to having almost identical expressions around the mouth and the eye area [21] of the face. These areas seem easily adjustable, as per a computer analysis.

The classifier was trained with the facial image obtained over the internet. The output values are significantly higher than the probability. Thus, the system accurately predicts the actual first name of research subjects. There was no extra user intervention in training. First names are not distributed arbitrarily among society's members [3].

2.0.1 Multi-Feature SVM

Figure 2.1 is the summary of the system. According to [5] (as cited in [3]), the test faces are first scaled with observed eye locations and then resampled to 150 x 120 pixels. After sampled on a dense grid with an interval of two-pixel, they extract SIFT descriptors [14] (as cited in [3]). After this, every 128 dimension SIFT descriptor is encoded to a 1024-dimensional code using the Locality-constrained Linear Coding (LLC) method, which is according to [20] (as cited in [3]). These encoded LLC codes are combined by applying max-pooling above a spatial pyramid [13] (as cited in [3]), resulting in a 1024-dimension vector at each of the twenty-one points of the pyramid grid. For each face, this results in a feature vector with 21 x 1024 = 21504 dimensions [3]. Thus, these twenty-one feature vectors could be thought of as originating from 21 complementing feature channels, and the authors suggest the Multi-Feature SVM (MFSVM) [3]. The algorithm is based on the AdaBoost [9] framework, where the classifiers are SVMs that work on multiple feature channels.

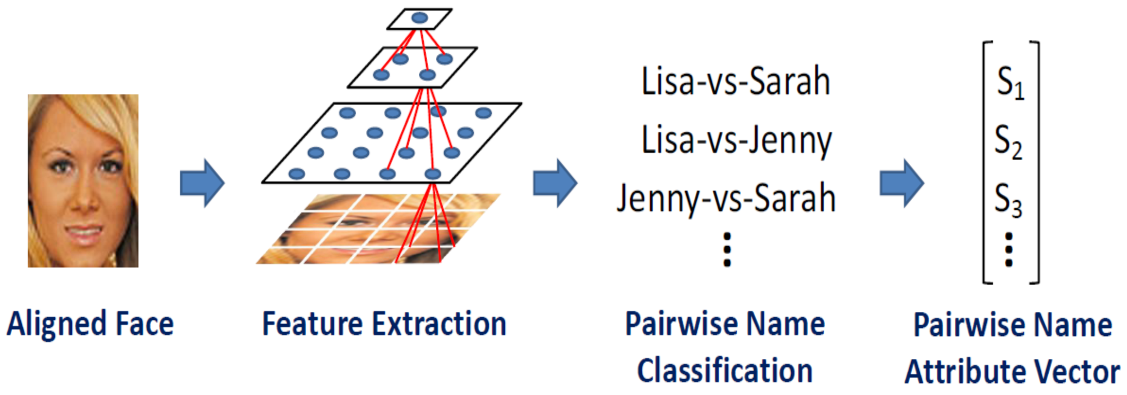


Figure 2.1: A summary of the system. The confidence scores produced by the pairwise name classifiers are referred to as the pairwise name attribute vector ©[2013] IEEE. Reprinted, with permission, from H. Chen, A. Gallagher, and B. Girod. What's in a name? First names as facial attributes. Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition., pages 3366–3373, 06 2013.

CHAPTER 3
Experimental setup and Implementation details

I'm gonna tell a real story. I'm gonna start with my name.

- Kendrick Lamar [15].

This chapter describes the algorithms used during the thesis work.

3.1 Hardware Configurations

Our training tasks are mostly done on Google Colaboratory. For training Vision Transformer, I have used GPU resources allocated by the institute. The resource's system configuration is 64 GB of RAM and Titan XP GPU with 12 GB of GPU memory. This system OS is Ubuntu 20.04.3 LTS. The following section discusses dataset details.

3.2 Dataset details

This section describes about the datasets used during the thesis work. We created the Indian dataset and it was used in our experiments.

3.2.1 Name100

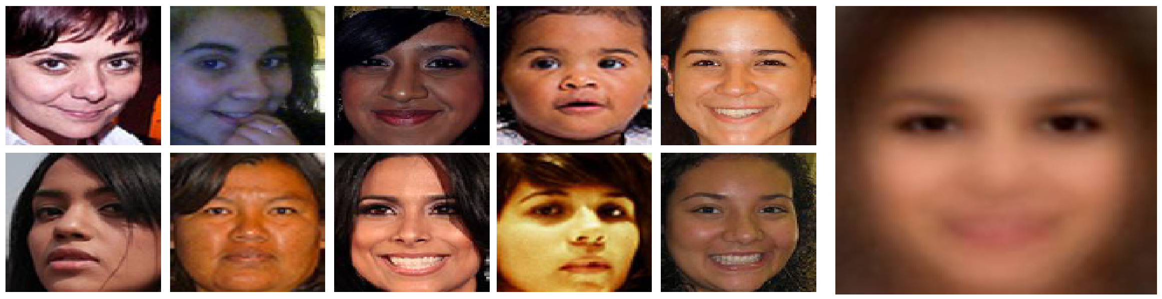
Name100 is the first dataset with Name and Face data. To determine the connection between first names and facial appearance, they generate a large dataset by selecting photographs and name tags from Flickr. The dataset comprises 800 faces, for each of the 100 most famous first names, based on information from the U.S. Social Security Administration (SSA) [19] (as cited in [3]). After completion, the dataset contains 48 men's names, 48 women's names, and 4 gender-neutral names. The names listed represent 20.35 per cent of all the Americans birthed from 1940 to 2010 [3].

When there are numerous people in a photo, name ambiguity occurs. Thus, images that contained multiple faces were eliminated, and it was verified whether the image tag included exactly one first name tag. Secondly, they removed the images that had celebrity names. The reason being this can result in a bias in their sampling. Assume that a search for "Brad" could produce a lot of photographs of the film actor "Brad Pitt" distorting the facial feature pattern for the name "Brad" [3].

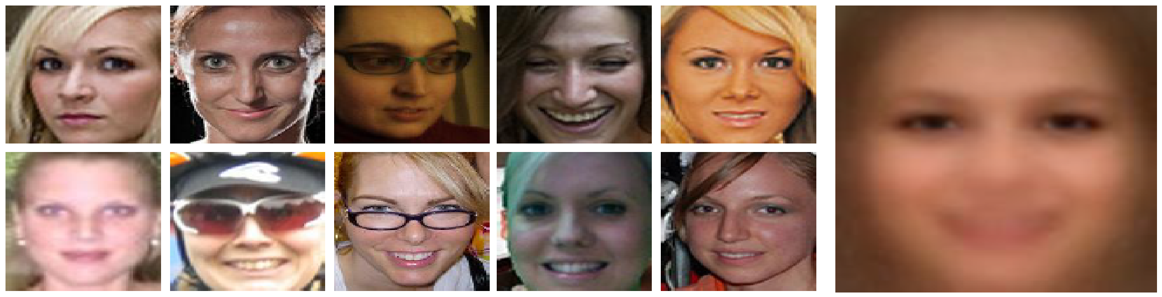
There was no additional manual labelling required for creating this dataset apart from the name tags available from the internet.

Sample image from the Name100 dataset

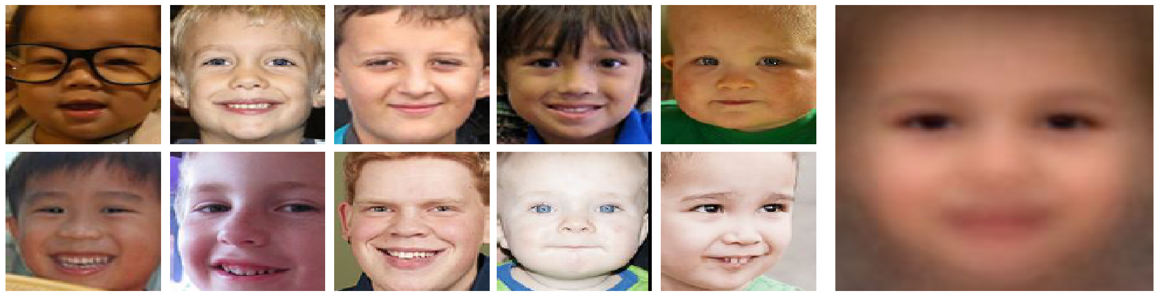
Figure 3.1 shows Alejandra's , Heather's and Ethan's face examples respectively. Alejandra's hair and skin are usually darker unlike Heather's. The name Ethan, which gained popularity in recent years, appears to be much young.



(a) Alejandra



(b) Heather



(c) Ethan

Figure 3.1: Sample images from Name100 images along with their average faces computed from 280 aligned faces ©[2013] IEEE. Reprinted, with permission, from H. Chen, A. Gallagher, and B. Girod. What's in a name? First names as facial attributes. Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition., pages 3366–3373, 06 2013.

3.2.2 Indian Dataset

We have created a dataset on Indian faces. We have scrapped the data from the internet. The dataset comprises five classes with 180 images of each class. Thus in total, this dataset has 900 images. The preprocessing was done using OpenCV.

Here we have considered five first names. We have divided them into two male names, two female names and one gender-neutral name. Here the first name Krishna is gender-neutral. For the male names, we have considered Rahul and Pranav. For the female names, we have taken Sonal and Priya.

Here, we also tried collecting data from Google Form. However, there was a lesser number of responses than we expected. The samples shown below are collected from Google Form, and we do have the consent of the owner of the photograph to use them.

Sample images from the Indian dataset

Here figure 3.2 shows Pranav's face examples. Similarly, figure 3.3, 3.4 and 3.5 show face examples of Krishna, Rahul and Sonal, respectively. These images are raw data samples that were collected directly from users. We thank them for their consent to use their pictures.

Figure 3.2: Raw Samples of Pranav's facial collected from Google Form.

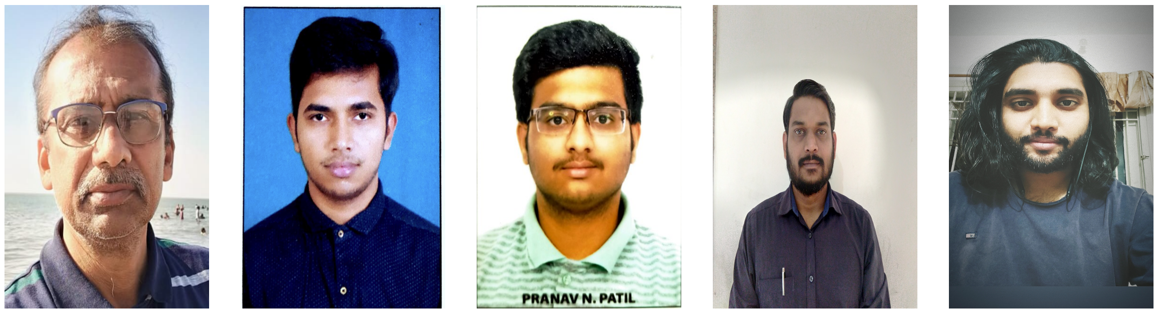


Figure 3.3: Raw Samples of Krishna's facial collected from Google Form.

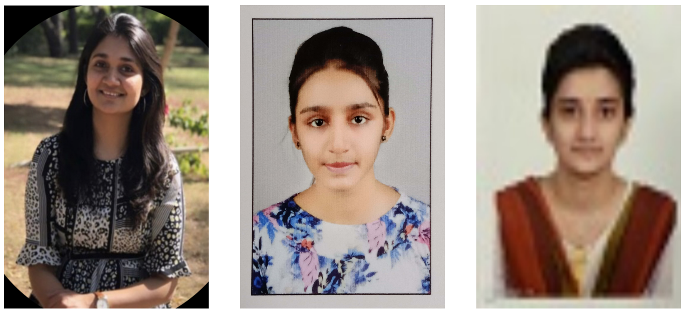


Figure 3.4: Raw Samples of Rahul's facial collected from Google Form.

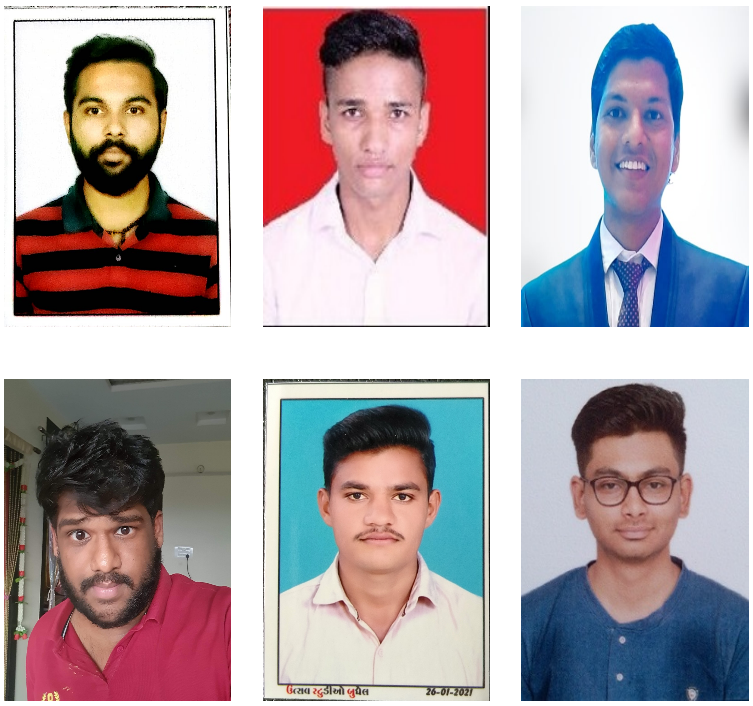
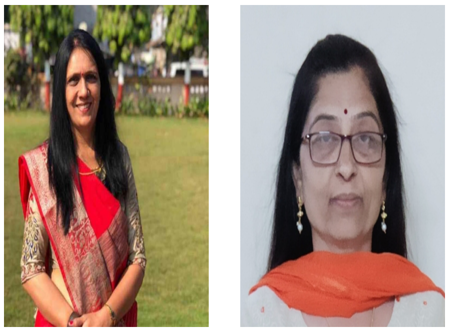


Figure 3.5: Raw Samples of Sonal's facial collected from Google Form.



3.3 Image classification techniques used

We have implemented various algorithms for the comparative analysis. We have implemented Vision Transformer [7], ensemble learning like Bagging [1] , and boosting algorithms like Adaboost [9] and XGBoost [4]. For extracting features, we have used Spatial pyramid pooling [10] and Deep Learning features. For Deep learning, it uses the Relu activation function.

In the domain of image classification, Vision transformer (ViT) [7] is state-of-the-art for ImageNet with a given particular condition. Thus we have chosen this model. Adaboost, a boosting technique, was used in [3]. And one knows boosting is a type of Ensemble learning technique. Therefore, we have attempted to follow in their footsteps and used various Ensemble learning techniques.

3.3.1 Spatial Pyramid Pooling for feature extraction

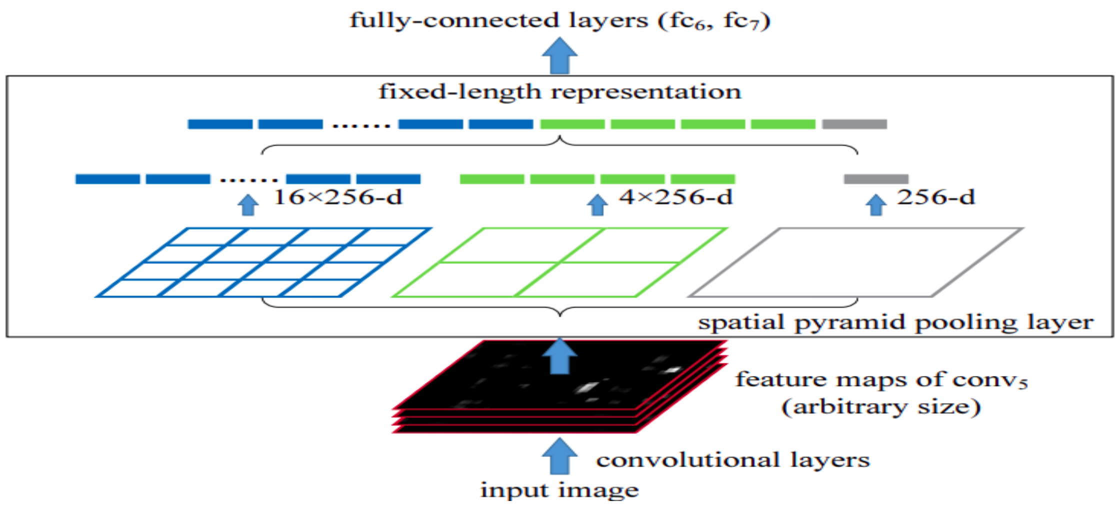

fully-connected layers (fc6, fc7)
fixed-length representation
16×256-d
4×256-d
256-d
spatial pyramid pooling layer
feature maps of conv5
(arbitrary size)
convolutional layers
input image

Figure 3.6: Spatial Pyramid Pooling Architecture ©[2015] IEEE. Reprinted, with permission, from K. He, X. Zhang, S. Ren and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 9, pp. 1904-1916, 1 Sept. 2015.

Figure 3.6 shows the architecture of Spatial pyramid pooling. In Spatial pyramid pooling [10] algorithm, 21 features are extracted in 3-levels. In Level-0, the sample remains undivided. In Level-1, the sample is split into four regions, and four features are extracted here. In Level-2 sample is split into sixteen regions, and sixteen features are extracted here. Thus these 21 extracted features are concatenated are used in classification.

3.3.2 Vision Transformer

In the domain of image classification, Vision transformer (ViT) [7] is state of the art for ImageNet with a given particular condition. The particular condition here is that ViT has an insatiable need to be pretrained on a large dataset to achieve higher results. And we applied this model and were able to achieve higher accuracy for a certain set of classes. This model has various variations. We have used ViT_B_16 with the image resolution of 224 x 224, and it was trained on the Imagenet21k dataset. Here B stands for Base. 16 are the number of patches. Figure 3.7 shows the architecture of the Vision transformer.

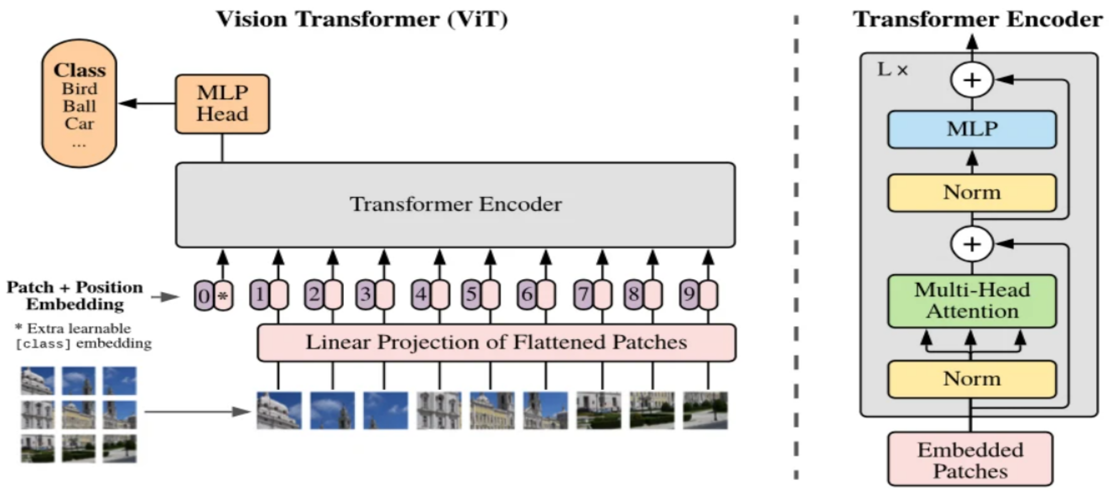


Figure 3.7: ViT Architecture [7]. Reprinted, with permission, from A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. CoRR, abs/2010.11929, 2020

3.3.3 Ensemble Learning

The ensemble learning technique combines various weak learners to achieve higher prediction accuracy. These weak learners can be any classifier algorithm. Some of the commonly used learners are Random forests. A few of the commonly used Ensemble learning techniques are bagging and boosting.

Bagging

In the bagging technique [1], all the classifiers learn independently and parallelly. The results of all these classifiers are combined to determine the final output.

Boosting

In boosting technique, classifiers are used iteratively. The weights for the incorrectly classified samples get elevated after each iteration. Similarly, the algorithm will drop the weights for the correctly classified samples after each iteration. These weight updations are done to provide more importance to the misclassified samples. And thus, it learns from its previous mistakes. XGBoost [4] and AdaBoost [9] are boosting algorithms.

CHAPTER 4
Results and Analysis

A person's first name is to him or her the sweetest and the most important sound in any language.

-Dale Carnegie [2].

This chapter summarizes and showcases the results. There is a comparison table that shows the classification accuracies of various models. There are confusion matrices for the best results. And the observations made from the experiments.

4.1 Results

Table 4.1: Result table of classification accuracy (%). The models with higher accuracies than the baseline are represented in **bold**. The accuracies are rounded off to 2 decimal places.

Sr. No.	Model	Name100 Subset 1	Name100 Subset 2	Indian
1	ViT [7]	35	**44.2**	**41.67**
2	Bagging SPP	35.63	37.38	**42.2**
3	Adaboost SPP	34.63	37.38	**41.6**
4	XGBoost SPP	32.75	37.25	**44.44**
5	Adaboost DL	27.5	31.62	**39.44**
6	Bagging DL	29.67	31.62	**39.44**

In the table 4.1 sufix SPP stands for Spatial pyramid pooling. These algorithms have used SPP for feature extraction. While the suffix DL stands for Deep learning feature.

Here, we have treated MFSVM [3] as the baseline. MFSVM uses boosting technique AdaBoost [9] with the classifier SVM. The baseline for five classes is 39.4%. The random chances for five classes would be one-fifth, i.e. 20%. With ViT model, we have achieved higher accuracy than this baseline for certain subsets of the dataset. While for 10 classes, the baseline of MFSVM is 23.5%. At the same time, the random chances for 10 classes are one-tenth, i.e., 10%. And with ViT for certain subsets of the dataset, it achieves an accuracy of 26%.

Name100 subset 1 stands for classes Chris, Jamie, Maggie, Stephen, and Tina. So the distribution of this subset includes 2 male names, 2 female names and 1 gender-neutral name. Name100 subset 2 stands for classes Abby, Amanda, Angela, Aaron, and Andrea. So the distribution of this subset includes 1 male name, 4 female names.

4.2 Confusion Matrices

We have achieved an almost higher accuracy score than the random chances with our experiments. Moreover, in some cases, the accuracy score even exceeded the baseline. We have also added the confusion matrix for the models with the highest accuracy for that particular dataset.

Moreover, from the results of other experiments, a pattern was followed. The first name is mainly misclassified among the first names of same-gender itself. For instance, male names are majorly misclassified into male names only. However, there are cases where the classification accuracy is lower than the random chances for some classes and models. But these types of instances are scarce.

Table 4.2: Confusion Matrix for Indian data. The model used is XGBoost [4]. The overall accuracy is 44.44%. The values are rounded off to 2 decimal places.

	Krishna	Pranav	Priya	Rahul	Sonal
Krishna	**30.56**	5.56	22.22	16.67	25.00
Pranav	8.33	**47.22**	2.78	41.67	0.00
Priya	38.89	2.78	**36.11**	0.00	22.22
Rahul	13.89	22.22	0.00	**63.89**	0.00
Sonal	16.67	0.00	36.11	2.78	**44.44**

From table 4.2, one can see that Pranav is majorly misclassified as Rahul. Moreover, Rahul is highly misclassified as Pranav. A male first name is mostly misclassified as other male names instead of the female names. Similarly, we can see the results for Priya and Sonal too. The female names are highly misclassified as other female names rathen than the male names. However, for the gender-neutral name Krishna, it is nondeterministic. Here, the overall accuracy is majorly higher than the random chances with some exceptions.

Table 4.3 uses Name100 subset 2 dataset. The Name100 subset 2 stands for classes Abby, Amanda, Angela, Aaron, and Andrea. So the distribution of this subset includes 1 male name and 4 female names. Here Aaron is the male name while the other names are female names.

Table 4.3: Confusion Matrix for Name100 Subset2. The model used is ViT [7]. The overall accuracy is 44.2%. The values are rounded off to 2 decimal places.

	Abby	Amanda	Angela	Aaron	Andrea
Abby	**55.00**	23.00	10.00	8.00	4.00
Amanda	23.00	**37.00**	19.00	5.00	15.00
Angela	23.00	30.00	**29.00**	6.00	12.00
Aaron	7.00	2.00	3.00	**81.00**	7.00
Andrea	24.00	20.00	11.00	29.00	**15.00**

From table 4.3 one can see that Andrea is highly misclassified as Aaron. Name-100 is the U.S. face-name dataset, and Andrea in the U.S. is used as a female name. But the name Andrea has Greek origin, and it means "strong and manly". It is used as a gender-neutral name in its native Italy and other European nations. As the dataset was collected from the Flickr platform, there are rare instances of male faces in the female name Andrea. The name Andrea was also highly misclassified as Aaron in many other models.

Table 4.4 uses 10 classes from Name100 dataset. These classes are Amanda, Brian, Aaron, Jamie, Brandon, Abby, Andrea, Anthony, Dylan, and Angela. So the distribution of this subset includes 5 male names, 4 female names and 1 gender-neutral name. Male names here are Brian, Aaron, Brandon, Anthony and Dylan. Female names here are Amanda, Abby, Andrea and Angela. The gender-neutral name here is Jamie.

From table 4.4 one can observe that seven classes here have classification accuracy higher than the random chance of one-tenth, i.e., ten per cent. While for the remaining three classes, one can observe lower classification accuracy. And one of these three classes is a gender-neutral name, Jamie. As it was already observed from tables 4.2 and 4.3, that gender-neutral names perform poorly.

Table 4.4: Confusion Matrix for Name100 subset containing 10 classes. The model used is ViT [7]. The overall accuracy is 26%. The values are rounded off to 2 decimal places.

	Amanda	*Brian*	*Aaron*	*Jamie*	*Brandon*	*Abby*	*Andrea*	*Anthony*	*Dylan*	*Angela*
Amanda	**0.43**	0.03	0	0.01	0.01	0.21	0.05	0.01	0.04	0.2
Brian	0.01	**0.46**	0.08	0.01	0.13	0.03	0.01	0.17	0.07	0.03
Aaron	0.03	0.26	**0.08**	0.01	0.16	0.05	0.01	0.16	0.2	0.03
Jamie	0.24	0.18	0.04	**0.03**	0.07	0.16	0.03	0.05	0.13	0.08
Brandon	0.03	0.24	0.08	0.04	**0.17**	0.03	0.01	0.15	0.21	0.05
Abby	0.32	0.01	0	0.01	0.01	**0.46**	0.01	0.03	0.07	0.08
Andrea	0.3	0.17	0.01	0.01	0.02	0.13	**0.02**	0.04	0.07	0.24
Anthony	0.03	0.23	0.08	0	0.13	0.06	0.02	**0.22**	0.17	0.07
Dylan	0.05	0.14	0.07	0.01	0.16	0.1	0.01	0.07	**0.33**	0.06
Angela	0.3	0.01	0.01	0.01	0.03	0.16	0.03	0.04	0.01	**0.4**

4.3 Visualized results

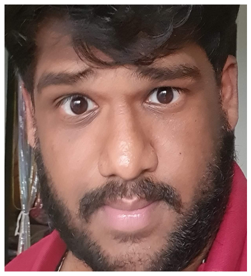
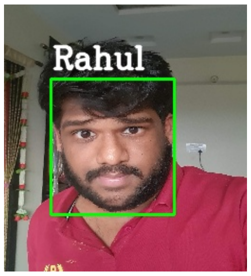
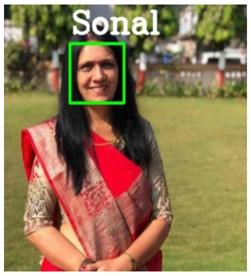
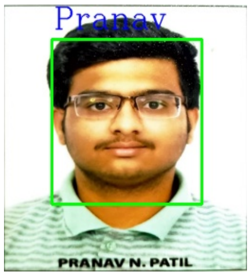
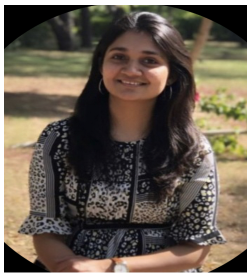
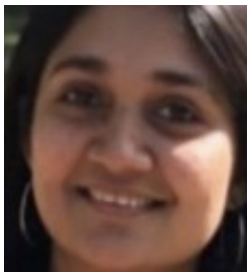
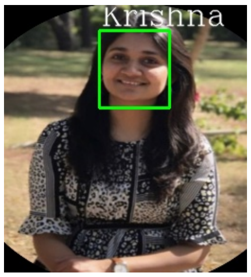
First Name	Raw Input image	Cropped Face	Name Association
Rahul			
Sonal			
Pranav			
Krishna			

Figure 4.1: Figure shows visualization of our approach on Indian dataset. First Names in **Blue** are correctly classified, while names in **Red** are misclassified.

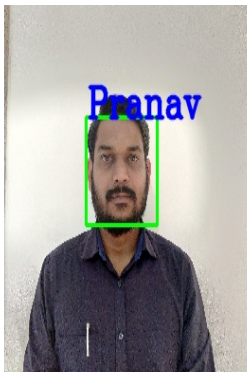
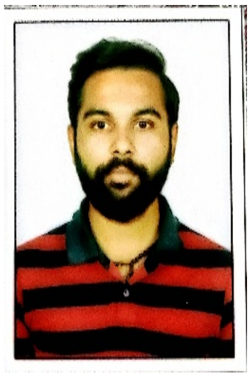
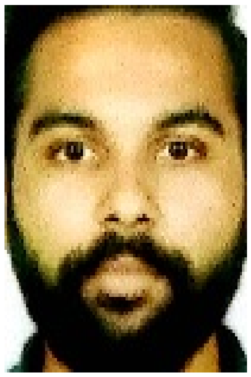
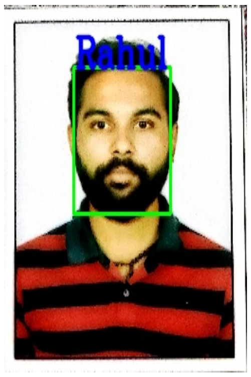
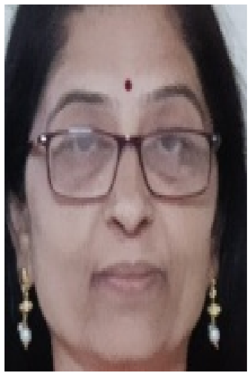
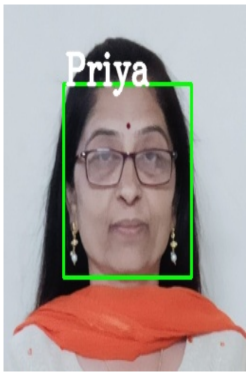
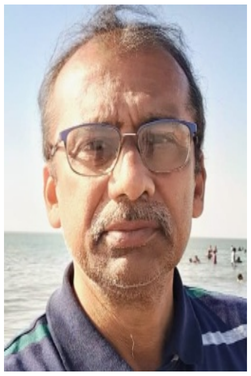
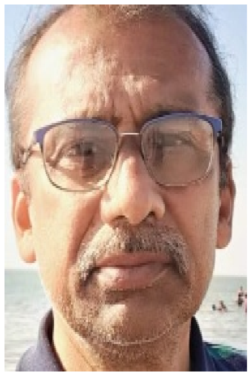
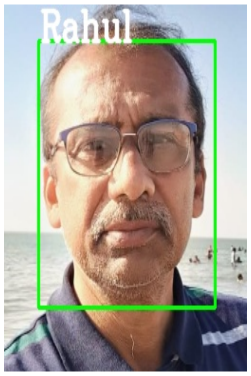
First Name	Raw Input image	Cropped Face	Name Association
Pranav			
Rahul			
Sonal			
Pranav			

Figure 4.2: Figure shows visualization of our approach on Indian dataset. First Names in **Blue** are correctly classified, while names in **Red** are misclassified.

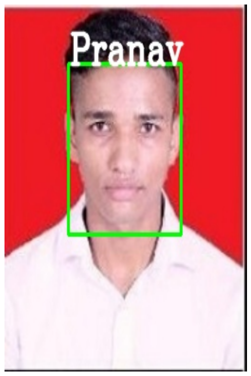
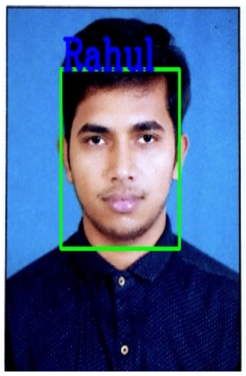
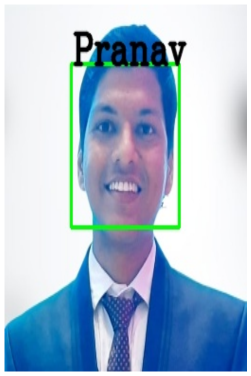
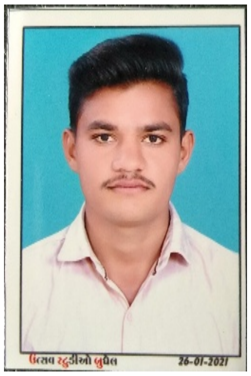
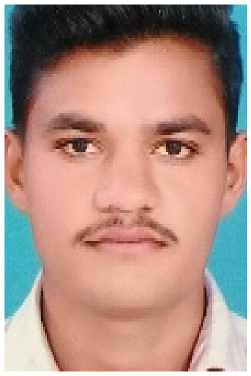
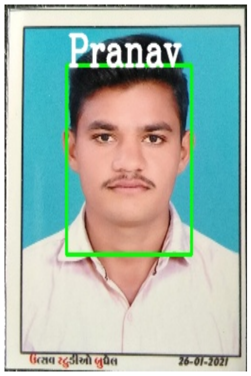
First Name	Raw Input image	Cropped Face	Name Association
Rahul			
Pranav			
Rahul			
Rahul			

Figure 4.3: Figure shows visualization of our approach on Indian dataset. First Names in **Blue** are correctly classified, while names in **Red** are misclassified.

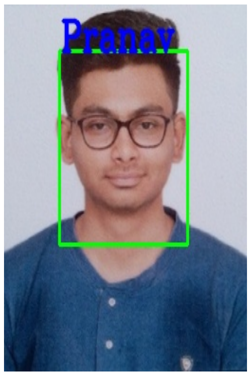
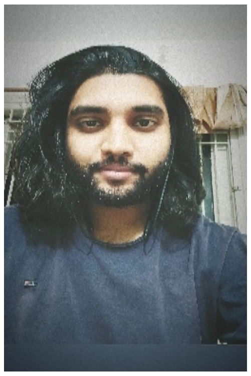
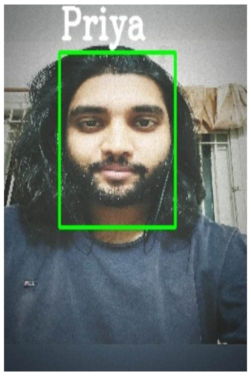
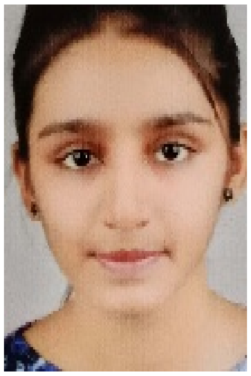
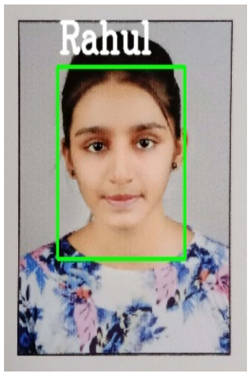
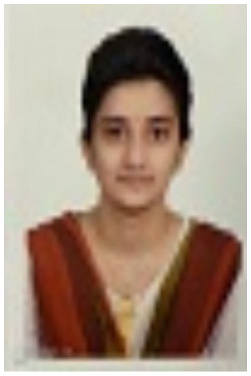
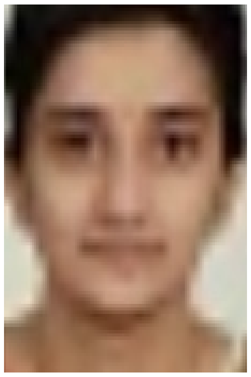
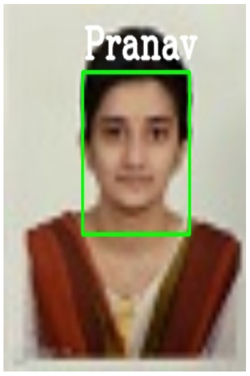
First Name	Raw Input image	Cropped Face	Name Association
Rahul			
Pranav			
Krishna			
Krishna			

Figure 4.4: Figure shows visualization of our approach on Indian dataset. First Names in **Blue** are correctly classified, while names in **Red** are misclassified.

CHAPTER 5
Conclusions and Future Work

Every name is real. That's the nature of Names.

-Jerry Spinelli [18].

In this chapter, we briefly discuss the conclusion drawn from the thesis work. And also describe the potential future work for this thesis.

5.1 Conclusion

In the discipline of computer vision, name prediction can be a tough and inspiring problem to solve. This work did not receive that much consideration and had very restricted efforts in terms of research. The name prediction system is not ideal for functioning well in all real-world situations. This thesis work summarises the results of the experiments that were conducted. We have also constructed the dataset for Indian faces. And with our experiments, we concluded that the patterns found in the Names100 dataset can also nearly be seen in Indian faces. Thus the current works have high prediction accuracy than random chance. The classes with gender-neutral names would require certain amount of future efforts to increase the prediction accuracy. But still plenty of work needs to be carried out to achieve higher efficiency and accuracy goals.

5.2 Future Works

As part of future work, we would want to enhance the size of the custom Indian dataset. We want to increase the number of classes, i.e. various unique first names. Moreover, we would like to raise the count of sample images per class. For the name selection procedure, we can do brainstorming, surveys, etc. We may approach different social media giants, educational institutes, or research institutes for the required data. We also would like to see and verify if this pattern exists for datasets of other nationalities. We would also like to improve the overall classification accuracy. And we would specifically like to increase the prediction accuracy for gender-neutral classes.

References

[1] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.

[2] D. Carnegie. *How to Have Rewarding Relationships Win Trust and Influence People: Dale Carnegie Success Series*. Manjul Publishing, 2018.

[3] H. Chen, A. C. Gallagher, and B. Girod. What's in a name? first names as facial attributes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2013.*, pages 3366–3373, 2013.

[4] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[5] T. Cootes, C. Taylor, D. Cooper, and J. Graham. Active shape models-their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.

[6] Faceprint definition & meaning. https://www.dictionary.com/browse/faceprint, 2021.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[8] K. Fulbeck, S. Lennon, and P. Spickard. *Part Asian, 100% Hapa*. Chronicle Books, 2006.

[9] T. Hastie, S. Rosset, J. Zhu, and H. Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[11] T. Kanade, T. Sakai, M. Nagao, and Y. ichi Ohta. Picture processing system using a computer complex. *Computer Graphics and Image Processing*, 2(3):207–215, 1973.

[12] R. Kramer and A. L. Jones. Do people's first names match their faces? *Journal of Articles in Support of the Null Hypothesis*, 12(1):1–8, 2015.

[13] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2169–2178, 2006.

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, volume 60, pages 91–110. Springer, 2004.

[15] D. Minter. *The Heart Of The King: The Quest Begins...* Gatekeeper Press, 2020.

[16] D. Parikh and K. Grauman. Interactively building a discriminative vocabulary of nameable attributes. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, pages 1681–1688. IEEE Computer Society, 2011.

[17] W. Shakespeare, H. Evans, and C. Praetorius. *Romeo and Juliet*. Works. C. Praetorius, 1886.

[18] J. Spinelli. *Stargirl*. Readers Circle. Laurel-Leaf Books, 2004.

[19] U.s. social security administration baby name database. `http://www.ssa.gov/oact/babynames`.

[20] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3360–3367, 2010.

[21] Z. Yonat, S. Anne-Laure, R. Nir, G. Jacob, and M. Ruth. We look like our names: The manifestation of name stereotypes in facial appearance. *Journal of Personality and Social Psychology*, 112(4):527–554, 2017.

Chapter A
Source Code

All the source code in this section are in Python language.

A.1 Preprocessing image using OpenCV

```python
##########################################################################
'''
    I am Kashyap Nirmal, M.Tech (ICT) student at DA-IICT.  I am working
    under the Thesis supervision of Prof. Manish Gupta, DA-IICT.
    This code file is a piece of work for my M.Tech thesis.

    The source will also be available at :
    https://github.com/Kashyap-Nirmal/Face_Name_Prediction

    => We have used following libraries in this code file:
        - OpenCV. Available at :https://opencv.org/
        - Python os. Available at:
        https://docs.python.org/3/library/os.html
        - Python sys. Available at:
        https://docs.python.org/3/library/sys.html

    => CODE and DATASET Directory structure details:

    - This code crops and resizes a face.
    - This code was executed on Google Colaboratory.
    - Create the OUTPUT_PATH Directory.
    - The dataset directory structure should be classwise directories.

        /Krishna/Krishna_1.jpg
        /Rahul/Rahul_1.jpg
        /Sonal/Sonal_1.jpg
        /Priya/Priya_1.jpg
'''
##########################################################################
import cv2
import sys
import os

#INSERT THE DATASET PATH HERE.
root_path = "DATASET_PATH"
save_path = "OUTPUT_PATH"
imagePath = root_path
```

```
for image_class in os.listdir(root_path):

    count = 0
    save_path_file = os.path.join(save_path,image_class)
    try:
      os.mkdir(save_path_file)
    except OSError as error:
      print(error)

    for img in os.listdir(os.path.join(root_path,image_class)):

      path_ = os.path.join(root_path,image_class,img)
      image = cv2.imread(path_)
      gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

      faceCascade = cv2.CascadeClassifier(
        cv2.data.haarcascades +
        "haarcascade_frontalface_default.xml"
      )
      faceDetected = faceCascade.detectMultiScale(
          gray,
          scaleFactor = 1.3,
          minNeighbors = 3,
          minSize = (30, 30)
      )

      for (x, y, w, h) in faceDetected:
          save_face = image[y:y + h, x:x + w]
          save_face = cv2.resize(save_face, (64, 64))
          filename = image_class + '_' + str(count) + '.jpg'
          cv2.imwrite(os.path.join(save_path_file,filename), save_face)

      count+=1
########################################################################
```

A.2 Vision Transformer

```
########################################################################
'''
    I am Kashyap Nirmal, M.Tech (ICT) student at DA-IICT.  I am working
    under the Thesis supervision of Prof. Manish Gupta, DA-IICT.
    This code file is a piece of work for my M.Tech thesis.

    The source will also be available at :
    https://github.com/Kashyap-Nirmal/Face_Name_Prediction

    => We have used following libraries in this code file:
        - Python time. Available at:
        https://docs.python.org/3/library/time.html
        - Python os. Available at:
        https://docs.python.org/3/library/os.html
        - datasets: Available at:
        https://huggingface.co/docs/datasets/index
        - numpy: Available at: https://numpy.org/
        - matplotlib: Available at: https://matplotlib.org/
        - seaborn: Available at: https://seaborn.pydata.org/
        - tensorflow: Available at: https://www.tensorflow.org/
        - keras: Available at: https://keras.io/
        - sklearn: Available at: https://scikit-learn.org/stable/
        - Python datetime. Available at:
        https://docs.python.org/3/library/datetime.html
        - pycm : Available at: https://www.pycm.io/
        - transformer: Available at:
        https://huggingface.co/docs/transformers/index

'''
#######################################################################
# This code block contains all the necessary imports.
import time
import os
import datasets
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import TensorBoard as \
TensorboardCallback, EarlyStopping

from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report,\
                                    accuracy_score,
                                    ConfusionMatrixDisplay

from datetime import datetime
from pycm import *

from transformers import ViTFeatureExtractor, ViTForImageClassification
from transformers import DefaultDataCollator
from transformers import TFViTForImageClassification, create_optimizer

#######################################################################

'''
  - Create a directory "0.Saved_Outputs"

  - This code block contains all the variables. Please set them as per
  as your convenience

  - The dataset directory structure should be classwise directories.

  /Krishna/Krishna_1.jpg
  /Rahul/Rahul_1.jpg
  /Sonal/Sonal_1.jpg
  /Priya/Priya_1.jpg

'''
```

```
model_id = "google/vit-base-patch16-224-in21k"

num_classes = 100
num_img_per_class = 800

#test size will be x% of train dataset
test_size = .20

num_train_epochs = 14
train_batch_size = 32
eval_batch_size = 32
learning_rate = 3e-5
weight_decay_rate=0.01
num_warmup_steps=0

class_data_labels = []
len_data = 0

# current dateTime
now = datetime.now()

# convert to string
date_time_str = now.strftime("%Y-%m-%d_%H:%M:%S")

#Name100 or Indian
dataset_name = "Name100"

#INSERT DATASET PATH HERE.
root_path = "./DATASET/"
filename_ = "ViT_"+dataset_name+"_"+str(num_classes)+ \
"_Classes_"+str(num_img_per_class)+"_images_"+date_time_str
save_path = "./0.Saved_Outputs/"
save_path += filename_ + '/'
os.mkdir(save_path)

######################################################################

feature_extractor = ViTFeatureExtractor.from_pretrained(model_id)

# basic processing (only resizing)
def process(examples):
    examples.update(feature_extractor(examples['img'], ))
    return examples

"""creates `Dataset` from image folder structure"""
def create_image_folder_dataset(root_path):

  global class_data_labels

  # get class names by folders names
  _CLASS_NAMES= os.listdir(root_path)
  # defines `datasets` features`
  features=datasets.Features(
    {
        "img" : datasets.Image(),
        "labels" : datasets.features.ClassLabel(
            names = _CLASS_NAMES
        ),
    }
  )
  # temp list holding datapoints for creation
  img_data_files = []
  label_data_files = []
  # load images into list for creation
  for img_class in os.listdir(root_path)[:(num_classes % 101)]:
    class_data_labels.append(img_class)
    for img in os.listdir(os.path.join(root_path,img_class))\ [:(
                                                    num_img_per_class % 801)]:
      path_=os.path.join(root_path,img_class,img)
      img_data_files.append(path_)
      label_data_files.append(img_class)

  global len_data
  len_data = len(img_data_files)
  # create dataset
  return datasets.Dataset.from_dict(
    {
        "img" : img_data_files,
        "labels" : label_data_files
    }, features=features
  )

#######################################################################

processed_dataset = create_image_folder_dataset(root_path)

processed_dataset = processed_dataset.map(process, batched = True)

processed_dataset = processed_dataset.shuffle().train_test_split(
    test_size = test_size
)

#######################################################################
'''
    Data collator that will dynamically pad the inputs received, as
    well as the labels.
'''
data_collator = DefaultDataCollator(return_tensors="tf")

# converting our train dataset to tf.data.Dataset
tf_train_dataset = processed_dataset["train"].to_tf_dataset(
   columns = ['pixel_values'],
   label_cols = ["labels"],
   shuffle = True,
   batch_size = train_batch_size,
   collate_fn = data_collator
)

# converting our test dataset to tf.data.Dataset
tf_eval_dataset = processed_dataset["test"].to_tf_dataset(
   columns = ['pixel_values'],
    label_cols = ["labels"],
    shuffle = False,
    batch_size = eval_batch_size,
    collate_fn = data_collator
)

#######################################################################

id2label = {str(i): label for i, label in enumerate(class_data_labels)}
label2id = {v: k for k, v in id2label.items()}

fp16 = True

if fp16:
  tf.keras.mixed_precision.set_global_policy("mixed_float16")

len_labels = len(class_data_labels)

# load pre-trained ViT model
model = TFViTForImageClassification.from_pretrained(
    model_id,
    #ignore_mismatched_sizes = True,
    num_labels = len_labels,
    id2label = id2label,
    label2id = label2id
)

# define loss
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)

# define metrics
metrics = [
    tf.keras.metrics.SparseCategoricalAccuracy(name = "accuracy"),
    tf.keras.metrics.SparseTopKCategoricalAccuracy(
        3,
        name = "top-3-accuracy"
    ),
]

num_train_steps = len(tf_train_dataset) * num_train_epochs
optimizer, lr_schedule = create_optimizer(
    init_lr=learning_rate,
    num_train_steps=num_train_steps,
    weight_decay_rate=weight_decay_rate,
    num_warmup_steps=num_warmup_steps,
)

# compile model
model.compile(
    optimizer = optimizer,
    loss=loss,
    metrics=metrics,
)

model.summary()
```

```
########################################################################

model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath = save_path,
    save_weights_only = True,
    monitor = 'val_accuracy',
    mode = 'max',
    save_best_only = True
)

early_Stopper = tf.keras.callbacks.EarlyStopping(
    monitor = "val_accuracy",
    patience = 1
)

########################################################################

train_results = model.fit(
    tf_train_dataset,
    validation_data=tf_eval_dataset,
    callbacks = [model_checkpoint_callback],#,early_Stopper
    epochs=num_train_epochs,
)

########################################################################

try:
  model.save(save_path + "Model_" + filename_)
except Exception as e:
  print("Error in saving model. Error msg" + str(e))

try:
  ypred = model.predict(tf_eval_dataset)
  ypred = ypred.logits.argmax(-1)

  true_categories = tf.concat([y for x, y in tf_eval_dataset] , axis=0)
  true_categories = np.array(true_categories)

except Exception as e:
  print("Error Model Predict" + str(e))

########################################################################

# Create CM From Data
try:

  cm = ConfusionMatrix(
    actual_vector = true_categories,
    predict_vector = ypred
  )

  try:

    cm.save_html(
        save_path + "Confusion_Matrix_" + filename_ ,
        color = "Blue" ,
        normalize = True
    )

  except Exception as e:

    print("Error in saving Html. Error msg" + str(e))

  try:

    cm.save_csv(
        save_path + "Confusion_Matrix_" + filename_ ,
        normalize=True
    )

  except Exception as e:

    print("Error in saving CSV. Error msg" + str(e))

  try:

    cm.save_obj(save_path + "Confusion_Matrix_" + filename_)

  except Exception as e:

    print("Error in saving obj. Error msg" + str(e))

except Exception as e:

  print("Error in CM 1 : ", e)

try:

  confusionmatrix = confusion_matrix(
    true_categories ,
    ypred ,
    normalize = 'true'
  )
  print("Confusion Matrix : ")
  print(confusionmatrix)
  cnt = 1
  for i in confusionmatrix:
    print(cnt)
    print(i)
    cnt += 1

except Exception as e:

  print("Error in CM 1 : ", e)

########################################################################

try:

  print("\nTotal number of classes: ", len(class_data_labels))
  if(len_data != 0):
    print("\nTotal number of images : ", len_data)
  print("\nList of class labels : ")
  print(class_data_labels)
  print("Train test split : {0} - {1}".format(
    100*(1-test_size),test_size*100)
  )
  print("Train Batch size : ", train_batch_size)
  print("Test Batch Size : ", eval_batch_size)
  print("Number of epochs : ", num_train_epochs)
  print("Model used here : ", model_id)
  print("Optimizer : Custom optimizer created.")
  print("Learning rate : "+str(learning_rate))
  print("\nClassification Report : ")
  print(classification_report(true_categories, ypred))
  ypred = ypred.reshape((ypred.shape[0], 1))
  true_categories = true_categories.reshape(
    (
        true_categories.shape[0],
        1
    )
  )
  print("Accuracy:", accuracy_score(true_categories, ypred))

except Exception as e:
  print("Error Print section" + str(e))

########################################################################
```

A.3 XGBoost with Spatial Pyramid Pooling for feature extraction

```
########################################################################
'''
    I am Kashyap Nirmal, M.Tech (ICT) student at DA-IICT.  I am working
    under the Thesis supervision of Prof. Manish Gupta, DA-IICT.
    This code file is a piece of work for my M.Tech thesis.

    The source will also be available at :
    https://github.com/Kashyap-Nirmal/Face_Name_Prediction

    => We have used following libraries in this code file:
        - tensorflow: Available at: https://www.tensorflow.org/
        - keras: Available at: https://keras.io/
        - sklearn: Available at: https://scikit-learn.org/stable/
        - xgboost: Available at:
        https://xgboost.readthedocs.io/en/stable/python/python_intro.
                                                    html
        - matplotlib: Available at: https://matplotlib.org/
        - seaborn: Available at: https://seaborn.pydata.org/
        - numpy: Available at: https://numpy.org/
        - Python random. Available at:
        https://docs.python.org/3/library/random.html
        - OpenCV. Available at :https://opencv.org/
        - Python os. Available at:
```

```
        https://docs.python.org/3/library/os.html
        - Python glob. Available at:
        https://docs.python.org/3/library/glob.html
        - Python math. Available at:
        https://docs.python.org/3/library/math.html

'''
######################################################################
# This code block contains all the necessary imports.
# This code was executed on Google Colaboratory.
from keras.preprocessing.image import img_to_array
from keras.layers import MaxPooling2D, Flatten, Dense
from keras.applications.vgg16 import VGG16
from keras.models import Model
from sklearn.metrics import confusion_matrix, classification_report,\
                                    accuracy_score

import xgboost as xgb
import matplotlib.pyplot as plot
import seaborn as sns
import numpy as npy
import random
import cv2
import os
import glob
import math

######################################################################

'''
  - The dataset directory structure should be "train" directory which
  contains all the training classes as subdirectories.
  - And "test" directory which contains all the testing classes as
  subdirectories.
  - Image resolution used is 64 * 64.

  => For instance:
  /DATASET/train/Krishna/Krishna_1.jpg
  /DATASET/train/Rahul/Rahul_1.jpg

  /DATASET/test/Krishna/Krishna_1.jpg
  /DATASET/test/Rahul/Rahul_1.jpg

'''

training_files = []
#INSERT THE TRAINING DATASET PATH HERE.
for filename in glob.glob(r"DATASET/train" + "/**/*", recursive= True):
    training_files.append(filename)

testing_files = []
#INSERT THE TRAINING DATASET PATH HERE.
for filename in glob.glob(r"DATASET/test" + "/**/*", recursive = True):
    testing_files.append(filename)

######################################################################
```

```
random.shuffle(training_files)
print(len(training_files))
print(len(testing_files))

######################################################################

#INSERT THE CLASS LABELS HERE.
class_labels = {
    "Krishna":0,
    "Pranav":1,
    "Priya":2,
    "Rahul":3,
    "Sonal": 4
}

######################################################################

training_image = []
testing_image = []
training_label = []
testing_label = []

for imgX in training_files:
  try:
    image = cv2.imread(imgX)
    image = cv2.resize(image, (64, 64))
    image = img_to_array(image)
    training_image.append(image)
    '''
        Suppose below is the naming convention used. We want class
        label 'Krishna'. So we will extract last second element of
        the list.

        C:\Files\gender_dataset_face\Krishna\Krishna_1.jpg.
    '''
    label = imgX.split(os.path.sep)[-2]
    l_index = class_labels[label]

    training_label.append([l_index])

  except:
    print("Corrupted.")

for imgX in testing_files:
  try:
    image = cv2.imread(imgX)
    image = cv2.resize(image, (64, 64))
    image = img_to_array(image)
    testing_image.append(image)

    label = imgX.split(os.path.sep)[-2]
    l_index = class_labels[label]

    testing_label.append([l_index])
  except:
    print("Corrupted.")

########################################################################

training_image = npy.array(training_image, dtype = "float")/255.0
testing_image = npy.array(testing_image, dtype = "float")/255.0

########################################################################

training_label = npy.array(training_label)
testing_label = npy.array(testing_label)

########################################################################

training_image.shape

########################################################################

#Feature extraction using SPP

base_model_ = VGG16(
    include_top = False,
    weights = 'imagenet',
    input_shape = (64, 64, 3)
)

win1 = math.ceil(16/1)
str1 = math.floor(16/1)

win2 = math.ceil(16/2)
str2 = math.floor(16/2)

win3 = math.ceil(16/4)
str3 = math.floor(16/4)

l1 = MaxPooling2D(
    pool_size = (win1),
    strides = str1,
    padding ="valid"
)(base_model_.layers[-10].output)
l2 = MaxPooling2D(
    pool_size = (win2),
    strides = str2,
    padding = "valid"
)(base_model_.layers[-10].output)
l3 = MaxPooling2D(
    pool_size = (win3),
    strides = str3,
    padding = "valid"
)(base_model_.layers[-10].output)

flat1 = Flatten()(l1)
flat2 = Flatten()(l2)
flat3 = Flatten()(l3)

for layer in base_model_.layers:
    layer.trainable = False

modelX = Model(
    inputs = base_model_.inputs,
    outputs = [flat1,flat2,flat3]
)

########################################################################

modelX.summary()

########################################################################

trainX_deep = modelX.predict(training_image)
testX_deep = modelX.predict(testing_image)

########################################################################

trainX_deep = npy.hstack(
    [
        trainX_deep[0],
        trainX_deep[1],
        trainX_deep[2]
    ]
)
testX_deep = npy.hstack([testX_deep[0], testX_deep[1], testX_deep[2]])

########################################################################

xgboost_classifier = xgb.XGBClassifier()

########################################################################

xgboost_classifier.fit(trainX_deep, training_label)

########################################################################

label_predicted = xgboost_classifier.predict(testX_deep)
print("Accuracy:", accuracy_score(testing_label, label_predicted))

########################################################################

confusionmatrix = confusion_matrix(
    testing_label,
    label_predicted,
    normalize = 'true'
)
axX = sns.heatmap(
    confusionmatrix ,
    cmap = 'Blues',
    annot = True ,
    cbar = True ,
    fmt ='.2%'
)
axX.figbox
list1 = []
```

```
for key in class_labels.keys():
  list1.append(key)
axX.xaxis.set_ticklabels(list1)
axX.yaxis.set_ticklabels(list1)
plot.setp(
    axX.get_xticklabels() ,
    rotation = 45 ,
    ha = "right",
    rotation_mode = "anchor"
)
axX.set_title('Confusion Matrix for Name Prediction\n\n');
plot.show()

#######################################################################

print(classification_report(testing_label, label_predicted))

#######################################################################
```

A.4 Bagging with Spatial Pyramid Pooling for feature extraction

```
#######################################################################
'''
    I am Kashyap Nirmal, M.Tech (ICT) student at DA-IICT.  I am working
    under the Thesis supervision of Prof. Manish Gupta, DA-IICT.
    This code file is a piece of work for my M.Tech thesis.

    The source will also be available at :
    https://github.com/Kashyap-Nirmal/Face_Name_Prediction

    => We have used following libraries in this code file:
        - tensorflow: Available at: https://www.tensorflow.org/
        - keras: Available at: https://keras.io/
        - sklearn: Available at: https://scikit-learn.org/stable/
        - xgboost: Available at:
        https://xgboost.readthedocs.io/en/stable/python/python_intro.
                                                    html
        - matplotlib: Available at: https://matplotlib.org/
        - seaborn: Available at: https://seaborn.pydata.org/
        - numpy: Available at: https://numpy.org/
        - Python random. Available at:
        https://docs.python.org/3/library/random.html
        - OpenCV. Available at :https://opencv.org/
        - Python os. Available at:
        https://docs.python.org/3/library/os.html
        - Python glob. Available at:
        https://docs.python.org/3/library/glob.html
        - Python math. Available at:
        https://docs.python.org/3/library/math.html

'''
#######################################################################
```

```
# This code block contains all the necessary imports.
# This code was executed on Google Colaboratory.

from keras.preprocessing.image import img_to_array
from keras.layers import MaxPooling2D, Flatten, Dense
from keras.applications.vgg16 import VGG16
from keras.models import Model
from sklearn.metrics import confusion_matrix, classification_report, \
                                    accuracy_score

import matplotlib.pyplot as plot
import seaborn as sns
import numpy as npy
import random
import cv2
import os
import glob
import math

from sklearn.ensemble import BaggingClassifier
from sklearn.svm import SVC

#######################################################################
'''
  - The dataset directory structure should be "train" directory which
  contains all the training classes as subdirectories.
  - And "test" directory which contains all the testing classes as
  subdirectories.
  - Image resolution used is 64 * 64.

  => For instance:
  /DATASET/train/Krishna/Krishna_1.jpg
  /DATASET/train/Rahul/Rahul_1.jpg

  /DATASET/test/Krishna/Krishna_1.jpg
  /DATASET/test/Rahul/Rahul_1.jpg

'''

training_files = []
#INSERT THE TRAINING DATASET PATH HERE.
for filename in glob.glob(r"DATASET/train" + "/**/*", recursive= True):
    training_files.append(filename)

testing_files = []
#INSERT THE TRAINING DATASET PATH HERE.
for filename in glob.glob(r"DATASET/test" + "/**/*", recursive = True):
    testing_files.append(filename)

#######################################################################

random.shuffle(training_files)
print(len(training_files))
print(len(testing_files))

#######################################################################
```

```python
#INSERT THE CLASS LABELS HERE.
class_labels = {
    "Krishna":0,
    "Pranav":1,
    "Priya":2,
    "Rahul":3,
    "Sonal": 4
}

########################################################################

training_image = []
testing_image = []
training_label = []
testing_label = []

for imgX in training_files:
  try:
    image = cv2.imread(imgX)
    image = cv2.resize(image, (64, 64))
    image = img_to_array(image)
    training_image.append(image)
    '''
        Suppose below is the naming convention used. We want class
        label 'Krishna'. So we will extract last second element of
        the list.

        C:\Files\gender_dataset_face\Krishna\Krishna_1.jpg.
    '''
    label = imgX.split(os.path.sep)[-2]
    l_index = class_labels[label]

    training_label.append([l_index])

  except:
    print("Corrupted.")

for imgX in testing_files:
  try:
    image = cv2.imread(imgX)
    image = cv2.resize(image, (64, 64))
    image = img_to_array(image)
    testing_image.append(image)

    label = imgX.split(os.path.sep)[-2]
    l_index = class_labels[label]

    testing_label.append([l_index])
  except:
    print("Corrupted.")

########################################################################

training_image = npy.array(training_image, dtype = "float")/255.0
testing_image = npy.array(testing_image, dtype = "float")/255.0
```

```
######################################################################

training_label = npy.array(training_label)
testing_label = npy.array(testing_label)

######################################################################

#Feature extraction using SPP

base_model_ = VGG16(
    include_top = False,
    weights = 'imagenet',
    input_shape = (64, 64, 3)
)

win1 = math.ceil(16/1)
str1 = math.floor(16/1)

win2 = math.ceil(16/2)
str2 = math.floor(16/2)

win3 = math.ceil(16/4)
str3 = math.floor(16/4)

l1 = MaxPooling2D(
    pool_size = (win1),
    strides = str1,
    padding = "valid"
)(base_model_.layers[-10].output)
l2 = MaxPooling2D(
    pool_size = (win2),
    strides = str2,
    padding = "valid"
)(base_model_.layers[-10].output)
l3 = MaxPooling2D(
    pool_size = (win3),
    strides = str3,
    padding = "valid"
)(base_model_.layers[-10].output)

flat1 = Flatten()(l1)
flat2 = Flatten()(l2)
flat3 = Flatten()(l3)

for layer in base_model_.layers:
    layer.trainable = False

modelX = Model(
    inputs = base_model_.inputs,
    outputs = [flat1, flat2, flat3]
)

######################################################################

modelX.summary()
```

```
######################################################################

svc = SVC(probability = True, kernel = "linear")
bagging_classifier = BaggingClassifier(
    base_estimator = svc,
    n_estimators = 21
)

######################################################################

trainX_deep = modelX.predict(training_image)
testX_deep = modelX.predict(testing_image)

######################################################################

trainX_deep = npy.hstack(
    [trainX_deep[0], trainX_deep[1], trainX_deep[2]]
)
testX_deep = npy.hstack([testX_deep[0], testX_deep[1], testX_deep[2]])

######################################################################

clfmodel = bagging_classifier.fit(trainX_deep, training_label)

######################################################################

label_predicted = clfmodel.predict(testX_deep)
print("Accuracy:", accuracy_score(testing_label, label_predicted))

######################################################################

confusionmatrix = confusion_matrix(
    testing_label,
    label_predicted,
    normalize = 'true'
)
axX = sns.heatmap(
    confusionmatrix ,
    cmap = 'Blues',
    annot = True ,
    cbar = True ,
    fmt ='.2%'
)
axX.figbox
list1 = []
for key in class_labels.keys():
  list1.append(key)
axX.xaxis.set_ticklabels(list1)
axX.yaxis.set_ticklabels(list1)
plot.setp(
    axX.get_xticklabels() ,
    rotation = 45 ,
    ha = "right",
    rotation_mode = "anchor"
)
```

```
axX.set_title('Confusion Matrix for Name Prediction\n\n');
plot.show()

######################################################################

print(classification_report(testing_label, label_predicted))

######################################################################
```

A.5 Adaboost with Spatial Pyramid Pooling for feature extraction

```
######################################################################
'''
    I am Kashyap Nirmal, M.Tech (ICT) student at DA-IICT.  I am working
    under the Thesis supervision of Prof. Manish Gupta, DA-IICT.
    This code file is a piece of work for my M.Tech thesis.

    The source will also be available at :
    https://github.com/Kashyap-Nirmal/Face_Name_Prediction

    => We have used following libraries in this code file:
        - tensorflow: Available at: https://www.tensorflow.org/
        - keras: Available at: https://keras.io/
        - sklearn: Available at: https://scikit-learn.org/stable/
        - xgboost: Available at:
        https://xgboost.readthedocs.io/en/stable/python/python_intro.
                                                    html
        - matplotlib: Available at: https://matplotlib.org/
        - seaborn: Available at: https://seaborn.pydata.org/
        - numpy: Available at: https://numpy.org/
        - Python random. Available at:
        https://docs.python.org/3/library/random.html
        - OpenCV. Available at :https://opencv.org/
        - Python os. Available at:
        https://docs.python.org/3/library/os.html
        - Python glob. Available at:
        https://docs.python.org/3/library/glob.html
        - Python math. Available at:
        https://docs.python.org/3/library/math.html

'''
######################################################################
# This code block contains all the necessary imports.
# This code was executed on Google Colaboratory.

from keras.preprocessing.image import img_to_array
from keras.layers import MaxPooling2D, Flatten, Dense
from keras.applications.vgg16 import VGG16
from keras.models import Model
from sklearn.metrics import confusion_matrix, classification_report, \
                                    accuracy_score
```

```python
import matplotlib.pyplot as plot
import seaborn as sns
import numpy as npy
import random
import cv2
import os
import glob
import math

from sklearn.ensemble import AdaBoostClassifier
from sklearn.svm import SVC

########################################################################
'''
  - The dataset directory structure should be "train" directory which
  contains all the training classes as subdirectories.
  - And "test" directory which contains all the testing classes as
  subdirectories.
  - Image resolution used is 64 * 64.

  => For instance:
  /DATASET/train/Krishna/Krishna_1.jpg
  /DATASET/train/Rahul/Rahul_1.jpg

  /DATASET/test/Krishna/Krishna_1.jpg
  /DATASET/test/Rahul/Rahul_1.jpg

'''

training_files = []
#INSERT THE TRAINING DATASET PATH HERE.
for filename in glob.glob(r"DATASET/train" + "/**/*", recursive= True):
    training_files.append(filename)

testing_files = []
#INSERT THE TRAINING DATASET PATH HERE.
for filename in glob.glob(r"DATASET/test" + "/**/*", recursive = True):
    testing_files.append(filename)

########################################################################

random.shuffle(training_files)
print(len(training_files))
print(len(testing_files))

########################################################################

#INSERT THE CLASS LABELS HERE.
class_labels = {
    "Krishna":0,
    "Pranav":1,
    "Priya":2,
    "Rahul":3,
    "Sonal": 4
}
```

```
#######################################################################

training_image = []
testing_image = []
training_label = []
testing_label = []

for imgX in training_files:
  try:
    image = cv2.imread(imgX)
    image = cv2.resize(image, (64, 64))
    image = img_to_array(image)
    training_image.append(image)
    '''
        Suppose below is the naming convention used. We want class
        label 'Krishna'. So we will extract last second element of
        the list.

        C:\Files\gender_dataset_face\Krishna\Krishna_1.jpg.
    '''
    label = imgX.split(os.path.sep)[-2]
    l_index = class_labels[label]

    training_label.append([l_index])

  except:
    print("Corrupted.")

for imgX in testing_files:
  try:
    image = cv2.imread(imgX)
    image = cv2.resize(image, (64, 64))
    image = img_to_array(image)
    testing_image.append(image)

    label = imgX.split(os.path.sep)[-2]
    l_index = class_labels[label]

    testing_label.append([l_index])
  except:
    print("Corrupted.")

#######################################################################

training_image = npy.array(training_image, dtype = "float")/255.0
testing_image = npy.array(testing_image, dtype = "float")/255.0

#######################################################################

training_label = npy.array(training_label)
testing_label = npy.array(testing_label)

#######################################################################

#Feature extraction using SPP
```

```
base_model_ = VGG16(
    include_top = False,
    weights = 'imagenet',
    input_shape = (64, 64, 3)
)

win1 = math.ceil(16/1)
str1 = math.floor(16/1)

win2 = math.ceil(16/2)
str2 = math.floor(16/2)

win3 = math.ceil(16/4)
str3 = math.floor(16/4)

l1 = MaxPooling2D(
    pool_size = (win1),
    strides = str1,
    padding = "valid"
)(base_model_.layers[-10].output)
l2 = MaxPooling2D(
    pool_size = (win2),
    strides = str2,
    padding = "valid"
)(base_model_.layers[-10].output)
l3 = MaxPooling2D(
    pool_size = (win3),
    strides = str3,
    padding = "valid"
)(base_model_.layers[-10].output)

flat1 = Flatten()(l1)
flat2 = Flatten()(l2)
flat3 = Flatten()(l3)

for layer in base_model_.layers:
    layer.trainable = False

modelX = Model(
    inputs = base_model_.inputs,
    outputs = [flat1, flat2, flat3]
)

########################################################################

modelX.summary()

########################################################################

svc = SVC(probability = True, kernel = "linear")
ADBclassifier = AdaBoostClassifier(
    n_estimators = 21,
    base_estimator = svc,
    learning_rate = 0.5,
    algorithm = 'SAMME.R'
)
```

```
########################################################################

trainX_deep = modelX.predict(training_image)
testX_deep = modelX.predict(testing_image)

########################################################################

trainX_deep = npy.hstack(
    [trainX_deep[0], trainX_deep[1], trainX_deep[2]]
)
testX_deep = npy.hstack([testX_deep[0], testX_deep[1], testX_deep[2]])

########################################################################

clfmodel = ADBclassifier.fit(trainX_deep, training_label)

########################################################################

label_predicted = ADBclassifier.predict(testX_deep)
print("Accuracy:", accuracy_score(testing_label, label_predicted))

########################################################################

confusionmatrix = confusion_matrix(
    testing_label,
    label_predicted,
    normalize = 'true'
)
axX = sns.heatmap(
    confusionmatrix ,
    cmap = 'Blues',
    annot = True ,
    cbar = True ,
    fmt ='.2%'
)
axX.figbox
list1 = []
for key in class_labels.keys():
  list1.append(key)
axX.xaxis.set_ticklabels(list1)
axX.yaxis.set_ticklabels(list1)
plot.setp(
    axX.get_xticklabels() ,
    rotation = 45 ,
    ha = "right",
    rotation_mode = "anchor"
)
axX.set_title('Confusion Matrix for Name Prediction\n\n');
plot.show()

########################################################################

print(classification_report(testing_label, label_predicted))

########################################################################
```

A.6 Bagging with Deep learning feature

```
########################################################################
'''
    I am Kashyap Nirmal, M.Tech (ICT) student at DA-IICT.  I am working
    under the Thesis supervision of Prof. Manish Gupta, DA-IICT.
    This code file is a piece of work for my M.Tech thesis.

    The source will also be available at :
    https://github.com/Kashyap-Nirmal/Face_Name_Prediction

    => We have used following libraries in this code file:
        - tensorflow: Available at: https://www.tensorflow.org/
        - keras: Available at: https://keras.io/
        - sklearn: Available at: https://scikit-learn.org/stable/
        - xgboost: Available at:
        https://xgboost.readthedocs.io/en/stable/python/python_intro.
                                                    html
        - matplotlib: Available at: https://matplotlib.org/
        - seaborn: Available at: https://seaborn.pydata.org/
        - numpy: Available at: https://numpy.org/
        - Python random. Available at:
        https://docs.python.org/3/library/random.html
        - OpenCV. Available at :https://opencv.org/
        - Python os. Available at:
        https://docs.python.org/3/library/os.html
        - Python glob. Available at:
        https://docs.python.org/3/library/glob.html
        - Python math. Available at:
        https://docs.python.org/3/library/math.html

'''
########################################################################
# This code block contains all the necessary imports.
# This code was executed on Google Colaboratory.

from keras.preprocessing.image import img_to_array
from keras.layers import MaxPooling2D, Flatten, Dense
from keras.applications.vgg16 import VGG16
from keras.models import Model
from sklearn.metrics import confusion_matrix, classification_report,\
                                    accuracy_score

import matplotlib.pyplot as plot
import seaborn as sns
import numpy as npy
import random
import cv2
import os
import glob
import math

from sklearn.ensemble import BaggingClassifier
from sklearn.svm import SVC

########################################################################
```

```
'''
  - The dataset directory structure should be "train" directory which
  contains all the training classes as subdirectories.
  - And "test" directory which contains all the testing classes as
  subdirectories.
  - Image resolution used is 64 * 64.

  => For instance:
  /DATASET/train/Krishna/Krishna_1.jpg
  /DATASET/train/Rahul/Rahul_1.jpg

  /DATASET/test/Krishna/Krishna_1.jpg
  /DATASET/test/Rahul/Rahul_1.jpg

'''

training_files = []
#INSERT THE TRAINING DATASET PATH HERE.
for filename in glob.glob(r"DATASET/train" + "/**/*", recursive= True):
    training_files.append(filename)

testing_files = []
#INSERT THE TRAINING DATASET PATH HERE.
for filename in glob.glob(r"DATASET/test" + "/**/*", recursive = True):
    testing_files.append(filename)

########################################################################

random.shuffle(training_files)
print(len(training_files))
print(len(testing_files))

########################################################################

#INSERT THE CLASS LABELS HERE.
class_labels = {
    "Krishna":0,
    "Pranav":1,
    "Priya":2,
    "Rahul":3,
    "Sonal": 4
}

########################################################################

training_image = []
testing_image = []
training_label = []
testing_label = []

for imgX in training_files:
  try:
    image = cv2.imread(imgX)
    image = cv2.resize(image, (64, 64))
    image = img_to_array(image)
    training_image.append(image)
    '''
        Suppose below is the naming convention used. We want class
        label 'Krishna'. So we will extract last second element of
        the list.

        C:\Files\gender_dataset_face\Krishna\Krishna_1.jpg.
    '''
    label = imgX.split(os.path.sep)[-2]
    l_index = class_labels[label]

    training_label.append([l_index])

  except:
    print("Corrupted.")

for imgX in testing_files:
  try:
    image = cv2.imread(imgX)
    image = cv2.resize(image, (64, 64))
    image = img_to_array(image)
    testing_image.append(image)

    label = imgX.split(os.path.sep)[-2]
    l_index = class_labels[label]

    testing_label.append([l_index])
  except:
    print("Corrupted.")

########################################################################

#Feature extraction using DL.

base_model = VGG16(
    include_top = False,
    weights = 'imagenet',
    input_shape = (64, 64, 3),
    classes = 5
)

flat1 = Flatten()(base_model.layers[-1].output)
class1 = Dense(4096, activation = 'relu')(flat1)

modelX = Model(inputs = base_model.inputs, outputs = class1)

########################################################################

modelX.summary()

########################################################################

training_image = npy.array(training_image, dtype = "float")/255.0
testing_image = npy.array(testing_image, dtype = "float")/255.0

########################################################################
```

```
training_label = npy.array(training_label)
testing_label = npy.array(testing_label)

#######################################################################

training_image.shape

#######################################################################

trainX_deep = modelX.predict(training_image)
testX_deep = modelX.predict(testing_image)

#######################################################################

svc = SVC(probability = True, kernel = "linear")
bagging_classifier = BaggingClassifier(
    base_estimator = svc,
    n_estimators = 50
)

#######################################################################

clfmodel = bagging_classifier.fit(trainX_deep, training_label)

#######################################################################

label_predicted = clfmodel.predict(testX_deep)
print("Accuracy:", accuracy_score(testing_label, label_predicted))

#######################################################################

confusionmatrix = confusion_matrix(
    testing_label,
    label_predicted,
    normalize = 'true'
)
axX = sns.heatmap(
    confusionmatrix ,
    cmap = 'Blues',
    annot = True ,
    cbar = True ,
    fmt ='.2%'
)
axX.figbox
list1 = []
for key in class_labels.keys():
  list1.append(key)
axX.xaxis.set_ticklabels(list1)
axX.yaxis.set_ticklabels(list1)
plot.setp(
    axX.get_xticklabels() ,
    rotation=45 ,
    ha="right",
    rotation_mode="anchor"
)
axX.set_title('Confusion Matrix for Name Prediction\n\n');
```

```
plot.show()

########################################################################

print(classification_report(testing_label, label_predicted))

########################################################################
```

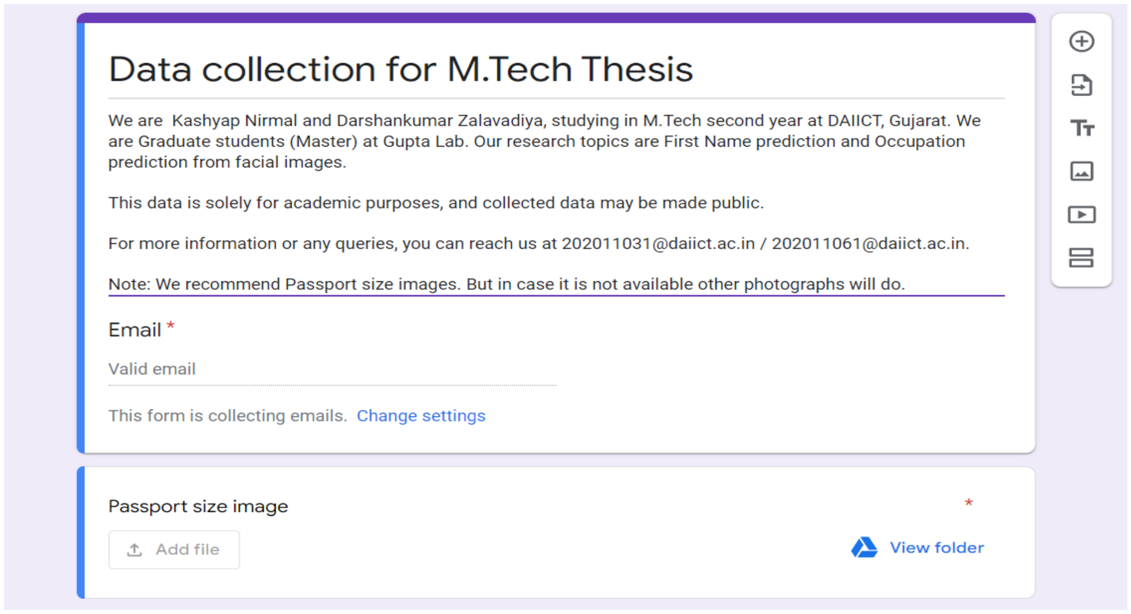
A.7 Adaboost with Deep learning feature

```
########################################################################
'''
    I am Kashyap Nirmal, M.Tech (ICT) student at DA-IICT.  I am working
    under the Thesis supervision of Prof. Manish Gupta, DA-IICT.
    This code file is a piece of work for my M.Tech thesis.

    The source will also be available at :
    https://github.com/Kashyap-Nirmal/Face_Name_Prediction

    => We have used following libraries in this code file:
        - tensorflow: Available at: https://www.tensorflow.org/
        - keras: Available at: https://keras.io/
        - sklearn: Available at: https://scikit-learn.org/stable/
        - xgboost: Available at:
        https://xgboost.readthedocs.io/en/stable/python/python_intro.
                                    html
        - matplotlib: Available at: https://matplotlib.org/
        - seaborn: Available at: https://seaborn.pydata.org/
        - numpy: Available at: https://numpy.org/
        - Python random. Available at:
        https://docs.python.org/3/library/random.html
        - OpenCV. Available at :https://opencv.org/
        - Python os. Available at:
        https://docs.python.org/3/library/os.html
        - Python glob. Available at:
        https://docs.python.org/3/library/glob.html
        - Python math. Available at:
        https://docs.python.org/3/library/math.html

'''
########################################################################
# This code block contains all the necessary imports.
# This code was executed on Google Colaboratory.
from keras.preprocessing.image import img_to_array
from keras.layers import MaxPooling2D, Flatten, Dense
from keras.applications.vgg16 import VGG16
from keras.models import Model
from sklearn.metrics import confusion_matrix, classification_report,\
                                    accuracy_score

import matplotlib.pyplot as plot
import seaborn as sns
import numpy as npy
import random
import cv2
import os
import glob
import math

from sklearn.ensemble import AdaBoostClassifier
from sklearn.svm import SVC

#######################################################################
'''
  - The dataset directory structure should be "train" directory which
  contains all the training classes as subdirectories.
  - And "test" directory which contains all the testing classes as
  subdirectories.
  - Image resolution used is 64 * 64.

  => For instance:
  /DATASET/train/Krishna/Krishna_1.jpg
  /DATASET/train/Rahul/Rahul_1.jpg

  /DATASET/test/Krishna/Krishna_1.jpg
  /DATASET/test/Rahul/Rahul_1.jpg

'''

training_files = []
#INSERT THE TRAINING DATASET PATH HERE.
for filename in glob.glob(r"DATASET/train" + "/**/*", recursive= True):
    training_files.append(filename)

testing_files = []
#INSERT THE TRAINING DATASET PATH HERE.
for filename in glob.glob(r"DATASET/test" + "/**/*", recursive = True):
    testing_files.append(filename)

#######################################################################

random.shuffle(training_files)
print(len(training_files))
print(len(testing_files))

#######################################################################

#INSERT THE CLASS LABELS HERE.
class_labels = {
    "Krishna":0,
    "Pranav":1,
    "Priya":2,
    "Rahul":3,
    "Sonal": 4
}

#######################################################################

training_image = []
testing_image = []
```

```
training_label = []
testing_label = []

for imgX in training_files:
  try:
    image = cv2.imread(imgX)
    image = cv2.resize(image, (64, 64))
    image = img_to_array(image)
    training_image.append(image)
    '''
        Suppose below is the naming convention used. We want class
        label 'Krishna'. So we will extract last second element of
        the list.

        C:\Files\gender_dataset_face\Krishna\Krishna_1.jpg.
    '''
    label = imgX.split(os.path.sep)[-2]
    l_index = class_labels[label]

    training_label.append([l_index])

  except:
    print("Corrupted.")

for imgX in testing_files:
  try:
    image = cv2.imread(imgX)
    image = cv2.resize(image, (64, 64))
    image = img_to_array(image)
    testing_image.append(image)

    label = imgX.split(os.path.sep)[-2]
    l_index = class_labels[label]

    testing_label.append([l_index])
  except:
    print("Corrupted.")

########################################################################
#Feature extraction using DL.

base_model_ = VGG16(
    include_top = False,
    weights = 'imagenet',
    input_shape = (64, 64, 3),
    classes = 5
)

flat1 = Flatten()(base_model_.layers[-1].output)
class1 = Dense(4096, activation = 'relu')(flat1)

modelX = Model(inputs=base_model_.inputs, outputs=class1)

########################################################################

modelX.summary()
```

```
########################################################################

training_image = npy.array(training_image, dtype = "float")/255.0
testing_image = npy.array(testing_image, dtype = "float")/255.0

########################################################################

training_label = npy.array(training_label)
testing_label = npy.array(testing_label)

########################################################################

training_image.shape

########################################################################

trainX_deep = modelX.predict(training_image)
testX_deep = modelX.predict(testing_image)

########################################################################

svc = SVC(probability = True, kernel = "linear")
adaboost = AdaBoostClassifier(
    n_estimators = 3,
    base_estimator = svc,
    learning_rate = 0.5,
    algorithm = 'SAMME.R'
)

########################################################################

clfmodel = adaboost.fit(trainX_deep, training_label)

########################################################################

label_predicted = clfmodel.predict(testX_deep)
print("Accuracy:", accuracy_score(testing_label, label_predicted))

########################################################################

confusionmatrix = confusion_matrix(
    testing_label,
    label_predicted,
    normalize = 'true'
)
axX = sns.heatmap(
    confusionmatrix ,
    cmap = 'Blues',
    annot = True ,
    cbar = True ,
    fmt ='.2%'
)
axX.figbox
list1 = []
for key in class_labels.keys():
  list1.append(key)
axX.xaxis.set_ticklabels(list1)
axX.yaxis.set_ticklabels(list1)
plot.setp(
    axX.get_xticklabels() ,
    rotation = 45 ,
    ha = "right",
    rotation_mode = "anchor"
)
axX.set_title('Confusion Matrix for Name Prediction\n\n');
plot.show()

########################################################################

print(classification_report(testing_label , label_predicted))

########################################################################
```

CHAPTER B
Additional snapshots



Data collection for M.Tech Thesis

We are Kashyap Nirmal and Darshankumar Zalavadiya, studying in M.Tech second year at DAIICT, Gujarat. We are Graduate students (Master) at Gupta Lab. Our research topics are First Name prediction and Occupation prediction from facial images.

This data is solely for academic purposes, and collected data may be made public.

For more information or any queries, you can reach us at 202011031@daiict.ac.in / 202011061@daiict.ac.in.

Note: We recommend Passport size images. But in case it is not available other photographs will do.

Email *

Valid email

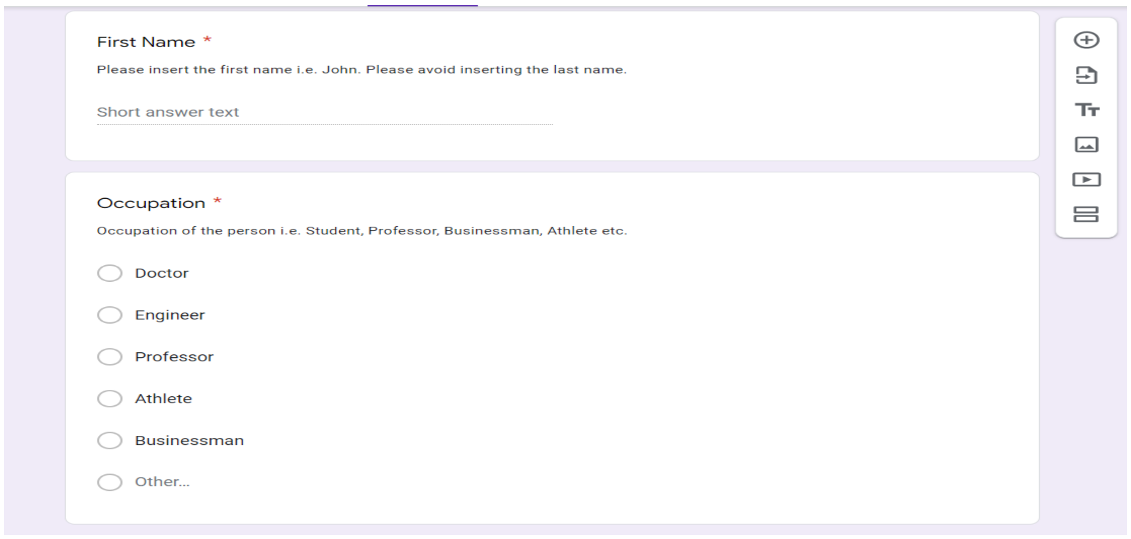
This form is collecting emails. Change settings

Passport size image *

Add file

View folder

Figure B.1: Google Form title and information fields snapshot.



First Name *

Please insert the first name i.e. John. Please avoid inserting the last name.

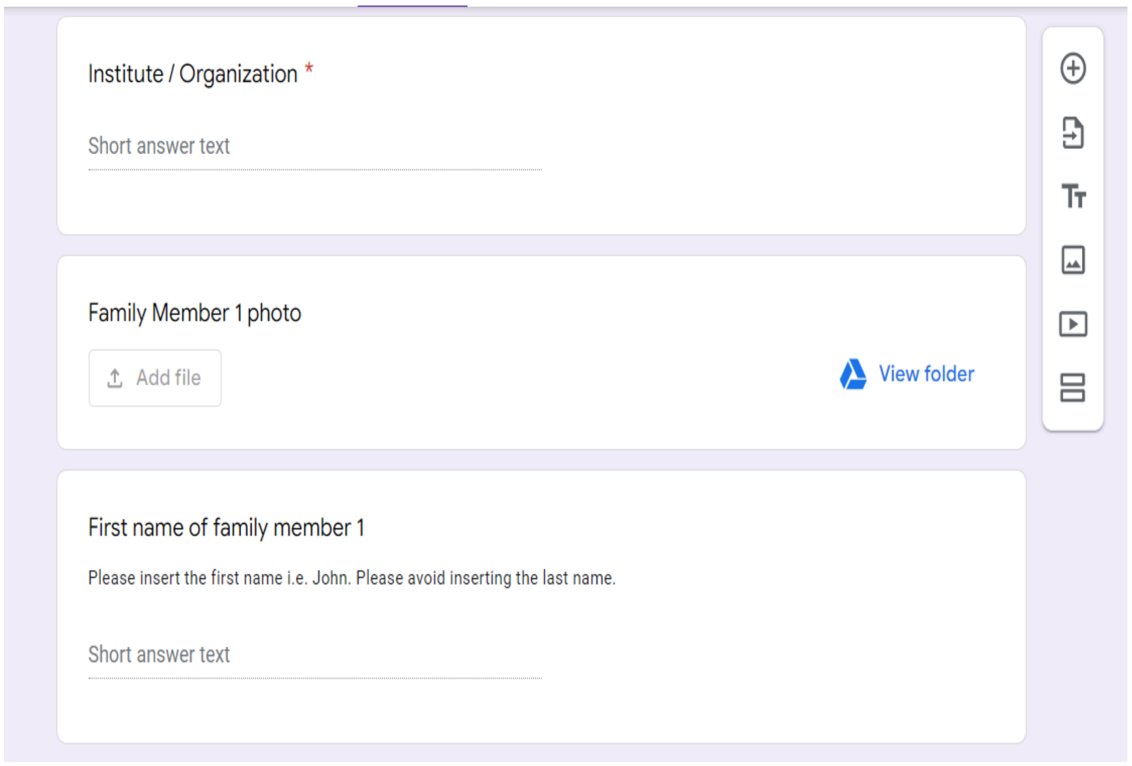
Short answer text

Occupation *

Occupation of the person i.e. Student, Professor, Businessman, Athlete etc.

- Doctor
- Engineer
- Professor
- Athlete
- Businessman
- Other...

Figure B.2: Google Form information fields snapshot.


Institute / Organization *

Short answer text

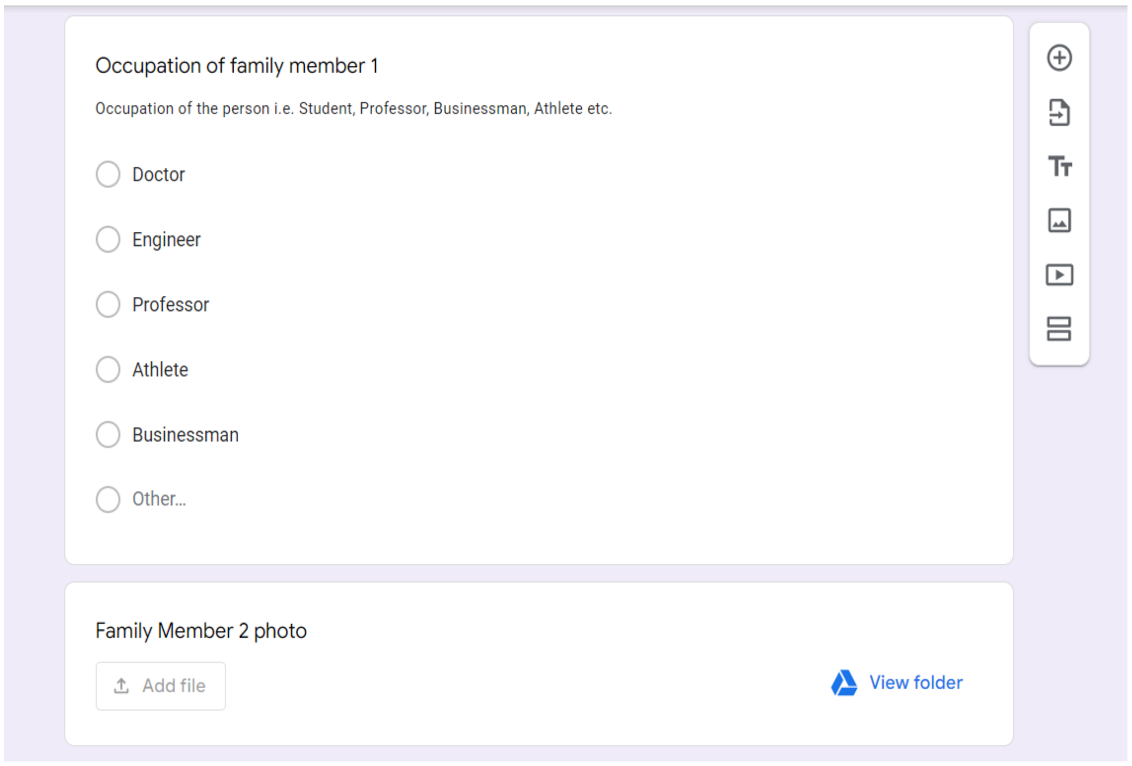
Family Member 1 photo

First name of family member 1

Please insert the first name i.e. John. Please avoid inserting the last name.

Short answer text

Figure B.3: Google Form information fields snapshot.

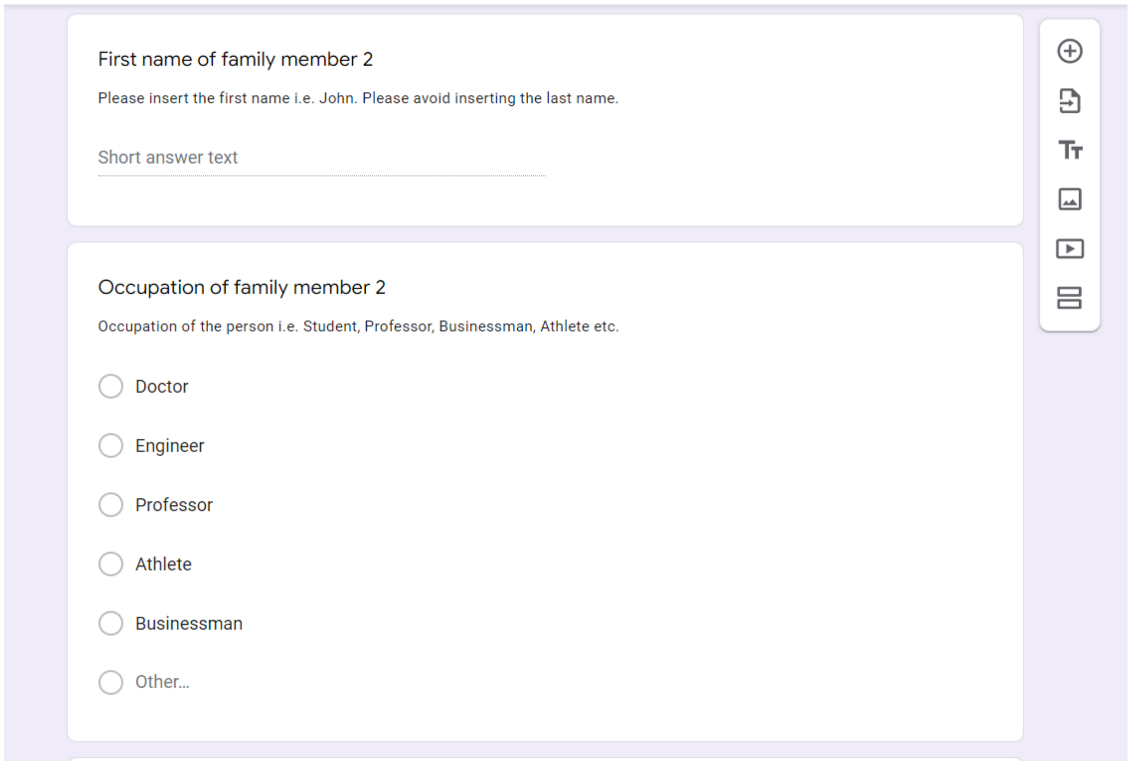

Occupation of family member 1

Occupation of the person i.e. Student, Professor, Businessman, Athlete etc.

- Doctor
- Engineer
- Professor
- Athlete
- Businessman
- Other...

Family Member 2 photo

Figure B.4: Google Form information fields snapshot.


First name of family member 2

Please insert the first name i.e. John. Please avoid inserting the last name.

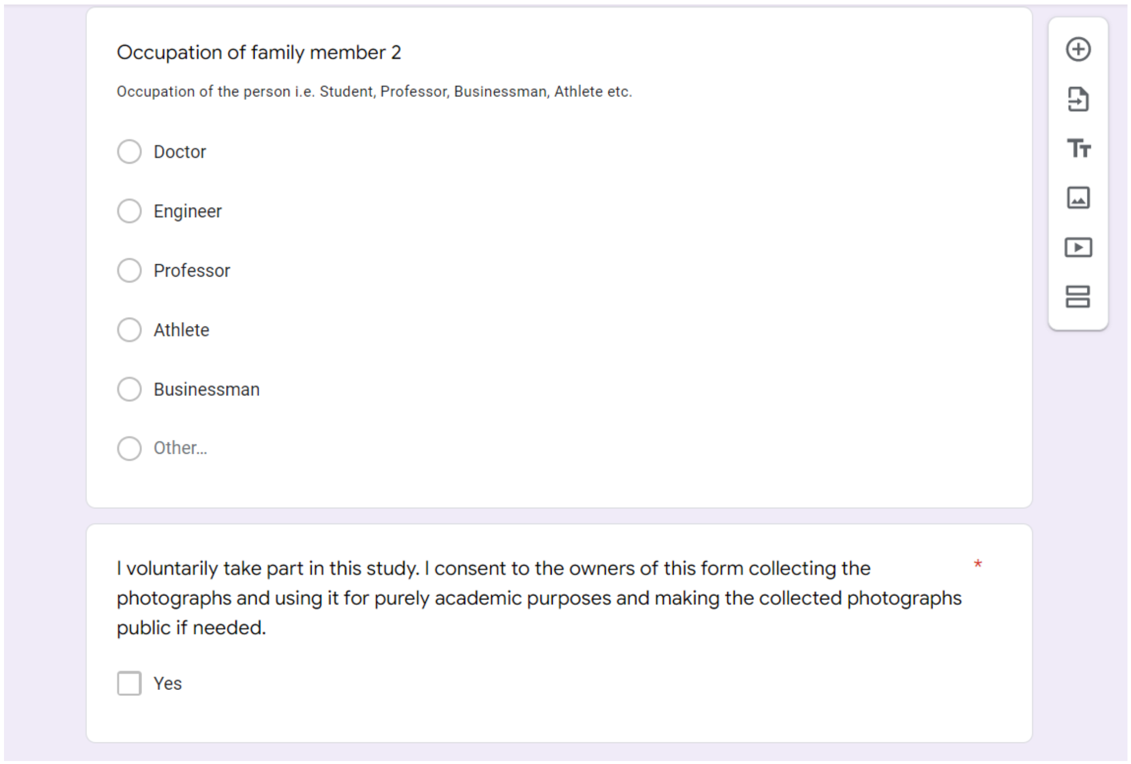
Short answer text

Occupation of family member 2

Occupation of the person i.e. Student, Professor, Businessman, Athlete etc.

- ◯ Doctor
- ◯ Engineer
- ◯ Professor
- ◯ Athlete
- ◯ Businessman
- ◯ Other...

Figure B.5: Google Form information fields snapshot.


Occupation of family member 2

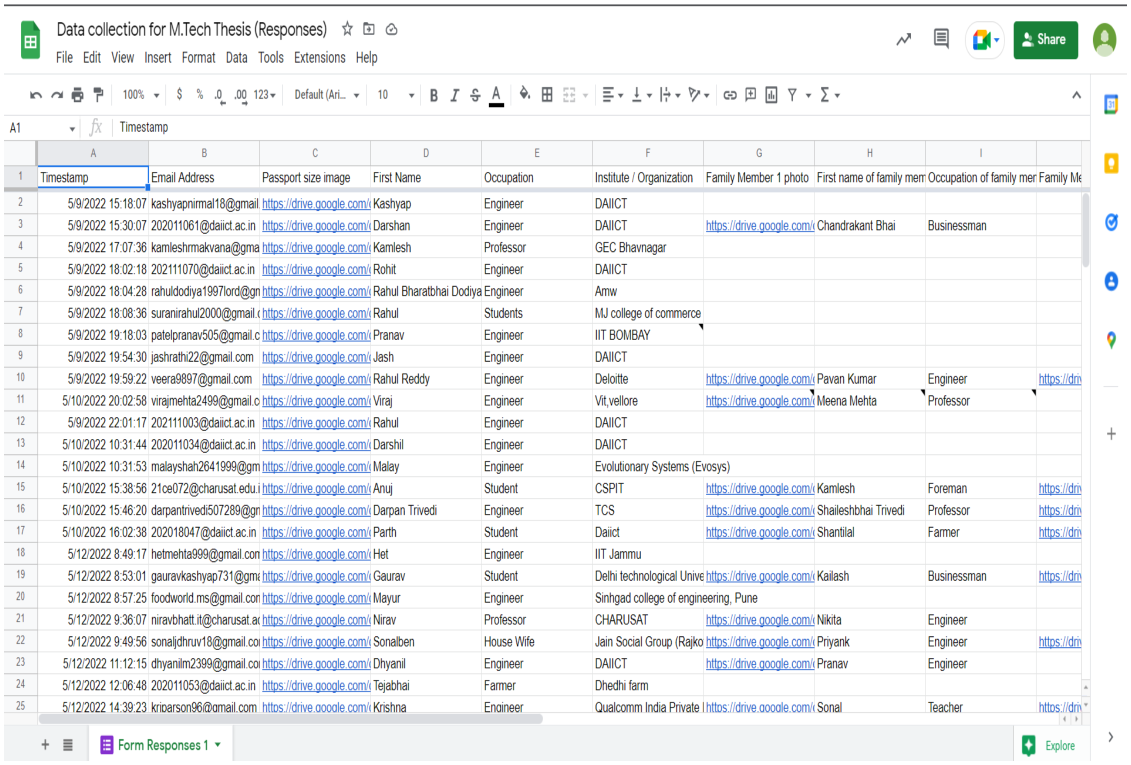
Occupation of the person i.e. Student, Professor, Businessman, Athlete etc.

- ◯ Doctor
- ◯ Engineer
- ◯ Professor
- ◯ Athlete
- ◯ Businessman
- ◯ Other...

I voluntarily take part in this study. I consent to the owners of this form collecting the photographs and using it for purely academic purposes and making the collected photographs public if needed. *

☐ Yes

Figure B.6: Google Form snapshot containing the Consent message.



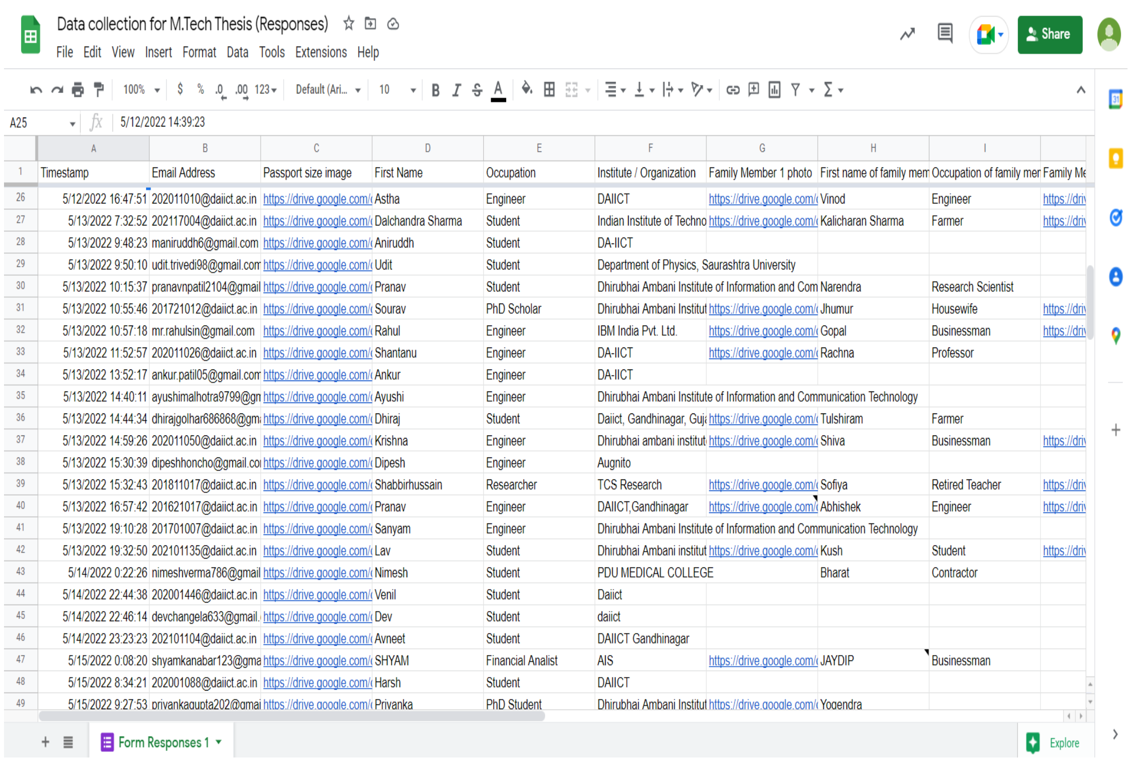
Data collection for M.Tech Thesis (Responses)

A1 Timestamp

	A	B	C	D	E	F	G	H	I	
1	Timestamp	Email Address	Passport size image	First Name	Occupation	Institute / Organization	Family Member 1 photo	First name of family mem	Occupation of family mer	Family Me
2	5/9/2022 15:18:07	kashyapnirmal18@gmail.	https://drive.google.com/	Kashyap	Engineer	DAIICT				
3	5/9/2022 15:30:07	202011061@daiict.ac.in	https://drive.google.com/	Darshan	Engineer	DAIICT	https://drive.google.com/	Chandrakant Bhai	Businessman	
4	5/9/2022 17:07:36	kamleshrmakvana@gma	https://drive.google.com/	Kamlesh	Professor	GEC Bhavnagar				
5	5/9/2022 18:02:18	202111070@daiict.ac.in	https://drive.google.com/	Rohit	Engineer	DAIICT				
6	5/9/2022 18:04:28	rahuldodiya1997lord@gn	https://drive.google.com/	Rahul Bharatbhai Dodiya	Engineer	Amw				
7	5/9/2022 18:08:36	suranirahul2000@gmail.c	https://drive.google.com/	Rahul	Students	MJ college of commerce				
8	5/9/2022 19:18:03	patelpranav505@gmail.c	https://drive.google.com/	Pranav	Engineer	IIT BOMBAY				
9	5/9/2022 19:54:30	jashrathi22@gmail.com	https://drive.google.com/	Jash	Engineer	DAIICT				
10	5/9/2022 19:59:22	veera9897@gmail.com	https://drive.google.com/	Rahul Reddy	Engineer	Deloitte	https://drive.google.com/	Pavan Kumar	Engineer	https://driv
11	5/10/2022 20:02:58	virajmehta2499@gmail.c	https://drive.google.com/	Viraj	Engineer	Vit,vellore	https://drive.google.com/	Meena Mehta	Professor	
12	5/9/2022 22:01:17	202111003@daiict.ac.in	https://drive.google.com/	Rahul	Engineer	DAIICT				
13	5/10/2022 10:31:44	202011034@daiict.ac.in	https://drive.google.com/	Darshil	Engineer	DAIICT				
14	5/10/2022 10:31:53	malayshah2641999@gm	https://drive.google.com/	Malay	Engineer	Evolutionary Systems (Evosys)				
15	5/10/2022 15:38:56	21ce072@charusat.edu.i	https://drive.google.com/	Anuj	Student	CSPIT	https://drive.google.com/	Kamlesh	Foreman	https://driv
16	5/10/2022 15:46:20	darpantrivedi507289@gn	https://drive.google.com/	Darpan Trivedi	Engineer	TCS	https://drive.google.com/	Shaileshbhai Trivedi	Professor	https://driv
17	5/10/2022 16:02:38	202018047@daiict.ac.in	https://drive.google.com/	Parth	Student	Daiict	https://drive.google.com/	Shantilal	Farmer	https://driv
18	5/12/2022 8:49:17	hetmehta999@gmail.con	https://drive.google.com/	Het	Engineer	IIT Jammu				
19	5/12/2022 8:53:01	gauravkashyap731@gma	https://drive.google.com/	Gaurav	Student	Delhi technological Unive	https://drive.google.com/	Kailash	Businessman	https://driv
20	5/12/2022 8:57:25	foodworld.ms@gmail.con	https://drive.google.com/	Mayur	Engineer	Sinhgad college of engineering, Pune				
21	5/12/2022 9:36:07	niravbhatt.it@charusat.ac	https://drive.google.com/	Nirav	Professor	CHARUSAT	https://drive.google.com/	Nikita	Engineer	
22	5/12/2022 9:49:56	sonaljdhruv18@gmail.co	https://drive.google.com/	Sonalben	House Wife	Jain Social Group (Rajko	https://drive.google.com/	Priyank	Engineer	https://driv
23	5/12/2022 11:12:15	dhyanilm2399@gmail.co	https://drive.google.com/	Dhyanil	Engineer	DAIICT	https://drive.google.com/	Pranav	Engineer	
24	5/12/2022 12:06:48	202011053@daiict.ac.in	https://drive.google.com/	Tejabhai	Farmer	Dhedhi farm				
25	5/12/2022 14:39:23	kriparson96@gmail.com	https://drive.google.com/	Krishna	Engineer	Qualcomm India Private I	https://drive.google.com/	Sonal	Teacher	https://driv

Form Responses 1

Figure B.7: Google Form Response snapshot with the response entries from 1-24



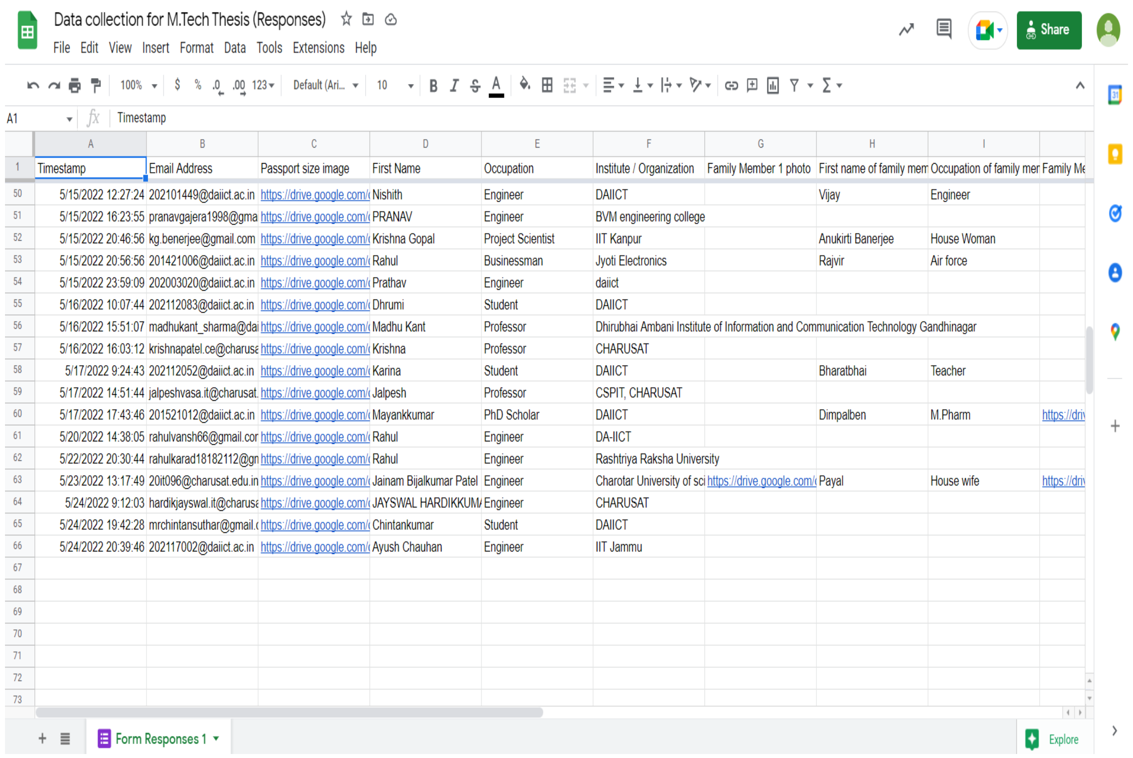
Data collection for M.Tech Thesis (Responses)

A25 5/12/2022 14:39:23

	A	B	C	D	E	F	G	H	I	
1	Timestamp	Email Address	Passport size image	First Name	Occupation	Institute / Organization	Family Member 1 photo	First name of family mem	Occupation of family mer	Family Me
26	5/12/2022 16:47:51	202011010@daiict.ac.in	https://drive.google.com/	Astha	Engineer	DAIICT	https://drive.google.com/	Vinod	Engineer	https://driv
27	5/13/2022 7:32:52	202117004@daiict.ac.in	https://drive.google.com/	Dalchandra Sharma	Student	Indian Institute of Techno	https://drive.google.com/	Kalicharan Sharma	Farmer	https://driv
28	5/13/2022 9:48:23	maniruddh6@gmail.com	https://drive.google.com/	Aniruddh	Student	DA-IICT				
29	5/13/2022 9:50:10	udit.trivedi98@gmail.com	https://drive.google.com/	Udit	Student	Department of Physics, Saurashtra University				
30	5/13/2022 10:15:37	pranavnpatil2104@gmail	https://drive.google.com/	Pranav	Student	Dhirubhai Ambani Institute of Information and Com		Narendra	Research Scientist	
31	5/13/2022 10:55:46	201721012@daiict.ac.in	https://drive.google.com/	Sourav	PhD Scholar	Dhirubhai Ambani Institut	https://drive.google.com/	Jhumur	Housewife	https://driv
32	5/13/2022 10:57:18	mr.rahulsin@gmail.com	https://drive.google.com/	Rahul	Engineer	IBM India Pvt. Ltd.	https://drive.google.com/	Gopal	Businessman	https://driv
33	5/13/2022 11:52:57	202011026@daiict.ac.in	https://drive.google.com/	Shantanu	Engineer	DA-IICT	https://drive.google.com/	Rachna	Professor	
34	5/13/2022 13:52:17	ankur.patil05@gmail.com	https://drive.google.com/	Ankur	Engineer	DA-IICT				
35	5/13/2022 14:40:11	ayushimalhotra9799@gn	https://drive.google.com/	Ayushi	Engineer	Dhirubhai Ambani Institute of Information and Communication Technology				
36	5/13/2022 14:44:34	dhirajgolhar686868@gma	https://drive.google.com/	Dhiraj	Student	Daiict, Gandhinagar, Guja	https://drive.google.com/	Tulshiram	Farmer	
37	5/13/2022 14:59:26	202011050@daiict.ac.in	https://drive.google.com/	Krishna	Engineer	Dhirubhai ambani institut	https://drive.google.com/	Shiva	Businessman	https://driv
38	5/13/2022 15:30:39	dipeshhoncho@gmail.co	https://drive.google.com/	Dipesh	Engineer	Augnito				
39	5/13/2022 15:32:43	201811017@daiict.ac.in	https://drive.google.com/	Shabbirhussain	Researcher	TCS Research	https://drive.google.com/	Sofiya	Retired Teacher	https://driv
40	5/13/2022 16:57:42	201621017@daiict.ac.in	https://drive.google.com/	Pranav	Engineer	DAIICT,Gandhinagar	https://drive.google.com/	Abhishek	Engineer	https://driv
41	5/13/2022 19:10:28	201701007@daiict.ac.in	https://drive.google.com/	Sanyam	Engineer	Dhirubhai Ambani Institute of Information and Communication Technology				
42	5/13/2022 19:32:50	202101135@daiict.ac.in	https://drive.google.com/	Lav	Student	Dhirubhai Ambani institut	https://drive.google.com/	Kush	Student	https://driv
43	5/14/2022 0:22:26	nimeshverma786@gmail	https://drive.google.com/	Nimesh	Student	PDU MEDICAL COLLEGE		Bharat	Contractor	
44	5/14/2022 22:44:38	202001446@daiict.ac.in	https://drive.google.com/	Venil	Student	Daiict				
45	5/14/2022 22:46:14	devchangela633@gmail.	https://drive.google.com/	Dev	Student	daiict				
46	5/14/2022 23:23:23	202101104@daiict.ac.in	https://drive.google.com/	Avneet	Student	DAIICT Gandhinagar				
47	5/15/2022 0:08:20	shyamkanabar123@gma	https://drive.google.com/	SHYAM	Financial Analist	AIS	https://drive.google.com/	JAYDIP	Businessman	
48	5/15/2022 8:34:21	202001088@daiict.ac.in	https://drive.google.com/	Harsh	Student	DAIICT				
49	5/15/2022 9:27:53	priyankagupta202@gmai	https://drive.google.com/	Priyanka	PhD Student	Dhirubhai Ambani Institut	https://drive.google.com/	Yogendra		

Form Responses 1

Figure B.8: Google Form Response snapshot with the response entries from 25-48.



Data collection for M.Tech Thesis (Responses)

A1 Timestamp

	A	B	C	D	E	F	G	H	I	
1	Timestamp	Email Address	Passport size image	First Name	Occupation	Institute / Organization	Family Member 1 photo	First name of family mem	Occupation of family mer	Family Me
50	5/15/2022 12:27:24	202101449@daiict.ac.in	https://drive.google.com/(	Nishith	Engineer	DAIICT		Vijay	Engineer	
51	5/15/2022 16:23:55	pranavgajera1998@gma	https://drive.google.com/(	PRANAV	Engineer	BVM engineering college				
52	5/15/2022 20:46:56	kg.benerjee@gmail.com	https://drive.google.com/(	Krishna Gopal	Project Scientist	IIT Kanpur		Anukirti Banerjee	House Woman	
53	5/15/2022 20:56:56	201421006@daiict.ac.in	https://drive.google.com/(	Rahul	Businessman	Jyoti Electronics		Rajvir	Air force	
54	5/15/2022 23:59:09	202003020@daiict.ac.in	https://drive.google.com/(	Prathav	Engineer	daiict				
55	5/16/2022 10:07:44	202112083@daiict.ac.in	https://drive.google.com/(	Dhrumi	Student	DAIICT				
56	5/16/2022 15:51:07	madhukant_sharma@dai	https://drive.google.com/(	Madhu Kant	Professor	Dhirubhai Ambani Institute of Information and Communication Technology Gandhinagar				
57	5/16/2022 16:03:12	krishnapatel.ce@charusa	https://drive.google.com/(	Krishna	Professor	CHARUSAT				
58	5/17/2022 9:24:43	202112052@daiict.ac.in	https://drive.google.com/(	Karina	Student	DAIICT		Bharatbhai	Teacher	
59	5/17/2022 14:51:44	jalpeshvasa.it@charusat.	https://drive.google.com/(	Jalpesh	Professor	CSPIT, CHARUSAT				
60	5/17/2022 17:43:46	201521012@daiict.ac.in	https://drive.google.com/(	Mayankkumar	PhD Scholar	DAIICT		Dimpalben	M.Pharm	https://driv
61	5/20/2022 14:38:05	rahulvansh66@gmail.cor	https://drive.google.com/(	Rahul	Engineer	DA-IICT				
62	5/22/2022 20:30:44	rahulkarad18182112@gn	https://drive.google.com/(	Rahul	Engineer	Rashtriya Raksha University				
63	5/23/2022 13:17:49	20it096@charusat.edu.in	https://drive.google.com/(	Jainam Bijalkumar Patel	Engineer	Charotar University of sci	https://drive.google.com/(	Payal	House wife	https://driv
64	5/24/2022 9:12:03	hardikjayswal.it@charusa	https://drive.google.com/(	JAYSWAL HARDIKKUM/	Engineer	CHARUSAT				
65	5/24/2022 19:42:28	mrchintansuthar@gmail.(	https://drive.google.com/(	Chintankumar	Student	DAIICT				
66	5/24/2022 20:39:46	202117002@daiict.ac.in	https://drive.google.com/(	Ayush Chauhan	Engineer	IIT Jammu				
67										
68										
69										
70										
71										
72										
73										

Figure B.9: Google Form Response snapshot with the response entries from 49-65.

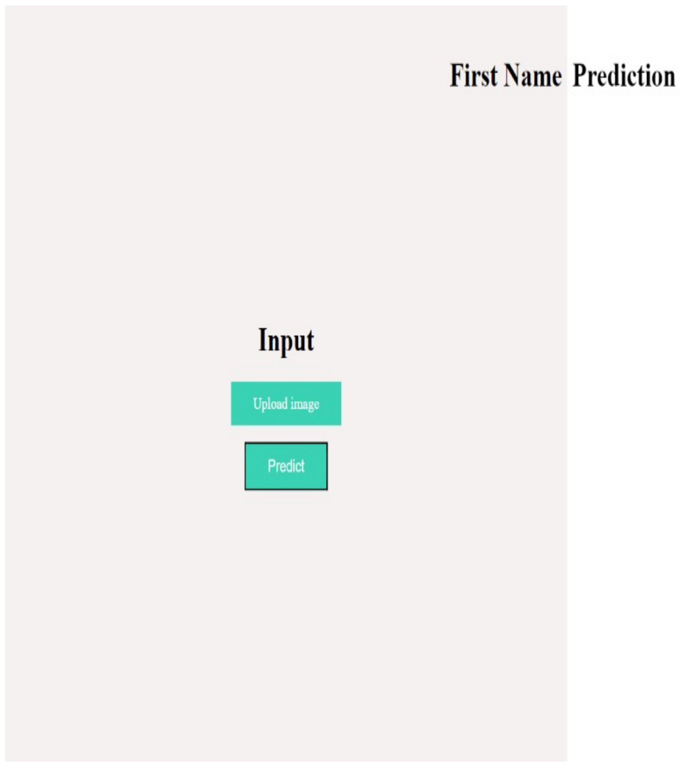


Figure B.10: A flask demo of the Name Association using name prediction.

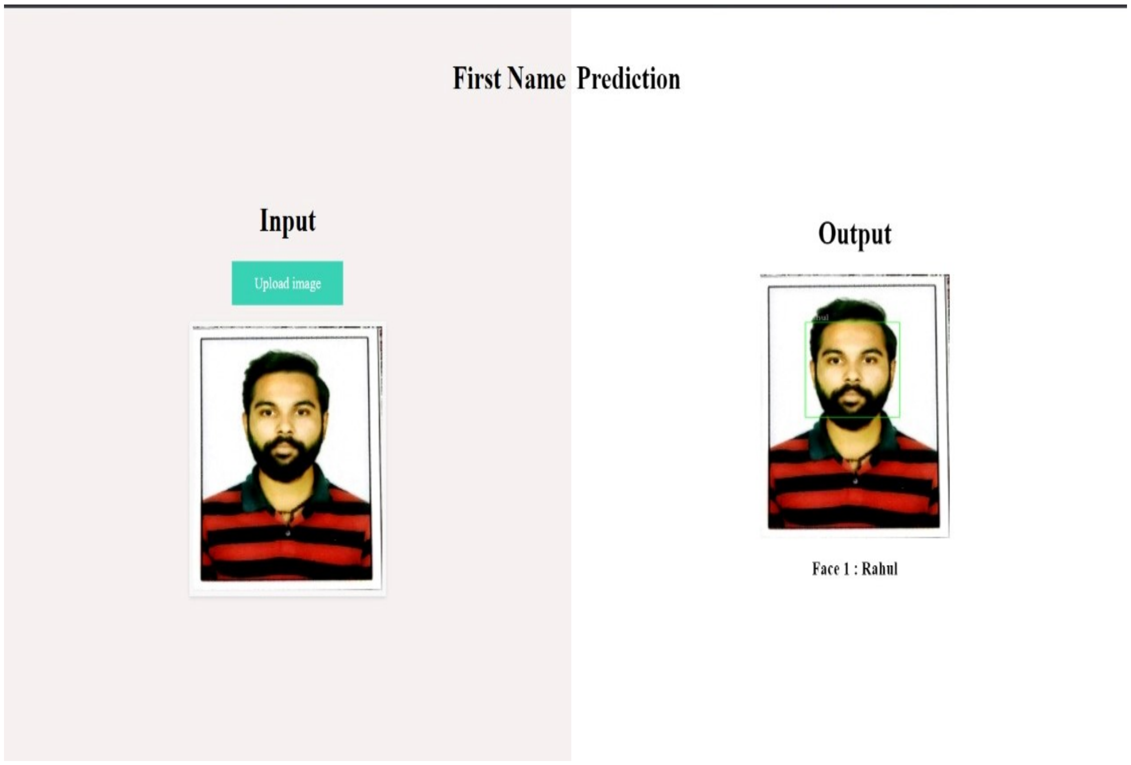


Figure B.11: Demonstration Output 1
Demonstration of the system using the Rahul's input image which is correctly classified as Rahul. And we can see the bounding box which associated the predicted name Rahul to the face.

Figure B.12: QR Code for Github repository.
URL : https://github.com/Kashyap-Nirmal/Face_Name_Prediction