Department of Electrical and Computer Engineering
North South University

Senior Design Project

P.A.G.E

Person Detection and Gender, Age and Ethnicity Classification System

Zahin Akram **ID # 1610618042**

Arifuzzaman Arman **ID # 1610551042**

Mohammad Rakib Imtiaz **ID # 1610294042**

Faculty Advisor

Mr. Syed Athar Bin Amir

Lecturer

ECE Department

Spring, 2020

DECLARATION

This is to certify that this Project is our original work. No part of this work has been submitted elsewhere partially or fully for the award of any other degree or diploma. Any material reproduced in this project has been properly acknowledged.

Students' name & Signature

1. **Zahin Akram**

2. **Arifuzzaman Arman**

3. **Mohammad Rakib Imtiaz**

APPROVAL

The capstone project entitled "**P.A.G.E. Person Detection and Gender, Age and Ethnicity Classification System**" by **Mohammad Rakib Imtiaz(ID:1610294042), Arifuzzaman Arman(ID:1610551042) and Zahin Akram(ID:1610618042)** is approved in partial fulfillment of the requirement of the Degree of Bachelor of Science in Computer Science and Engineering on Spring 2020 and has been accepted as satisfactory.

Supervisor's Signature

Syed Athar Bin Amir
Lecturer
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh.

Department Chair's Signature

Dr. Mohammad Rezaul Bari
Associate Professor & Chair
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh.

ACKNOWLEDGMENT

First of all, we wish to express our gratitude to the Almighty for giving us the strength to perform our responsibilities and complete the report.

The capstone project program is very helpful to bridge the gap between theoretical knowledge and real-life experience as part of the Bachelor of Science (BSc) program. This report has been designed to have practical experience through theoretical understanding.

We also acknowledge our profound sense of gratitude to all the teachers who have been instrumental in providing us the technical knowledge and moral support to complete the project with full understanding.

It is imperative to show our appreciation for our honorable faculty member Syed Athar Bin Amir for his undivided attention and help to achieve this milestone. Also, our gratefulness is divine to the North South University, ECE department for providing us a course such as CSE 499 in which we could really work on this project and materialize it the way we have dreamt of.

We thank our friends and family for their moral support to carve out this project and always offer their support.

Abstract

In this report, we present a software that is capable of identifying humans from images along with the ability to classify Age, Race, and Gender of a person up to a limited capacity. This is achieved by training two independent models through machine learning. The first model was trained to identify people from an image and classifies any object as a person from a variety of angles using a range of features as markers. The second model is capable of detecting and extracting features from any possible faces of people within an image. Through the extraction of facial features, the model then predicts the age, race, and gender. The entire system, through the two models, then presents images as output with identified people encapsulated in boxes and their predicted attributes displayed under the face. The results are also presented in a statistical format for easier and quicker understanding and evaluation.

Table of Contents

Chapter 1
Project Overview

1.1 Introduction

In this project we are performing two fundamental tasks. The first of which is object detection in the form of person detection and the classification of a person's age, race and gender. Object detection, or sometimes referred to as object recognition, is the culmination of image classification and object localization. Image classification usually contains a specific object in clear detail and attempts to predict the name or label of the image. Object classification broadly refers to locate an object in an image and surround it with a box. Our work fundamentally represents these two ideas. Within our object detection not only do we classify objects within the image to our target, we also mark them by a box even when they are not the primary or the only subject of the image. Where image classification requires the label object to be the subject through object detection not only do we surround them in a bounding box we also detect the target from a variety of angles. Our work is capable of detection from many perspectives including many poses and despite some level of obstruction. To do this we used the model from the R-CNN (Region-Based Convolution Neural Network)**[4]** family, particularly the FRCNN model, including a varied dataset that stretches from VOC)**[3]** and COCO)**[6]**dataset for person detection and UTK)**[10]** to FairFace**[5]**for feature classification.

This project can collect and provide statistical data to a wide variety of enterprises. It can provide the number of visitors in a restaurant, the age, race and gender of customers at the counter of a shop to the number of people crossing the streets. These statistical data can help improve business and provide data for future events.

1.2 Project Description

In our project, we present the ability to the user input either images or a video through which to parse. The input can consist of multiple images of varying angles and perspectives. In the event of a video, the system will look through the video and perform its functionality at specific intervals and output data accordingly. The output will consist of both images and data. In the images, the output will be presented via the marking of the subject using a bounding box. Wherever possible, the system will also predict the Age, Race, and Gender of a person provided that the face is visible and there is some information to be extracted. The other form of data will be presented in the form of a file that consists of the number of people within the pictures and their data such as the number of males or females, the number of people in age category and race.

1.3 Background and Motivation

While such products already exist they are rarely for the collection of statistical data. Often all person detection is usually focused on the face or the body of a human separately. However, in a real-world environment often those robust but specific detection models fail in providing a high standard. In our work, we explore the need for both cost-effectively and efficiently. Often large and highly accurate systems sacrifice a lot of resources and time to perform detection. Ours presents a highly optimum and efficient solution. Even though our project is not the most accurate or the most efficient it certainly meets the standard of both. It is not too expensive or time-consuming yet it manages to keep a high moderation of quality.

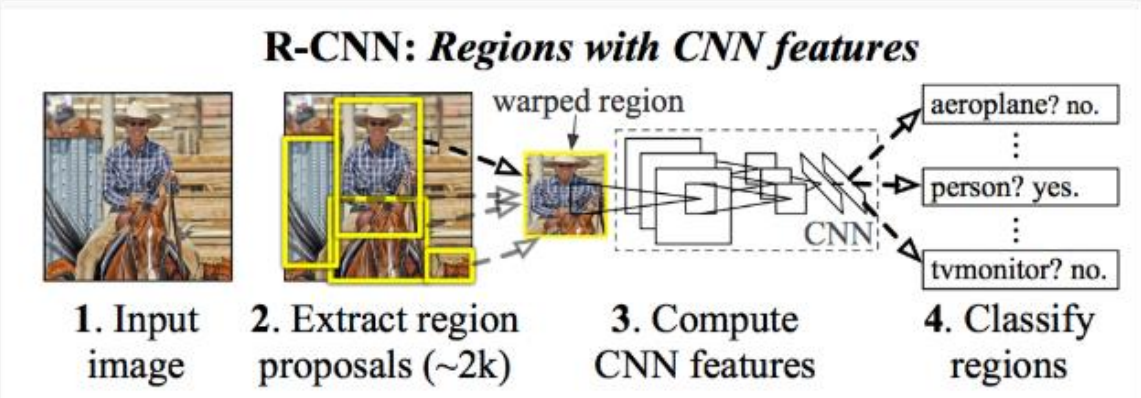
The motivation behind our project is to be able to provide a tool that serves everyone from small business owners to large corporations owning chain stores to the government to collect and tabulate data. Data analysis plays a vital role in today's world. It can show a store the customers it is attracting and the core base of its revenue. However, for data analysis to happen, there must be data to be processed. This collection of data is made simple by our system. It can provide data about the real world in real-time leading to a better analysis of the world.

Chapter 2
Related Works

2.1 Introduction

Object detection, particularly people, have been of particular interest in Computer vision for a long time. However, due to the sheer variety of human motion, their appearance, and different level of obstruction and blockage have rendered this task extremely difficult [1]. One of the many approaches is the use of a Histogram of Oriented Gradient (HOG) [9]. It is imperative to note here this algorithm has gone through many different changes since its original conception. A brief discussion regarding this algorithm will take place further along.

Another new and important development is the use of Region-Based Convolutional Neural Network (RCNN) which acts as a traditional Convolutional Neural Network except for images. It extracts information from the bounded region and uses a classifier to label the region into one of its known classes. Again, further discussion will occur in more relevant places.



Summary of the R-CNN Model ArchitectureTaken from Rich feature hierarchies for accurate object detection and semantic segmentation.

2.2 Existing Work

Representations of person detection are quite variable and sometimes easily found. One such example can be the face filter often used in camera apps on phone. Even in many editing software used for making animations or videos the background blurring system has to isolate the person and place it onto the foreground. However, such systems suffer from its limit to scope. The face filter software almost always is entirely dependent on the face of its user. Even slight alterations such as variance in angles can easily render them obsolete. They are also limited by the angle of the subject. If the subject or person is facing sideways or any other way other than straight to the camera the system is apt to malfunction. Similarly, editing software capable of distinguishing a person onto the foreground will be limited if the subject is slightly obscured. They are also limited by perspective or angle as mentioned above. Our work is different because it is capable of dealing with a certain level of obstruction and verifies a person's body and face independently.

The software used for self-driving cars, particularly to its ability in detecting pedestrians, is a suitable comparison for our system. Such a system must be capable of detecting a person from multiple angles and be able to differentiate depth to make a suitable judgment. Our system is similarly capable of detecting a person regardless of angle and overlapping to a degree. Our system can also infer classification for upto a limited supply of input data.

Chapter 3
Technical Design

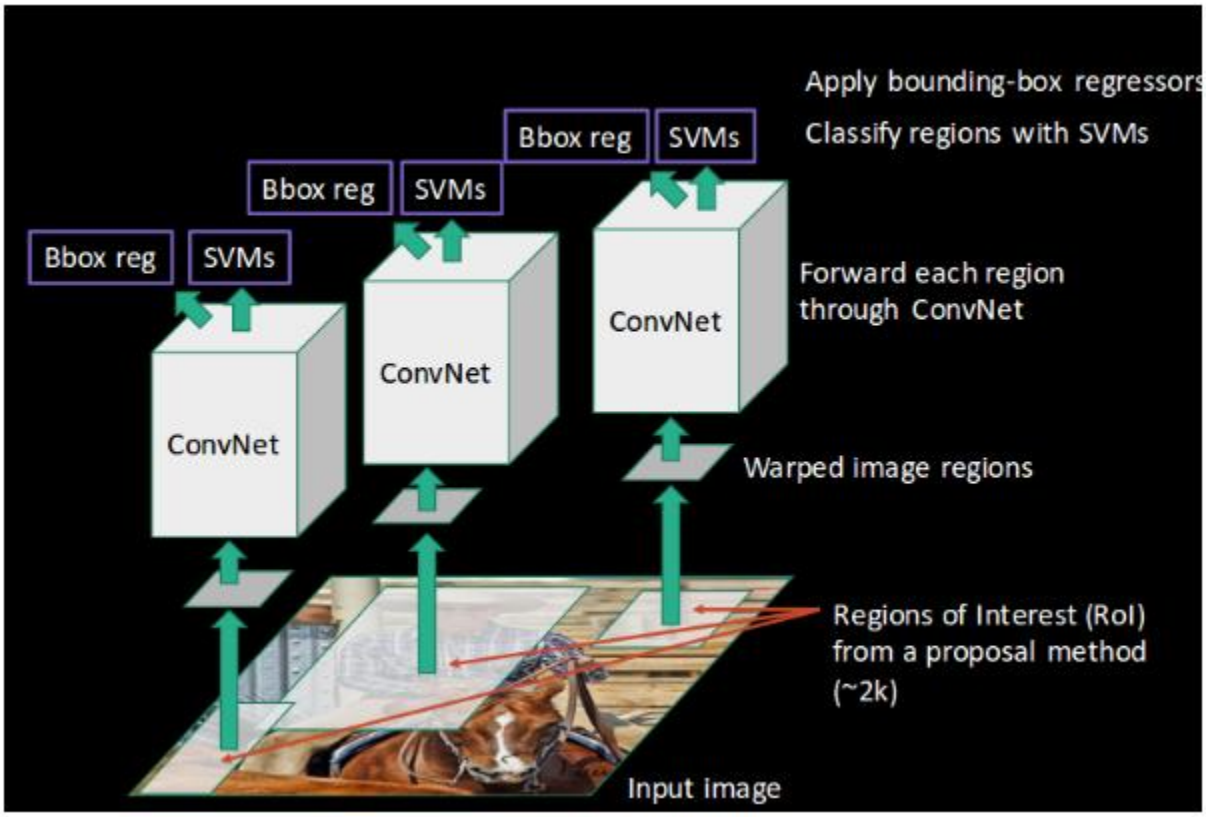
3.1 Introduction

In this chapter we discuss the techniques and methods that are in overall usage for object detection and the methods that are of priority in this project.

3.2 Region-Based Convolutional Neural Network (RCNN)

A standard Convolutional Neural Network (CNN) an image is passed to the network. The image is then divided into regions and passed through a filter. The filter is started randomly and is continuously updated as training goes on. We therefore take an image and pass it through a trained CNN to see if the regions in the input, when passed through the filters, matches with any of output known by the CNN. The problem is the lack of scope and difference this allows. A traditional CNN has difficulty in understanding where the region of an object begins or ends. Any obscurity of the target object, slight differences in perspectives, or alterations to shape can throw off a CNN because of its limitations. RCNN deals with this problem by first segmentations on the image. These segmentations are looked over and combined based on a variety of factors. These factors can range anywhere from color, texture similarity, size matching and similarity of edges. Then, these specific regions are taken and passed through a CNN. Even if we identify the region of image we are interested in classification for training the segmentation technique tries its best to isolate only the part the object we are interested in. These segmentation attempts to rule out obscurity and since the target region is marked the mapped segments presents cohesive objects to be processed. These selections are referred to as Regions of Interest (ROI). Since ROI's can be of varying size and shape and a CNN expects inputs of specific size and shape, these ROI are reduced to a similar size in a process called ROI pooling. These feature maps are

passed through a CNN with a classifier at the end to determine whether the region was a target object. A regression model is used for bounding purposes.

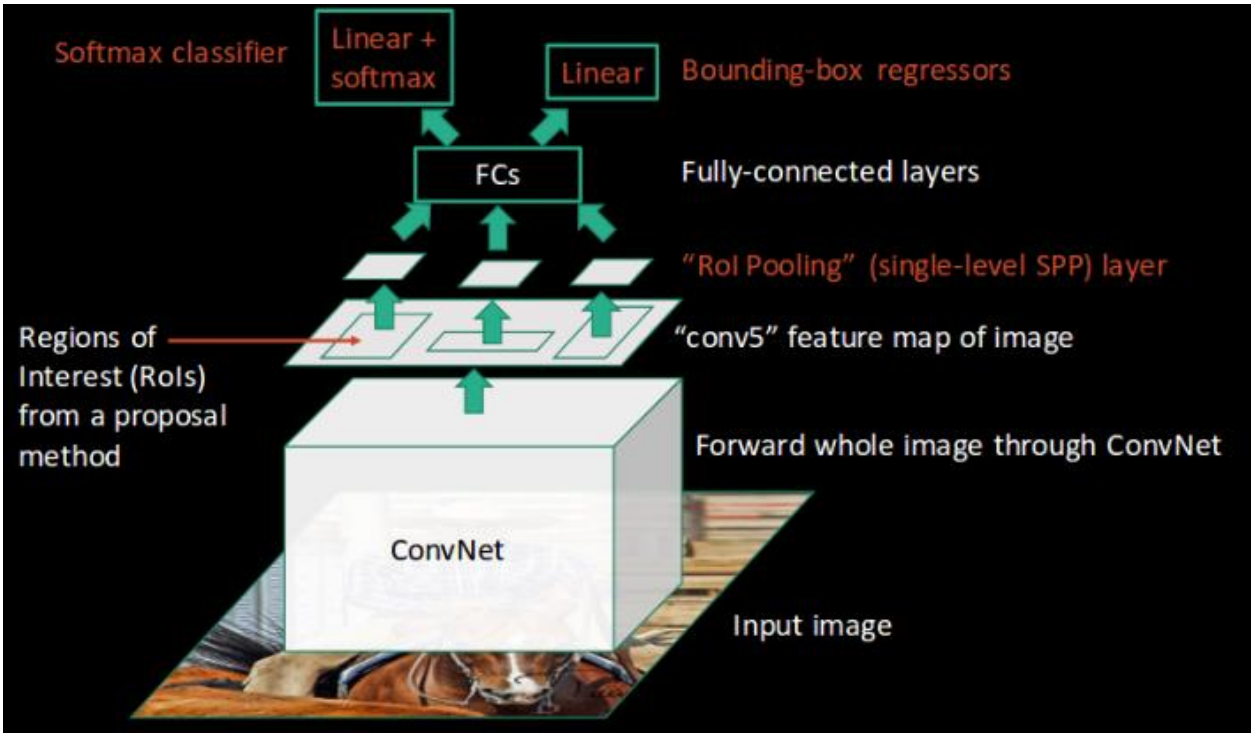


3.3 Issues of RCNN

RCNN, while effective, is very slow. Simply said the segmentations divide an image into many parts of individual objects. These segmentations undergo feature extraction for the CNN network. A single image contributes to many segmentations which slows down an already expensive CNN network.

3.4 Faster Region-Based Convolutional Neural Network (FRCNN)

FRCNN [8] tries to minimize the issue of time consumption and efficiency. It tries to build upon the RCNN by simple alterations. The primary issue is the large volume of segmentations that occur and the time it takes to process them. Instead of running segmentations that needs to be processed separately, FRCNN came up with a way to share the feature mapping of the segmentations across all images. This reduces the need to running each segmentation individually. Imagine using a general method of classification instead of having to run it every time.

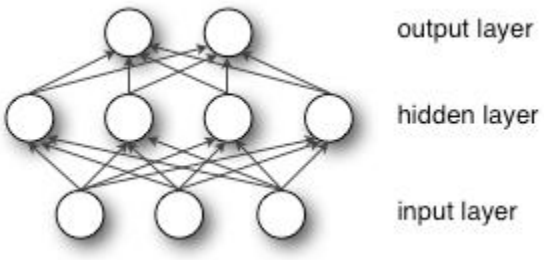


FRCNN works with first passing the image through a CNN to generate Regions of Interest (ROI). Since all the images are different and therefore ROI's are different ROI pooling is used to format them all to a size acceptable by CNN. Each of these regions is processed by a fully

connected layer. A softmax is applied upon the output of the Fully Connected layer for classification. A regression layer is used simultaneously to generate a bounding box.

3.5 Multilayer Perceptron Layer (MLP)

MLP is one of the fundamentals of Machine Learning. It tries to imitate the human cognition. We start by having a single neuron that is capable of assigning a value or a weight to a value and then determining an outcome of the variable with that weight. The neuron forms its own marker or condition or equation with that weight and substitutes the value with its actual value and using the weight comes to a solution. These weights are randomly started and via supervised training of many data determine the condition and the possible weight for which an answer can be determined. An MLP consists of many such neurons often it layers to deal with complex problems and to have many attributes it must watch out for.



A single layer MLP

Chapter 4
System Design

4.1 Introduction

In the previous section we discussed the theory of what we were using in this project. All discussion took place in simple terms without diving too deep into details. In this section we will take a look how each of these applies to this project directly and how they are utilized.

4.2 Components of Project

This project can be divided into a couple of core components. The first of which is the person detection. And the second is the classification of Age, Race and Gender based on facial features. These components will be further explained upon in the next subsections

4.2. A Components of Project: Person Detection

We have previously discussed FRCNN [8]and its architecture. This is the structure used for Person Detection. It consists of the standard VGG16 model with other layers embedded on top that are designed for the purposes of object classification and the drawing of bounding boxes as discussed previously. This FRCNN model was trained upon the COCO Dataset [6]. However, the training was not limited to the "Person" class only. Experimentation showed that having a few more classes allowed mitigation of wrongful identification of objects that were prone to appearing in images of people. Objects such as cars, bikes, lamps and food among others appeared in images consisting of persons and would sometimes be wrongly classified of people.

4.2. B Components of Project: Gender, Age and Ethnicity classification

The architecture for this part of the project is the MLP classifier. This classifier has been discussed above and is simplistic in nature. However, it does not achieve classification on its own. The MLP classifier is trained upon the features of the face extracted by the Face Recognition Library of Python[12]. The Face Recognition Library detects the face and extracts the facial features of the face. The facial features consist of values in a variety of variables. These variables are presented in a table of numbers where the each column represents the variable, with each row having the values per image. This table of features is then fed into the MLP classifier for training. The reason such a simplistic model is capable of detecting Gender, Age and Ethnicity is that once these variables are presented it no longer needs to deal with an image. The MLP classifier also does not need to be concerned with background data or even noisy data. It classifies upon the categories based solely on the values of variables per image. These technique works simply because a face is rather distinguishable because of its characteristics like eyes, ears, nose, etcetera. However, unlike Person Detection, the classification portion is limited by the assumption that the face is clearly visible in the data and facing the camera.

4.3. Classification of Person

Here classification is solely whether an object detected is a Person or not. While we do have other objects that we classify they are not of interest and therefore not shown. Only detected Persons are shown through a red bounding box

4.4. Classification of Gender, Age and Ethnicity

The classification of Gender is limited to either "Male" or "Female". The classification of Age is not a number but a label that is used to describe the range of age the person can be interpreted to be based on their appearance. The labels are Child (Age: 0-2), Adolescent (Age: 3-9), Teen (Age: 10-19), Young Adult (Age: 20-29), Adult (Age: 30-39), Old Adult (Age: 40-59) and Senior (Age: 60+). The classification of Age was not decided upon what we think a person should be at their age. The decision of range was decided solely upon the data available. For example: people of and over the age of 18 should be adult but there was not enough data that separated them from those of Age 17 or lower. Similarly we lacked a large number of data that distinguished children from teenagers. This was particularly difficult as we needed a large number of data for ages across multiple races. And even across one race of people children and adults often look younger or older and different from each other. The decision of the Age Range was based solely on data available and what gave the most accurate predictions.

The classification of Ethnicity for this project was limited to: "White", "Black", "Indian", "Asian" and "Other". The classification of "Indian" covers the South East Asia or the Indian Subcontinent. The classification of "Asian" is for the people of East Asia. For people who don't fall under "White", "Black", "Indian", "Asian" we classify as "Other". Again, our choice was limited to the data at hand. We understand that the classification is very broad and offensive to those wrongfully identified. The intention is not to suggest what a person is or should be but rather to present a prediction of what is most likely based solely on the data at hand.

4.5. The Limitations of Classification

The models were trained upon FairFace Dataset [5] and the UTKFace Dataset [10]. As a small team with little resource it was impossible for us to gather enough data that could encompass the whole world and all of its people. We do not suggest what a person is or should be but rather present an estimation of where a person might fall within the data of our 70000 people only.

Chapter 5
Software Design

5.1 Introduction

In this section we will discuss how the individual components come together into a cohesive singular project. This section will essentially lay out the pipeline of the final project.

5.2 Pipeline

The Person Detection and classification Gender, Age and Ethnicity are done by independent models and separately. But, they do work together in a specific series to form an organized output.

The system starts with the detection of face and then the classification according to Gender, Age and Ethnicity. The classifications are added into the image right below where the faces were detected. To keep them inconspicuous the classifications in all categories appear in small white letters. The information is also stored in a csv file for further evaluation and presentation. Now over that image the Person Detection is run to form the bounding boxes that outline the body of a person.

5.3 Example

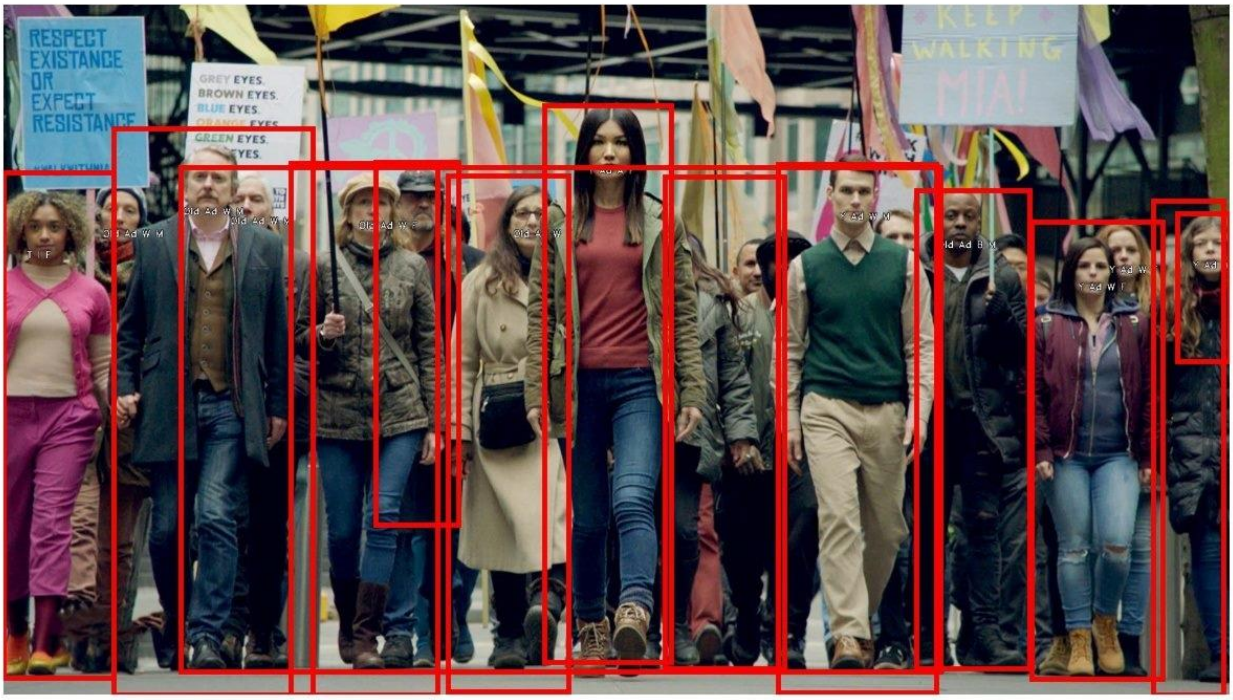
The following image is an example of the whole process:

We start with the raw image:

We run it through the Gender, Age and Ethnicity classification:

They may be small but the classifications appear in small letters and are kept track of:

Next the image is run through the Person Detection:



And thus we have a complete set of information about the people within the picture.

Chapter 6
Design Specifics

6.1 Introduction

In this section we will discuss the specifics of the system. We have already talked about the architecture briefly above but there were specifics within that architecture that needed to be modified to be complaint with our system.

6.2 Parameters

Something important that needs to be mentioned is the parameters of the work presented in this paper. They have already been mentioned above but to be precise: we wanted a system that is capable of performing under resource constraint and present accurate output efficiently. Accuracy and particularly efficiency played vital roles when it came to tweaking the model.

6.3 FRCNN Class Discussion

FRCNN has already been used for detection of objects. In fact it is built for that singular purpose. And, as the "Faster" part of "Faster Region-Based Convolutional Neural Network" would suggest it was also much faster than the standard models. However, the FRCNN model available in the official GitHub repository consisted of around 80 classes. While 80 class classifications meant that wrongful or false positives were fewer, it also increased the amount of time taken to process per image. This time taken increased significantly as the specs of the computer digressed. Our aim was to make it efficient and capable of running under simpler computers. Therefore the first step was the limitation of classes. But limiting the whole system to a mere 1 class gave way to false positives. This was because some objects like cars and food

appeared only in vicinity of people and therefore leads to wrong classifications. These led to the issue of accuracy vs. efficiency. It made the system faster but less accurate. We had to attempt a variety of combinations for classes for the one we found the most optimal. We settled on a total of 20 classes. These classes are:

'aeroplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus', 'car', 'cat', 'chair', 'cow', 'diningtable', 'dog', 'horse', 'motorbike', 'person', 'pottedplant', 'sheep', 'sofa', 'train', 'tvmonitor'

The decision for classes were made empirically and based solely on our dataset for training. All of these objects were selected because they appear in the same images as people do.

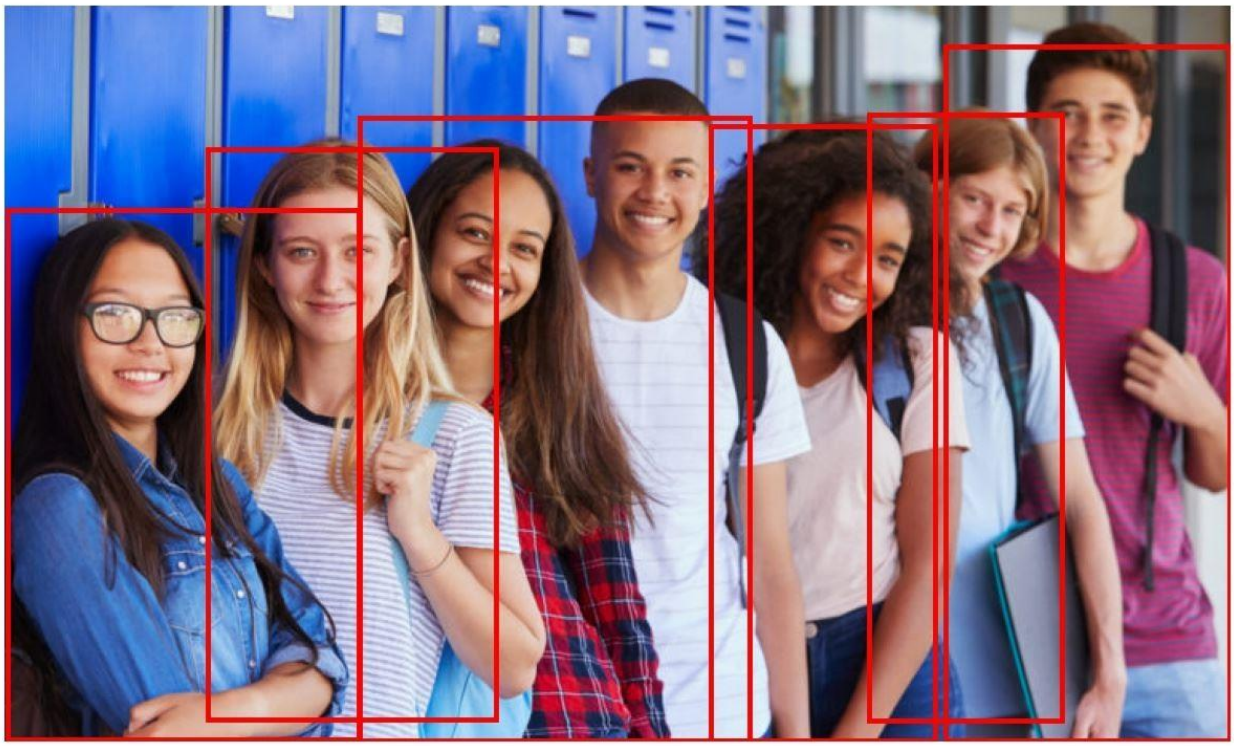
6.4 FRCNN Image Information Discussion

Accuracy is a difficult topic to cover because evaluating it seems highly situational. Detection depends on available information which is never consistent. Our system can detect people from all around a person given there is enough information, for example: the image being clear. Also another variable is height. Detection also depends on the angle at which the picture was taken. Orientation also presents another issue. Images with peoples bodies tilted cause the system to misidentify the query.

6.5 FRCNN Overlapping Discussion

This is an issue that has plagued person detection for as long as object detection has been in development. Objects detection within the depth of an image, which is in 2D, has proven difficult in most scenarios [9]. Often referred to the overlapping error, this occurs when 2 objects

of interest are behind one another. The object that is fully in view is detected and wrapped in a bounding box. The object behind the initial object does not represent enough information to be classified as a separate entity. An example can be seen here:



We can see that detection is quite accurate except when it comes to the 2 individuals in the middle; mainly the girl wearing the red fennel shirt and the boy wearing the white t-shirt. In this case there was a lack of information that clearly outlined the separation of individuals. Here it may look like the model is functioning accurately separating the other individuals but it has to be said that this error occurs on a case by case basis. Certain positioning, background, resolution of image, height of person among many others contributes to the decisions made by the model. This problem is addressed by the Max Suppression Algorithm which looks over the overlapping of the bounding boxes. Sometimes data within the first bounding box is ignored when considering the next box. Other times they are still considered. In this scenario the girl with the red shirt did not

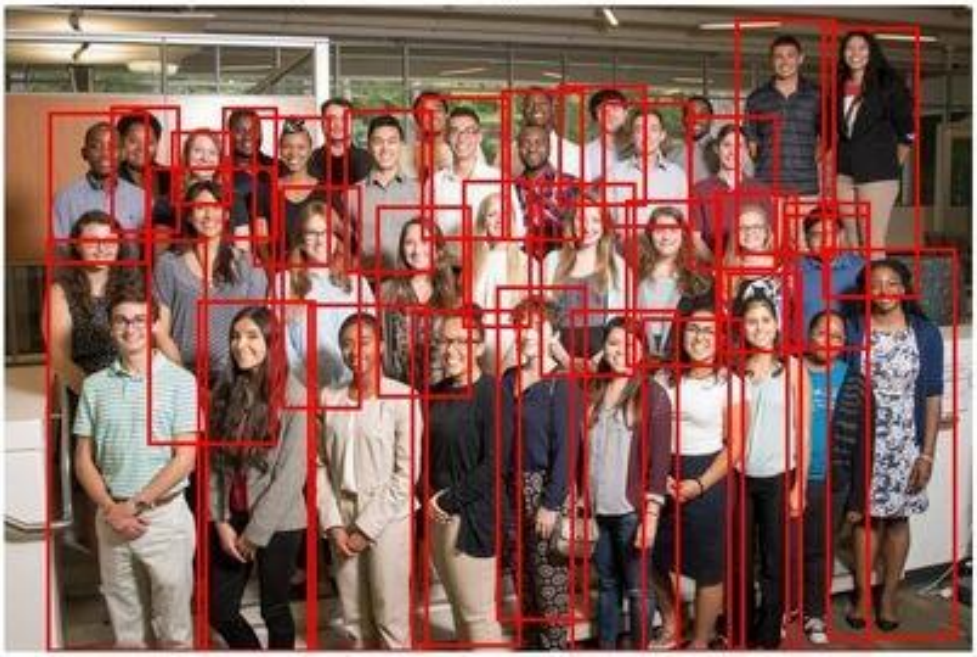
provide enough information to be classified as a separate entity and therefore got merged with another person. We have mitigated this issue by specifically training for these issue and calibrating variables where we see them and we can see further examples in the next section.

6.6 FRCNN Accuracy and Efficiency Discussion

As discussed above many factors contribute to accuracy and efficiency. Even classifying accuracy is difficult as we need to constitute what is a 100% accurate. We took the image above as a boundary. Bodies clearly visible, not densely packed, with decent resolution and clarity became the benchmark. The correct value being judged is the number of person detected. We also judge the answer to be in a range that constitutes as to being within the range of 15% of the correct number after rounding (flooring for least and roof value for the max). In the example above the exact number of people is 7 and therefore the range is 5-8 people. Therefore, our output of 6 people can be considered successful. In an example of 10 people the range would be 8-12.

As for efficiency per image, contributing factors are number of people in image, size of image, density of people. In an image of 1080x900 of 2 people standard apart and in view, our model takes 1.68 seconds to process with complete accuracy. In an image of 865x768 with 23 people took 2.056 seconds to run. Compared to the fully developed FRCNN model available on their Github repository the same images take around 4 and 5 seconds respectively simply because they are matching to so much data and information.

In the following image 31 people out of 36 were detected and the process took 2.8 seconds



**All images and comparisons were made in the same PC to keep time consistent.

6.7 MLP Classifier and Face Recognition Specifics

The MLP classifier works in tandem with the Face Recognition Library to identify the face and generate facial encodings based on features to classify Gender, Age and Ethnicity. The primary concern was to have a dataset wide and varied enough to being able to identify the people world over. The entire segment is obviously dependent on the face being clearly visible and having enough resolution for the encodings to be generated. That means that people who are seen sideways or have glasses or masks are not identified. These limitations could not be overcome as the only other way for classification in the categories given could be done via body shape, apparel, among others, but proved far too difficult a task to be challenged. One further dependency of the classification is the accuracy of the Face Recognition Library which is open

sourced, not trained by us, and made publicly available through the Python libraries. Therefore accuracy is severely determined by the proficiency of this library.

Chapter 7
Training the Models

7.1 Introduction

In this section we will touch upon how the individual models were trained and how they were trained. We will also touch upon the datasets which were used

7.2 Training the FRCNN Model

The FRCNN model was trained on the COCO Dataset [6]. The training dataset consisted of 20000 images. The model was tested upon 5000images. The final iteration of model was trained for 1000 epoch where each epoch is going through the entire dataset once.

7.3 Training the MLP Classifier

The MLP classifier was trained upon a combination of the publicly available FairFace Dataset [5] and the UTKFace Dataset [10]. The training consisted for 3000 epochs. The training dataset consisted of 60000 images and the testing consisted of 10000 images. Over the entire dataset of 60000 images we also 15% random images set apart for validation. These validation and train images were randomly selected for each epoch.

Chapter 8
Design Impact

8.1 Introduction

In this section we look upon the implications of our project. We will try to explain further the motivations and goals of the project and discuss the unintended side effects of our work. We also take a look at the social impact and how the project is limited to very few categories when it comes to Gender and Ethnicity and how they pertain to the wider world.

8.2 The Intended Audience

The intention behind our project was to serve the community at both the level of local businesses and governments. The ultimate output provides a statistical output of foot traffic through any location across the whole day. Our system presents a statistical analysis of the number of people detected across a surveyed region and provides analysis of the passersby. Businesses can use this project to figure out the number of customers in their store and have an analysis of who their customers are. The governments can use our work to easily survey public areas and see the density of people passing that region. Business owners can see the data and make changes to their store to suit their customers or bring in new ones.

Through our project, a business owner or manager can get detailed information such as whether the store is favored by men or women and can decide to cater to their existing type of customers or attempt to bring in the others. Large stores can also use this system to figure out which section of their store gets the most foot traffic. Similarly, the government can figure out which section of the city sees the most people and invest more in that region.

8.3 The Issue of Classification of Gender

We would like to start with saying that the people who worked on this project respects the right of every person to their own identity and understand that no person's gender or identity should be guessed solely on their appearance or face. The task of narrowing down a person to their gender based on a guess will be inaccurate in most sense. The purpose of this project is not to identify or narrow down a person but rather to provide a sort of statistical data that can be interpreted upon.

The model and algorithm is based upon the data of 70000 people who identified as "Male" or "Female". It would have been improbable of us to collect data of everyone within the fluid spectrum of gender. Therefore, the classification is limited to the broad spectrum of "Male" or "Female". These classifications are made solely on facial features that are most commonly seen on our binary spectrum.

The purpose of this project is not to suggest or state the identity of a person. It is merely intended to create a prediction upon the likelihood of the person falling upon the algorithmic binary spectrum that is limited upon the 70000 people within the training dataset.

8.4 The Issue of Classification of Age

For many people guessing the age of another person is rather intuitive. We can make educated guesses based on appearances such as wrinkles or skins but they are never truly accurate. A lot of people often don't look their age. This is something everyone knows and again our project was limited only to the dataset available. It will be impossible to find images of every possible appearance of any age. The goal is to find the most acceptable answer to what is the age of the target person. And its accuracy is limited to both environmental and human limitations.

8.5 The Issue of Classification of Ethnicity

There is a wide number of Ethnicities that people identify with. Many people are also born in places that is far away from where their ancestors were. It is wrong to outright suggest or call the Ethnicity of a person. The intention of this project was to suggest the Ethnicity of a Person. This project provides a prediction of what might be the Ethnicity of person within its dataset of 70000 people or resembles the most within that data. This is an algorithm that is limited by the examples from which it learned and is limited to only that in capacity.

8.6 The Right to Privacy

The works of this project can serve to survey people both publicly and privately. The project is not intended for the purposes of violating a person's right to privacy. It is intended to provide statistical data to serve businesses and the community. The usage of this system should be publicly announced wherever it is being utilized to give people the right to deny the use of their likeliness if they are inclined to do so especially when used to survey public places.

8.7 The Unintended Consequences

Like all works this project is limited by its users. However, the intention behind its creation is simply to provide statistical data and information even if limited to a certain capacity. It is not meant to harass or criticize or suggest what a person should be. It must be remembered that this work is limited by the data used to train and test the algorithm. The predications are limited and should not be enforced upon anyone.

Chapter 9
Result and Evaluation

9.1 Introduction

In this section we will look in depth at the results of the testing and discuss them accordingly.

9.2 Person Detection Accuracy

Firstly we need to present the parameters. They have already been discussed above, but, to reiterate:

The system cannot detect people separately if they are packed in an incredibly dense manner like in a concert or stadium. There needs to be enough that can be used to separate people. However, the amount of data needed varies a lot in situations. From our testing, the system can be judged being reliably accurate if the number of people in the image is up to 50. Similarly, another limitation is the angle of view can be up to 75 degrees above or below the eye level. Furthermore, detections can be considered accurate if the number of person detected is within a range of 15% of the actual number of people in picture. For example, a picture of 10 individuals would have a correct range of 8-12, with the least value being floored and the max being roofed.

With all these in mind, and testing on 5000 images of varying size, shape, quality and number of individuals of different race, age and ethnicities and a number of individuals in almost all pictures our final accuracy for Persons were: 77.5%

Here is a comparison with other state of the art detection models:

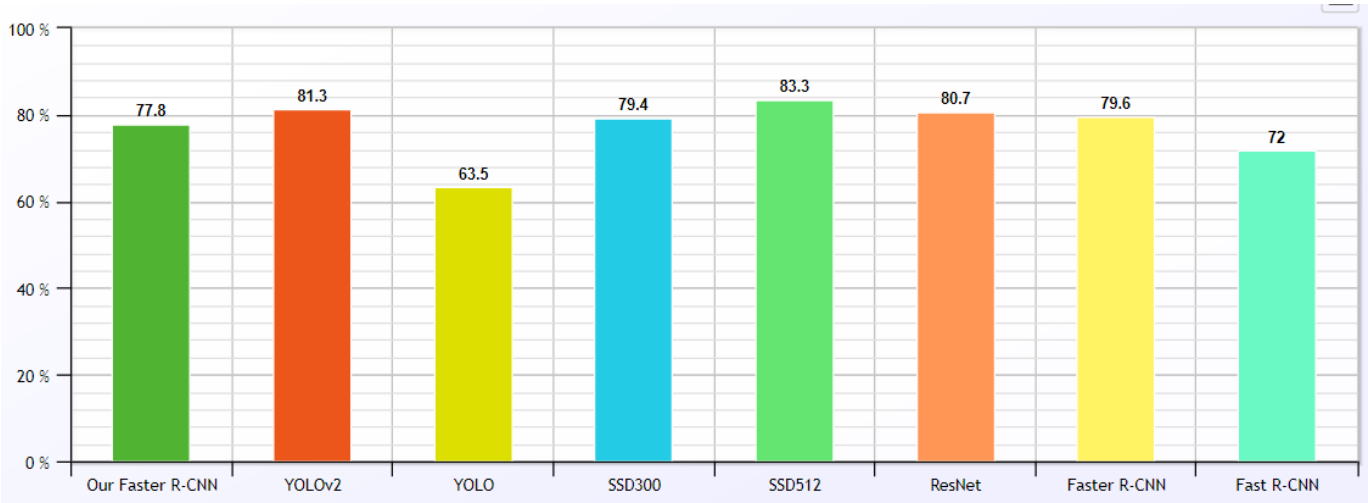


Table: Comparison of our model accuracy (Furthest to the left; green) vs. other state of the art

9.3 Evaluation

We can see that our model is within range of many of the state of the art models for detection. The 2 largest reasons for difference in accuracy is: Data used for Training and the amount of classes available. The effect of more data in training is that the model now has a better chance of recognition and more data to match with or pull from. Having many more classes, almost all other examples here have used above 80 classes, makes it easier for the models to identify and therefore eliminate other objects within the Region of Interest.

Our FRCNN pulls at 77.8% while the standard and trained FRCNN pulls at 79.6%. The differences between ours and state of the art FRCNN are that the base RNN has been altered, parameters changed to suit lower classes and slightly different implementations of the Max

Suppression system. Another notable change is the speed at which ours performs. Out of all the models presented above, ours performs faster than all the others. This holds true even when adjusted for the lower number of classes. The most accurate that has so far been seen is the SSD512 model which is also the slowest among all known detection models. YOLOv2 has a far better mix of efficiency and accuracy even if it is very resource hungry. Provided with a PC that has the architecture to handle large computations YOLOv2 is very fast but it does require a lot of power. At cases of when such resources are not available the falloff is much steeper. The purpose of our work was to balance efficiency and accuracy and we believe we have

9.4 Gender, Age and Ethnicity Accuracy:

In this section we will determine the Accuracy in all 3 categories. We used the same pictures across every category. And, the images were chosen at random. The distribution across all images is given below:

Male	Female	Total
5202	4798	10000

Table I: Number of Male and Female in Testing

Child Age: 0-2	Adolescent Age: 3-9	Teen Age: 10-19	Young Adult Age: 20-29	Adult Age:30-39	Old Adult Age:4 0-59	Senior Age: 60+	Total
1498	780	1480	1800	1800	1800	842	10000

Table II: Number of People in each Age group

White	Black	Asian	Indian	Other	Total
3000	1993	1992	2008	1007	10000

Table III: Number of people in all Ethnic Categories

9.5 A Gender Detection Accuracy:

The Results done for 10000 images for Gender are given below:

	Actual	Detected	Correct Detection	Wrong Detection
Male	5202	5235	4597/88.4%	638
Female	4798	4765	4160/86.7%	605
Total	10000	10000	8757/87.6%	1243

Table IV: Number of Male, Female and Total Detection Accuracy

Here, we can see that we have 88.4% accuracy for Male Detection, 86.7% accuracy for Female and an overall 87.6% accuracy. The Column of Wrong Detection identifies the false positives. 638 Males were classified that were actually Female and thus wrongfully Detected. We can see that the number of false positives is relatively similar for both Male and Female for all Ethnicities.

9.5 B Age Detection Accuracy:

	Actual	Detected	Correct Detection	Wrong Detection
Child 0-2	1498	1337	1177	160
Adolescent 3-9	780	1300	576	747
Teen 10-19	1480	681	501	180
Young Adult 20-29	1800	3598	1519	2079
Adult 30-39	1800	1091	455	636
Old Adult 40-59	1800	1546	930	616
Senior 60+	842	424	397	27
Total	10000	10000	5555	4445

Table V: Detection within Age Range

Now, at a first glance it may appear our Age Classification is severely mistaken. The Category for Young Adult for example has over 2000 mistakes. First of all 3598 Young Adults were detected where only 1519 were correct. The mistakes were that other age groups were wrongly classified as "Young Adult". Let us look at the total classification for Young Adult:

Child	Adolescent	Teen	Young Adult (Correct)	Adult	Old Adult	Senior
8	8	535	1519	1173	349	6

Table VI: Detection of Young Adult within the entire Age Range

We can see that some Teens were wrongfully classified as Young Adults and many Adults were wrongfully classified as Young Adults. In fact most of the error occurred for between the ages of 27-33. Since the separation of Young Adult and Adult is at the age of 30, therefore, we believe that these false positives are understandable. This is a region of age that is very difficult to classify even for humans. As for the errors for Teens, many Teens are often mistakenly called Young Adults, particularly for the Age Range 18-23 where most of those false positives were seen. In fact for every category mistakes occurred with the adjacent category. Children and Adolescent were confused. Adults were confused with Young Adults and Old Adults and so on. We believe this is one category where accuracy cannot be judged on just numbers of correct and wrong.

8.5 C Ethnicity Detection Accuracy:

The Results for 10000 images are given below:

	Actual	**Detected**	**Correct Detection**	**Wrong Detection**
White	3000	2748	2317	431
Black	1993	1655	1549	106
Asian	1992	2127	1839	288
Indian	2008	1367	1076	291
Other	1007	2103	524	1579
Total	10000	10000	7305	2695

Table VII: Detection of People in all Ethnicities

We can see that the results overall seems close especially for people of White, Black and Asian descent. The accuracy suffers when it comes to Indian people but particularly when it comes to Other ethnicities. One particular reason for this is that the dataset did include a portion of Hispanic and Latino descent. However, they were not included as their addition was a detriment over the total accuracy as often people of the Subindian continent were often classified as Latino. There were some images like this which entered the test set. Majority of the Wrong Detection were in the Others category. We can further discuss this category with the following table:

White	Black	Asian	Indian	Other (Correct)
573	195	72	739	524

Table VIII: Detection of People in all Ethnicities

The Others category severely misclassified White and Indian Ethnicity. One of the reasons was Age. We lacked data for all ethnicities for different ages and therefore when the Ethnicity detection saw old people during test, it failed to put them in the correct category and so put them in others. The same is true for children and young people. Often people with tattoo's or sunglasses or alternate angles were misclassified.

Chapter 10
Future Works

10.1 Introduction

In this section we look at how we want to progress further in this project. We will present our aims and ideas for the future pertaining to this work.

10.2 Person Detection on Mobile Application

We wish to be able to deploy our Person Detection Model on mobile applications so that the detection can work on apps offline. It is so that we can offer our services on an even simpler platform and provide accurate and efficient predictions on an even simpler platform

10.3 Mask Detection

During a time of global pandemic, the use of Masks is encouraged by doctors and scientists to prevent the spread of contagious diseases. However, detection of faces is severely limited when a person is wearing a mask. We wish to be able to provide a system that build upon our classifier to detect whether a person is wearing a mask so that social rules during a pandemic can be enforced. It is also an interesting challenge whether a mask can not only be correctly identified but also being ran in a simple device so that it may be used anywhere no matter the device

10.4 Mask Detection for Mobile Application

As stated previously the detection of Masks is often vital and making them simple enough to run anywhere is a notable achievement. To that end, we want to deploy our mask detector on mobile application and make the model run on almost any device.

10.5 Graphical User Interface

At the moment the project works on Windows Consoles and can be difficult to use for someone who has limited experience using consoles on computers. We want to make it user friendly through the implementation of a User Interface, making its use simple and intuitive.

10.6 Online Deployment of Project

A large scale deployment for it to be utilized online makes the works available to a much wider audience. It also makes the user device redundant if the project is being used through servers. We want to be able to make the system highly accessible and make it available through any platform and provide a wide range of ways for it to be accessed. Making the project browser based bypasses almost all device restrictions.

10.7 Improve Further upon the Results

While we believe the project to be accurate and efficient at this point in time, models are only getting faster and better. We want to keep working further by collecting more data and further streamlining the output to make it even faster and better. There is nowhere to go but up.

Bibliography

References

[1] R. Benenson, M. Omran, J. Hosang and B. Schiele, "Ten years of pedestrian detection, what have we learned?," in *European Conference on Computer Vision*, 2014.

[2] C. Chen and A. Ross, "Evaluation of gender classification methods on thermal and near-infrared face images," in *2011 International Joint Conference on Biometrics (IJCB)*, 2011.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, p. 303–338, 6 2010.

[4] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014.

[5] K. Kärkkäinen and J. Joo, "FairFace: Face Attribute Dataset for Balanced Race, Gender, and Age," *arXiv preprint arXiv:1908.04913*, 2019.

[6] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, 2014.

[7] Z. Liu, Z. Chen, Z. Li and W. Hu, "An Efficient Pedestrian Detection Method Based on YOLOv2," *Mathematical Problems in Engineering*, vol. 2018, 2018.

[8] S. Ren, K. He, R. Girshick and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015.

[9] T. Surasak, I. Takahiro, C.-h. Cheng, C.-e. Wang and P.-y. Sheng, "Histogram of oriented gradients for human detection in video," in *2018 5th International Conference on Business and Industrial Research (ICBIR)*, 2018.

[10] S. Y. Zhang and H. Qi, "Age Progression/Regression by Conditional Adversarial Autoencoder," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Website Links of libraries, frameworks and datasets used:

https://pypi.org/project/face-recognition/

https://www.tensorflow.org/

https://matplotlib.org/

http://host.robots.ox.ac.uk/pascal/VOC/

https://arxiv.org/abs/1908.04913

https://susanqq.github.io/UTKFace/

Appendix
Software Listing

Gender Detection

create_label.py file

```
import csv

# Get rid of failed images by checking feature.csv and create a new label file

with open('feature.csv') as file:
    reader = csv.reader(file)
    with open('Fairlabel.csv') as file2:
        reader2 = csv.reader(file2)
        with open('final_label3.csv', 'w', newline='') as file3:
            writer = csv.writer(file3)
            for line in reader:
                file2.seek(0)
                for line2 in reader2:
                    if line[0] == line2[0]:
                        writer.writerow(line2)
                        break
```

feature-extration.py file

```
import os
import csv
import pandas as pd
import argparse
import dlib
import face_recognition
from face_recognition import face_locations



def main():
    parser = argparse.ArgumentParser()

    parser.add_argument(
        '--save_feature',
        type=str,
        default='feature.csv',
        help='path to the feature file to be save (default: feature.csv)')

    parser.add_argument(
        '--read_csv',
        type=str,
        default='Fairlabel.csv',
        help='path to the csv file to read (default: Fairlabel.csv)')
    args = parser.parse_args()

    print("extracting face features from images")
    feature_vecs = []
```

```
fnames = []
final_fnames = []
failed_fnames = []

# Open the csv file and take the image location from each row
with open(args.read_csv) as file:
    reader = csv.reader(file)
    next(reader)
    for line in reader:
        fnames.append(line[0])

for fname in fnames:
    img_path = fname

    # face detection
    print(img_path)
    X_img = face_recognition.load_image_file(img_path)
    X_faces_loc = face_locations(X_img)
    # if the number of faces detected in a image is not 1, ignore the image
    if len(X_faces_loc) == 0:
        failed_fnames.append(fname)
        continue
    # extract 128 dimensional face features
    final_fnames.append(fname)
    faces_encoding = dlib.face_recognition_model_v1(
        X_img, known_face_locations=X_faces_loc)[0]
    feature_vecs.append(faces_encoding)
# Save the features in a csv file
df_feat = pd.DataFrame(feature_vecs, index=final_fnames)
df_feat.sort_index(inplace=True)
```

```
    df_feat.to_csv(args.save_feature)
    # Print file names where a single face was not detected
    print(failed_fnames)


if __name__ == "__main__":
    main()
```

train.py file

```
import argparse
import os
import face_recognition
import numpy as np
import pickle
from face_recognition import face_locations
from tqdm import tqdm
import cv2
import pandas as pd
import csv

COLS = [
    'Gender',
    'Asian',
    'White',
    'Black',
    'Indian',
```

```
    'Others',
    'Child',
    'Adolescent',
    'Teen',
    'Young Adult',
    'Adult',
    'Old Adult',
    'Senior']
N_UPSCLAE = 1


def extract_features(img_path):
    """Exctract 128 dimensional features
    """
    X_img = face_recognition.load_image_file(img_path)
    locs = face_locations(X_img, number_of_times_to_upsample=N_UPSCLAE)
    if len(locs) == 0:
        return None, None
    face_encodings = face_recognition.face_encodings(
        X_img, known_face_locations=locs)
    return face_encodings, locs


def predict_one_image(img_path, clf, labels):
    """Predict face attributes for all detected faces in one image
    """
    face_encodings, locs = extract_features(img_path)
    if not face_encodings:
```

```
        return None, None
    pred = pd.DataFrame(clf.predict_proba(face_encodings),
                        columns=labels)
    pred = pred.loc[:, COLS]
    return pred, locs


def draw_attributes(img_path, df):
    """Write predicted face attributes on the image
    """
    races = ['A', 'W', 'B', 'I', 'O']
    ages = ['C', 'Adst', 'T', 'Y Ad', 'Ad', 'Old Ad', 'Se']
    img = cv2.imread(img_path)
    data = {
        'Face': 0,
        'Male': 0,
        'Female': 0,
        'A': 0,
        'W': 0,
        'B': 0,
        'I': 0,
        'O': 0,
        'C': 0,
        'Adst': 0,
        'T': 0,
        'Y Ad': 0,
        'Ad': 0,
        'Old Ad': 0,
```

```
    'Se': 0
}
for row in df.iterrows():
    #
    top, right, bottom, left = row[1][13:].astype(int)
    data['Face'] = data['Face'] + 1
    if row[1]['Gender'] <= 0.5:
        gender = 'M'
        data['Male'] = data['Male'] + 1
    else:
        gender = 'F'
        data['Female'] = data['Female'] + 1
    race = races[int(np.argmax(row[1][1:6]))]
    age = ages[int(np.argmax(row[1][6:13]))]
    data[race] = data[race] + 1
    data[age] = data[age] + 1
    text_showed = f"{age} {race} {gender}"
    # No face boundary now
    #cv2.rectangle(img, (left, top), (right, bottom), (0, 0, 255), 2)
    # Write predicted attributes on the image at bottom left corner of the
    # image
    font = cv2.FONT_HERSHEY_DUPLEX
    img_width = img.shape[1]
    font_scale = cv2.getFontScaleFromHeight(font, img.shape[1], 1)
    cv2.putText(img, text_showed, (left + 6, bottom - 6),
                font, 0.5, (0, 0, 0), 2)
    cv2.putText(img, text_showed, (left + 6, bottom - 6),
                font, 0.5, (255, 255, 255), 1)
```

```
    data['img'] = img
    return data


def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('--img_dir', type=str,
                        default='test/',
                        help='input image directory (default: test/)')
    parser.add_argument(
        '--output_dir',
        type=str,
        default='results/',
        help='output directory to save the results (default: results/')
    parser.add_argument(
        '--model',
        type=str,
        default='face_model_Fair_6.pkl',
        help='path to trained model (default: face_model_Fair_6.pkl)')

    args = parser.parse_args()
    output_dir = args.output_dir
    input_dir = args.img_dir
    model_path = args.model

    if not os.path.isdir(output_dir):
        os.mkdir(output_dir)
    # load the model
```

```
with open(model_path, 'rb') as f:
    clf, labels = pickle.load(f, encoding='latin1')

print("classifying images in {}".format(input_dir))
data_list = []
for fname in tqdm(os.listdir(input_dir)):
    img_path = os.path.join(input_dir, fname)
    try:
        pred, locs = predict_one_image(img_path, clf, labels)
    except BaseException:
        print("Skipping {}".format(img_path))
    if not locs:
        continue
    locs = \
        pd.DataFrame(locs, columns=['top', 'right', 'bottom', 'left'])
    df = pd.concat([pred, locs], axis=1)
    data = draw_attributes(img_path, df)
    img = data['img']
    data['img'] = fname
    data_list.append(data)
    cv2.imwrite(os.path.join(output_dir, fname), img)
    os.path.splitext(fname)[0]
    output_csvpath = os.path.join(output_dir,
                                  os.path.splitext(fname)[0] + '.csv')
    df.to_csv(output_csvpath, index=False)
with open(os.path.join(output_dir, 'output.csv'), 'w', newline='') as file:
    writer = csv.DictWriter(
        file,
```

```
            fieldnames=[
                'img',
                'Face',
                'Male',
                'Female',
                'A',
                'W',
                'B',
                'I',
                'O',
                'C',
                'Adst',
                'T',
                'Y Ad',
                'Ad',
                'Old Ad',
                'Se'])
        writer.writeheader()
        for dic in data_list:
            writer.writerow(dic)
    return 0


if __name__ == "__main__":
    main()
```

pred.py file

```
import argparse
import os
import face_recognition
import numpy as np
import pickle
from face_recognition import face_locations
from tqdm import tqdm
import cv2
import pandas as pd
import csv

COLS = [
    'Gender',
    'Asian',
    'White',
    'Black',
    'Indian',
    'Others',
    'Child',
    'Adolescent',
    'Teen',
    'Young Adult',
    'Adult',
    'Old Adult',
    'Senior']
N_UPSCLAE = 1


def extract_features(img_path):
```

```
    """Exctract 128 dimensional features
    """
    X_img = face_recognition.load_image_file(img_path)
    locs = face_locations(X_img, number_of_times_to_upsample=N_UPSCLAE)
    if len(locs) == 0:
        return None, None
    face_encodings = face_recognition.face_encodings(
        X_img, known_face_locations=locs)
    return face_encodings, locs


def predict_one_image(img_path, clf, labels):
    """Predict face attributes for all detected faces in one image
    """
    face_encodings, locs = extract_features(img_path)
    if not face_encodings:
        return None, None
    pred = pd.DataFrame(clf.predict_proba(face_encodings),
                        columns=labels)
    pred = pred.loc[:, COLS]
    return pred, locs


def draw_attributes(img_path, df):
    """Write predicted face attributes on the image
    """
    races = ['A', 'W', 'B', 'I', 'O']
    ages = ['C', 'Adst', 'T', 'Y Ad', 'Ad', 'Old Ad', 'Se']
    img = cv2.imread(img_path)
    data = {
        'Face': 0,
```

```
    'Male': 0,
    'Female': 0,
    'A': 0,
    'W': 0,
    'B': 0,
    'I': 0,
    'O': 0,
    'C': 0,
    'Adst': 0,
    'T': 0,
    'Y Ad': 0,
    'Ad': 0,
    'Old Ad': 0,
    'Se': 0
}
for row in df.iterrows():
    #
    top, right, bottom, left = row[1][13:].astype(int)
    data['Face'] = data['Face'] + 1
    if row[1]['Gender'] <= 0.5:
        gender = 'M'
        data['Male'] = data['Male'] + 1
    else:
        gender = 'F'
        data['Female'] = data['Female'] + 1
    race = races[int(np.argmax(row[1][1:6]))]
    age = ages[int(np.argmax(row[1][6:13]))]
    data[race] = data[race] + 1
    data[age] = data[age] + 1
    text_showed = f"{age} {race} {gender}"
    # No face boundary now
```

```
        #cv2.rectangle(img, (left, top), (right, bottom), (0, 0, 255), 2)
        # Write predicted attributes on the image at bottom left corner of the
        # image
        font = cv2.FONT_HERSHEY_DUPLEX
        img_width = img.shape[1]
        font_scale = cv2.getFontScaleFromHeight(font, img.shape[1], 1)
        cv2.putText(img, text_showed, (left + 6, bottom - 6),
                    font, 0.5, (0, 0, 0), 2)
        cv2.putText(img, text_showed, (left + 6, bottom - 6),
                    font, 0.5, (255, 255, 255), 1)
    data['img'] = img
    return data


def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('--img_dir', type=str,
                        default='test/',
                        help='input image directory (default: test/)')
    parser.add_argument(
        '--output_dir',
        type=str,
        default='results/',
        help='output directory to save the results (default: results/')
    parser.add_argument(
        '--model',
        type=str,
        default='face_model_Fair_6.pkl',
        help='path to trained model (default: face_model_Fair_6.pkl)')

    args = parser.parse_args()
```

```
output_dir = args.output_dir
input_dir = args.img_dir
model_path = args.model

if not os.path.isdir(output_dir):
    os.mkdir(output_dir)
# load the model
with open(model_path, 'rb') as f:
    clf, labels = pickle.load(f, encoding='latin1')

print("classifying images in {}".format(input_dir))
data_list = []
for fname in tqdm(os.listdir(input_dir)):
    img_path = os.path.join(input_dir, fname)
    try:
        pred, locs = predict_one_image(img_path, clf, labels)
    except BaseException:
        print("Skipping {}".format(img_path))
    if not locs:
        continue
    locs = \
        pd.DataFrame(locs, columns=['top', 'right', 'bottom', 'left'])
    df = pd.concat([pred, locs], axis=1)
    data = draw_attributes(img_path, df)
    img = data['img']
    data['img'] = fname
    data_list.append(data)
    cv2.imwrite(os.path.join(output_dir, fname), img)
    os.path.splitext(fname)[0]
    output_csvpath = os.path.join(output_dir,
                                  os.path.splitext(fname)[0] + '.csv')
```

```
        df.to_csv(output_csvpath, index=False)
    with open(os.path.join(output_dir, 'output.csv'), 'w', newline='') as file:
        writer = csv.DictWriter(
            file,
            fieldnames=[
                'img',
                'Face',
                'Male',
                'Female',
                'A',
                'W',
                'B',
                'I',
                'O',
                'C',
                'Adst',
                'T',
                'Y Ad',
                'Ad',
                'Old Ad',
                'Se'])
        writer.writeheader()
        for dic in data_list:
            writer.writerow(dic)
    return 0


if __name__ == "__main__":
    main()
```

gui.py file

```
from tkinter import *
from tkinter import filedialog
from tkinter import messagebox
from subprocess import check_call
import csv
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

root = Tk()
root.title("Project")
root.geometry("600x260")

# # Initializes the entries on user interface
input_entry = Entry(
    root,
    text="",
    fg="black",
    bg="white",
    readonlybackground="white",
    disabledforeground="black",
    state="readonly",
    width=60,
    borderwidth=5)
output_entry = Entry(
    root,
    text="",
    fg="black",
```

```
    bg="white",
    readonlybackground="white",
    disabledforeground="black",
    width=60,
    borderwidth=5)
model_entry = Entry(
    root,
    text="",
    fg="black",
    bg="white",
    readonlybackground="white",
    disabledforeground="black",
    width=60,
    borderwidth=5)
input_dir = ''
output_dir = ''
model_dir = ''

# options store user settings in option.csv file
if os.path.exists("option.csv"):
    with open('option.csv') as file:
        reader = csv.DictReader(file)
        line = next(reader)
        input_dir = line['input_dir']
        output_dir = line['output_dir']
        model_dir = line['model_dir']



def update():
    """Updates the values on option.csv file
```

```
    """
    global input_dir
    global output_dir
    global model_dir

    data = {
        "input_dir": input_dir,
        "output_dir": output_dir,
        "model_dir": model_dir
    }

    with open('option.csv', 'w', newline='') as file:
        writer = csv.DictWriter(
            file,
            fieldnames=[
                "input_dir",
                "output_dir",
                "model_dir"])
        writer.writeheader()
        writer.writerow(data)


def choose_dir(option):
    """Let user pick a directory as input/output/model directory, put the new value on display
     and calls update to save the new value.
    """

    global input_dir
    global output_dir
    global model_dir
```

```
    if option == 0:
        input_dir = filedialog.askdirectory(
            parent=root,
            initialdir=sys.path[0],
            title='Please select input directory')
    if option == 1:
        output_dir = filedialog.askdirectory(
            parent=root,
            initialdir=sys.path[0],
            title='Please select output directory')
    if option == 2:
        model_dir = filedialog.askopenfilename(
            initialdir=sys.path[0],
            title='Please select model file',
            filetypes=[('pkl files', '*.pkl')])

    if input_dir:
        input_entry['state'] = 'normal'
        input_entry.delete(0, END)
        input_entry.insert(0, input_dir)
        input_entry['state'] = 'readonly'

    if output_dir:
        output_entry['state'] = 'normal'
        output_entry.delete(0, END)
        output_entry.insert(0, output_dir)
        output_entry['state'] = 'readonly'


if model_dir:
        model_entry['state'] = 'normal'
```

```
        model_entry.delete(0, END)
        model_entry.insert(0, model_dir)
        model_entry['state'] = 'readonly'

    update()


def return_output():
    """return the values from output.csv file
    """

    global output_dir
    return pd.read_csv(output_dir + "/output.csv")


def Person_per_Image():
    """Graph function for Create Graph 1 button
    """

    df = return_output()
    Person = df.Face
    Images = df.img
    plt.style.use("fivethirtyeight")

    plt.figure(figsize=(20, 10))

    x_indexes = np.arange(len(Images))
    width = 0.25

    plt.bar(
        x_indexes - width,
```

```
        Person,
        width=width,
        color="#444444",
        label="Person")

    plt.legend()

    plt.title("Per Image Person Distribution")
    plt.xlabel("Images")
    plt.ylabel("Count")

    plt.tight_layout()
    plt.show()


def Male_and_Female_per_Image():
    """Graph function for Create Graph 2 button
    """

    df = return_output()
    images = df.img
    Male = df.Male
    Female = df.Female
    plt.style.use("fivethirtyeight")

    plt.figure(figsize=(20, 10))

    x_indexes = np.arange(len(images))
    width = 0.25

    plt.bar(x_indexes, Male, width=width, color="#444444", label="Male")
```

```
    plt.bar(
        x_indexes + width,
        Female,
        width=width,
        color="#e5ae38",
        label="Female")

    plt.legend()

    plt.title("Per Image Gender Distribution")
    plt.xlabel("Images")
    plt.ylabel("Count")

    plt.tight_layout()

    plt.show()


def Person_Age_per_Image():
    """Graph function for Create Graph 3 button
    """

    df = return_output()

    Images = df['img']
    Adst = df['Adst']
    T = df['T']
    Y_Ad = df['Y Ad']
    Ad = df['Ad']
    Old_Ad = df['Old Ad']
```

```
Se = df['Se']

plt.style.use("fivethirtyeight")

plt.figure(figsize=(20, 10))

x_indexes = np.arange(len(Images))
width = 0.25
shift1 = 1.75
shift2 = 2.00
shift3 = 2.25
shift4 = 2.50
shift5 = 2.75
shift6 = 3.00

plt.bar(
    x_indexes +
    shift1,
    Adst,
    width=width,
    color="#444444",
    label="Adolescent")
plt.bar(
    x_indexes +
    shift2,
    T,
    width=width,
    color="#e5ae38",
    label="Teenager")
plt.bar(
    x_indexes +
```

```
    shift3,
    Y_Ad,
    width=width,
    color="#FF5733",
    label="Young Adult")
plt.bar(
    x_indexes +
    shift4,
    Ad,
    width=width,
    color="#3349FF",
    label="Adult")
plt.bar(
    x_indexes +
    shift5,
    Old_Ad,
    width=width,
    color="#13F3DE",
    label="Old Adult")
plt.bar(
    x_indexes +
    shift6,
    Se,
    width=width,
    color="#BC3FB8",
    label="Senior")

plt.legend()

plt.title("Per Image Age Distribution")
plt.xlabel("Images")
```

```
    plt.ylabel("Count")

    plt.tight_layout()

    plt.show()


def run_model():
    """Runs the face classifiacation model
    """
    global input_dir
    global output_dir
    global model_dir
    try:
        check_call(['python', 'pred.py', "--img_dir", input_dir,
                    "--output_dir", output_dir, "--model", model_dir])
    except BaseException:
        messagebox.showerror("Result", "Model did not run successfully!")
    else:
        messagebox.showinfo("Result", "Model ran successfully!")
        graph_btn_1['state'] = 'active'
        graph_btn_2['state'] = 'active'
        graph_btn_3['state'] = 'active'


# Initializes the buttons on user interface
input_btn = Button(
    root,
    text="Enter Input Directory",
    command=lambda: choose_dir(0),
    width=17)
```

```
output_btn = Button(
    root,
    text="Enter Output Directory",
    command=lambda: choose_dir(1),
    width=17)
model_btn = Button(
    root,
    text="Enter Model File",
    command=lambda: choose_dir(2),
    width=17)
submit_btn = Button(root, text="Run Model", command=run_model, width=17)
graph_btn_1 = Button(
    root,
    text="Create Graph 1",
    command=Person_per_Image,
    width=17,
    state="disabled")
graph_btn_2 = Button(
    root,
    text="Create Graph 2",
    command=Male_and_Female_per_Image,
    width=17,
    state="disabled")
graph_btn_3 = Button(
    root,
    text="Create Graph 3",
    command=Person_Age_per_Image,
    width=17,
    state="disabled")

# Places buttons and entries on the user interface
```

```
input_btn.grid(row=0, column=0, pady=10, padx=20)
output_btn.grid(row=1, column=0, pady=10, padx=20)
model_btn.grid(row=2, column=0, pady=10, padx=20)
submit_btn.grid(row=3, column=1, pady=10)
graph_btn_1.grid(row=4, column=0, pady=10)
graph_btn_2.grid(row=4, column=1, pady=10)
graph_btn_3.grid(row=4, column=2, pady=10)

input_entry.grid(row=0, column=1, pady=10, padx=20, columnspan=2)
output_entry.grid(row=1, column=1, pady=10, padx=20, columnspan=2)
model_entry.grid(row=2, column=1, pady=10, padx=20, columnspan=2)

# Shows the values on the entries on the interface
input_entry['state'] = 'normal'
input_entry.insert(0, input_dir)
input_entry['state'] = 'readonly'

output_entry['state'] = 'normal'
output_entry.insert(0, output_dir)
output_entry['state'] = 'readonly'

model_entry['state'] = 'normal'
model_entry.insert(0, model_dir)
model_entry['state'] = 'readonly'

# Runs the gui
root.mainloop()
```

Person Detection

train.py file

```
from __future__ import division
from __future__ import print_function
from __future__ import absolute_import



import _init_paths
from model.train_val import get_training_roidb, train_net
from model.config import cfg, cfg_from_file, cfg_from_list, get_output_dir, get_output_tb_dir
from datasets.factory import get_imdb
import datasets.imdb
import argparse
import pprint
import numpy as np
import sys

import tensorflow as tf
from nets.vgg16 import vgg16
from nets.resnet_v1 import resnetv1
def roidb(imdb_names):

  def pull_roidb(imdb_name):

    imdb = get_imdb(imdb_name)
    print('Dataset Number `{:s}`'.format(imdb.name))
    imdb.set_proposal_method(cfg.TRAIN.PROPOSAL_METHOD)
    print('Method: {:s}'.format(cfg.TRAIN.PROPOSAL_METHOD))
    roidb = get_training_roidb(imdb)
```

```
    return roidb

  roidbs = [pull_roidb(s) for s in imdb_names.split('+')]
  roidb = roidbs[0]
  if len(roidbs) > 1:
    for r in roidbs[1:]:
      roidb.extend(r)
    tmp = get_imdb(imdb_names.split('+')[1])
    imdb = datasets.imdb.imdb(imdb_names, tmp.classes)
  else:
    imdb = get_imdb(imdb_names)
  return imdb, roidb


if __name__ == '__main__':

  args = parse_args()

  print('Argument call:')
  print(args)

  if args.cfg_file is not None:
    cfg_from_file(args.cfg_file)
  if args.set_cfgs is not None:
    cfg_from_list(args.set_cfgs)

  print('Config:')
  pprint.pprint(cfg)

  np.random.seed(cfg.RNG_SEED)
  imdb, roidb = roidb(args.imdb_name)
```

```
print('{:d} ROIDB'.format(len(roidb)))


output_dir = get_output_dir(imdb, args.tag)
print('Model `{:s}`'.format(output_dir))


tb_dir = get_output_tb_dir(imdb, args.tag)
print('Log `{:s}`'.format(tb_dir))

orgflip = cfg.TRAIN.USE_FLIPPED
cfg.TRAIN.USE_FLIPPED = False
_, valroidb = roidb(args.imdbval_name)
print('{:d} validation roidb entries'.format(len(valroidb)))
cfg.TRAIN.USE_FLIPPED = orgflip


if args.net == 'vgg16':
  net = vgg16()
elif args.net == 'res50':
  net = resnetv1(num_layers=50)
elif args.net == 'res101':
  net = resnetv1(num_layers=101)
else:
  raise NotImplementedError

train_net(net, imdb, roidb, valroidb, output_dir, tb_dir,
          pretrained_model=args.weight,
          max_iters=args.max_iters)
```

test.py file

```
from __future__ import division
from __future__ import print_function

import _init_paths
from model.test import test_net
from model.config import cfg, cfg_from_file, cfg_from_list
from datasets.factory import get_imdb
import argparse
import pprint
import time, os, sys

import tensorflow as tf
from nets.vgg16 import vgg16
from nets.resnet_v1 import resnetv1
from nets.mobilenet_v1 import mobilenetv1



if __name__ == '__main__':
  args = parse_args()

  print('Called with args:')
  print(args)

  if args.cfg_file is not None:
    cfg_from_file(args.cfg_file)
  if args.set_cfgs is not None:
    cfg_from_list(args.set_cfgs)
```

```
print('Using config:')
pprint.pprint(cfg)


if args.model:
  filename = os.path.splitext(os.path.basename(args.model))[0]
else:
  filename = os.path.splitext(os.path.basename(args.weight))[0]

tag = args.tag
tag = tag if tag else 'default'
filename = tag + '/' + filename

imdb = get_imdb(args.imdb_name)
imdb.competition_mode(args.comp_mode)

tfconfig = tf.ConfigProto(allow_soft_placement=True)
tfconfig.gpu_options.allow_growth=True


sess = tf.Session(config=tfconfig)

if args.net == 'vgg16':
  net = vgg16()
elif args.net == 'res50':
  net = resnetv1(num_layers=50)
elif args.net == 'res101':
  net = resnetv1(num_layers=101)

  net = mobilenetv1()
else:
```

```
  raise NotImplementedError


net.create_architecture("TEST", imdb.num_classes, tag='default',
                        anchor_scales=cfg.ANCHOR_SCALES,
                        anchor_ratios=cfg.ANCHOR_RATIOS)

if args.model:
  print(('Loading model check point from {:s}').format(args.model))
  saver = tf.train.Saver()
  saver.restore(sess, args.model)
  print('Loaded.')
else:
  print(('Loading initial weights from {:s}').format(args.weight))
  sess.run(tf.global_variables_initializer())
  print('Loaded.')

test_net(sess, net, imdb, filename, max_per_image=args.max_per_image)

sess.close()
```

run.py file

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import _init_paths
from model.config import cfg
from model.nms_wrapper import nms
from model.test import im_detect

from nets.vgg16 import vgg16
from nets.resnet_v1 import resnetv1
from PIL import Image
import PIL

from utils.timer import Timer
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import os, cv2
import argparse
import logging
import subprocess



def sendmessage(title, message):
    pynotify.init("Test")
    notice = pynotify.Notification(title, message)
    notice.show()
```

```
    return

def sendmessage2(message):
    subprocess.Popen(['notify-send', message])
    return

CLASSES = ('__background__',
           'aeroplane', 'bicycle', 'bird', 'boat',
           'bottle', 'bus', 'car', 'cat', 'chair',
           'cow', 'diningtable', 'dog', 'horse',
           'motorbike', 'person', 'pottedplant',
           'sheep', 'sofa', 'train', 'tvmonitor')


NETS = {'vgg16': ('vgg16_faster_rcnn_iter_65000.ckpt',)}
DATASETS= {'pascal_voc_0712': ('voc_2007_trainval+voc_2012_trainval',)}


def detection(im, class_name, dets, thresh=0.5):

    inds = np.where(dets[:, -1] >= thresh)[0]
    if len(inds) == 0:
        return

    im = im[:, :, (2, 1, 0)]
    fig, ax = plt.subplots(figsize=(12, 12))
    ax.imshow(im, aspect='equal')
    class_count=0
    for i in inds:
        bbox = dets[i, :4]
        score = dets[i, -1]
```

```
            ax.add_patch(
                plt.Rectangle((bbox[0], bbox[1]),
                              bbox[2] - bbox[0],
                              bbox[3] - bbox[1], fill=False,
                              edgecolor='red', linewidth=3.5)
                )
            class_count = class_count + 1

        ax.set_title(('{} detections with '
                      'p({} | box) >= {:.1f}').format(class_name, class_name,
                                                      thresh),
                      fontsize=14)

        plt.axis('off')
        plt.tight_layout()
        plt.draw()
        return class_count

def run(sess, net, image_name):


        im_file = os.path.join(cfg.DATA_DIR, 'run', image_name)
        im = cv2.imread(im_file)



        timer = Timer()
        timer.tic()
        scores, boxes = im_detect(sess, net, im)
        timer.toc()
        print('Detection took {:.3f}s for 20 object classes'.format(timer.total_time, boxes.shape[0]))
```

```
    CONF_THRESH = 0.8
    NMS_THRESH = 0.3
    for cls_ind, cls in enumerate(CLASSES[1:]):
        person_count=0
        cls_ind += 1
        cls_boxes = boxes[:, 4*cls_ind:4*(cls_ind + 1)]
        cls_scores = scores[:, cls_ind]
        dets = np.hstack((cls_boxes,
                          cls_scores[:, np.newaxis])).astype(np.float32)
        keep = nms(dets, NMS_THRESH)
        dets = dets[keep, :]
        if str(cls)=='person':
            person_count = detection(im, cls, dets, thresh=CONF_THRESH)
            if person_count is None:
                person_count = 0
            print(str(cls) + ' : ' + str(person_count))
            if person_count > 0:
                f = open("log.txt","a+")
                f.write(str(image_name) + ' ' + str(person_count) + '\n')
                f.close()


if __name__ == '__main__':

    cfg.TEST.HAS_RPN = True
    args = parse_args()

    demonet = args.demo_net
    dataset = args.dataset
```

```
tfmodel = os.path.join('output', demonet, DATASETS[dataset][0], 'default',
                       NETS[demonet][0])


if not os.path.isfile(tfmodel + '.meta'):
    raise IOError(('{:s} not found.\nDownload correct network from '
                   'Put the network in correct directory').format(tfmodel + '.meta'))

tfconfig = tf.ConfigProto(allow_soft_placement=True)
tfconfig.gpu_options.allow_growth=True


sess = tf.Session(config=tfconfig)


if demonet == 'vgg16':
    net = vgg16()
elif demonet == 'res101':
    net = resnetv1(num_layers=101)
else:
    raise NotImplementedError
net.create_architecture("TEST", 21,
                        tag='default', anchor_scales=[8, 16, 32])
saver = tf.train.Saver()
saver.restore(sess, tfmodel)

print('Network loading{:s}'.format(tfmodel))

im_names = os.listdir('data/demo/')
print(im_names)
```

```
for im_name in im_names:

    print('Demo for data/demo/{}'.format(im_name))
    demo(sess, net, im_name)
    plt.savefig('results/result_{}'.format(im_name), bbox_inches='tight')
```