B.TECH PROJECT END-TERM REPORT

Project Title	**ML Based Surveillance using drones**
Name of the Student/s with Roll Number/s	**1. Tanishq Joshi (B19EE083)** **2. Suyash SIngh (B19EE100)**
Name of the Supervisor/s	**Prof. Binod Kumar**

Contents of the report:

A. Abstract

As security concerns are increasing in the world, it is of utmost importance that the surveillance is done in an efficient way. In this project, we have developed an end-to-end system for such surveillance from a drone using Machine Learning Technologies. We have used various detection and recognition models and compared their performance in order to find the best suitable model for the given use case. As the real world conditions may vary a lot depending on the lighting conditions in which the drone is flying or the quality of the video feed from the drone, it is of utmost importance that our system is robust and can tackle all these situations well. Thus, we have employed various data augmentation and fine tuning techniques in order to make our model robust so that it can perform well in all situations.

B. Motivation

Surveillance using drones is extremely important nowadays amidst rising security concerns. Integrating Machine Learning Techniques with drone based surveillance enables us to have autonomous response systems. Humans are prone to fatigue and errors but autonomous response system as in ML based surveillance techniques can be programmed to operate reliably and continuously over long periods of time

Intelligent surveillance systems can easily stream live a security feed or follow fleeing suspects and provide the concerned authorities with real time tracking information of the

suspects. Drones can serve as a better alternative to guards by patrolling threat sites, capturing aerial footages, securing perimeters and preventing break ins. Intelligent surveillance systems can easily maintain real time surveillance 24 hours a day

C. Methodology

I. Problem Statement

The problem statement revolves around the development of an efficient algorithm for performing surveillance using a drone and its live camera feed, this involves the development of algorithms like face detection and face recognition which can then be fed into the FPGA boards of the drone to perform computation in the drone itself.

II. Literature Review and Exploring Existing Solutions

There exist many solutions to the similar problem statement stated above and thus before developing our own in-house solution, it is necessary to look at all the available state-of-the-art solutions.
The project can be divided into 2 parts:

1) Developing an efficient face detection algorithm
2) Developing an efficient face recognition algorithm

Below is the description of some popular widely known solutions and their comparison with each other.

(i) Detection:

➔ *Opencv*

Face detection using OpenCV is performed using Haar Cascade, an open-source library written in C language. Speedwise, as the library is written in C, it is extremely fast to use. Haar Cascade was able to detect the straight-angle faces easily but it had difficulty in detecting the side faces. The algorithm also performed poorly in bad lighting conditions

➔ *YOLO*

YOLO is a state-of-the-art object detection algorithm. It uses convolutional neural networks for detection. It can detect 80 classes with high accuracy but we use it for detecting only 1 class which is the face. The main problem of this model is that it is computationally consuming and may take more time to run. The advantage is high accuracy without any flaws.

➔ *SSD*
Single Shot Detection algorithm is also a deep learning algorithm for face detection. The main advantage of this model is that it can detect faces in various lighting and various poses. Its accuracy is decent but not as good as the YOLO model. Its speed is good but still not suitable to run on a CPU

➔ *MTCNN*
MTCNN is also an algorithm based on Deep Learning techniques. It uses Deep Cascaded Convolutional Neural Networks for detection. It has better accuracy than the OpenCV model but correspondingly takes more time to run. The MTCNN model is very accurate for face detection applications but it is very slow. For this we have used the FastMTCNN algorithm which is a bit inaccurate but is very suitable for real time applications.

(ii) Recognition:

➔ *face_recognition library*
The face recognition library is a simple library to recognize and manipulate faces from the command line. This library is based on dlib's state-of-the-art recognition system. The model has an accuracy of 98.38% on a benchmark faces dataset. This library provides us with functions to recognize faces using facial landmarks. Then the main task of this library is to recognize the detected faces. The library contains functions to encode the found faces using deep learning. Encoding the image provides us with an easy way to compute the distance between the 2 images and thereby recognize the image by finding the closest face in the database.

➔ *facenet library*
The facenet library is a popular implementation of the face recognition problem statement, and it uses a deep learning-based model to perform face recognition, the deep learning model used is Inception Resnet which is used to convert an image to a 128-dimensional feature space of embeddings, these embeddings can then be used to compute the similarity between 2 images and thus perform face recognition.

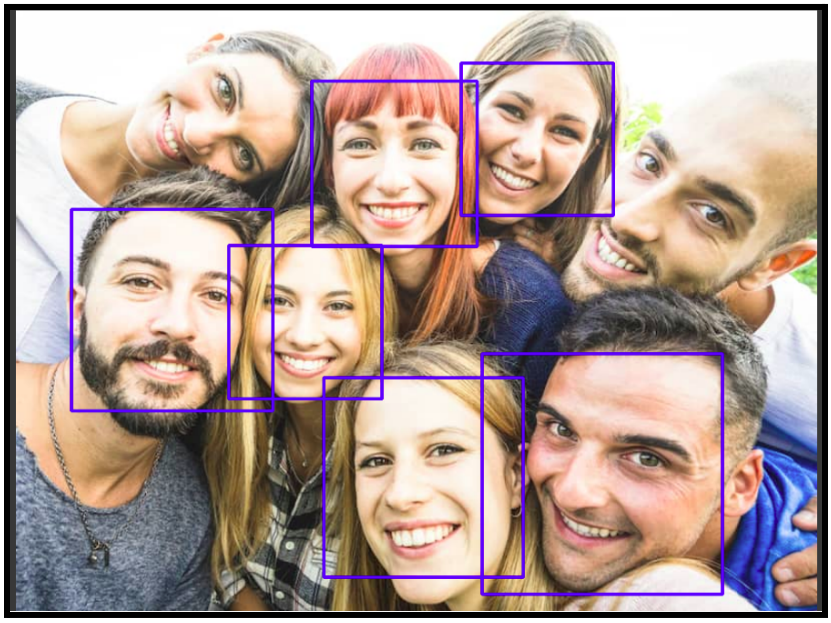
D. Results and Discussion

I. Initial Approach:

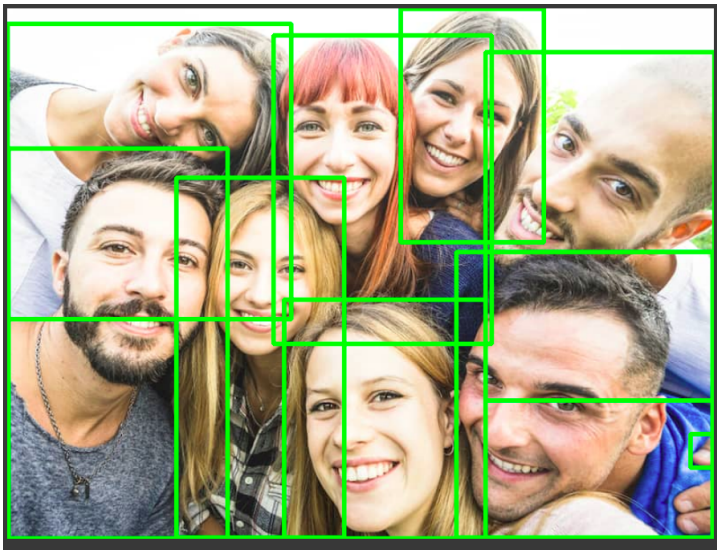
Our Initial approach involved the identification of the open-source and state-of-the-art solution and its implementation, the workflow was as follows:

1. *Implementation of the detection models:*

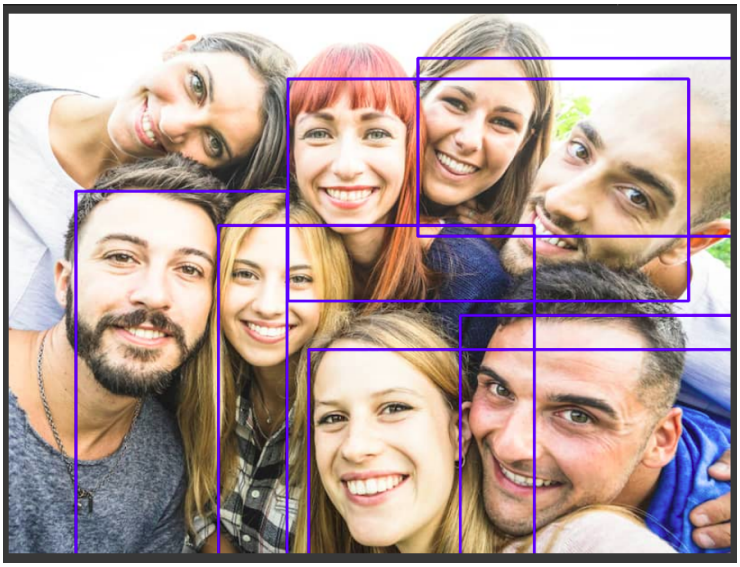
   The best detection models were implemented by using the existing pretrained model available in PyTorch and OpenCV libraries and the results of each of the models was compared by testing on some of the challenging images of faces. The results obtained by using the OpenCV detection model (using haar cascade) are given below:



   The results obtained by using the YOLOv5 model are given below:



The results obtained by using the FastMTCNN model are given below:



➔ As visible from the above results both FastMTCNN and YOLO models work very well and outperform the OpenCV model, although they yield low fps on CPU but by using hardware acceleration through GPUs the inference time can be reduced drastically.

➔ Another important fact to note is that YOLO is a comparatively superior model since its inference speed is much lower than MTCNN with good accuracy so for applications only involving general detection YOLO may appear to be a better choice.

- ➔ On the other hand, MTCNN has a very special application in the field of face detection as it is specially trained to not only detect faces but also detect various features of a face i.e the nose, eyes, and lips, thus proving to perform better in complicated scenarios where there are occlusions in the face image due to masks, sunglasses, etc.

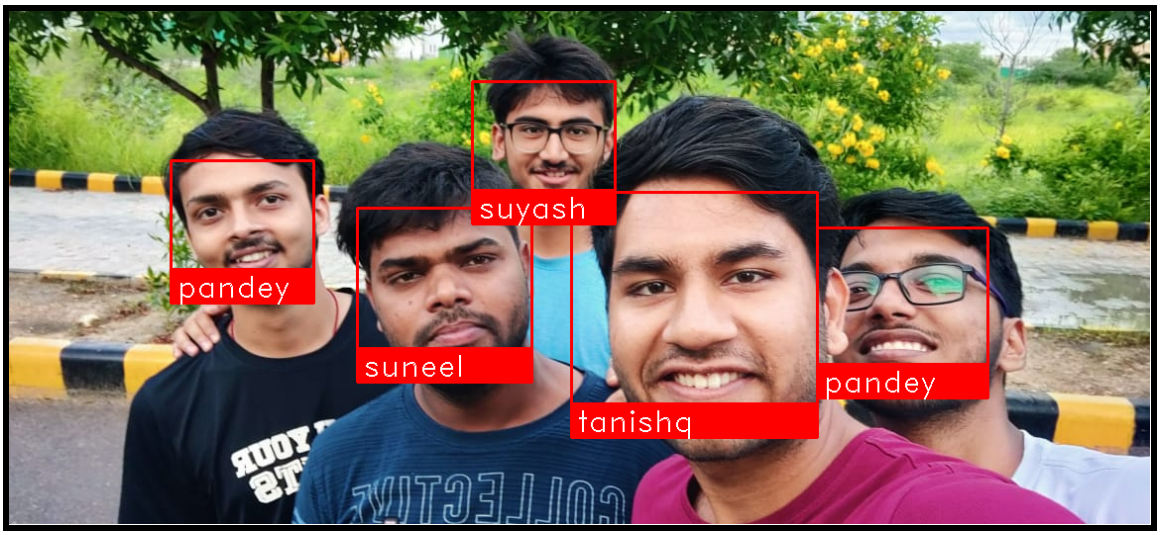
- ➔ Since our application goes much beyond traditional detection and also involves the face recognition part, the MTCNN model proves to be a better choice for us in this case but since we need it for real time applications, a lighter version of MTCNN: FastMTCNN is used.

- ➔ Although we are choosing FastMTCNN in our final implementation, we will consider the performance of the YOLO model too while deciding on the final model as it looks very promising with certain changes.

2. *Implementation of the recognition models:*

   Similar to the detection models, the recognition models were tested separately by providing cropped detected images to the recognition model and testing its accuracy by comparing the image with a set of different labels of the faces of different people in the dataset.

   The results obtained by testing the OpenCV recognition model are as follows:



The results obtained by testing Inception ResNet model for recognition are as follows:

- ➔ From the above-obtained results, it is very evident that the inception ResNet model performs way better than the naive OpenCV model in terms of accuracy even when the number of classes, i.e the number of people has increased significantly, especially in the case of live feed videos in place of images.
- ➔ The only shortcoming of the Inception ResNet model is that it is a very compute-intensive model thus requires a lot of processing and also results in low fps, thus optimizing on that part is required which should be kept in mind while choosing the final end-to-end face detection and recognition model.

III. Final Model Used and its implementation

Currently, the final model used comprises the MTCNN algorithm for face detection and the Inception Resnet model for face recognition.
A popular implementation of the above models is found in the facenet library which makes the pretrained model of the above-discussed AI models available.
The complete implemented pipeline in step by step manner can be summarized as follows:

1. Maintaining an initial database containing images of faces of different people for face recognition.

2. Receiving a live feed from an IP camera or a mobile camera as input for the processing part.

3. Performing face detection through the MTCNN detection algorithm and also maintaining a threshold value for detection.

4. Extracting the individual cropped faces and storing it for it to be compared with the database images.

5. Performing face recognition by comparing the embeddings of the input face images with the embeddings of the images present in the database, the model used to find the embedding values is Inception Resnet in our case.

6. Based on the difference in the embedding values calculated by using the 2-p norm between both the input and the database values, the candidate with the minimum value and with a value below a threshold value gets selected as the person detected.

7. A bounding box is then formed on the continuous live feed based on the output received from the detection algorithm and the name received from the recognition algorithm acts as the label for the same.

The overall flow of the pipeline

IV. Feedback Received and problems with the provided solution

1. Fine Tuning on the different datasets:
   The recognition models might be trained on datasets that are far different in reality than the actual scenario. The recognition models are trained on datasets that are ideal. These might include cases where the faces are clearly visible and are bright and the lighting conditions are also perfect and also the full face of the individual is

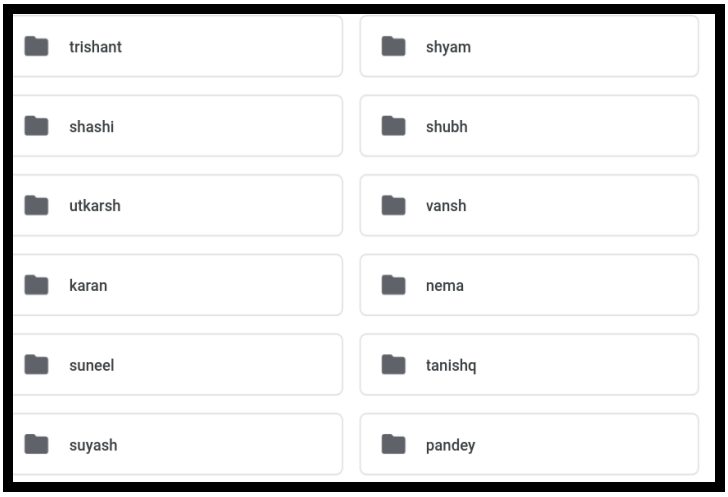
visible. Thus, the model needs to be given a taste of the actual scenario and hence, needs to be trained on the dataset in hand. For this, fine-tuning of the model should be done with the available dataset.

2. Size of the network:
   The model used currently is very heavy to use as it contains a large number of layers and a large number of nodes in each layer. Due to this heavy architecture, the power requirements of this model are significantly high. As the model is to be planted on a chip on a drone, the power consumption by the model should be low else most of the power supplied to the drone from the battery would be used by the model for its running instead of the energy being used for flying the drone. Thus, even though the model gave us good accuracy, its power requirements are pretty high

3. Robustness of the model:
   The present dataset contains pictures clicked in good lighting conditions and the situation is such that the full face of the individual is clearly visible. This does not represent an actual scenario as in real life many times the suspect would be wearing some time of occlusion like a cap or sunglasses, etc. The lighting conditions would also vary drastically in the real scenario. Sometimes the drone might capture the suspect in the dark or some shadow might be coming on the face of the suspect. So these are all the real-world scenarios that should be tackled by the model. Thus, one more shortcoming of the present model is that it does not account for the real-life situation even though it gives good results in ideal cases

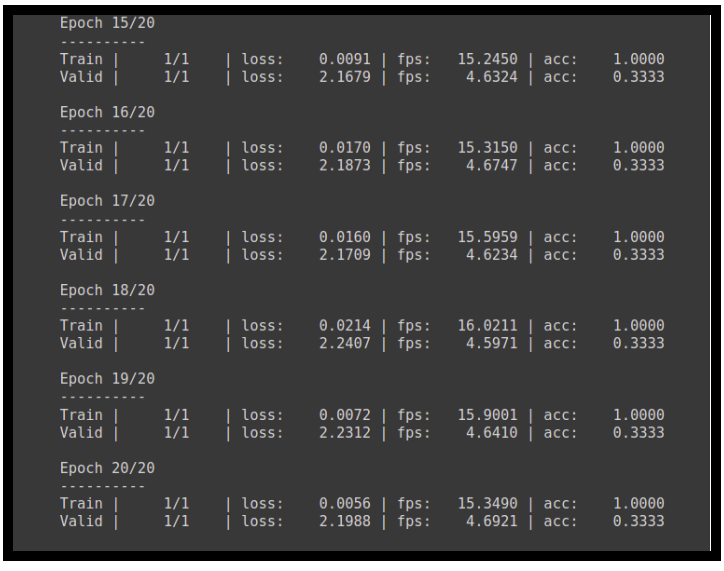
V. Work Done on Feedback Received

1. We created a dataset where every person had 10-15 images added. There were a total of 12-15 people in the dataset. The images that were added were taken in varying lighting conditions and from different altitudes. All the images that were taken depicted real-world scenarios where the drone would actually be working. The creation of this dataset was very helpful in making the model robust and training the model such that it works well in all the different situations that it will face



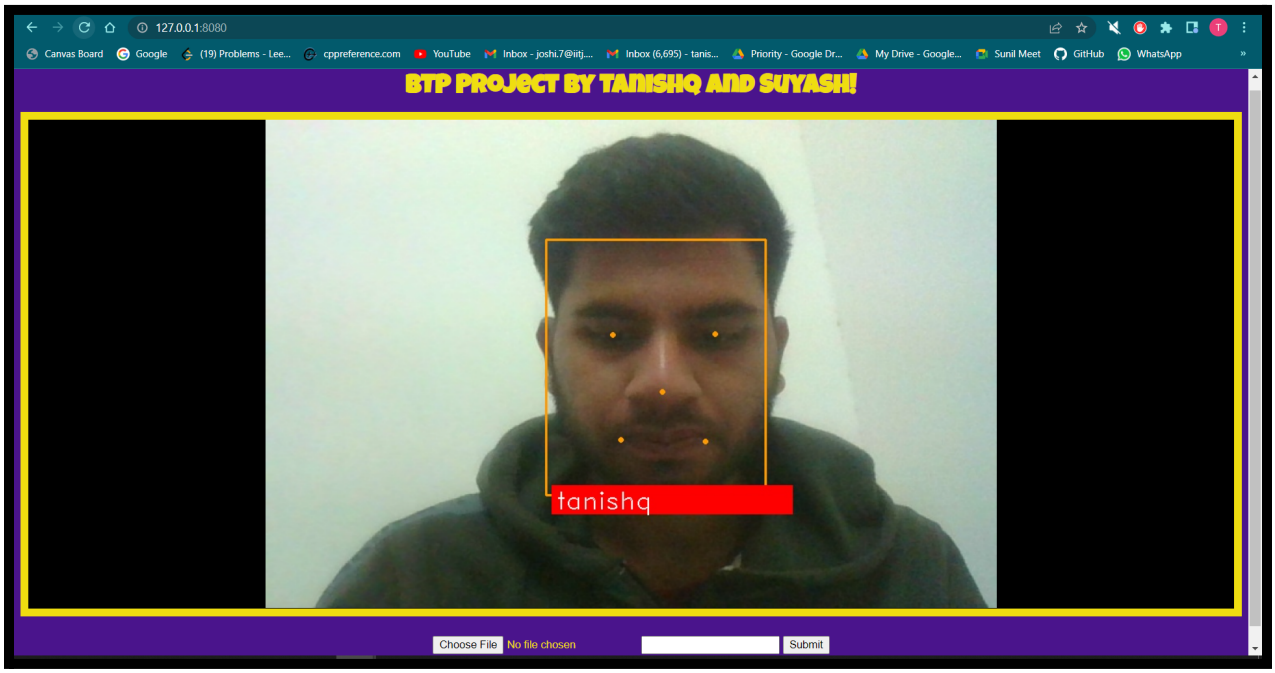
Snapshot of our Dataset

2. We finetuned the model on the dataset that we created. The dataset contains images in varying lighting conditions and from varying altitudes. Fine Tuning the model helped a lot to improve the performance of our system as the model could understand the data at hand even more. This helped to make our model more robust and better performing on the real-world data.

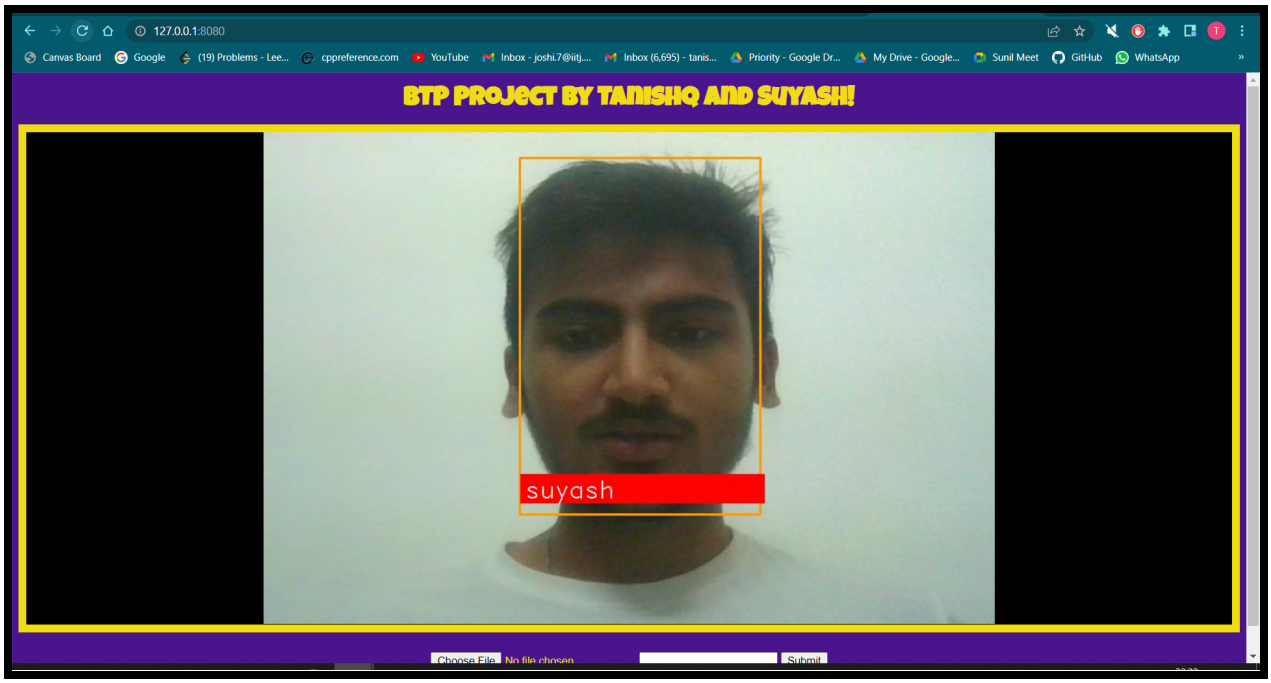


Improving Accuracy after Fine Tuning

3. We created an interface that makes it easier for the common user to add new images to the dataset or to watch the live stream from the camera. It gives all the controls to the admin at one location. The interface contains features to watch the live stream from the drone. Modification of the database can also be done very easily with the help of this interface, the admin can easily add images of individuals from the database. Further work can be done in this interface which is to create a notification system that would alert the concerned authorities if a suspect is detected by the system such that in this case necessary action can be taken



MTCNN + Inception Resnet



Yolo + Inception Resnet

4. The dataset images were primarily in uniform lighting conditions and angles but this might not always be the case, thus a set of different augmentations were applied, like Random Rotation, Gaussian Blur, Random Horizontal Flip, Random Vertical Flip, Color Jitter, Random Crop, Random Invert. It was found that the random rotation combined with gaussian blur performed the best. The results of the model before and after augmentation is given below.

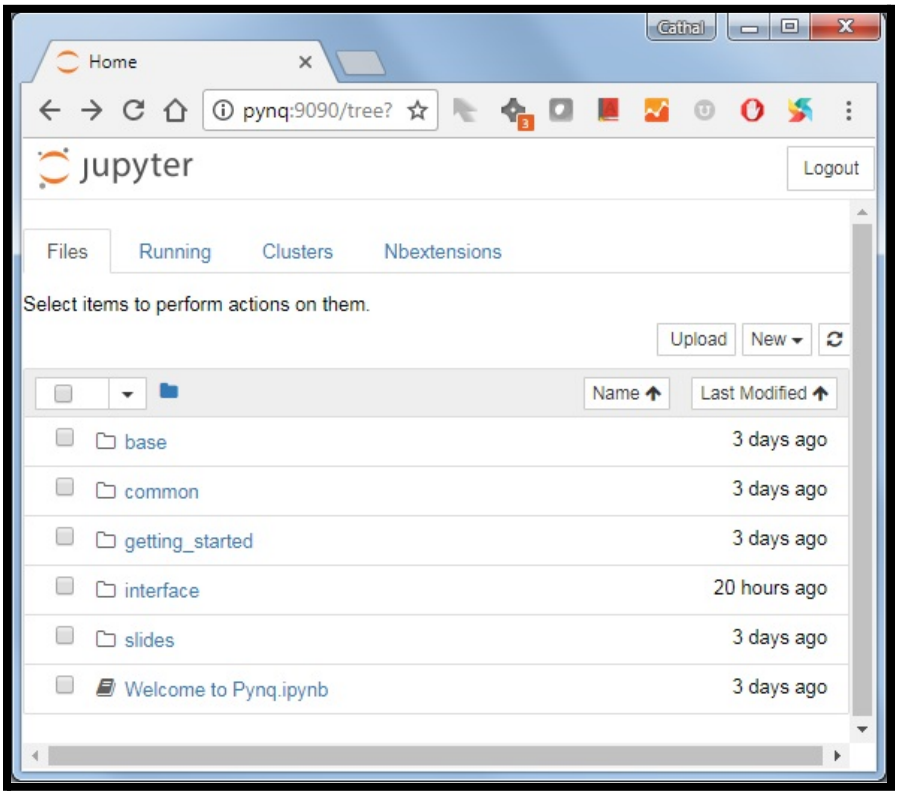
Experiment	Train Set Accuracy	Validation Set Accuracy
Baseline model (no augmentation)	100 %	33.33 %
Baseline Model (with augmentation)	100 %	85.71 %

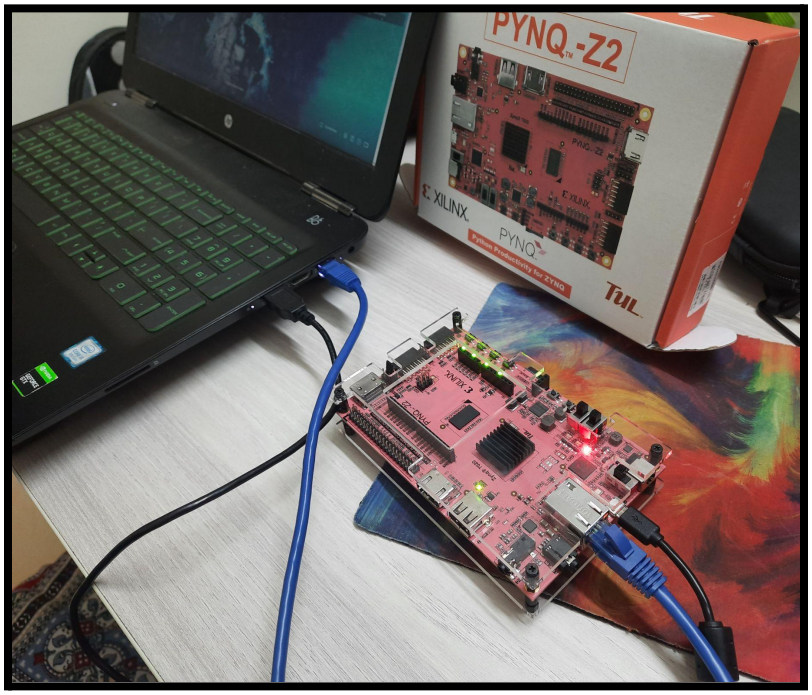
VI. Hardware Implementation

- ➔ The hardware implementation involved the use of the FPGA board i.e the pynqz2 board.
- ➔ The pynq board has python support and thus the software part can be deployed on the board.

➔ The board can then be put on the drone so with connections to the drone camera thus the detection and recognition algorithm can be applied to the drone system in such a fashion.

➔ The jupyter system on the pynq board and the hardware setup is shown in the following images.



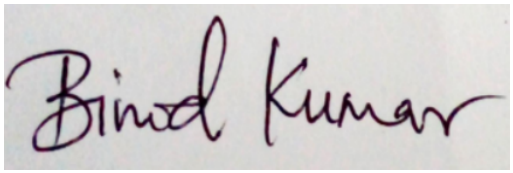


E. Conclusion

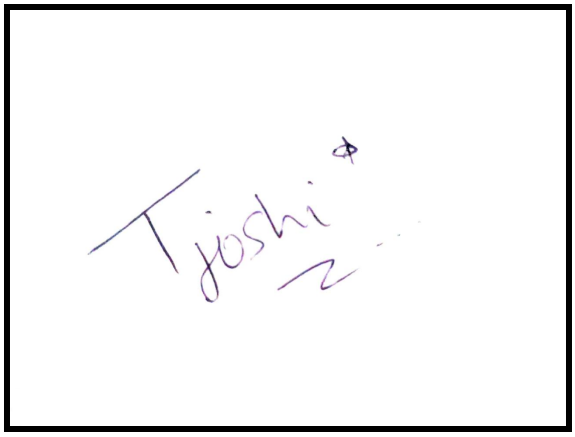
- ➔ We implemented the complete pipeline of face detection and recognition by taking input from the ip camera and detecting using several detection algorithms like yolo and mtcnn and various recognition algorithms like Inception Resnet and face recognition library
- ➔ Our project can be considered as a valuable contribution to tackle the increasing security challenges of the modern world. We have implemented a system which does surveillance in an efficient manner
- ➔ As the real world conditions vary a lot depending on the lighting conditions in which the drone is flying or the quality of the video feed for the drone, we have used various state of the art techniques like data augmentation, creating a dataset which depicts the real world conditions, and finetuning our system on the created dataset so that our system is very robust and can work well in real world scenarios
- ➔ In conclusion, we can say that we have created a state of the art surveillance system using machine learning techniques which can be employed on a drone

F. References

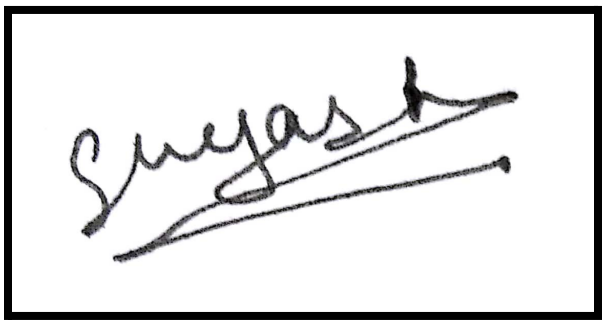
- ➔ Literature Review (different techniques): https://www.engati.com/blog/facial-recognition-systems
- ➔ FaceNet documentation: https://github.com/timesler/facenet-pytorch
- ➔ Mtcnn Documentation : https://github.com/ipazc/mtcnn
- ➔ Yoloface v5 Documentation : https://github.com/elyha7/yoloface
- ➔ Yoloface v3 Documentation: https://pypi.org/project/yoloface/



Signature of Supervisor



Signature of Student 1
(Tanishq Joshi)



Signature of Student 2
(Suyash Singh)