

Detecting Impersonators In Examination Halls Using AI

*A Project Report Submitted in the
Partial Fulfillment of the Requirements
for the Award of the Degree of*

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

A Vishal 17881A0559

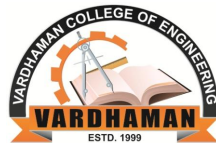
T Nitish Reddy 17881A0520

P Prahasit Reddy 17881A0523

SUPERVISOR

Dr. S Shitharth

Assistant Professor

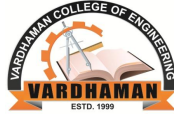


Department of Computer Science and Engineering

VARDHAMAN COLLEGE OF ENGINEERING, HYDERABAD

An Autonomous Institute, Affiliated to JNTUH

May, 2021



VARDHAMAN COLLEGE OF ENGINEERING, HYDERABAD

An Autonomous Institute, Affiliated to JNTUH

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project titled **Detecting Impersonators In Examination Halls Using AI** is carried out by

A Vishal	17881A0559
T Nitish Reddy	17881A0520
P Prahasit Reddy	17881A0523

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** during the year 2020-21.

Signature of the Supervisor
Dr. S Shitharth
Assistant Professor

Signature of the HOD
Dr. Rajanikanth Aluvalu
Professor and Head, CSE

Acknowledgement

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We wish to express our deep sense of gratitude to **Dr. S Shitharth**, Assistant Professor and Project Supervisor, Department of Computer Science and Engineering, Vardhaman College of Engineering, for his able guidance and useful suggestions, which helped us in completing the project in time.

We are particularly thankful to **Dr. Rajanikanth Aluvalu**, the Head of the Department, Department of Computer Science and Engineering, his guidance, intense support and encouragement, which helped us to mould our project into a successful one.

We show gratitude to our honorable Principal **Dr. J.V.R. Ravindra**, for providing all facilities and support.

We avail this opportunity to express our deep sense of gratitude and heartfelt thanks to **Dr. Teegala Vijender Reddy**, Chairman and **Sri Teegala Upender Reddy**, Secretary of VCE, for providing a congenial atmosphere to complete this project successfully.

We also thank all the staff members of Computer Science and Engineering department for their valuable support and generous advice. Finally thanks to all our friends and family members for their continuous support and enthusiastic help.

A Vishal

T Nitish Reddy

P Prahasit Reddy

Abstract

Detecting impersonators in examination halls is very significant to conduct the examination equitably. Recently there were occasions of potential impersonators taking tests instead of the intended person. To untangle this issue, we require an efficacious method with less manpower. With the advancement of machine learning and AI technology, we can overcome this issue. In this project, we are developing an AI system where images of students are saved and we are using the transfer learning process so that the output generated is accurate. If the student is an allowed one, it shows the hall ticket number and name of the student otherwise it appears unknown tag.

Keywords: Face Detection; Face Recognition; Transfer Learning; Convolutional neural network; MobileNetV2; Hyperparameter Tunning

Table of Contents

Title	Page No.
Acknowledgement	i
Abstract	ii
List of Figures	v
Abbreviations	v
CHAPTER 1 Introduction	1
1.1 Problem Statement	1
1.2 Existing System	1
1.3 Observation	2
1.3.1 Proposed Model	2
1.3.2 Advantages of Proposed Model	2
CHAPTER 2 Literature Survey	3
CHAPTER 3 Technology Study	5
3.1 Dataset and Methods	5
3.2 Pre-processing	5
3.3 Image Classification	6
3.4 Face Recognition Using Caffe Model	7
3.5 Algorithms	8
3.5.1 Convolutional Neural Network	8
3.5.2 Transfer Learning	10
3.5.3 Hyperparameter Tuning	10
3.6 Train/Test Split	12
3.7 MobileNetV2	12
3.8 Model Fitting	13
CHAPTER 4 System Design	14
4.1 Process Model	14
4.2 Architectural Diagram	15
4.3 Activity Diagram	16
4.4 Hardware Requirements	17
4.5 Software Requirements	17

CHAPTER 5	Results and Discussion	18
5.1	Validation	18
5.1.1	Training data v/s validation loss	18
5.1.2	Training V/s Value accuracy	19
5.2	Verification	20
5.2.1	Verification of Authorized image:	20
5.2.2	Verification of Unauthorized image:	20
5.2.3	Verification of authorized image through video stream:	21
5.2.4	Verification of Unauthorized image through video stream:	22
5.2.5	Confusion Matrix	22
CHAPTER 6	Conclusions and Future Scope	24
6.1	Conclusion	24
6.2	Future Scope	24
REFERENCES		25

List of Figures

3.1	image count of students	6
3.2	images after applying pre-processing	7
3.3	working of convet	9
3.4	working of transfer learning	10
3.5	Implementation of Hyperparameter Tuning	11
3.6	Implementation of Train/Test Split	12
3.7	Implementation of MobileNetV2	13
4.1	process model	14
4.2	Architectural Diagram	15
4.3	activity diagram	16
5.1	Training data V/s Validation loss	18
5.2	Training V/s Value accuracy	19
5.3	Authorized student after verification procedure	20
5.4	Unthorized student after verification procedure	21
5.5	Verification of known video stream	21
5.6	Verification of unknown person in video stream	22
5.7	confusion matrix	23

Abbreviations

Abbreviation	Description
VCE	Vardhaman College of Engineering
CNN	Convolution Neural Network
AI	Artificial Intelligence
RELU	Rectified Linear Unit

CHAPTER 1

Introduction

1.1 Problem Statement

As we all know, India is a developing country and the youth of the nation plays a crucial role in the country's development, so it is very important to conduct exams fairly and honestly. Examinations are conducted to measure the knowledge of a student and the result of an examination plays an important role in a student's professional career. So, the examinations should be conducted fairly. Conducting an examination is a hectic task for an educational institute from distributing the hall tickets to verifying them at the time of examination. In real-time, manual checks of hall tickets happens to verify the student is certified or an impersonator. Manual checks need more human power and we cannot get maximum accuracy as there have been instances of students who cheated exams and are in a very crucial stage where a minor mistake can be costly. To avoid this we have proposed an AI system that detects impersonators in examination halls.

1.2 Existing System

In the Existing system, students are identified by the details present in their hall ticket, whether the student is the intended person or not. Sometimes because of being late to the examination hall, students rush into the test centers or test room so, identifying each of them becomes difficult. Hence, many of them enter the test room without being authenticated so, there might be more chances of impersonators entering the examination hall without being verified. There were instances of hall ticket morphing where manual checks could not play a vital role, which leads to compromise the purpose of conducting an examination honestly. Recently, there were cases of students

who did not get caught during the manual authentication of the hall tickets. So there is a need for a system to be implemented to untangle this issue.

1.3 Observation

1.3.1 Proposed Model

To overcome the problem faced in the existing model, we propose an AI system for detecting impersonators in the examination hall. In the proposed system first, images of each student are collected by using the haarcascade-frontalface classifier which is used to identify the face of the students, and each student is given a unique value at the time of the creation of hall tickets. Each dataset comprises 100 images of every student, and then we use image augmentation to increase the count of images so that the accuracy of the model increases. We use the transfer learning concept in which we take a pre-trained model which is already trained with large datasets. In our system, we use the mobilenet-v2 model(which is a pre-trained model which has been trained for millions of datasets) and train it with our dataset by using CNN, and the model is saved in the system.

Whenever a student sets foot in the classroom, the student should look at the camera and enter class, thereafter the faces of the students are recognized by our model, and later it will compare the faces of the student with the previously trained data. The result is produced in the form of the name and hall ticket number if the face matches with the dataset or else it shows unknown. As we know the entry time of the examination is around 30 minutes so our model runs for 30 minutes and we record this video and save it for further verification.

1.3.2 Advantages of Proposed Model

We can ensure that examinations are conducted fairly with less manpower required to verify hall tickets at the entry to the examination hall if we use our methodology. We can add other techniques in the future, such as setting off an alarm if an impostor is found, ensuring immediate action.

CHAPTER 2

Literature Survey

In Early 1994 [1]. Have implemented neural networks for detecting faces. He considered forming CNN to detect whether there is a face in the image window or not and explore the entire image at all sites. In the year 2002, Garcia et al [2] have developed a system to detect semi-frontal faces of the human being. Their system can also detect the face of the person in a fluctuating image in any environment.

In this study, the creators (Prof. P Y Kumbhar and Shubham Dhere, 2017) [3] utilize Raspberry Pi, Haar Filter, and OpenCV for continuous face identification. The classifier utilized here is a classifier for Haar. By checking the head positions, SimpleCV and OpenCV are utilized for face identification. The AdaBoost calculation is utilized as a calculation for face discovery. The progression of this technique for face location goes this way: Starting with the webcam, Capturing and putting away pictures in the database, recognizing the face and confirming the face personality from the current data set, distinguishing the face, preparing the distinguished face picture that is an extraction of the element, discovering the match of the face in the current data set, showing the comparing result if the display ID and confirmation have not been recognized. A video outline that runs in a circle that gathers the pictures from the camera and those pictures are put away in transitory capacity, at that point utilized for grouping and identification, is utilized in the execution of this strategy (Prof. P Y Kumbhar, 2017).

Face recognition has a large body of literature, which is spread across numerous domains. In video, face recognition necessitates face tracking, which has multiple dimensions, and face recognition has issues. We'll compare and contrast these various head positions. It should be installed in a testing environment that can recognise human faces. Extract the shape, histogram, texture, color and different options from the face . so as to attain correct

implementation, matching should eliminate communication noise, background pictures and other factors through the employment of image preprocessing . External value detection could be a technology that solves the matter of finding patterns that don't correspond to expected traditional behavior within the data . It takes longer to notice fraud, that isn't enough, that the computer file is video, however solely images. Identity recognition includes a transparent comparison of a person' face with all the folks in the information to ascertain identity and verification, and its feature is that the correspondence between the face and also the face hold on in the database .

CHAPTER 3

Technology Study

3.1 Dataset and Methods

One of the major challenges in the creation of a dataset is the collection of student images. As a result, we captured the images through our laptop's webcam. The combination of these pictures forms a dataset for a face recognition system. The data preparation stage includes data collecting, filtering, and pre-processing the data to organize it for modeling as needed. This includes all the steps needed to prepare the final data set, such as normalization, collection and transformation of attributes, classification, and so forth.

Data preparation involves capturing the images of students at different angles. If a student wears spectacles, then his images are taken with and without spectacles. Our dataset consists of 1000 images of each student taken in various angles. This is accomplished by using the haarcascade-frontalface classifier and OpenCV libraries.

3.2 Pre-processing

Pre-processing is the first step to be applied to the image data after the images have been taken. It is applied to improve the image data and by reducing image distortions to process the data accurately.

- Resizing: Input images are resized to 224*224 pixels as our models require this input shape.
- Convert to arrays: Each image is converted to numpy array with a shape of [224,224,3], wherein each pixel value ranges from 0 to 255.
- Scaling: As pixel values are ranging from 0 to 255, the data is scaled in the range of 0 to 1.

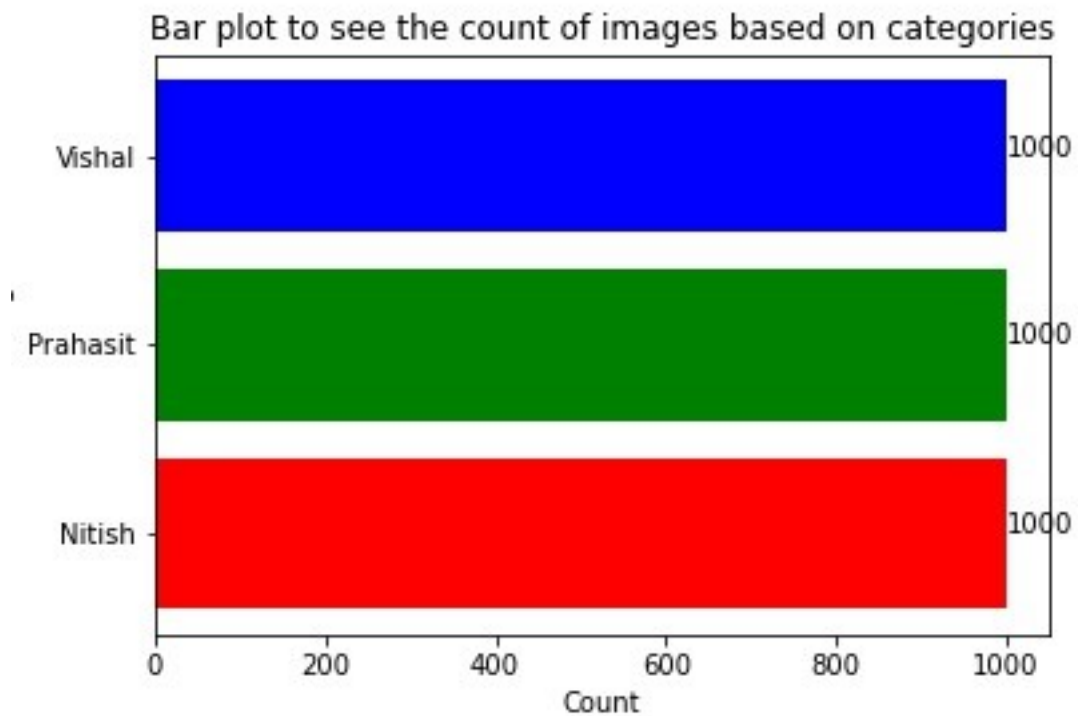


Figure 3.1: image count of students

- Labelling: Based on the folder images, output label is assigned as ‘Nitish’: 0, ‘Prahasit’: 1, and ‘Vishal’: 2.
- to-categorical: Converting the output array of labelled data (from 0 to number-of-classes-1) to one-hot vector.

3.3 Image Classification

MobileNets is one of the best architectures for low-power machines. MobileNets could be a neural network architecture, the first-generation development product MobileNetV1. MobileNetV2 uses 2 blocks: residual and downsizing. There are 3 layers of those two block forms. The primary layer uses the trigger to operate ReLU6 for 1x1 convolution, the second layer uses deep convolution and ReLU for the second layer, and the last layer uses a linear trigger function to form a 1x1 convolution layer. MobileNetV2 will improve the potency of process load and shorten the iteration time compared to MobileNetV2.

MobileNetV2 is a design that uses pre-trained models to represent neural



Figure 3.2: images after applying pre-processing networks. Some style of an artificial neural network already has its parameters and may add layers, or is typically known as fine-tuning. MobileNetV2 artificial neural network layer isn't subjected by value changes throughout the execution during the training process, the fine-tuning layer is carried out by adjusting the values in line to what the values are.

3.4 Face Recognition Using Caffe Model

The Caffe Model is used to obtain image characteristics. Using previously qualified instruction, these features are obtained. Some of the skills used include recognition of objects, learning an image's semantic attributes, and identification of objects. A reference model for solving visual problems is given by Caffe.

In this analysis, the Caffe Model is used in the image generated by the input device to perform a face selection. This needs to be achieved in order to generate the desired output of the prediction process that has been carried out at the training data level. In order to use the Caffe Model, several files required by OpenCV must be used: the .prototxt extension file containing the neural network configuration, and the *.caffemodel file containing the

previously trained model's weight values.

3.5 Algorithms

3.5.1 Convolutional Neural Network

[4] In our brain, we have four parts where each part does a specific work. One of them is the Cerebral cortex which consists of the visual cortex which has several layers that work on images. Each layer performs various functions like identifying the outlines of the objects, detecting the objects from the image, and so on.[5] At last, the visual cortex will summarize the total image and send the response to other parts of the brain.

Incorporating the above information into a machine is done by Computer Neural Network. CNN is one of the deep learning techniques where input images are moved through a sequence of layers known as convnets. Every layer transforms one volume to another volume through a differentiable function. Convnet consists of the following layers:

- **Input layer:** It contains raw images in the form of an image array. Here we take an image array of size $224 \times 224 \times 3$.
- **Convolutional layer with filters:** The image array is used as an input in the convolution layer, and we apply certain filters [horizontal or vertical filters] to it to produce an output array of the size $(n-f+1) \times (n-f+1) \times 3$, where 'n' is the size of the image array (here we used $224 \times 224 \times 3$), and f is the size of the filter [it depends on the size of the input array]. We employ padding [where we add extra columns on all sides and the formula changes to $(n+2p-f+1) \times (n+2p-f+1) \times 3$ where p is the extra columns that we add on one side] to produce the same size output array. This layer is mainly used to find the sharp edges in the image array(face outline).
- **Activation function layer:** The output of the convolutional layer is taken as input and we will apply element-wise activation function and it is used to activate or deactivate a neuron i.e pixel. We took Rectified

Linear Unit(ReLU) as an activation function because it does not activate all the neurons at a time. This layer will activate the neurons detected in the convolutional layer i.e the sharp edges and deactivate the neurons which are not detected in the convolutional layer.

- **Pooling layer:** This layer reduces the size of volume hence, making the computation fast by reducing memory and prevents overfitting. In this layer, we will increase the intensity of the sharp edges by taking the average pooling of size (7,7).
- **Fully connected layer:** It takes input from the previous layer and it computes class score and it flattens it into a one-dimensional array whose size is equal to the number of classes taken. Later, we apply a Softmax classifier to classify the object with values 0 or 1.

In real-time, there will be a camera placed at the entrance of every examination hall. Whenever students enter the classroom, our model will detect the face of the student through the camera, and it will compare with the dataset. Then the output value is generated if it is a known value, it displays the name and hall ticket number of the student, or else an unknown tag is displayed.

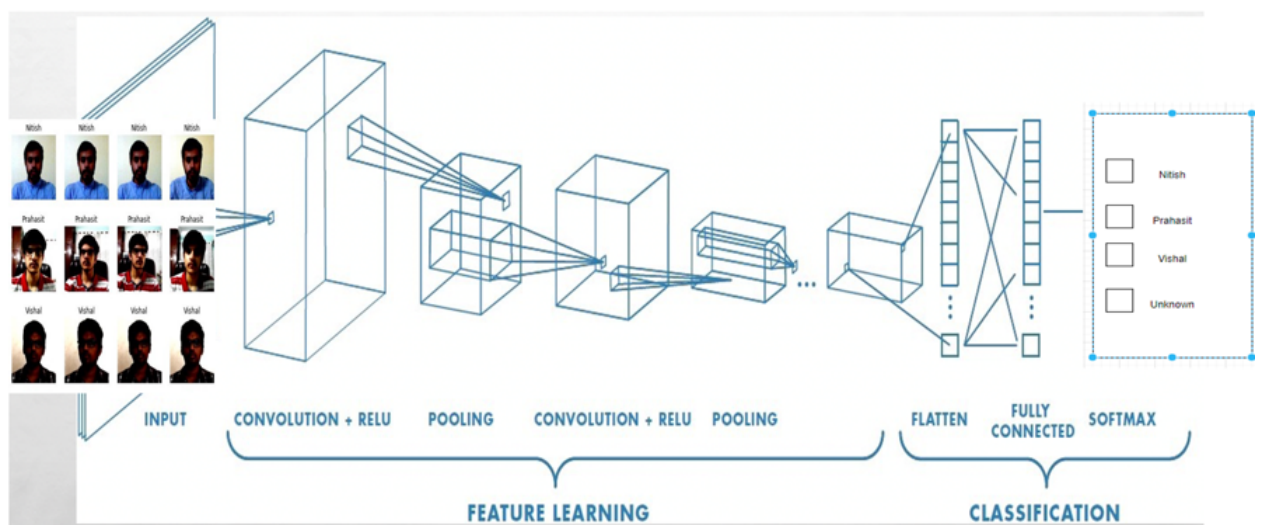


Figure 3.3: working of convet

3.5.2 Transfer Learning

Transfer learning is a method where a model is trained for one particular task and is used for other similar tasks as a starting point.[6] It is a process that uses a pre-trained model that was trained for a dataset and is used in predicting another dataset. The advantage of using Transfer learning is that it reduces the training time of a model as well as the errors associated with the prediction.

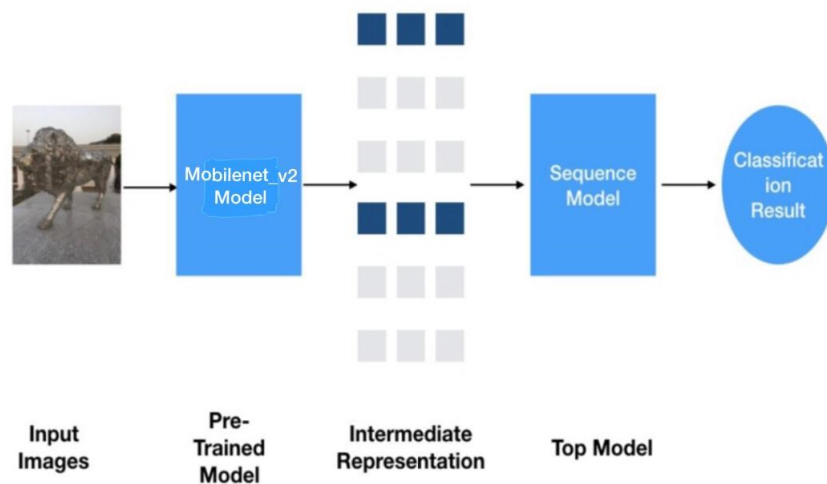


Figure 3.4: working of transfer learning

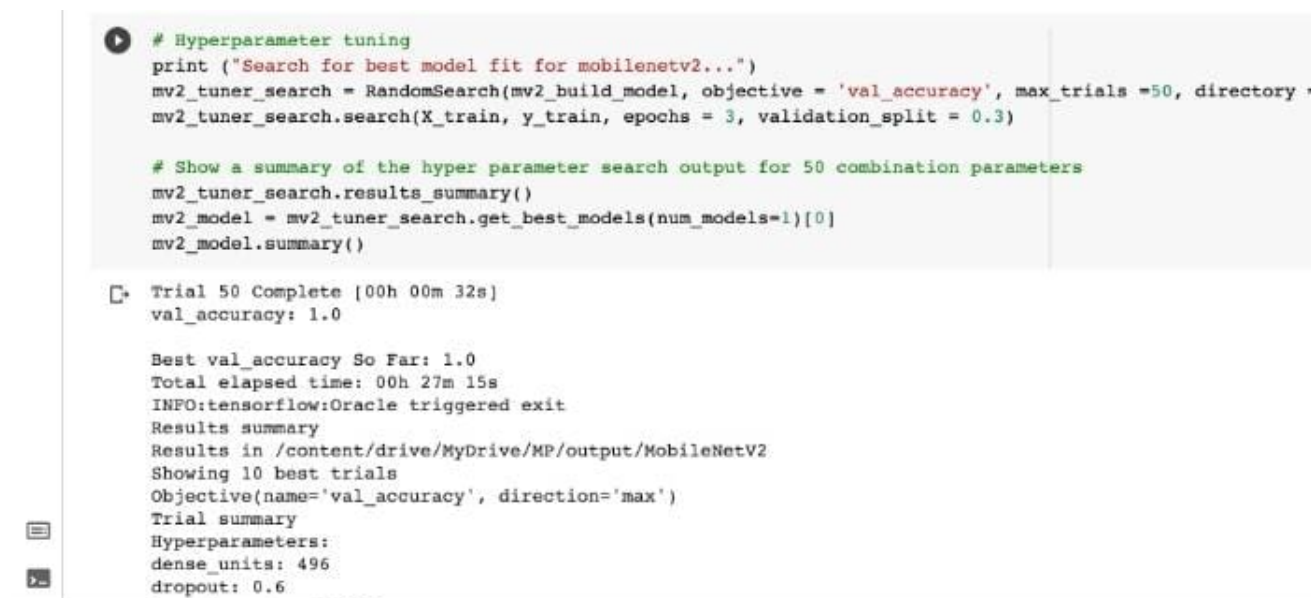
3.5.3 Hyperparameter Tuning

Keras-tuner, which is a meta-optimization assignment, is used to tune hyperparameters. For each trial, a model and an internal optimization algorithm are trained for a specific hyperparameter environment.[7] Hyperparameter tweaking produces the best hyperparameter setting, while model training produces the best model parameter setting. The set is produced by the hyperparameter tuner.

Parameters:

- Fully connected layer - neurons (128 to 1024).
- Dropout layer (0.4 to 0.8) count.
- Learning Rate (0.0001 to 0.01).

We used the “imagenet” dataset to train our model, and the weights assigned to them are also from the “imagenet” dataset. At the fully connected layer, we used the ‘relu’ activation function to avoid vanishing gradients and adjust weights in backpropagation, and at the output layer, we used the ‘softmax’ activation function to solve multi-label classification issues.[8] Furthermore, we chose the Adam optimizer over the Gradient Descent or Stochastic Gradient Descent (SGD) optimizers because of its momentum feature, which helps to reduce noise while converging to global minima, and its RMS prop capability in a controlled change of learning rate, which aids in faster convergence of loss value to global minima. Dropout values are set between 0.1 and 0.8 based on several research articles, with 0.1 to 0.8 being considered a reasonable range for avoiding model overfitting. While tuning and loss are monitored using ‘categorical cross-entropy,’ the learning rate is chosen from [0.0001, 0.0005, 0.001, 0.005, 0.01] values. We tuned with max trials = 50, which means we tried 50 distinct parameter combinations for three epochs each and then chose the best model based on ‘validation accuracy.’ Later, we trained the best model chosen after tuning for 50 epochs on the complete dataset (training and validation), halting early based on the ‘validation loss’.



```
# Hyperparameter tuning
print ("Search for best model fit for mobilenetv2...")
mv2_tuner_search = RandomSearch(mv2_build_model, objective = 'val_accuracy', max_trials =50, directory = 'hyperparameter_tuning')
mv2_tuner_search.search(X_train, y_train, epochs = 3, validation_split = 0.3)

# Show a summary of the hyper parameter search output for 50 combination parameters
mv2_tuner_search.results_summary()
mv2_model = mv2_tuner_search.get_best_models(num_models=1)[0]
mv2_model.summary()
```

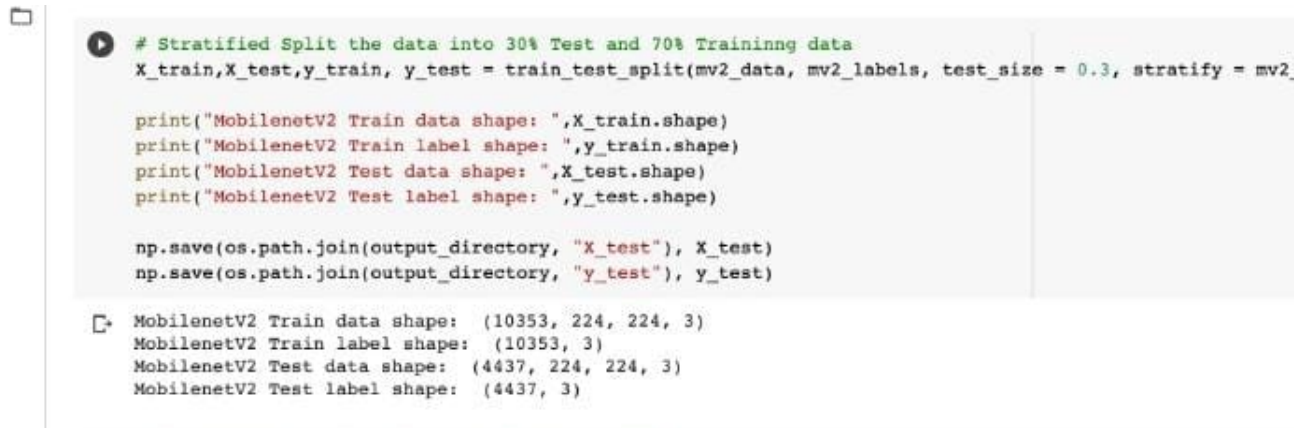
```
Trial 50 Complete [00h 00m 32s]
val_accuracy: 1.0

Best val_accuracy So Far: 1.0
Total elapsed time: 00h 27m 15s
INFO:tensorflow:Oracle triggered exit
Results summary
Results in /content/drive/MyDrive/MP/output/MobileNetV2
Showing 10 best trials
Objective(name='val_accuracy', direction='max')
Trial summary
Hyperparameters:
dense_units: 496
dropout: 0.6
```

Figure 3.5: Implementation of Hyperparameter Tuning

3.6 Train/Test Split

In our model, the dataset is divided into a training dataset and test dataset in the ratio 70 percent and 30 percent respectively.



```
# Stratified Split the data into 30% Test and 70% Traininng data
X_train,X_test,y_train, y_test = train_test_split(mv2_data, mv2_labels, test_size = 0.3, stratify = mv2_labels)

print("MobilenetV2 Train data shape: ",X_train.shape)
print("MobilenetV2 Train label shape: ",y_train.shape)
print("MobilenetV2 Test data shape: ",X_test.shape)
print("MobilenetV2 Test label shape: ",y_test.shape)

np.save(os.path.join(output_directory, "X_test"), X_test)
np.save(os.path.join(output_directory, "y_test"), y_test)
```

```
MobilenetV2 Train data shape: (10353, 224, 224, 3)
MobilenetV2 Train label shape: (10353, 3)
MobilenetV2 Test data shape: (4437, 224, 224, 3)
MobilenetV2 Test label shape: (4437, 3)
```

Figure 3.6: Implementation of Train/Test Split

[9]

3.7 MobileNetV2

MobileNetV2 is one of the pre-trained models for image classification trained on the Imagenet dataset. MobileNetV2 is extracted from keras.applications library. It is built on an inverted residual structure, with residual connections between bottleneck levels. As a source of non-linearity, the intermediate expansion layer employs lightweight depth-wise convolutions to filter features. In our system, we will first remove the last layer of MobileNetv2 which consists of thousands of output. Later we will train the model with an imagenet dataset for the last time and then we will follow the following steps. The input is sent to the base model with average pooling size (7,7), then Flatten, then to the Dense layer with a size of 128 and the ReLU activation function, then the arrays are dropped by 0.4 and sent to the Dense layer of 2 with the SoftMax activation function (MobileNetV2 — Light Weight Model (Image Classification), 2019). Later as we know that the layers in MobileNetV2 is already trained so we need to make them as not trainable as shown in fig.3.7. and we will add our layers [which depends on the number of categories in

the dataset]. Addition of new layers is done by Sequential Convolution Layer and we will compile the model using adam optimizer[wit handles sparse and noisy values and optimizes our system].

```
# Build Hyper Tunning model for MobileNetV2
def mv2_build_model(hp):
    baseModel = MobileNetV2(weights="imagenet", include_top=False, input_tensor=Input(shape=(224, 224, 3)))
    headModel = baseModel.output
    headModel = keras.layers.AveragePooling2D(pool_size=(7, 7))(headModel)
    headModel = keras.layers.Flatten()(headModel)
    headModel = keras.layers.Dense(units = hp.Int(name = 'dense_units', min_value=64, max_value=512, step=16), activation = 'relu')(headModel)
    headModel = keras.layers.Dropout(hp.Float(name = 'dropout', min_value = 0.1, max_value = 0.8, step=0.1, default=0.5))(headModel)
    headModel = keras.layers.Dense(noc, activation = 'softmax')(headModel)
    model = Model(inputs=baseModel.input, outputs=headModel)
    # loop over all layers in the base model and freeze them so they will
    # not be updated during the first training process
    for layer in baseModel.layers:
        layer.trainable = False
    model.compile(optimizer = keras.optimizers.Adam(hp.Choice('learning_rate', values = [5e-2, 1e-2, 5e-3, 1e-3, 5e-4])),
                  loss = 'categorical_crossentropy',
                  metrics = ['accuracy'])
    return model
```

Figure 3.7: Implementation of MobileNetV2

3.8 Model Fitting

We use various types of pre-processing in this project, including augmentation, which seeks to prevent overfitting so that the software can still generate accurate predictions if it encounters an issue with micro differences.

Augmentation is the practice of using training data sets to increase variability. The training set and the testing set are made up of pre-processed mask data with no photographs of the mask form. The training set is the part of the dataset where the training and testing set classification models are used, and where the trained model is tested and evaluated to determine the accuracy of the dataset to predict the appropriate face form.

Several augmentation factors have been added to this project, including rotation, flip, and picture alteration from the initial stage (range shift). These augmentation parameters are essential since the changes in picture capture utilizing the camera are so varied: face tilt, camera flip direction, and images that shift at a location other than a certain spot.

Random Search describes a search space as a bounded domain of hyper-parameter values in which points are randomly sampled to discover the best model from a set of training parameters.

CHAPTER 4

System Design

4.1 Process Model

In this section, we are going to discuss the procedure that we followed in developing our model. The following process is followed to design our system. Images of students are taken a few days before the examinations to be utilized for later authentication. After that, we use our model, which is a facial recognition model, to train the data. Subsequently, the trained model is then saved to our system. While students are entering the classroom, photos of them are taken in real-time and utilized to forecast the student's names and accompanying hall ticket numbers, which are subsequently delivered to the system dashboard. The video file is then saved in the system for future use. If a student is a legitimate candidate, his name and hall ticket number will be displayed and if the student is unapproved, an unknown tag will be displayed.

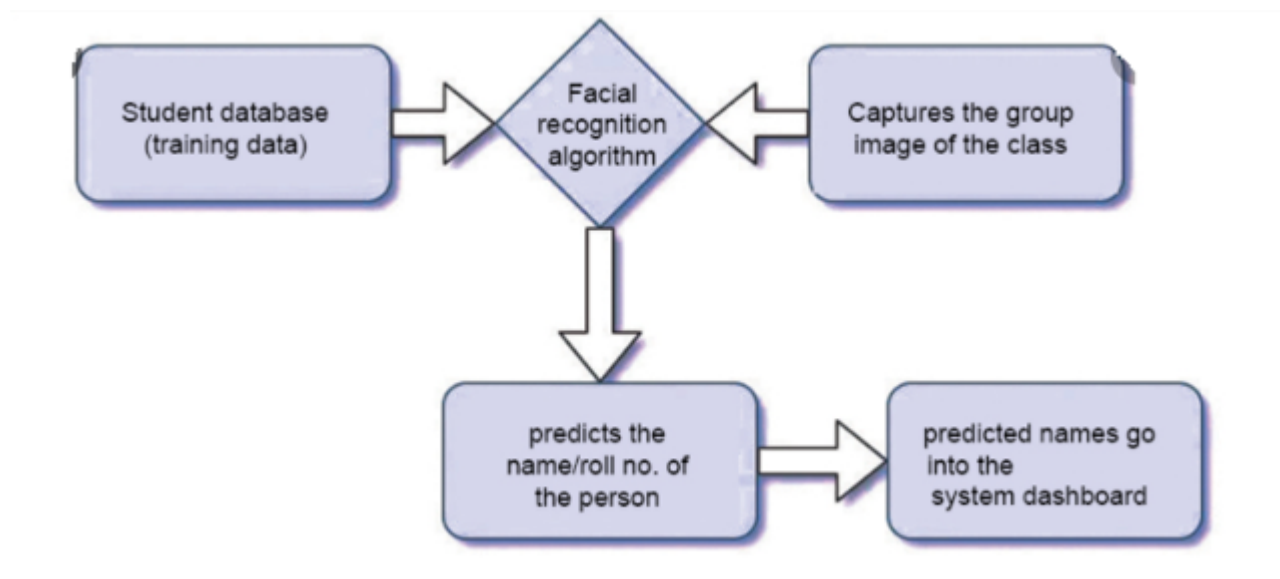


Figure 4.1: process model

4.2 Architectural Diagram

Our system is built in two stages: first, we train the model, and then we apply it to real-world use cases. So, Architecture Diagram consists of two main steps, namely training the model and applying that trained model in the real world.

1. Training the Model:

The first step in Training is to import our dataset of student images. We gathered images here with the help of the OpenCV library and image augmentation. Later, the image numbers are increased by data augmentation techniques such as scaling, rotating, and shuffling the images. Using transfer learning, we then take the MobileNetV2 pre-trained model, which has been trained on imagenet datasets, and train it on our dataset. Finally, the model is saved to the System. The trained model is then used.

2. Applying the model:

In this step, we will first load the model from the system. Later, in real-world scenarios, we use the Caffe model to detect faces in video streams. We then send these images to our model, which classifies the faces and displays the results.

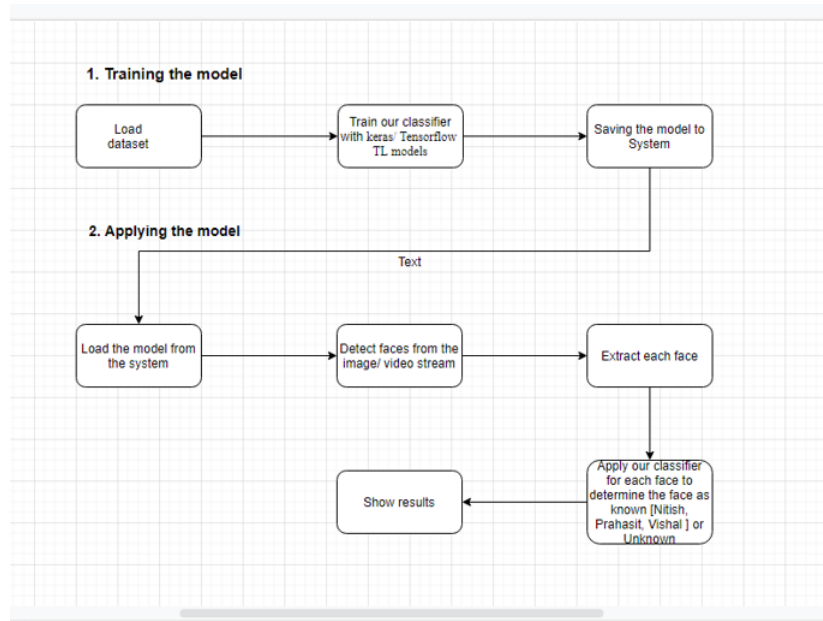


Figure 4.2: Architectural Diagram

4.3 Activity Diagram

The first step is the acquisition of student images. Next, data pre-processing is applied to remove any distortions in the image to make the data processed in a better way. Then we acquire the MobileNetV2 model which is a pre-trained model which has been trained for a large number of similar datasets. We then train our model with MobileNetV2 model. After that, we tune the model using a Convolutional Neural Network. Next, we validate the data. Following this, we get the results, if a student is recognized then, his name and his hall ticket number are showed and if a student is not recognized then an unknown tag is showed.

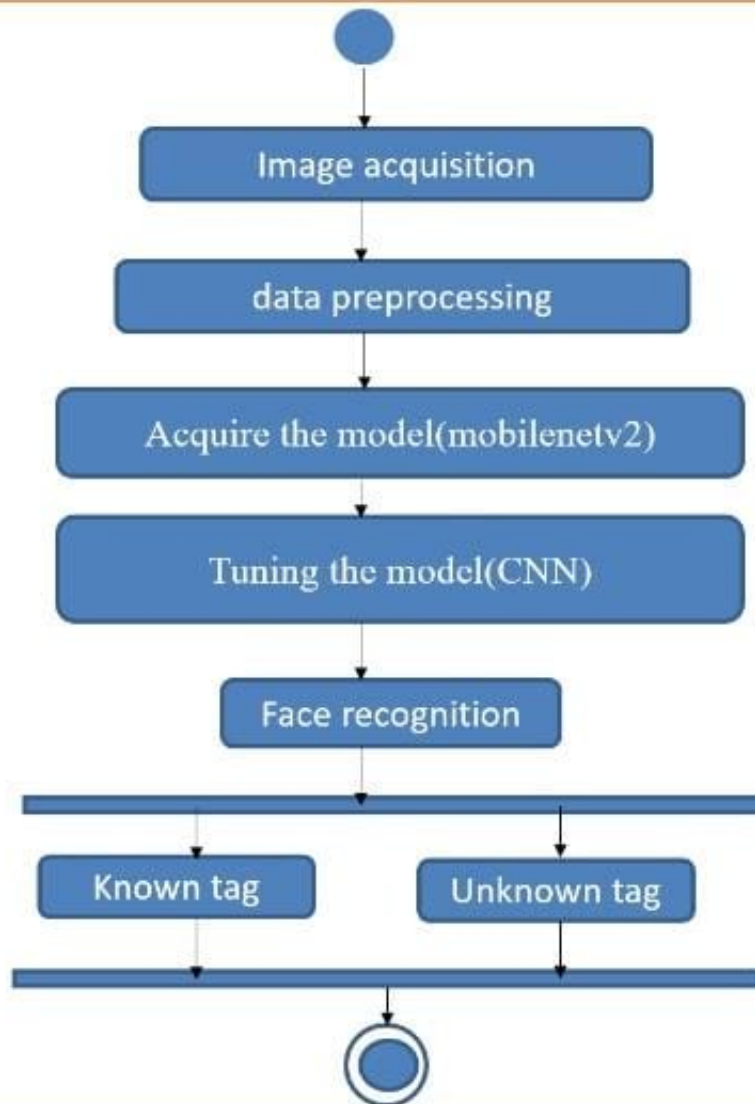


Figure 4.3: activity diagram

4.4 Hardware Requirements

These are the minimum hardware requirements for our system to function properly.

- Operating System: Windows, MAC, Linux
- Hard Disk: 500 GB
- Mouse: Logitech
- RAM: 2 GB
- Processor: Intel core I3

4.5 Software Requirements

The following are the software requirements for our system to function correctly.

- We created our model using the Python programming language.
- To develop our model, we used the Anaconda platform, which is one of the best platforms for data science.
- We used several different libraries, including OpenCV, Keras, pickle, matplotlib, NumPy, pandas, Keras-tuner, and sklearn.
 - OpenCv is a Computer Vision library which deals with images.
 - Keras is a deep learning library that includes pre-trained models such as MobileNetV2, algorithms such as Sequential CNN, and other approaches for model development.
 - pickle library is used to save and load data.
 - Numpy libraries handle mathematical operations such as arrays, Pandas handle DataFrames, and Mathplotlib handles plots.
 - keras-tuner is used for hyper-parameter tuning
- While creating the hall ticket, a data set is generated.

CHAPTER 5

Results and Discussion

5.1 Validation

The proposed approach has been evaluated by measuring the precision, recall, and accuracy metrics of the face detection model and our classifier for MobileNetV2.

5.1.1 Training data v/s validation loss

Loss is the result of a poor forecast. Its value reflects a model's performance. Our model suffered a loss of '3.2509e-08' which is a tiny value. The blue line in the figure 5.1 indicates training loss, whereas the yellow line depicts validation loss. As shown in figure 5.1, the difference between validating loss and training loss is minimal, indicating that our model has been properly trained. The training loss was initially 0.001, but as the training progressed, it was reduced to 0. Later on, the training loss rose slightly, which could be due to image quality or an image change. Finally, because it has already been trained on some data, the training has been reduced to 0.

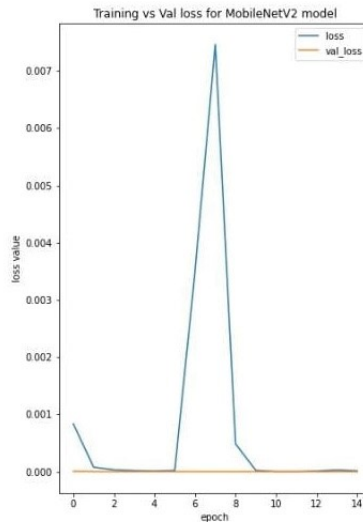


Figure 5.1: Training data V/s Validation loss

5.1.2 Training V/s Value accuracy

Accuracy is one of the measures to calculate the algorithm's performance. In the figure 5.2, the blue line shows the training accuracy whereas, the yellow line shows the validation accuracy. Our model got '1.0' accuracy. As shown in the figure 5.3, as there is an increase in the number of epochs, there was an increase in the accuracy of the model from 0.998 to 1. But, at a certain point, accuracy has decreased a bit. Later, the accuracy has improved. However, the difference between training accuracy and validating accuracy is very less hence, our model is not overfitted or underfitted.

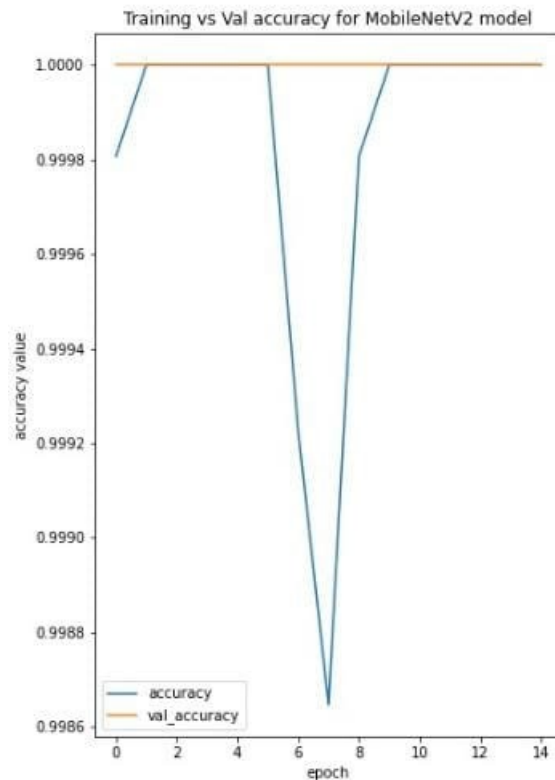


Figure 5.2: Training V/s Value accuracy

5.2 Verification

Verification can be done in two ways: directly with photos or through a video stream. For verification, we sent the student's photographs to the model, which verifies them.

5.2.1 Verification of Authorized image:

We gave the student's photograph that are present in the data set to the model, and we were correct in our prediction.

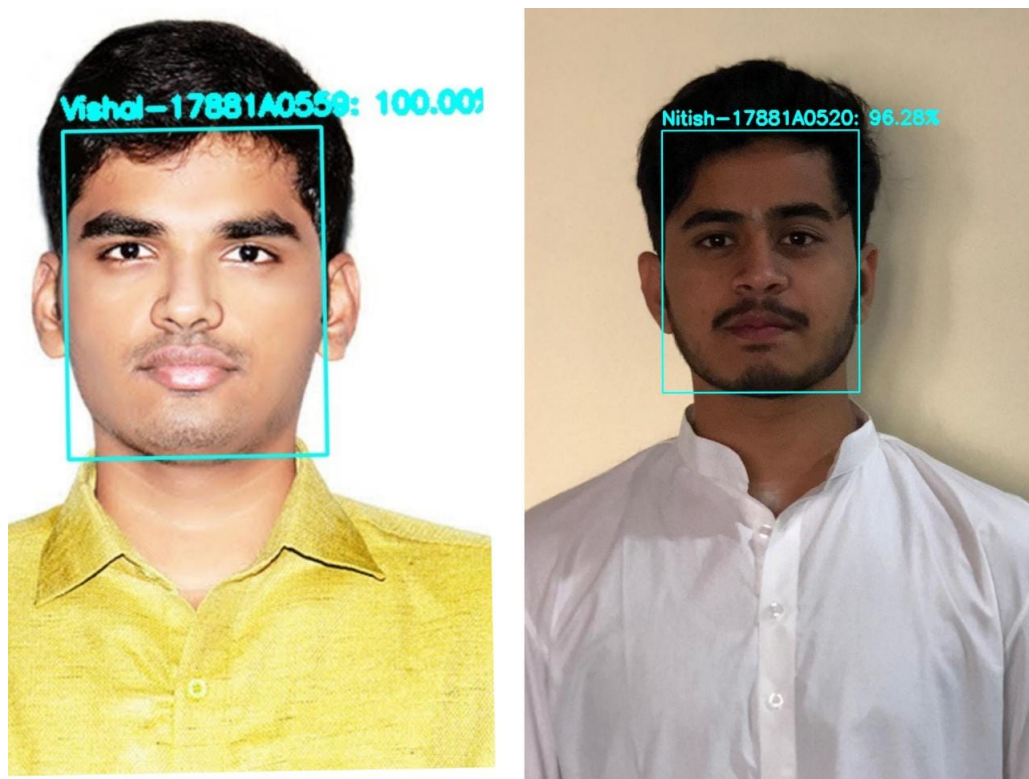


Figure 5.3: Authorized student after verification procedure

5.2.2 Verification of Unauthorized image:

We gave the student's photograph that are absent in the data set to the model, and we were correct in our prediction.



Figure 5.4: Unauthorized student after verification procedure

5.2.3 Verification of authorized image through video stream:

In verification, we have taken input from the video stream and for the authorized students our model predicted correctly.

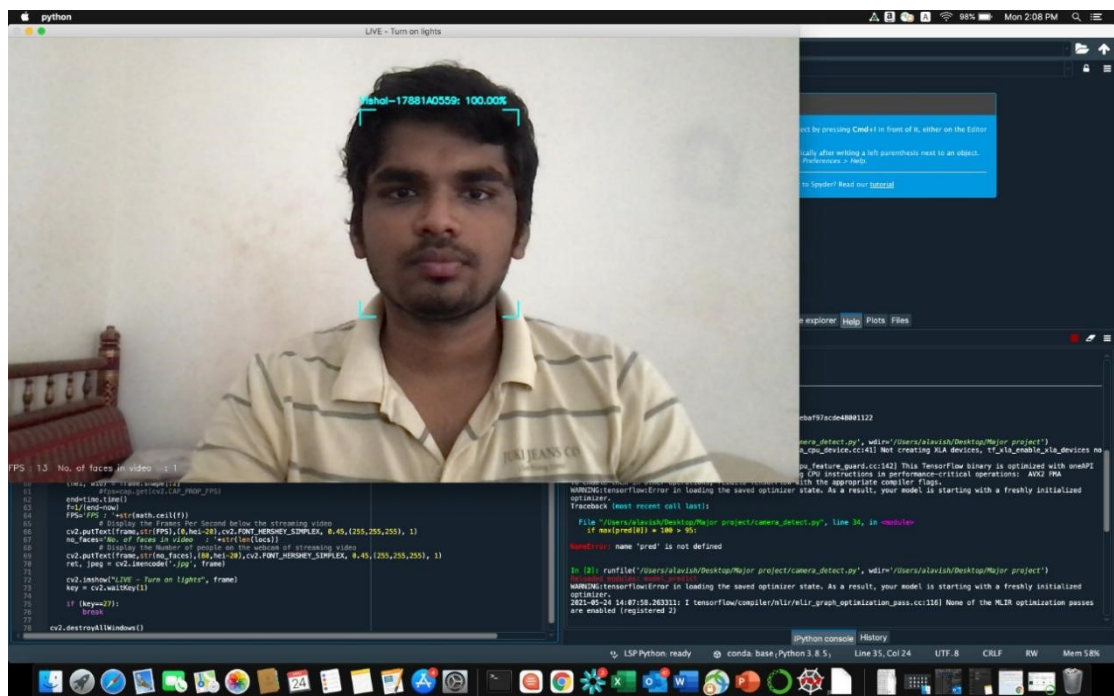


Figure 5.5: Verification of known video stream

5.2.4 Verification of Unauthorized image through video stream:

In verification, we have taken input from the video stream and for the unauthorized students i.e they are not present in the dataset our model predicted correctly.



Figure 5.6: Verification of unknown person in video stream

5.2.5 Confusion Matrix

As we can see in the below confusion matrix, the confusion matrix is between the actual and predicted values. Our model as predicted 1478 'Nitish' images and 1479 'Prahasit' images and 1479 'Vishal' images and there are no wrong predictions.

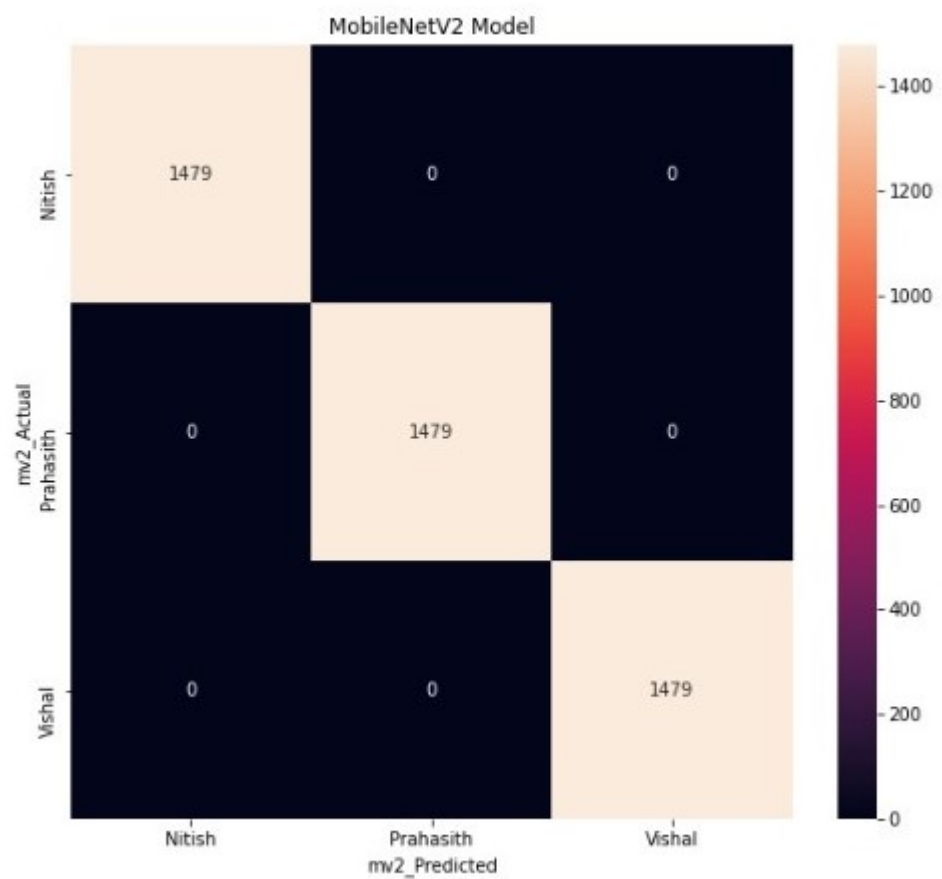


Figure 5.7: confusion matrix

CHAPTER 6

Conclusions and Future Scope

6.1 Conclusion

In recent times, because of an upsurge in cases of impersonators in examinations, there is a need to develop a system which can detect impersonators and them from taking examinations. Hence, we have developed this model to reduce malpractices and improve the quality of examinations. In this project, we adopted the mobilenet-v2 model which gives greater efficiency that the other models.

6.2 Future Scope

In near future, by this model, we will try to reduce the manpower of cross-checking the hall tickets. In our present system, our model identifies impersonators and a person is required to alert the higher authorities. In the future, our model is going the alert the chief examiner directly when an impersonator is found.

REFERENCES

- [1] C. Monrocq R. Vaillant and Y. Le Cun. “Original approach for the localisation of objects in images.” In: *IEE Proceedings Vision, Image and Signal Processing* (1994).
- [2] C. Garcia and M. Delakis. “A neural architecture for fast and robust face detection”. In: *16th International Conference* (2002).
- [3] *Real Time Face Detection and Tracking Using OpenCV*. URL: <https://ijrest.net/downloads/volume-4/issue-4/pid-ijrest-44201715.pdf>.
- [4] *CNN*. URL: <https://medium.com/@RaghavPrabhu/understanding-ofconvolutional-neural-network-cnn-deep-learning-99760835f148>.
- [5] *TransferLearning*. URL: <https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/>.
- [6] *Hyperparameter Tunning*. URL: <https://towardsdatascience.com/hyperparameter-tuning-c5619e7e6624>.
- [7] *Keras-tuner*. URL: https://www.tensorflow.org/tutorials/keras/keras_tuner.
- [8] *Keras*. URL: <https://keras.io/api/applications/>.
- [9] *MobileNetV2*. URL: <https://towardsdatascience.com/review-mobilenetv2-light-weightmodel-image-classification-8febb490e61c>.