Face Mask Detection

Yashwanth Telukuntla
(ytelukuntla@albany.edu)

Prasanth Nagisetty
(pnagisetty@albany.edu)

S R K N S Bharadwaj Nidamarthy
(snidamarthy@albany.edu)

Work done so far

- Platform used
- Packages used
- Exploratory Data Analysis
- Plotting random images
- Class distribution
- Data Augmentation
- Model building
- Accuracy and loss
- Classification reports, confusion matrix
- Model Evaluation
- Image Prediction
- Detector
- Facemask detection using camera

Platform used

- Python
- Google colab pro
- Google drive(to store dataset and models we trained)
- Visual Studio code

Packages Used

- numpy
- pandas
- matplotlib(plotting graphs)
- tensorflow(MobileNetV2, data augmentation…)
- sklearn(accuracy_score, precision_score, recall_score, f1_score)
- Keras
- Seaborn
- Dlib

Exploratory Data Analysis

- Data set use Face Mask Detection ~12K Images Dataset | Kaggle
- Exploratory data analysis is performed to gain different useful information and hidden insights from dataset. In this section different statistical techniques have been used to gain insights and then being visualized into appropriate charts and plots
- With get_all_images_from_subdirectory_to_dataframe() above function, we will traverse all the images from each directory (train, test and val) and store into a list with class (mask or without mask), image name and image path. After that we will show in pandas dataframe. The dataframe representing this information is shown below.

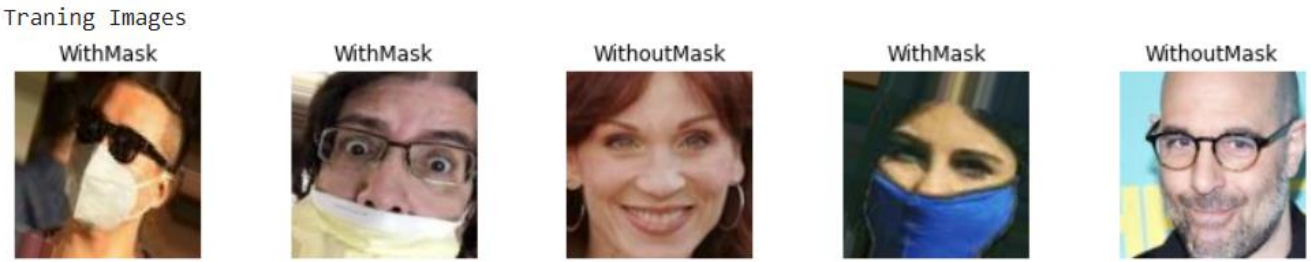
```
df_list = get_all_images_from_subdirectory_to_dataframe(train_dir)
df_train = pd.DataFrame(data=df_list, columns=['class', 'image_name', 'image_path'])

df_list = get_all_images_from_subdirectory_to_dataframe(test_dir)
df_test = pd.DataFrame(data=df_list, columns=['class', 'image_name', 'image_path'])

df_list = get_all_images_from_subdirectory_to_dataframe(val_dir)
df_val = pd.DataFrame(data=df_list, columns=['class', 'image_name', 'image_path'])
```

	class	image_name	image_path
0	WithoutMask	2083.png	/content/Face Mask Dataset/Train/WithoutMask/2...
1	WithoutMask	4529.png	/content/Face Mask Dataset/Train/WithoutMask/4...

Plotting random images

- This analysis is about plotting the random images from dataset. This plot_random_images() function is used to plot the random images from specific dataset folder (train, test and val) with class name (mask or without mask).

```
[ ]  1 print("Traning Images")
     2 plot_random_images(df_train, 5)
```



Class distribution

- We have two classes(with Mask and WithoutMask) that is equally distributed among the train, test and validation data sets.
- Training set hove total of 10000 images
- Testing set have total of 992 images
- Validation set have total of 800 images

training distribution

class Distribution

	Category	classPercentage
0	WithMask	50
1	WithoutMask	50

testing distribution

class Distribution

	Category	classPercentage
0	WithoutMask	51.31
1	WithMask	48.69

validation distribution

class Distribution

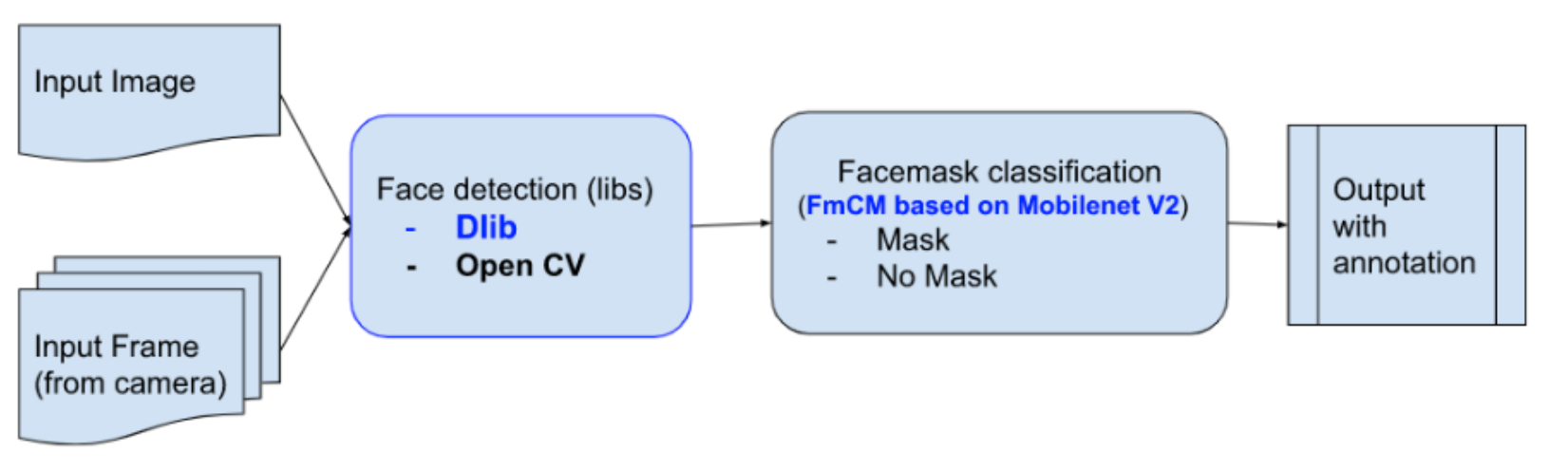
	Category	classPercentage
0	WithMask	50
1	WithoutMask	50

Data Augmentation

- Data Augmentation in play. A convolutional neural network that can robustly classify objects even if it's placed in different orientations is said to have the property called invariance. More specifically, a CNN can be invariant to translation, viewpoint, size or illumination (Or a combination of the above).
- We have a directory in which two folder mask and without mask. Therefore, we will use the tensorflow provide flow from directory method. The flow_from_directory() method takes a path of a directory and generates batches of augmented data. The directory structure is very important when you are using flow_from_directory() method. The flow_from_directory() assumes: The root directory contains at least two folders one for train and one for the test.

```
4
5 train_datagen = ImageDataGenerator( horizontal_flip=True, rotation_range=10,height_shift_range=0.2,
6                                      shear_range=0.2,width_shift_range=0.2,brightness_range=[0.2,1.2],
7                                     rescale= 1./255,
8                                     zoom range=0.2,)
```

Facemask detection process



Face detector

1. Dlib

- **HOG and Linear SVM:**

  This method is based on histogram of oriented gradients and linear support vector machine. It provides a very fast recognition result of frontal faces but has limited recognition on side-profile faces This method is suitable for the case in which we can get the direct view of the face (ID systems). This method is better than Haar-cascade (OpenCV) but need more computational power.

  >>>usage: dlib.get_frontal_face_detector()

- **Max-Margin Mod Object Detection (MNMOD) CNN** face detector: this method has capability of capturing face(s) at angles. However, this model needs GPU or will be super slow in CPU.

  >>>usage: dlib.cnn_face_detection_model_v1(CNN model)

  >>> CNN model: mmod_human_face_detector.dat

Face detector (cont..)

2. OpenCV

- **Using Haar-Cascade:** use Haar cascade algorithm to detect objects in images. We can use a pre-trained cascade (xml file) to detect frontal faces. From experimental results, this method is a lightweight face detector: model size is around 930KB, fast detection. However, it returns many 'false positive' detections, i.e., return 'face' that is not a face

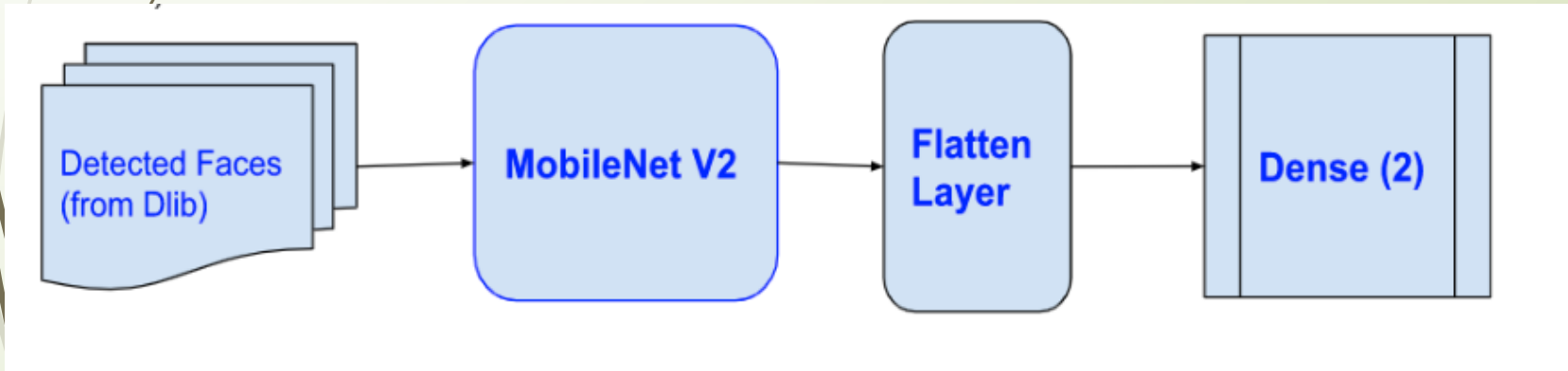
  >>>usage: **face_cascade = cv2.CascadeClassifier ('haarcascade_frontalface_default.xml')**

- **Deep neural network:** We use Open CV's deep neural network model to detect face. The model includes a .**prototxt** file which defines the model architecture (layers of DNN) and a .**caffemodel** which contains the pre-trained weights of the DNN. It is based on Single-Shot-Multibox detector and uses ResNet-10 Architecture as backbone. The model was trained using images available from the web, but the source is not disclosed.

  >>>usage: **dnn_net = cv2.dnn.readNetFromCaffe (facenet_dnn.prototxt, facenet_dnn.caffemodel)**

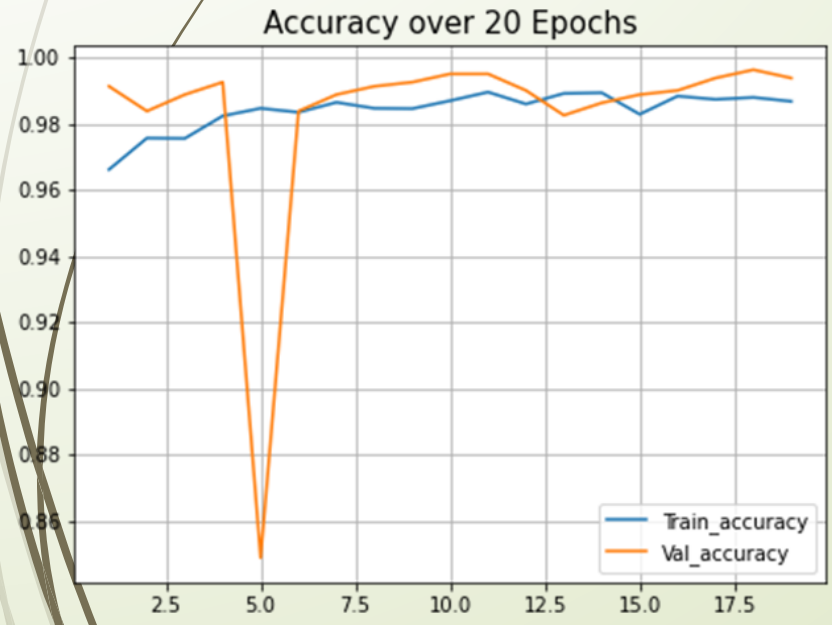
Model building

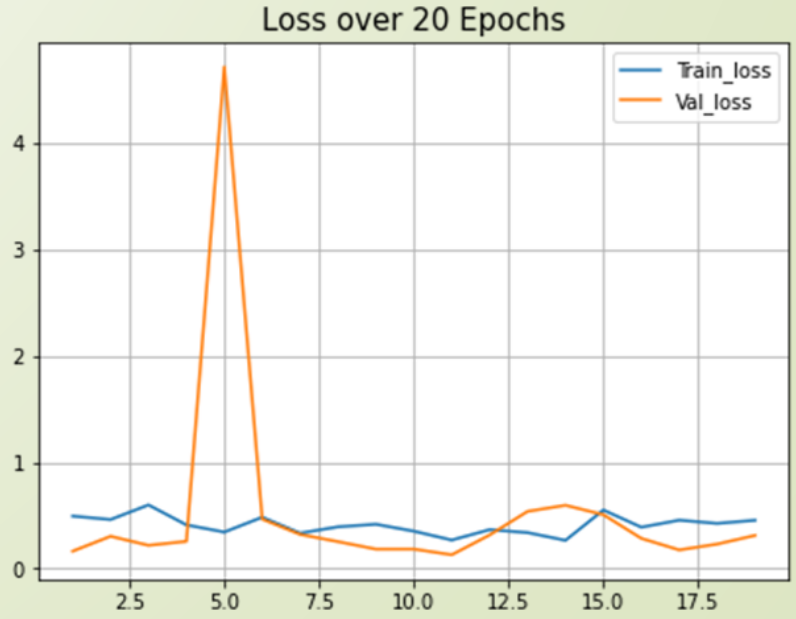
- We are using MobileNetV2
- Initially we used the below model, and it didn't perform well, we got issues in training after 10 epochs, training_loss keeps above 0.4 and does not reduce



Accuracy and loss of old model

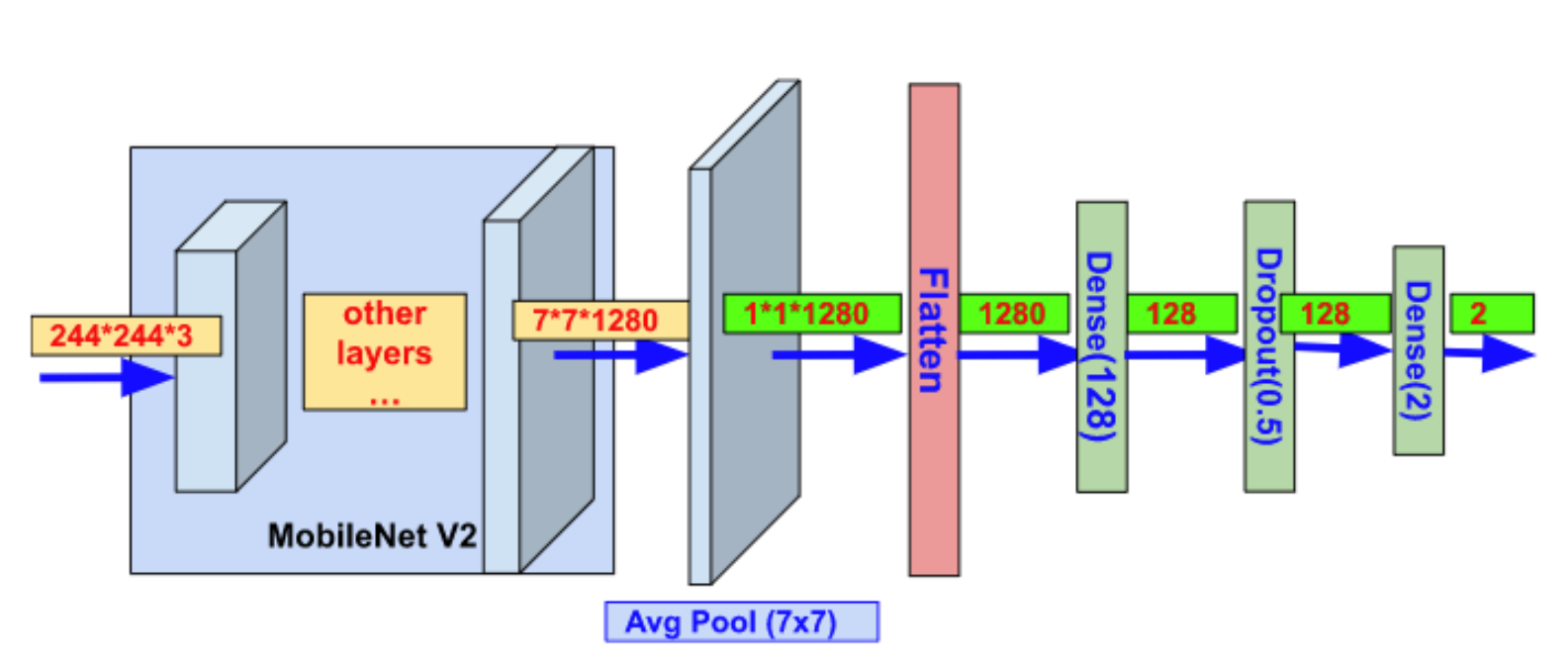
- We have plotted accuracy and loss for the three models MobileNetV2 over 20 epochs.
- No development in Loss and accuracy





Solution for issues of old model

- **Average Pooling 2D (7x7) module**
- **Flatten module**
- **Dense(128) module**
- **Dropout(0.5)**
- **Dense(2)**

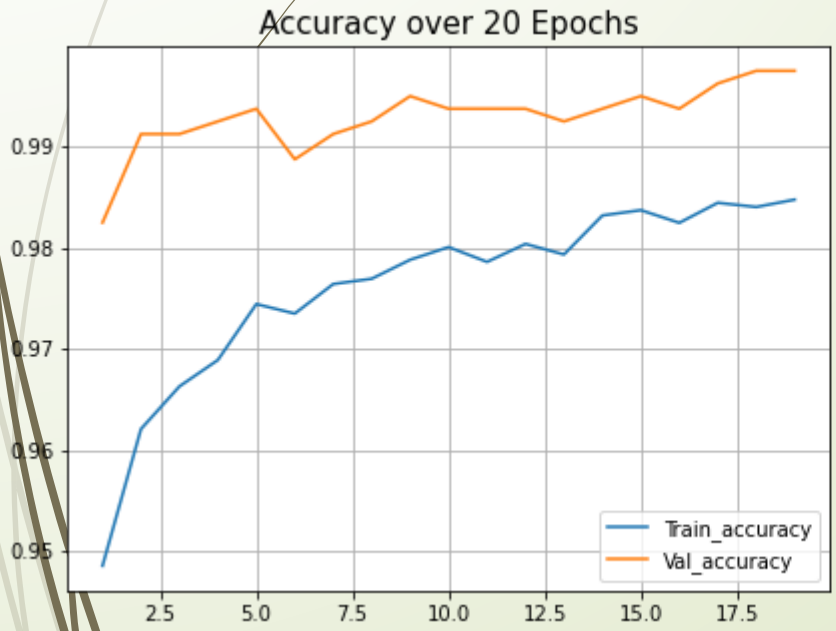


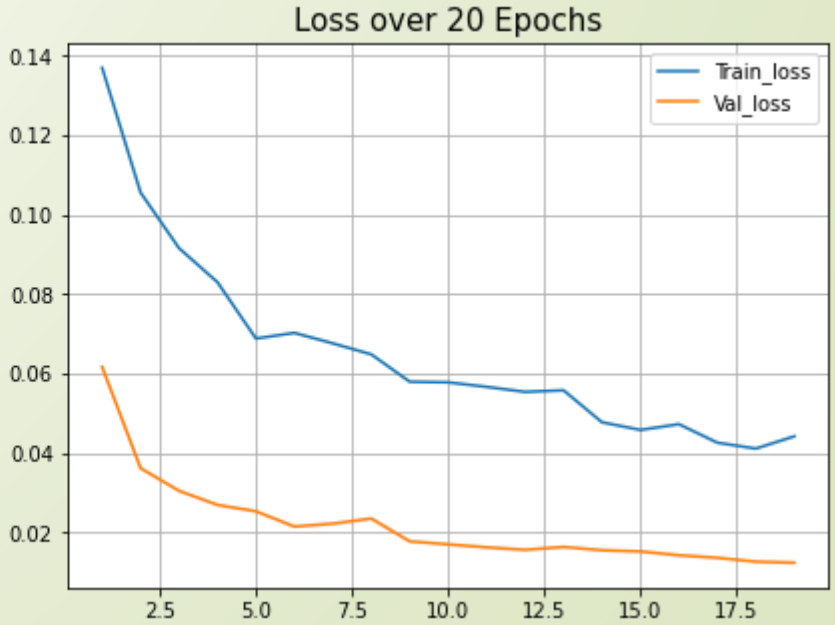
Configuring the parameters

- We used Adam optimizer with INIT_LR=0.0001 and decay=INIT_LR/ number_of_epochs)
- And used setps_per_epoch = 300 in the fit model.

Accuracy and loss of new model

- We have plotted accuracy and loss for the model MobileNetV2 over 20 epochs with step_per_epoch = 300
- We also generated classification reports, confusion matrix for the model





Model Evaluation

- We have evaluated all the three models based on accuracy, precision, recall, f1-score for MobileNetV2

Accuracy on validation set: 0.9929

Classification report :

	precision	recall	f1-score	support
0	0.99	1.00	0.99	504
1	1.00	0.99	0.99	488
accuracy			0.99	992
macro avg	0.99	0.99	0.99	992
weighted avg	0.99	0.99	0.99	992

Model Evaluation

- Confusion matrix

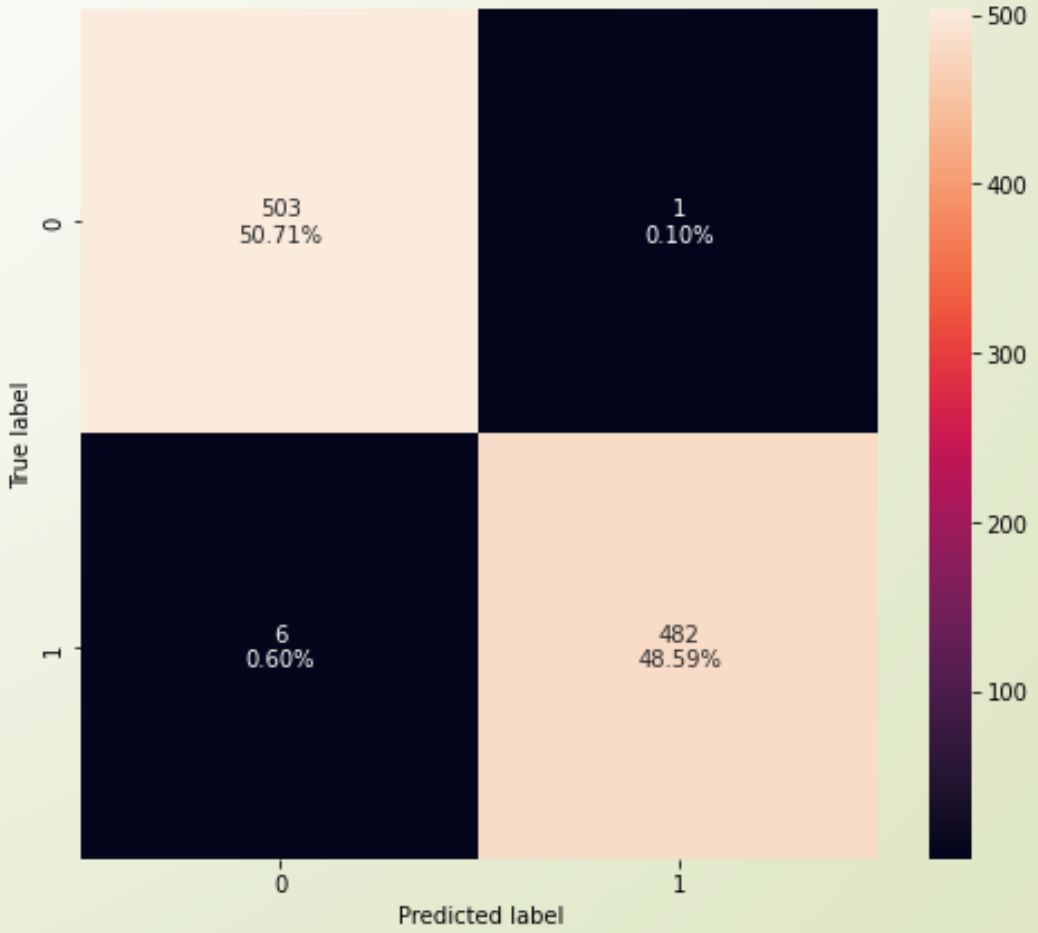
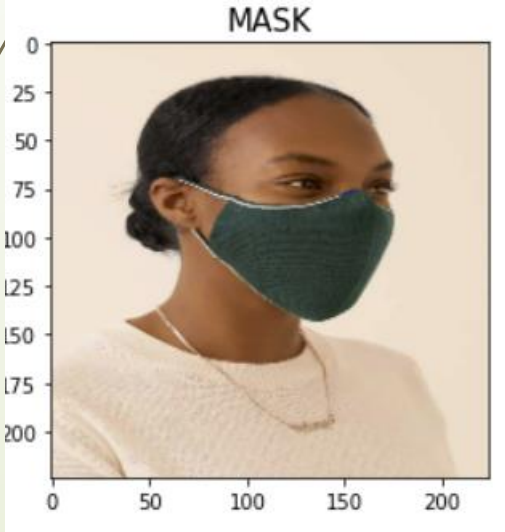


Image classification

- We can classify an image with the help of models we have developed
- We are tested with MobileNetV2



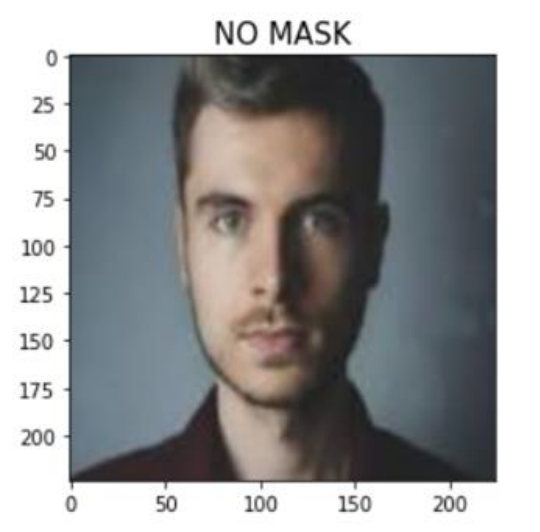
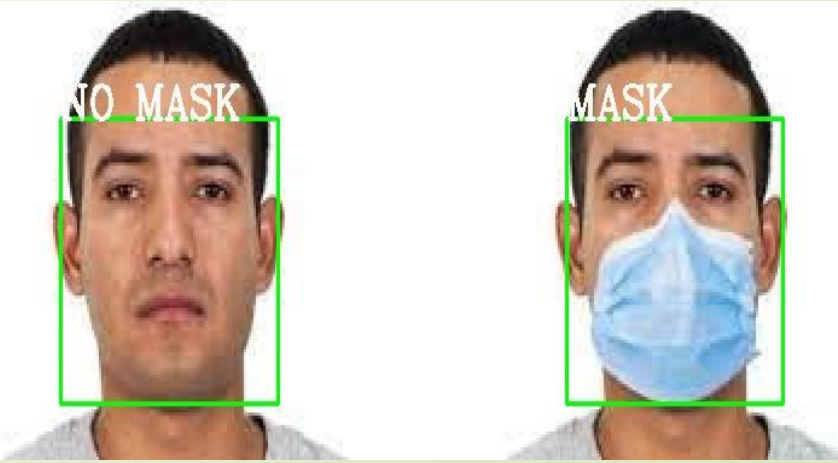


Image Prediction

- We are able to detect the people in a image/came and able to classify whether they are wearing mask or not.
- We are able to predict multiple people at a time.



Thanks