HUAWEI

NTI
National Telecommunication Institute

Mask Detection

Presented to:

Dr: Ghazal A. Fahmy

Dr: Mohammed El-zorkany

Presented by:

Ahmed Gamal Simeda

Amr Abo-Alyazied Zaki

Ibrahim Mohamed Basal

Mohamed Talaat Kandel

Zakaria Abdallah Halim

CONTENT

Abstract:

COVID-19 pandemic has rapidly affected our life disrupting the world trade and movements. Wearing a protective face mask has become a new normal. As we know it's a dangerous virus which can be spreading from humans to humans through droplets and airborne and as for the prevention, wearing a face mask is essentials while going outside or meeting to others. However, some irresponsible people refuse to wear face mask with so many excuses. Therefore, face mask detection has become a crucial task to help global society. This documentation presents a simple approach to achieve this purpose using some basic Machine Learning packages like TensorFlow, Keras, OpenCV, and Scikit-Learn. With the help of OpenCV we were able to capture the video feed from different sources like webcam, video file or an IP camera. The proposed method detects the face from the image correctly and then identifies if it has a mask on or not. As a surveillance task performer, it can also detect a face along with a mask in motion. The method attains average accuracy up to 99.6% and 99.8% respectively on two different datasets.

Introduction

Coronavirus disease 2019 (COVID-19) has globally infected over 216 million people causing over 4.5 million deaths. Individuals with COVID-19 have had a wide scope of symptoms reported – going from mellow manifestations to serious illness. Respiratory problems like shortness of breath or difficulty in breathing is one of them. Elder people having lung disease can possess serious complications from COVID-19 illness as they appear to be at higher risk.

To curb certain respiratory viral ailments, including COVID-19, wearing a clinical mask is very necessary. The public should be aware of whether to put on the mask for source control or aversion of COVID-19. Potential points of interest of the utilization of masks lie in reducing vulnerability of risk from a noxious individual during the "pre-symptomatic" period and stigmatization of discrete persons putting on masks to restraint the spread of virus. Therefore, face mask detection has become a crucial task in present global society.

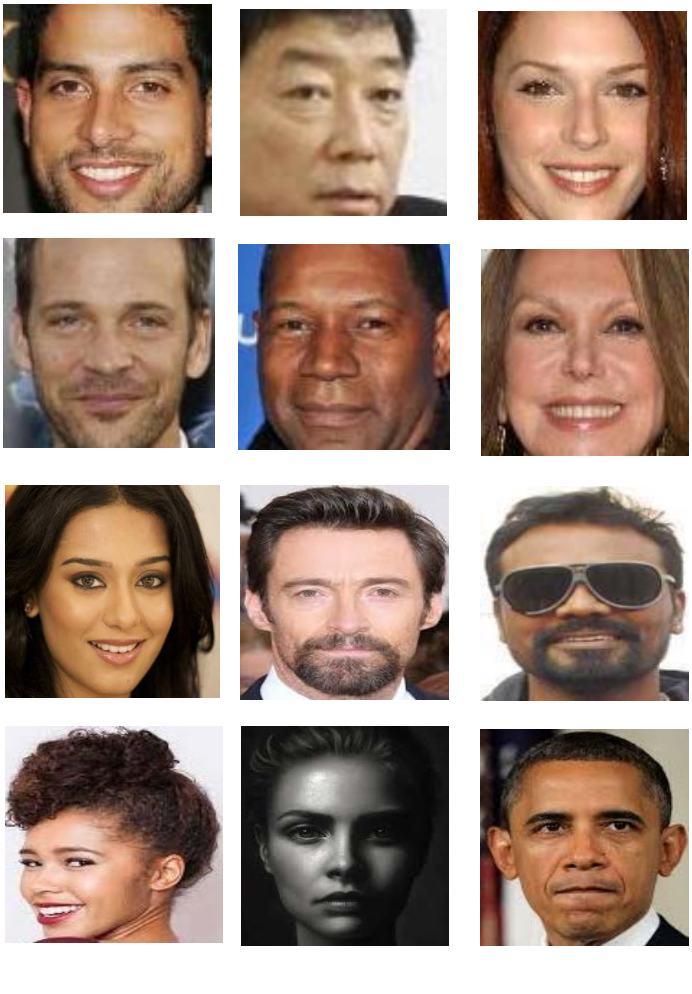
Face mask detection involves in detecting the location of the face and then determining whether it has a mask on it or not. The issue is proximately cognate to general object detection to detect the classes of objects. Face identification categorically deals with

distinguishing a specific group of entities i.e., Face. It has numerous applications, such as autonomous driving, education, surveillance, and so on. To overcome this situation, a robust face mask detection needs to be developed. In order to detect a face mask, the object detection algorithm can be implemented. The state of art of object detection algorithm which has a robust performance is the You Only Look Once (YOLO). A few years later, YOLO method was improved to YOLO V2 which was able to detect over 9000 object categories. In this version, a novel, multi-scale training method was developed [6]. Thereafter, Kim, et al., implemented the YOLO-V2 for image recognition and other testbenches for a CNN accelerator. In this work, the YOLO V2 method has been applied through the simulation and FPGA experiment. Harisankar, et al. on the other hand, modified the YOLO V2 to detect and localize the pedestrian by adding the Model H architecture. In order to detect the pedestrian precisely, they used ZED camera and created the depth map. Within two years Redmon, et al., introduced the YOLO V3 although the architecture is a little bigger than the last time, this version is accurately as SSD algorithm but three times faster. Therefore, Hu, et al. implemented this version to detect the workers with or without helmets in videos. The first, YOLO V3 model was to identify and intercept the worker video then made the sample model of wearing and not wearing helmet. The theorical analysis and experimental results verify that the proposed algorithm is able to detect the helmet with high detection accuracy. Because of the YOLO method is open source, then it allows everyone to improve the algorithm of the YOLO method. As presented in, Alexey, et al. introduced the YOLOv4 with optimal speed and accuracy. In this work, they assumed that such universal features such as Weighted-Residual-Connections (WRC), Cross-Stage-Partial connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish activation are able to give high detection accuracy. According to the state of arts and the results which have already been tested by some researcher, in this work we developed a face mask detection for COVID-19 prevention by using the YOLO V4 algorithm. the YOLO v4 algorithm will be used to detect the face mask. The face mask detection will be applied in real-time application to detect all types of commercial face mask. By adding the YOLO V4, it is hoped that this device able to detect whether the users are wearing a face mask or not.

Dataset

Training set
5300/5300

Without Mask



With Mask

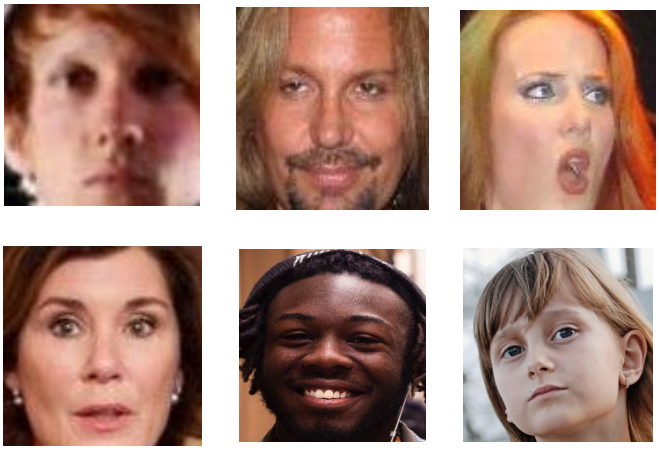
Validation set
553/553

Without Mask

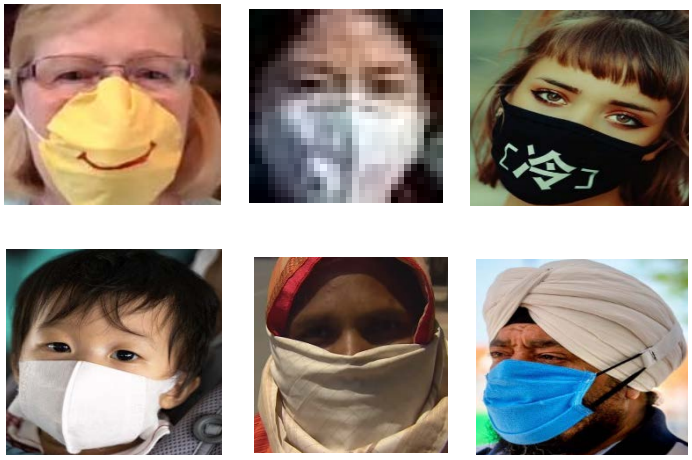
With Mask

Test set
533/559

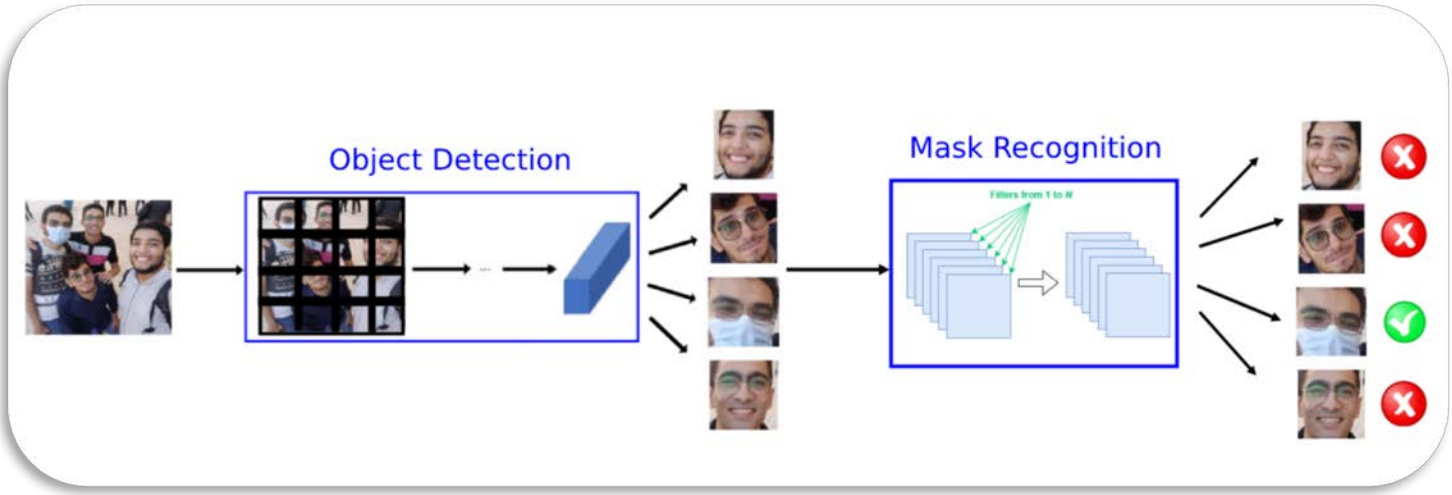
Without Mask



With Mask



Methodology



Face Detection:

Input: Dataset including faces with and without masks

For each image in dataset do

Convert them to RGB

Detect the faces in the whole image and crop them

resize the result face into (128, 128)

Feed them to the model classification

end

Output: faces (128, 128)

Mask Detection:

Input: Dataset including one face with or without masks -- OR -- The Output Of Face Recognition

For each image in dataset do

Resizing images into (128, 128) (if necessary)

normalizing the image and converting them to array

Flattening them

end

Output: Either Masked face or Unmasked

A brief look over the models

- **Object detection**

```
# model loading
def load_mobilenetv2_224_075_detector(path):
    input_tensor = Input(shape=(224, 224, 3))
    output_tensor = MobileNetV2(weights=None, include_top=False, input_tensor=input_tensor, alpha=0.75).output
    output_tensor = ZeroPadding2D()(output_tensor)
    output_tensor = Conv2D(kernel_size=(3, 3), filters=5)(output_tensor)

    model = Model(inputs=input_tensor, outputs=output_tensor)
    model.load_weights(path)

    return model

mobilenetv2 = load_mobilenetv2_224_075_detector(r"facedetection-mobilenetv2-size224-alpha0.75.h5")
```

With the help of some functions:

- Non-max suppression
- Union suppression
- … And others

We have also put all that in one class to make the code cleaner and more efficient.

- **Mask Recognition**

AlexNet

```
#Building AlexNet Model
model = Sequential()

model.add(Conv2D(96, kernel_size=(11, 11), strides=(4,4), padding='valid', activation='relu', input_shape=(128,128,3)))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))

model.add(Conv2D(256, kernel_size=(5, 5), strides=(1,1), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
model.add(Conv2D(384, kernel_size=(3, 3), strides=(1,1), activation='relu', padding='same'))
model.add(Conv2D(384, kernel_size=(3, 3), strides=(1,1), activation='relu', padding='same'))
model.add(Conv2D(256, kernel_size=(3, 3), strides=(1,1), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
model.add(Dropout(0.5))


#model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4096, activation='relu'))
model.add(Dense(2, activation='softmax'))
```

ResNet_101

```
Category_count = 2
SHAPE = (128, 128, 3)

base_resnet = ResNet101(weights='imagenet', include_top = False, input_shape = SHAPE)
base_resnet.trainable = False

model = Sequential()
model.add(base_resnet)

model.add(Flatten())
model.add(Activation('relu'))
model.add(Dense(Category_count))
model.add(Activation('softmax'))
```

DenseNet_201

```
base_model = tf.keras.applications.densenet.DenseNet201(weights='imagenet', input_shape = (128,128,3),include_top=False)
for layer in base_model.layers:
    layer.trainable = False
#building the model
model=Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Flatten())
model.add(Dense(300, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(100,activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2,activation="softmax"))
```

MobileNet_v2

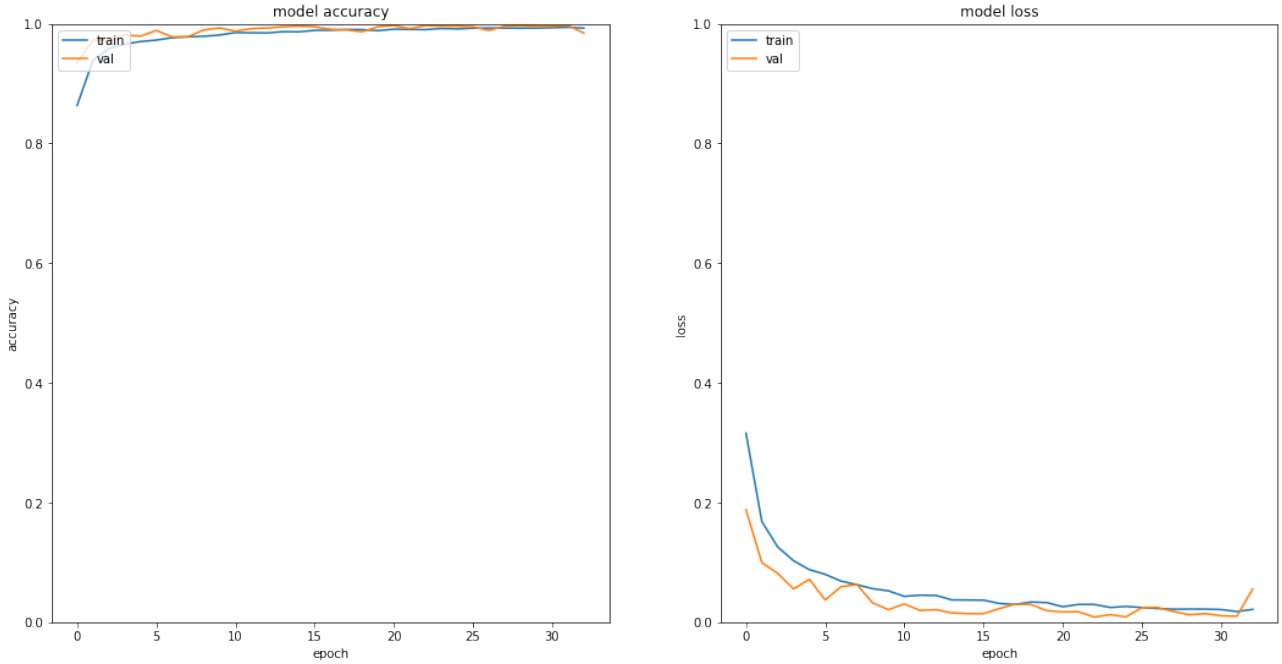
```
baseModel = MobileNetV2(weights="imagenet", include_top=False, input_shape=(128, 128, 3))
for layer in baseModel.layers:
    layer.trainable = False

headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(3, 3))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

model = Model(inputs=baseModel.input, outputs=headModel)
```

We have used Generators for data augmentation to prepare the data before feeding it to the models.

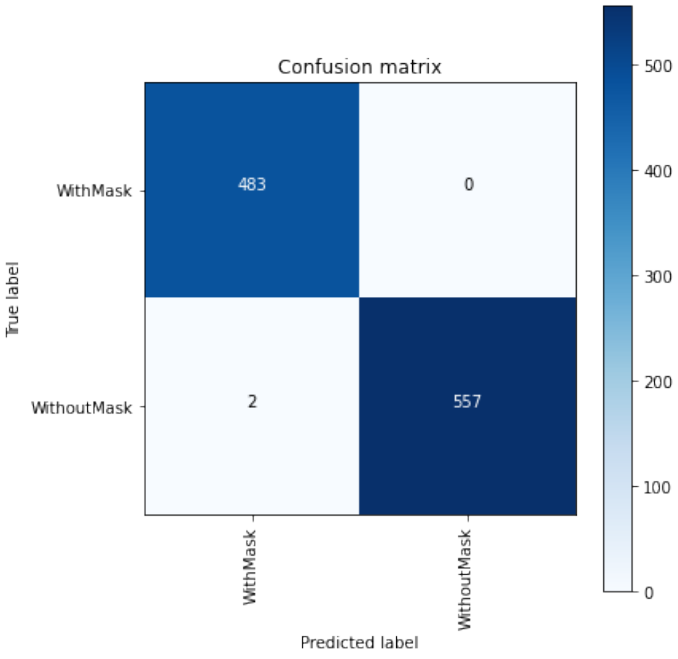
Results

AlexNet

Training Curves:

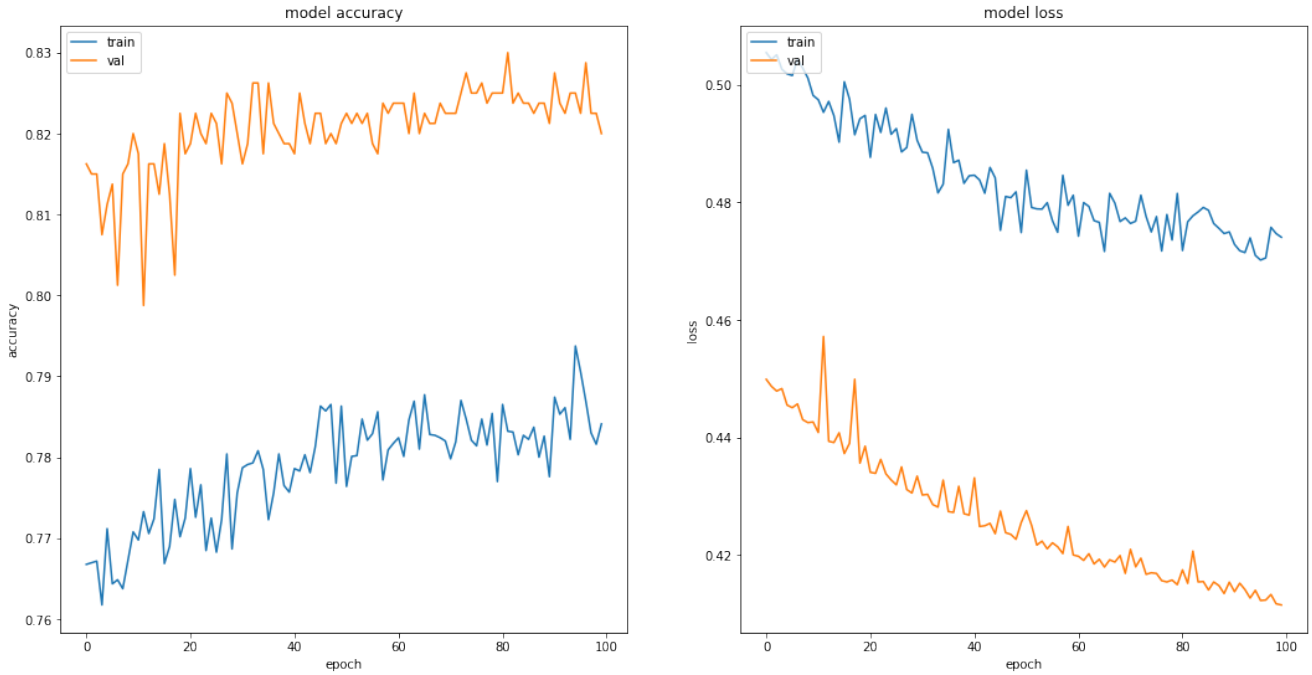


Confusion Matrix:

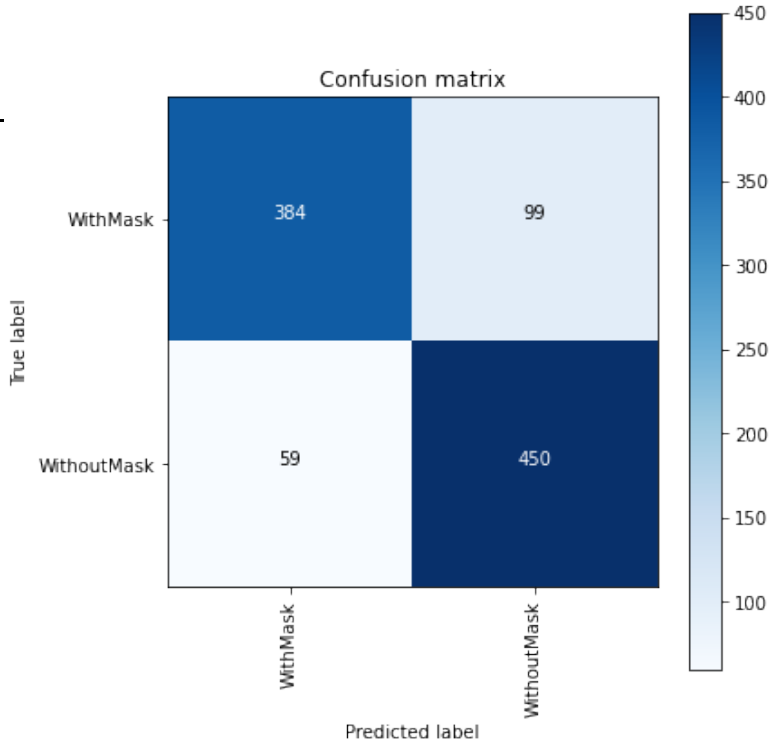


ResNet101

Training Curves:

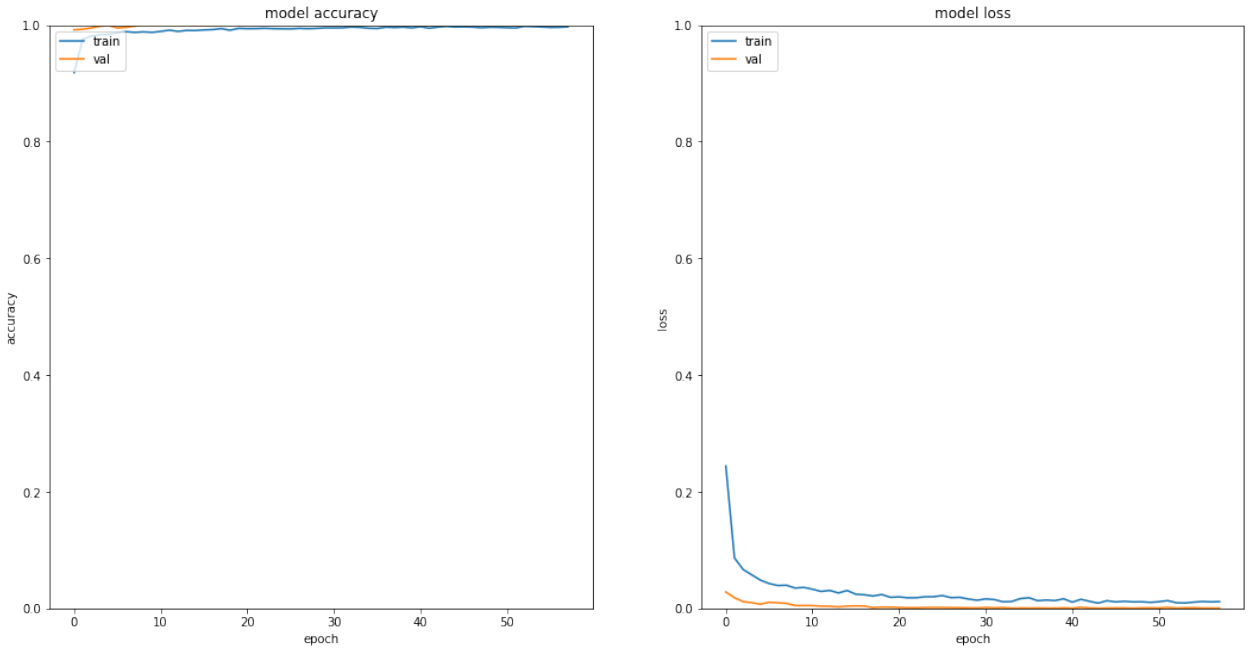


Confusion Matrix:

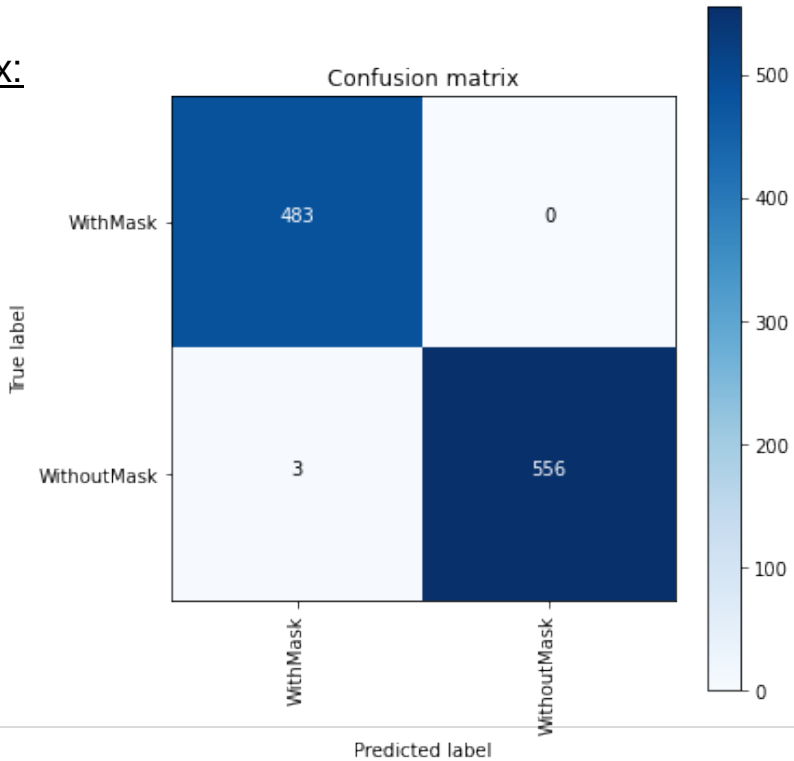


DenseNet201

Training Curves:

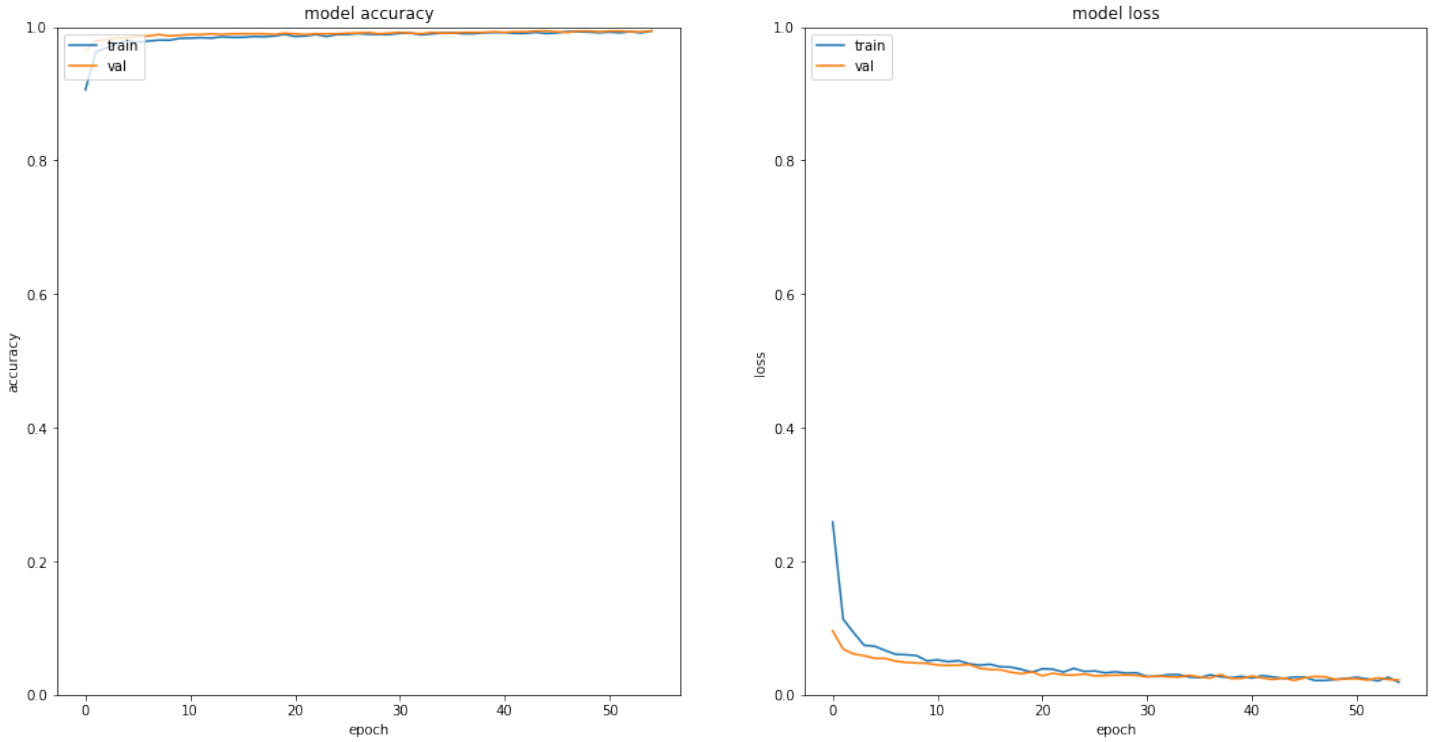


Confusion Matrix:

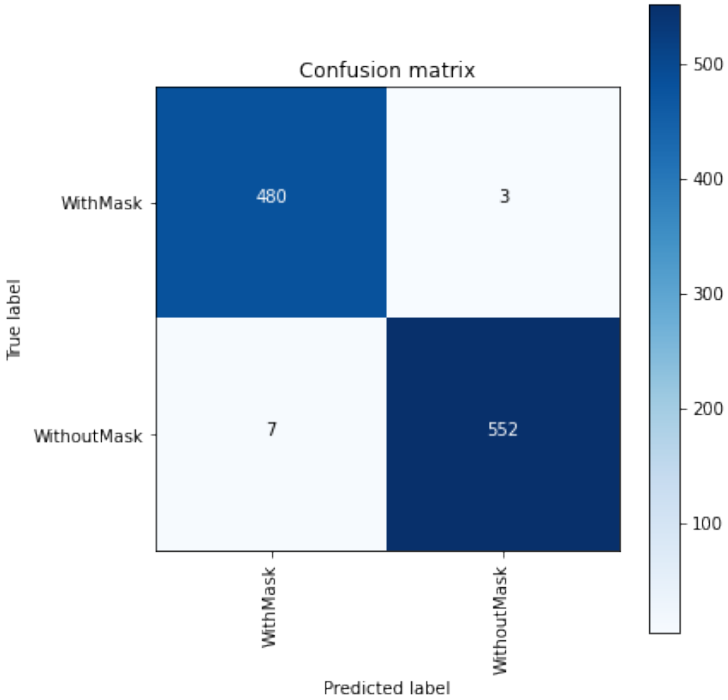


MobileNet_v2

Training Curves:



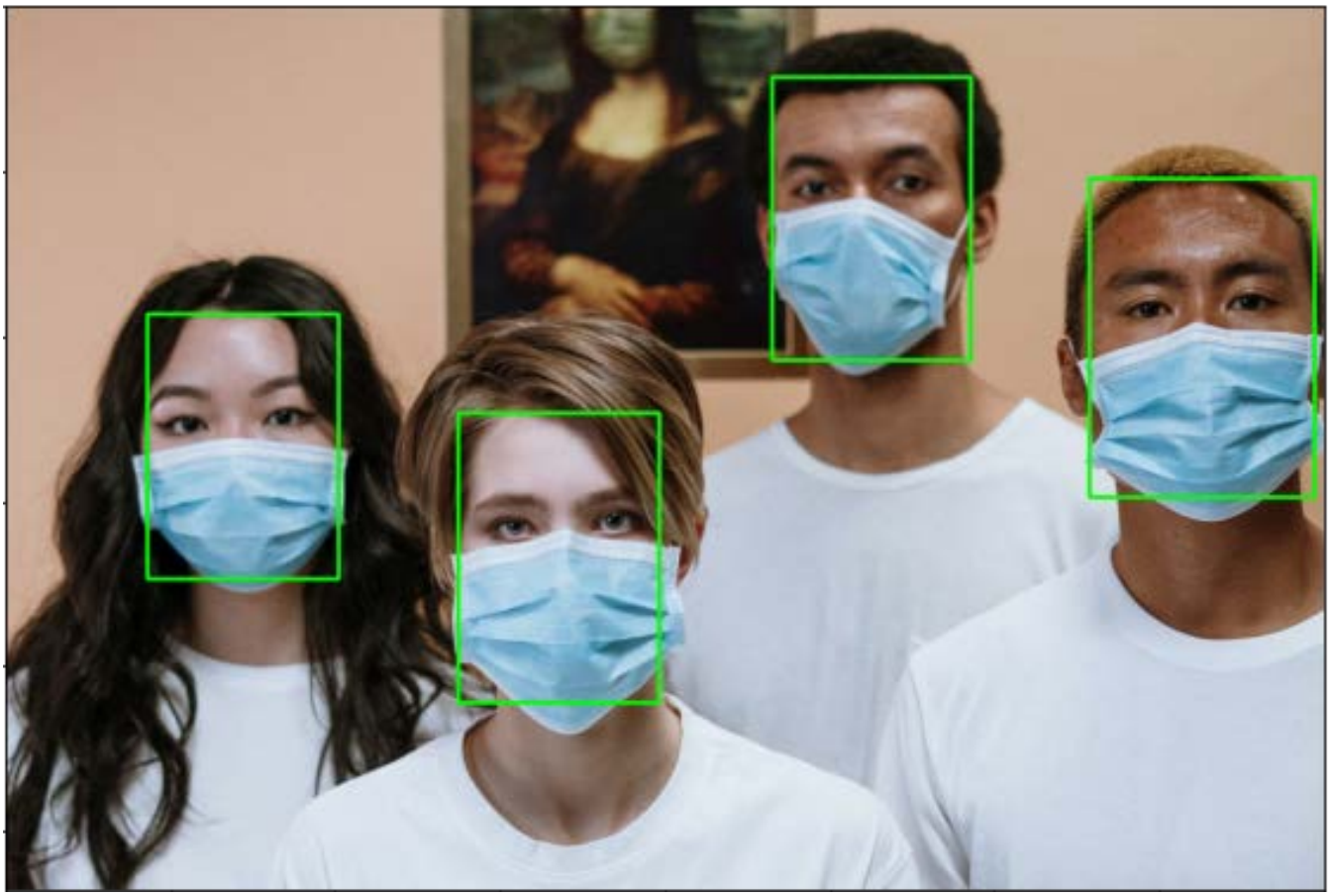
Confusion Matrix:



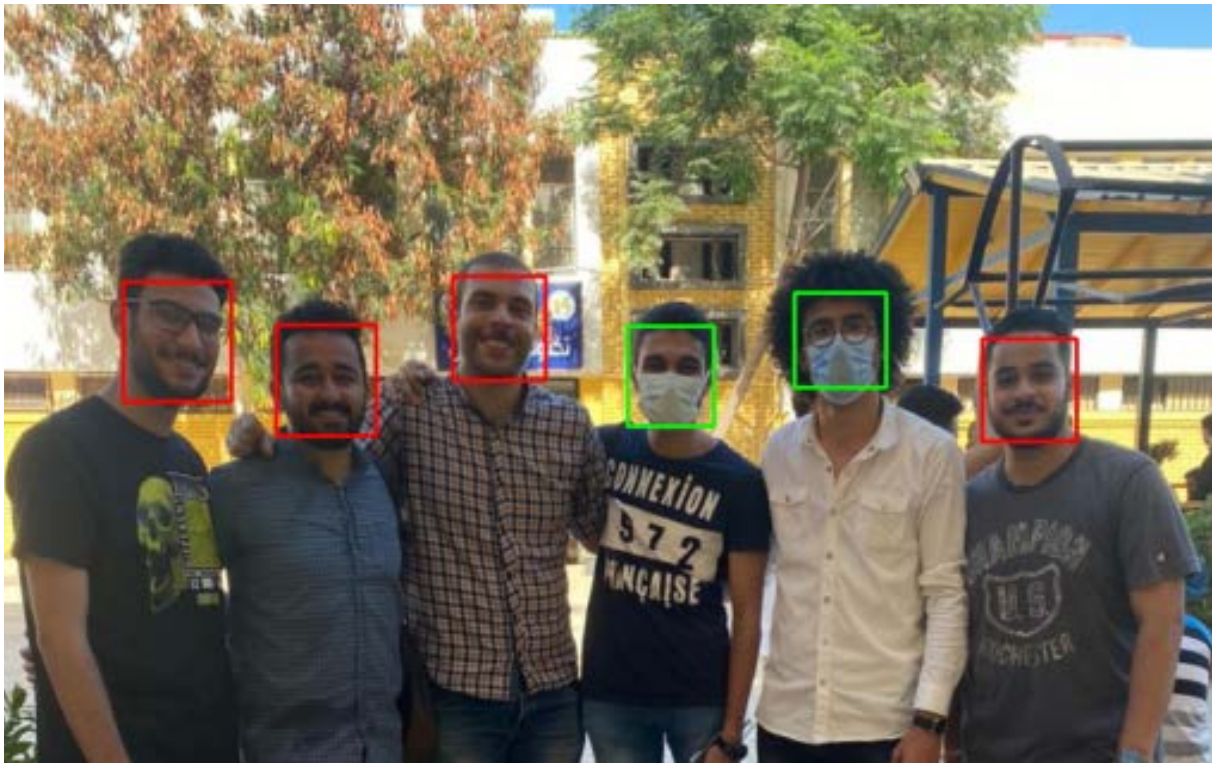
Results Summary:

	Test Accuracy(%)	Precision(%)	Recall(%)	F1 Score(%)	TP	TN	FP	FN
AlexNet	**99.8**	**99.59**	**1**	**99.79**	**483**	**557**	**2**	**0**
ResNet_v2	84.07	86.68	79.5	82.93	384	450	59	99
DenseNet201	99.71	99.38	1	99.69	483	556	3	0
Mobile Net_v2	99.04	98.56	99.38	98.97	480	552	7	3

- *Let's take a closer look on some case studies:*



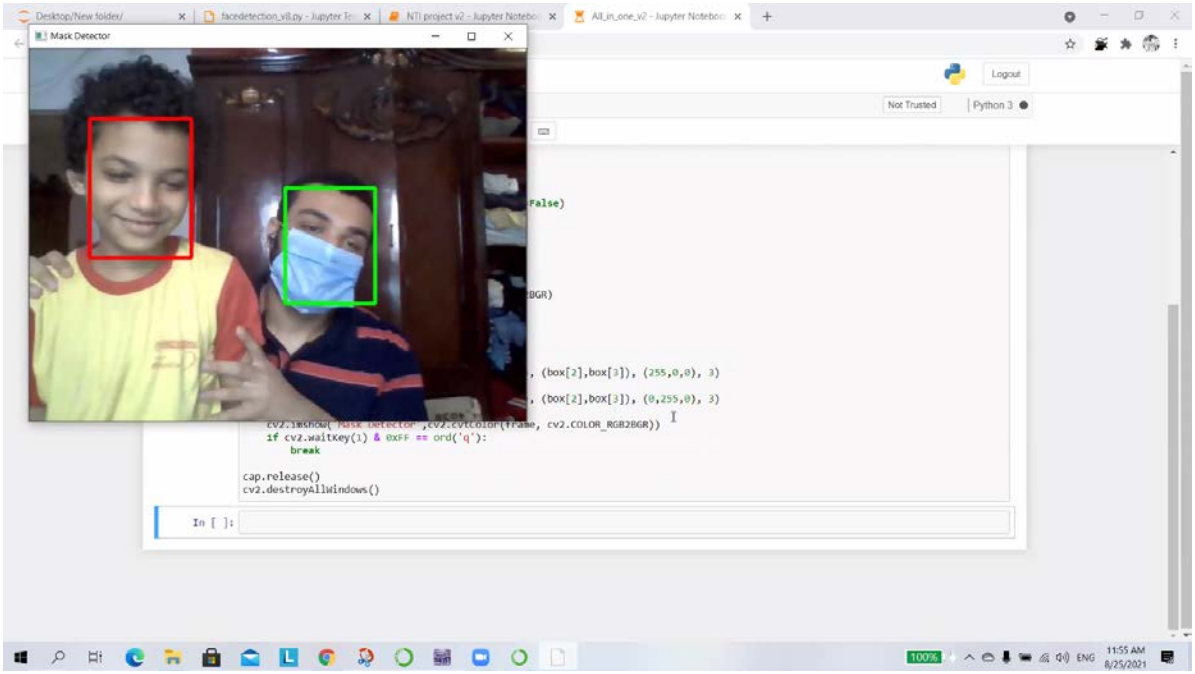

Emirates
Emirates
Emirates


CONNEXION
372
FRANÇAISE


11
2
10
6
5
4


Samsung Triple Camera
MR Zeco

➢ *Application on video cam:*



Conclusion

During this work, we have trained so many models and finally we've come up with 4 models that represent the different stages we've come across until we reached these results. The final model has been developed to come up with an efficient way for detecting and notifying officials when a person does not follow the COVID 19 safety protocols in a workplace, business establishments ...etc.

In the future work, we will try to:

- Add additional class (wearing the mask not properly).
- Face Recognition
- More people detection