1. INTRODUCTION

Rapid advancements in the fields of Science and Technology have led us to a stage where we are capable of achieving feats that seemed improbable a few decades ago. Technologies in fields like Machine Learning and Artificial Intelligence have made our lives easier and provide solutions to several complex problems in various areas. Modern Computer Vision algorithms are approaching human-level performance in visual perception tasks. From image classification to video analytics, Computer Vision has proven to be a revolutionary aspect of modern technology. In a world battling against the Novel Coronavirus Disease (COVID-19) pandemic, technology has been a lifesaver. With the aid of technology, 'work from home' has substituted our normal work routines and has become a part of our daily lives. However, for some sectors, it is impossible to adapt to this new norm.

As the pandemic slowly settles and such sectors become eager to resume in-person work, individuals are still sceptical of getting back to the office. 65% of employees are now anxious about returning to the office (Woods, 2020). Multiple studies have shown that the use of face masks reduces the risk of viral transmission as well as provides a sense of protection (Howard et al., 2020; Verma et al., 2020). However, it is infeasible to manually enforce such a policy on large premises and track any violations. Computer Vision provides a better alternative to this. Using a combination of image classification, object detection, object tracking, and video analysis, we developed a robust system that can detect the presence and absence of face masks in images as well as videos.

In this paper, we propose a two-stage CNN architecture, where the first stage detects human faces, while the second stage uses a lightweight image classifier to classify the faces detected in the first stage as either 'Mask' or 'No Mask' faces and draws bounding boxes around them along with the detected class name. This algorithm was further extended to videos as well. The detected faces are then tracked between frames using an object tracking algorithm, which makes the detections robust to the noise due to motion blur. This system can then be integrated with an image or video capturing device like a CCTV camera, to track safety violations, promote the use of face masks, and ensure a safe working environment.

1.1 MOTIVATION

The motivation for this project came out to be according to the real world demands to compensate the needs of software nowadays. In this time of pandemic, there can't be a better project than going for Mask detection system which detects whether a person has mask on his/her or not. This can be handy for regulating the stricter measures and therefore flexible algorithm required solve this problem.

Also Motivated by the works of,

1. Adrian Rosebrock
2. sentdex
3. Prajna Bhandary

1.2 RELATED WORKS

- Traditional Object Detection:

The problem of detecting multiple masked and unmasked faces in images can be solved by traditional object detection model. The process of object detection mainly involves localizing the objects in images and classifying them (in case of multiple objects). Traditional algorithms like Haar Cascade (Viola and Jones, 2001) and HOG (Dalal and Triggs, 2005) have proved to be effective for such tasks, but these algorithms are heavily based on Feature Engineering. In the era of Deep learning, it is possible to train Neural Networks that outperform these algorithms, and do not need any extra Feature Engineering.

- Convolutional Neural Networks:

Convolutional Neural Networks (CNNs) (LeCun et al., 1998) is a key aspect in modern Computer Vision tasks like pattern object detection, image classification, pattern recognition tasks, etc. A CNN uses convolution kernels to convolve with the original images or feature maps to extract higher-level features, thus resulting in a very powerful tool for Computer Vision tasks.

- Modern Object Detection Algorithms:

CNN based object detection algorithms can be classified into 2 categories: Multi-Stage Detectors and Single-Stage Detectors.

1.3 OBJECTIVE

We propose a two-stage CNN architecture, where the first stage detects human faces, while the second stage uses a lightweight image classifier to classify the faces detected in the first stage as either 'Mask' or 'No Mask' faces and draws bounding boxes around them along with the detected Class name. This algorithm was further extended to videos as well. The detected faces are then tracked between frames using an object tracking algorithm, which makes the detections robust to the noise due to motion blur. This system can then be integrated with an image or video capturing device like a CCTV camera, to track safety violations, promote the use of face masks, and ensure a safe working environment.

1.4 APPLICATION

The system can be used in the following places to identify people with or without masks:

1. Offices – Manufacturers, retail, other SMEs and corporate giants
2. Hospitals/healthcare organizations
3. Airports and railway stations
4. Sports venues
5. Entertainment and hospitality industry
6. Densely populated areas

Analysing the current scenario, government and private organizations want to make sure that everyone working or visiting a public or private place is wearing masks throughout the day. The face mask detection platform can quickly identify the person with a mask, using cameras and analytics. Depending upon the requirements, the system is also adaptable to the latest technology and tools i.e.; you can add contact numbers or email addresses in the system to send an alert to the one who has not worn the mask. You can also send an alert to the person whose face is not recognizable in the system.

2. PROJECT PLANNING

Agenda

To enlist and visualize the steps to finalize the project to break down the problem into a number of smaller tasks, allocating tasks among members and setting tentative deadlines for completion.

2.1 TOOLS:

1. Python (Jupyter Notebook)

Python is a remarkably powerful dynamic, object-oriented programming language that is used in a wide variety of application domains. It offers strong support for integration with other languages and tools, and comes with extensive standard libraries. To be precise, the following are some distinguishing features of Python:
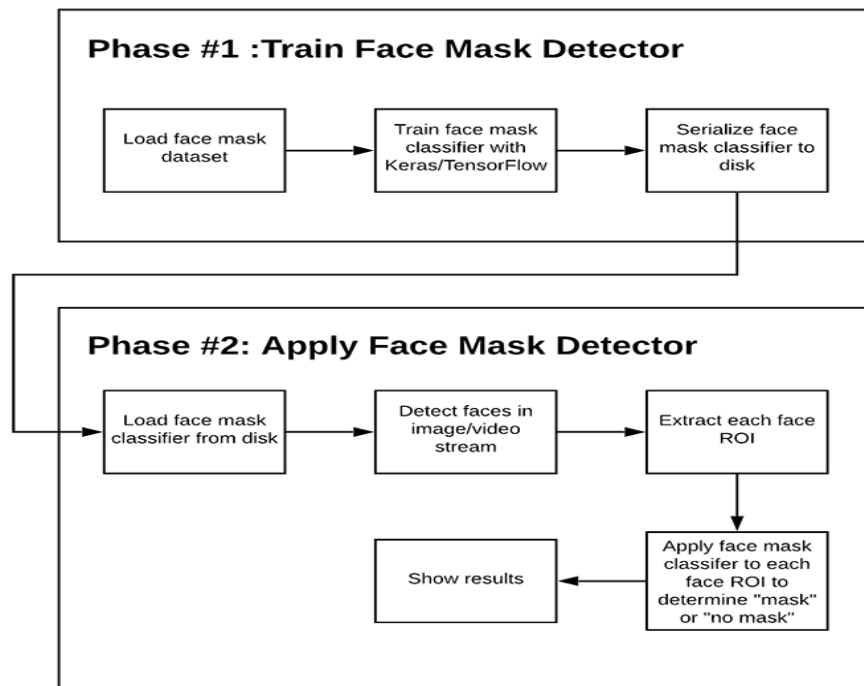
- Very clear, readable syntax.

- Strong introspection capabilities.

- Full modularity.

- Exception-based error handling.

- High level dynamic data types.

- Supports object oriented, imperative and functional programming styles.
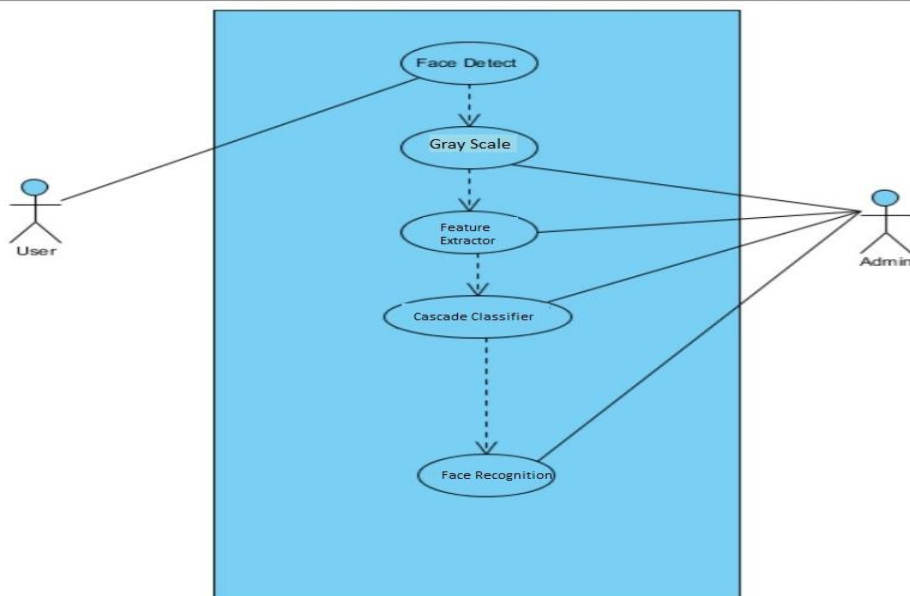
- Embeddable.

- Scalable Mature.

2. OpenCV

OpenCV is a library of programming functions for real time computer vision originally developed by Intel and now supported by Willogarage. It is free for use under the open source BSD license. The library has more than five hundred optimized algorithms. It is used around the world, with forty thousand people in the user group. Uses range from interactive art, to mine inspection, and advanced robotics. The library is mainly written in C, which makes it portable to some specific platforms such as Digital Signal Processor. Wrappers for languages such as C, Python, Ruby and Java (using JavaCV) have been developed to encourage adoption by a wider audience. The recent releases have interfaces for C++. It focuses mainly on real-time image processing. OpenCV is a cross-platform library, which can run on Linux, Mac OS and Windows. To date,

2.2 SYSTEM MODEL

First we loaded the data from the dataset. Then, we classified into mask and no mask. After classification, we stored them in Dataset and Target set. After this, we train the dataset with the required model and got the accuracy 96%. Then, we loaded the model and loaded as casket classifier for face detection. Then we read the video and converted them into Gray Scale. We resized the image and reshaped into 4 dimensions.

Phase #1 :Train Face Mask Detector

Load face mask dataset → Train face mask classifier with Keras/TensorFlow → Serialize face mask classifier to disk

Phase #2: Apply Face Mask Detector

Load face mask classifier from disk → Detect faces in image/video stream → Extract each face ROI → Apply face mask classifer to each face ROI to determine "mask" or "no mask" → Show results

Usecase Diagram

User

Admin

Face Detect → Gray Scale → Feature Extractor → Cascade Classifier → Face Recognition

2.3 WORKING

The collected RAW data was passed through: -
STAGE 1 (FACE DETECTOR) AND THE INTERMEDIATE PROCESSING BLOCK OF THE ARCHITECTURE.
This process was carried out to ensure that the
distribution and nature of training data for Stage 2 match the expected input for Stage 2 during the final deployment.

Data Preprocessing
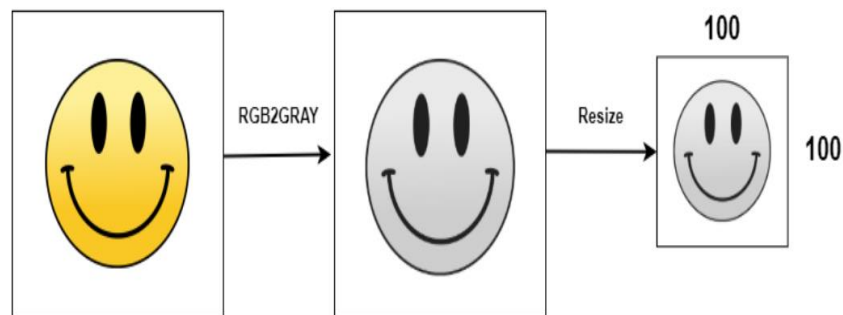
STAGE 2

For the second stage, three CNN classifiers were trained for classifying images as masked or unmasked. The models were trained using the Keras framework.

Data augmentation was performed using the ImageDataGenerator class in Keras. The input image size was set as 100X100. We Selected an initial learning rate of 0.001. Besides this, the training process included checkpointing the weights for best loss, reducing the learning rate on the plateau, and early stopping. Each model was trained for the dataset and the lowest validation loss was selected

Convolutional Neural Network Architecture
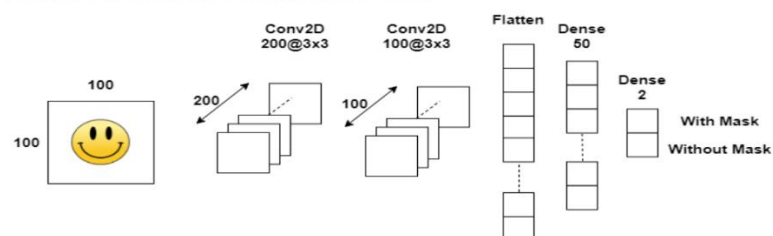
STAGE 3

Detecting Faces with and without masks
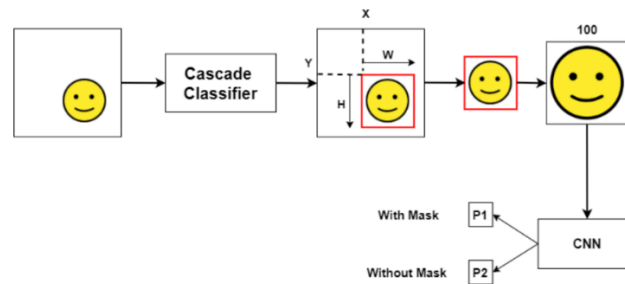
Combining all the components of our architecture, we thus get a highly accurate and robust Face Mask Detection System.

Retina Face was selected as our Face Detector in Stage 1, while the CNN based model was selected as our Face Mask. Classifier in Stage 2. The resultant system exhibits high performance and has the capability to detect face masks in images with mask faces over a wide range of angles.

Until now, we have seen that our system shows high performance over images,

Overcoming most of the issues commonly faced in object detection in images. For real-world scenarios, it is beneficial that the detection system to work well over video feeds as well.

3. IMPLEMENTATION

3.1 DATASET:

The three face mask classifier models were trained on our dataset. The dataset images for masked and unmasked faces were collected from image datasets available in the public domain, along with some data scraped from the Internet. Masked images were obtained from the Real-world Masked Face Recognition
Dataset and Face Mask Detection dataset on Kaggle
Our dataset also includes images of improperly worn face masks or hands covering the face, which get classified as non-masked faces.

The Dataset
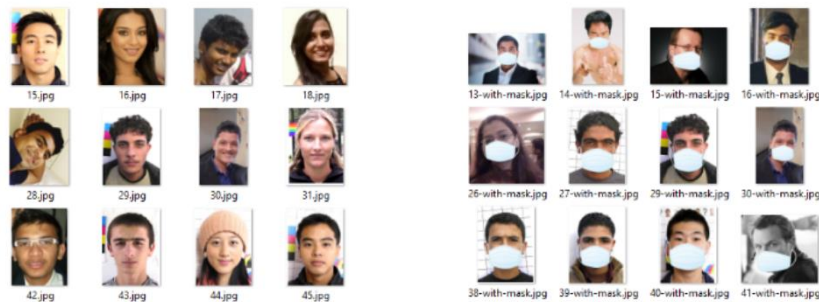
3.2 METHODOLOGY:

Under Machine Learning this problem falls under the Image processing problem and could be solved using OpenCv.

With the help of the comparison in accuracy and efficiency of the several algorithms used based on the research done we came to a conclusion to examine the problem using OpenCv image processing algorithms.

We have used a sample image and video for training the model. Steps involved in training the model are as follows:

Preprocess i.e. Re-sizing and changing the image to gray scale.

Localise i.e. converting the image to binary for segmentation

3.3 SCREENSHOTS OF THE CODE

1. Data Preprocessing

```
In [6]:  import tensorflow
```

```
In [2]:  import cv2,os

         data_path='dataset'
         categories=os.listdir(data_path)
         labels=[i for i in range(len(categories))]

         label_dict=dict(zip(categories,labels))

         print(label_dict)
         print(categories)
         print(labels)

         {'with mask': 0, 'without mask': 1}
         ['with mask', 'without mask']
         [0, 1]
```

```
In [3]:  img_size=100
         data=[]
         target=[]

         for category in categories:
             folder_path=os.path.join(data_path,category)
             img_names=os.listdir(folder_path)
```

```
         for category in categories:
             folder_path=os.path.join(data_path,category)
             img_names=os.listdir(folder_path)

             for img_name in img_names:
                 img_path=os.path.join(folder_path,img_name)
                 img=cv2.imread(img_path)

                 try:
                     gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
                     #Coverting the image into gray scale
                     resized=cv2.resize(gray,(img_size,img_size))
                     #resizing the gray scale into 100x100, since we need a fixed common size for all the images in the dataset
                     data.append(resized)
                     target.append(label_dict[category])
                     #appending the image and the label(categorized) into the list (dataset)

                 except Exception as e:
                     print('Exception:',e)
                     #if any exception rasied, the exception will be printed here. And pass to the next image
```

```
In [7]:  import numpy as np

         data=np.array(data)/255.0
         data=np.reshape(data,(data.shape[0],img_size,img_size,1))
         target=np.array(target)

         from keras.utils import np_utils

         new_target=np_utils.to_categorical(target)
```

```
In [9]:  np.save('data',data)
         np.save('target',new_target)
```

```
In [1]:  from keras.models import load_model
         import cv2
         import numpy as np
```

```
In [2]:  model = load_model('model-017.model')

         face_clsfr=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

         source=cv2.VideoCapture(0)

         labels_dict={0:'MASK',1:'NO MASK'}
         color_dict={0:(0,255,0),1:(0,0,255)}
```

```
In [ ]:  while(True):

             ret,img=source.read()
             gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
             faces=face_clsfr.detectMultiScale(gray,1.3,5)

             for x,y,w,h in faces:

                 face_img=gray[y:y+w,x:x+w]
                 resized=cv2.resize(face_img,(100,100))
                 normalized=resized/255.0
                 reshaped=np.reshape(normalized,(1,100,100,1))
                 result=model.predict(reshaped)

                 label=np.argmax(result,axis=1)[0]

                 cv2.rectangle(img,(x,y),(x+w,y+h),color_dict[label],2)
                 cv2.rectangle(img,(x,y-40),(x+w,y),color_dict[label],-1)
                 cv2.putText(img, labels_dict[label], (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)


             cv2.imshow('LIVE',img)
             key=cv2.waitKey(1)

             if(key=='q'):
                 break

         cv2.destroyAllWindows()
         source.release()
```

2. Training the CNN


```
Epoch 1/20
31/31 [==============================] - ETA: 0s - loss: 0.6939 - accuracy: 0.4889WARNING:tensorflow:From c:\users\govind yadav
\appdata\local\programs\python\python37\lib\site-packages\tensorflow\python\training\tracking\tracking.py:111: Model.state_upda
tes (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
WARNING:tensorflow:From c:\users\govind yadav\appdata\local\programs\python\python37\lib\site-packages\tensorflow\python\traini
ng\tracking\tracking.py:111: Layer.updates (from tensorflow.python.keras.engine.base_layer) is deprecated and will be removed i
n a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
INFO:tensorflow:Assets written to: model-001.model\assets
31/31 [==============================] - 60s 2s/step - loss: 0.6939 - accuracy: 0.4889 - val_loss: 0.6935 - val_accuracy: 0.467
7
Epoch 2/20
31/31 [==============================] - 59s 2s/step - loss: 0.6931 - accuracy: 0.5091 - val_loss: 0.6936 - val_accuracy: 0.467
7
Epoch 3/20
31/31 [==============================] - 58s 2s/step - loss: 0.6931 - accuracy: 0.5091 - val_loss: 0.6937 - val_accuracy: 0.467
7
Epoch 4/20
31/31 [==============================] - 59s 2s/step - loss: 0.6930 - accuracy: 0.5091 - val_loss: 0.6937 - val_accuracy: 0.467
7
Epoch 5/20
31/31 [==============================] - 56s 2s/step - loss: 0.6930 - accuracy: 0.5091 - val_loss: 0.6940 - val_accuracy: 0.467
7
Epoch 6/20
28/31 [=========================>...] - ETA: 4s - loss: 0.6933 - accuracy: 0.5011
```

```python
In [21]: from matplotlib import pyplot as plt

plt.plot(history.history['loss'],'r',label='training loss')
plt.plot(history.history['val_loss'],label='validation loss')
plt.xlabel('# epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```



```python
In [22]: plt.plot(history.history['accuracy'],'r',label='training accuracy')
plt.plot(history.history['val_accuracy'],label='validation accuracy')
plt.xlabel('# epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

```python
from keras.models import Sequential
from keras.layers import Dense,Activation,Flatten,Dropout
from keras.layers import Conv2D,MaxPooling2D
from keras.callbacks import ModelCheckpoint

model=Sequential()

model.add(Conv2D(200,(3,3),input_shape=data.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
#The first CNN layer followed by Relu and MaxPooling layers

model.add(Conv2D(100,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
#The second convolution layer followed by Relu and MaxPooling layers

model.add(Flatten())
model.add(Dropout(0.5))
#Flatten layer to stack the output convolutions from second convolution layer
model.add(Dense(50,activation='relu'))
#Dense layer of 64 neurons
model.add(Dense(2,activation='softmax'))
#The Final layer with two outputs for two categories

model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```python
from sklearn.model_selection import train_test_split

train_data,test_data,train_target,test_target=train_test_split(data,target,test_size=0.1)
```

```python
checkpoint = ModelCheckpoint('model-{epoch:03d}.model',monitor='val_loss',verbose=0,save_best_only=True,mode='auto')
history=model.fit(train_data,train_target,epochs=20,callbacks=[checkpoint],validation_split=0.2)
```

```
Epoch 1/20
31/31 [==============================] - ETA: 0s - loss: 0.6939 - accuracy: 0.4889WARNING:tensorflow:From c:\users\govind yadav
\appdata\local\programs\python\python37\lib\site-packages\tensorflow\python\training\tracking\tracking.py:111: Model.state_upda
tes (from tensorflow.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
WARNING:tensorflow:From c:\users\govind yadav\appdata\local\programs\python\python37\lib\site-packages\tensorflow\python\traini
ng\tracking\tracking.py:111: Layer.updates (from tensorflow.python.keras.engine.base_layer) is deprecated and will be removed i
n a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
INFO:tensorflow:Assets written to: model-001.model\assets
31/31 [==============================] - 60s 2s/step - loss: 0.6939 - accuracy: 0.4889 - val_loss: 0.6935 - val_accuracy: 0.467
```
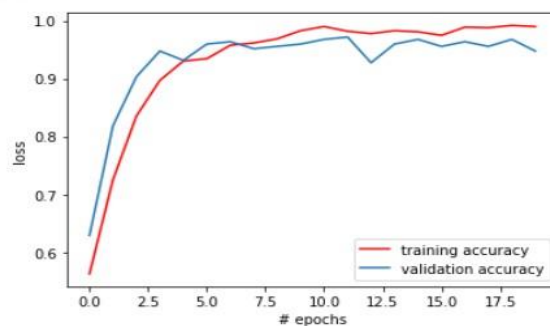
3. Detecting Masks

```python
In [22]:  plt.plot(history.history['accuracy'],'r',label='training accuracy')
          plt.plot(history.history['val_accuracy'],label='validation accuracy')
          plt.xlabel('# epochs')
          plt.ylabel('loss')
          plt.legend()
          plt.show()
```
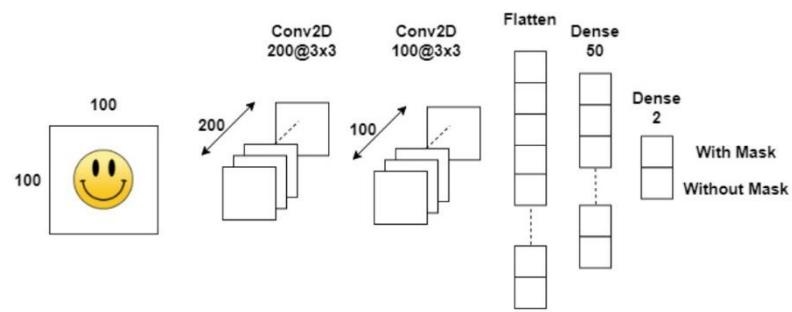


```python
In [23]:  print(model.evaluate(test_data,test_target))
```

```
138/138 [==============================] - 6s 44ms/step
[0.14019694376358952, 0.9637681245803833]
```

Convolutional Neural Network Architecture


```
In [1]: import numpy as np

        data=np.load('data.npy')
        target=np.load('target.npy')

        #Loading the save numpy arrays in the previous code
```

```
In [2]: from keras.models import Sequential
        from keras.layers import Dense,Activation,Flatten,Dropout
```

4. RESULTS AND DISCUSSIONS

The following results were observed after running the project on Open CV: -

4.1   With Mask

4.2   Without Mask

5. CONCLUSION

Corporate giants from various verticals are turning to AI and ML, leveraging technology at the service of humanity amid the pandemic. Digital product development companies are launching mask detection API services that enable developers to build a face mask detection system quickly to serve the community amid the crisis. The technology assures reliable and real-time face detection of users wearing masks. Besides, the system is easy to deploy into any existing system of a business while keeping the safety and privacy of users' data. So the face mask detection system is going to be the leading digital solution for most industries, especially retail, healthcare, and corporate sectors.

6. FUTURE WORKS

There are a number of aspects we plan to work on shortly:

- Currently, the model gives 5 FPS inference speed on a CPU. In the future, we plan to improve this up to 15 FPS, making our solution deployable for CCTV cameras, without the need of a GPU.

- The use of Machine Learning in the field of mobile deployment is rising rapidly. Hence, we plan to port our models to their respective TensorFlow Lite versions.

- Our architecture can be made compatible with TensorFlow Run Time (TFRT), which will increase the inference performance on edge devices and make our models efficient on multithreading CPUs.

- Stage 1 and Stage 2 models can be easily replaced with improved models in the future, that would give better accuracy and lower latency.

7. SAMPLE INDIVIDUAL CONTRIBUTION REPORT:

MASK DETECTION

DATTATTRAYA DEB

1705304

Abstract: Face Detection has evolved as a very popular problem in Image processing and Computer Vision. Many new algorithms are being devised using convolutional architectures to make the algorithm as accurate as possible. These convolutional architectures have made it possible to extract even the pixel details.

The end of 2019 witnessed the outbreak of Coronavirus Disease 2019 (COVID-19), which has continued to be the cause of plight for millions of lives and businesses even in 2020. As the world recovers from the pandemic and plans to return to a state of normalcy, there is a wave of anxiety among all individuals, especially those who intend to resume in-person activity. Studies have proved that wearing a face mask significantly reduces the risk of viral transmission as well as provides a sense of protection. However, it is not feasible to manually track the implementation of this policy. Technology holds the key here. We introduce a Deep Learning based system that can detect instances where face masks are not used properly. Our system consists of a dual-stage Convolutional Neural Network (CNN) architecture capable of detecting masked and unmasked faces and can be integrated with pre-installed CCTV cameras. This will help track safety violations, promote the use of face masks, and ensure a safe working environment.

Individual contribution and findings: Ideation, Objective, Results and Discussions
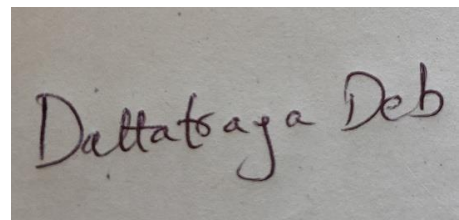
Individual contribution to project report preparation: Implementation, Project Planning, Conclusion, Future Scope, References.

Individual contribution for project presentation and demonstration: Presentation Introduction, Working, Future Scope.

Full Signature of Supervisor: Full Signature of Student:

……………………………………

MASK DETECTION

GOVIND YADAV

1705310

Abstract: Face Detection has evolved as a very popular problem in Image processing and Computer Vision. Many new algorithms are being devised using convolutional architectures to make the algorithm as accurate as possible. These convolutional architectures have made it possible to extract even the pixel details.

The end of 2019 witnessed the outbreak of Coronavirus Disease 2019 (COVID-19), which has continued to be the cause of plight for millions of lives and businesses even in 2020. As the world recovers from the pandemic and plans to return to a state of normalcy, there is a wave of anxiety among all individuals, especially those who intend to resume in-person activity. Studies have proved that wearing a face mask significantly reduces the risk of viral transmission as well as provides a sense of protection. However, it is not feasible to manually track the implementation of this policy. Technology holds the key here. We introduce a Deep Learning based system that can detect instances where face masks are not used properly. Our system consists of a dual-stage Convolutional Neural Network (CNN) architecture capable of detecting masked and unmasked faces and can be integrated with pre-installed CCTV cameras. This will help track safety violations, promote the use of face masks, and ensure a safe working environment.

Individual contribution and findings: Ideation, Coding, Implementation, System Testing, Project Planning.

Individual contribution to project report preparation: Abstract, Dataset, working

Individual contribution for project presentation and demonstration: Project Planning, Introduction and Future Scope, Rectification.

Full Signature of Supervisor: Full Signature of Student:

Govind Yadav

…………………………………

8. REFERENCES

1. Chollet, F., & others, (2015), Keras, https://keras.io,
    Available at: https://github.com/fchollet/keras.

2. Deng, J., Guo, J., Ververas, E., Kotsia, I., and Zafeiriou, S., RetinaFace: Single-Shot Multi-Level Face Localisation in the Wild, 2020, IEEE/CVF   Conference on Computer Vision and Pattern Recognition

3. Ejaz, M.S., Islam, M.R., Sifatullah, M., Sarker, A., Implementation of principal component analysis on masked and non-masked face recognition, 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), 2019, pp. 1–5.

4. Girshick, R., Donahue, J., Darrell, T., and Malik, J., Rich feature hierarchies for accurate object detection and semantic segmentation, Proceedings of the IEEE Conference on Computer Vision and pattern recognition, 2014, pp. 580–587. Girshick, R., Fast R-CNN, Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.

5. Adrian Rosebrock, sentdex, Prajna Bhandary

6. AI with Thakshilla

7. https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/