University of Regina

ENIN 880CA
ADVANCED ARTIFICIAL NEURAL NETWORK

PROJECT REPORT
FACE MASK DETECTION SYSTEM USING CNN AND TRANSFER LEARNING

SUBMITTED TO

Dr. RENE MAYORGA
PROFESSOR, INDUSTRIAL SYSTEMS ENGINEERING

SUBMITTED BY

DHAVAL BHAILALBHAI PATEL [200439819]
VEDANG PANDYA [200427798]

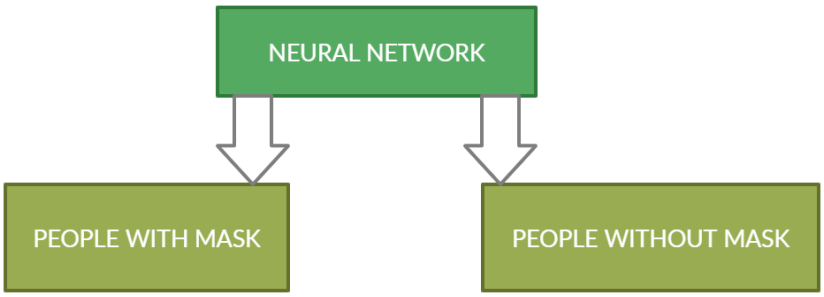
FACULTY OF GRADUATE STUDIES AND RESEARCH
FACULTY OF ENGINEERING AND APPLIED SCIENCES

INDEX

PROBLEM STATEMENT

Area of interest – health care, computer vision.

As we all know covid19 outbreak took place in China in 2019 and since then the cases are rising exponentially in every country. Several measures were introduced in order to curb the spread of the contagious disease like social distancing, wearing mask in public places, etc. Social distancing is something which is often neglected by the people whereas the mask requirement atleast serve the purpose. However, people still evade the restrictions using several tricks. This is where neural network comes in handy. We can build a neural network for detecting the mask in public places.

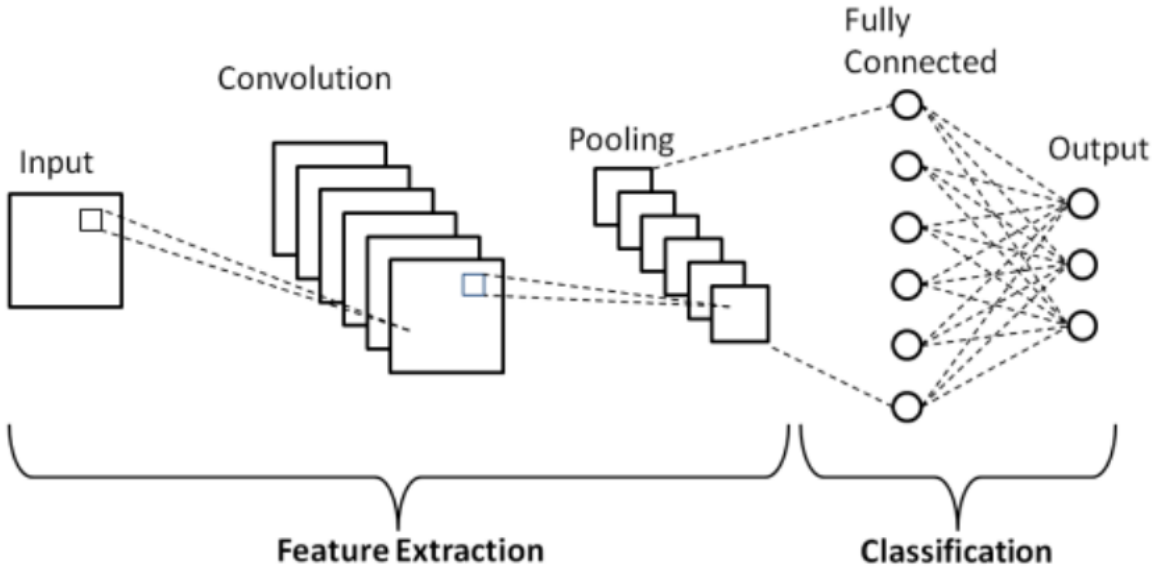


The above diagram shows the decision-making process which neural network will be performing.

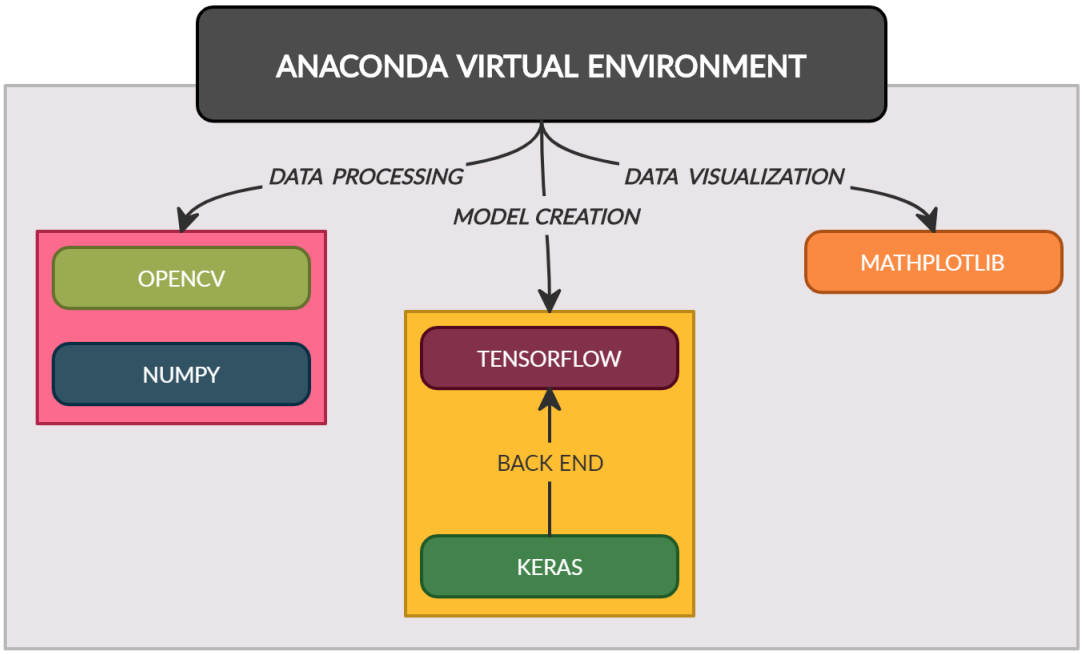
Legislation regarding face mask in Canada:

Public health officials are requesting people to wear mask depending on rates of infection and transmission [5]. The use of face mask is mandatory by various jurisdictions in many indoor public spaces and even on public transportation. Even the public gatherings are restricted in some provinces depending upon the number of the active cases.

CONVOLUTIONAL NEURAL NETWORK



ESSENTIAL LIBRARIES AND TOOLS REQUIRED



PROPOSED ALGORITHM

Our aim for the proposed solution of the Live mask detection system is to detect if the subject has worn mask or not from live feed. Proposed algorithm for the solution is as follows:

1] Train the neural network on the dataset and save the best trained CNN model.

2] Take the live feed from video source and split it into images

3] Apply classifier on the image to isolate area of interest [i.e the face regions]

4] Apply preprocessing on the classified image.

5] Feed the image in the best trained CNN model and identify if the face is with mask or without mask.

DATASET

The data-source for this project is from **KAGGLE** prepared by **Omkar Gurav**. The data comprised a **total of 7553** augmented images with RGB channels. These are divided further into two sub-folders namely masked and non-masked images folder with **3725 images and 3825 images** respectively. We have gathered images from internet and added them to the existing dataset.

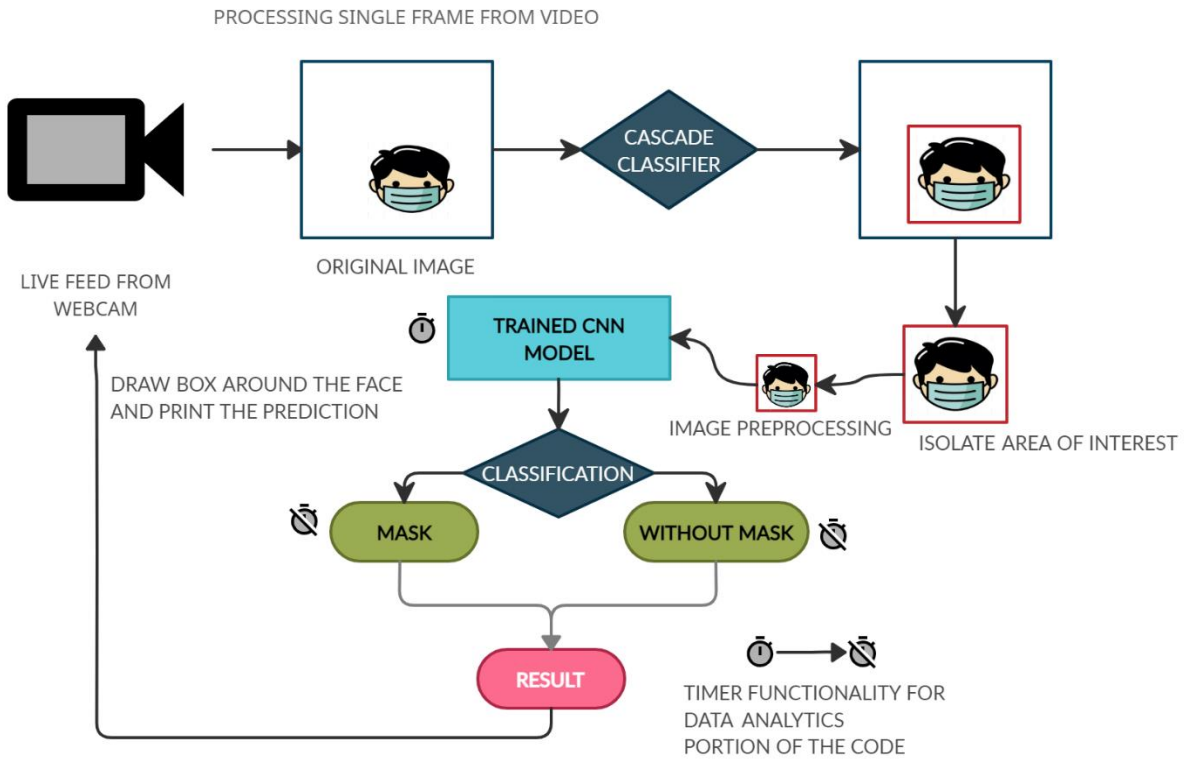
CODE

We tried two approaches to train the NN for this problem

1] Training Convolutional neural network for mask detections from scratch.

2] Apply transfer learning on pretrained MobileNetV2 model

METHOD 1



WORKFLOW FOR PROPOSED CNN

We solution can be divided into three stages:

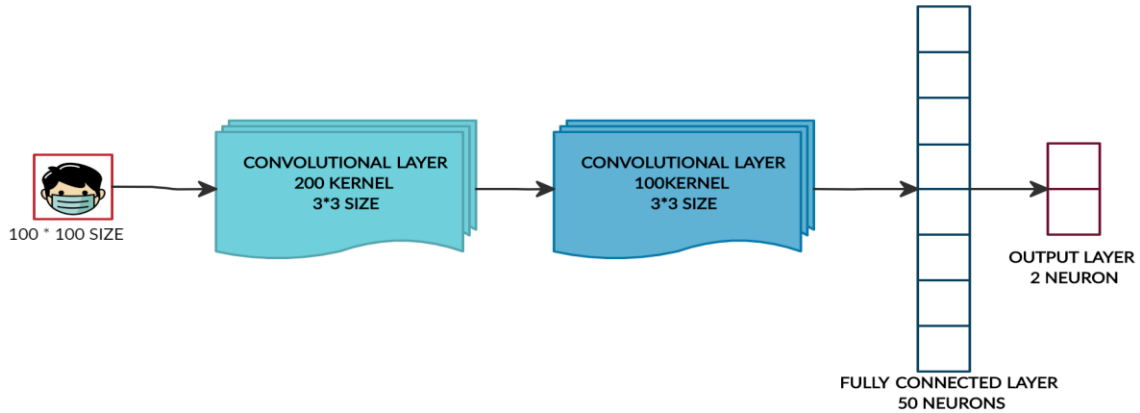
1) Stage 1: The first stage is the data preprocessing and it is shown below:

```
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)
img_size=100
data=[]
target=[]
counter=0
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)
        print(img_name)
        counter=counter +1
        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)
    print(counter)
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)
np.save('data',data)
np.save('target',new_target)
```

The above code converts the input image into a 100 * 100 greyscale images. The reason being that, we consider that in mask detection our main objective is not to identify the color of the mask rather we need to decrease the training time for our NN model. So, by removing the color and by resizing the original image we can focus on feature that helps us to identify required features for detection.

2) Stage 2: In the second stage we used the pre-processed images from the first stage as an input for training our NN.



```
import numpy as np
data=np.load('data.npy')
target=np.load('target.npy')
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=['accu
racy'])

from sklearn.model_selection import train_test_split
train_data,test_data,train_target,test_target=train_test_split(data,target,te
st_size=0.1)
checkpoint =
ModelCheckpoint('model{epoch:03d}.model',monitor='val_loss',verbose=0,save_be
st_only=True,mode='auto')
history=model.fit(train_data,train_target,epochs=10,callbacks=[checkpoint],va
lidation_split=0.2)
```

The skeleton of the network is same as shown in figure. We have used Softmax function as an activation function because softmax functions performs best when we have to normalize the output and convert tensors/list from previous layer into weight sum probability whose total probability sum up to unity [Best case scenarios binary classification tasks].linear and sigmoidal functions fails to achieve this.

3) Stage 3: Deployment of the model using live stream from WEBCAM as an input

```
from keras.models import load_model
import cv2
```

```
import numpy as np
import tkinter
from tkinter import messagebox
import smtplib
from datetime import datetime
import matplotlib.pyplot as plt
import cv2
#we load the best trained model
model = load_model('model-10.model')

labels_dict={0:'MASK',1:'NO MASK'}
color_dict={0:(0,255,0),1:(0,0,255)}
m_count=0
nm_count=0
m_list=[]
nm_list=[]
time_list=[]
counter=0
#start of the timer for the session
StartdateTimeObj = datetime.now()
#Enable live feed from the web camera
videocapture = cv2.VideoCapture(0)
scale_factor = 1.3

while 1:
    #load haarcascade classifier
    face_clsfr=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    source=cv2.VideoCapture(0)
    ret,img=source.read()
    gray=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    #Segregate the faces using haarcascade classifier from live feed
    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))
        #feed a 100 by 100 grayscale image to the network
        result=model.predict(reshaped)

        label=np.argmax(result,axis=1)[0]
        #Code to draw label face and predictions from model
        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)
       #only update if face is detected with mask in a frame
       if(label==0):
            m_count = m_count+1
            counter=counter + 1
        #only update if face is detected without mask in a frame
        if(label==1):
            nm_count = nm_count + 1
            counter=counter + 1
        if counter==2:
            pltEndTime=datetime.now()
```

```
            m_list.append(m_count)
            nm_list.append(nm_count)
            time_list.append(((StartdateTimeObj-pltEndTime).seconds))
            counter = 0
    cv2.imshow('face', img)
    k = cv2.waitKey(30) & 0xff
    if(k == 27):
        break

cv2.destroyAllWindows()
source.release()
#end of the time sequence for the session
EnddateTimeObj = datetime.now()
```

The In the stage 3 we have applied pre-train model on the live feed from webcam. Where there were no real time operations to be calculated for data analytics purpose, we have experienced a very high frame rate and our configuration was adequate to run the code. But as we enabled the analytics feature in the code, we experienced a medium frame rate that was at acceptable standards when we consider feature-frame rate trade-off.

RESULTS OF PROPOSED NEURAL NETWORK:

Training Neural Network Plots

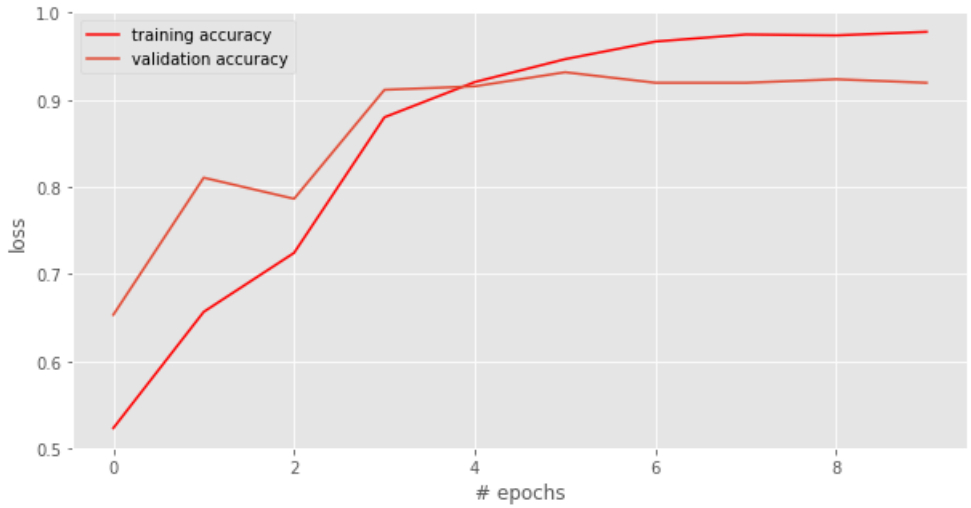


Figure: plot of training accuracy and validation accuracy.

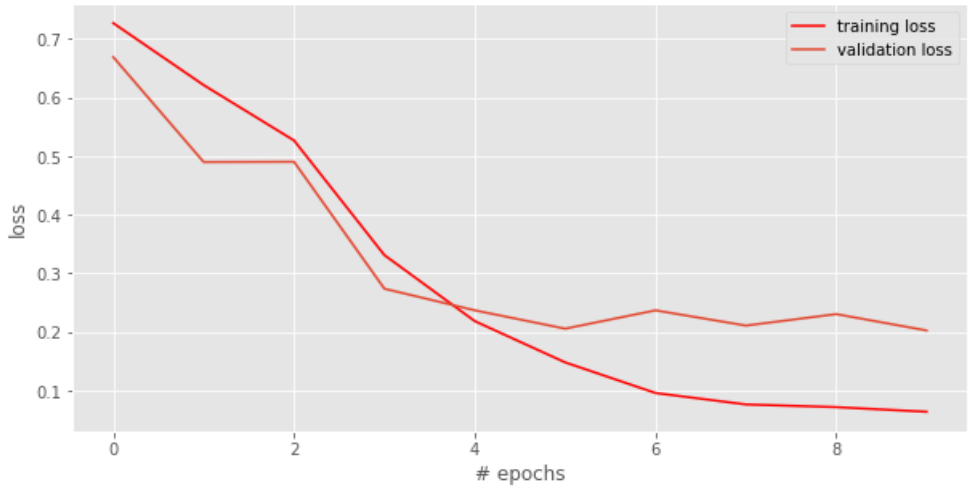
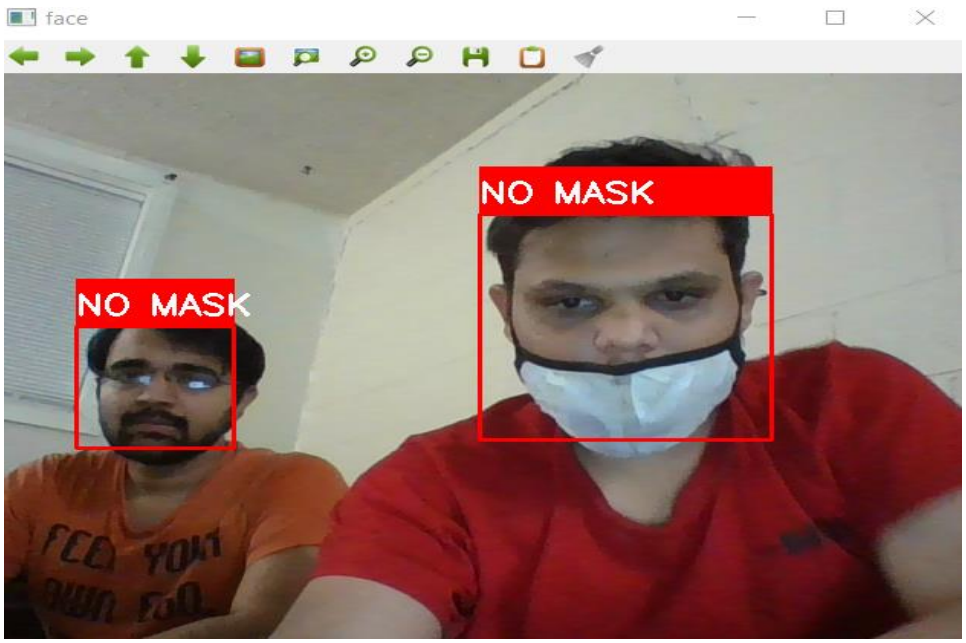


Figure: plot of training loss and validation loss.

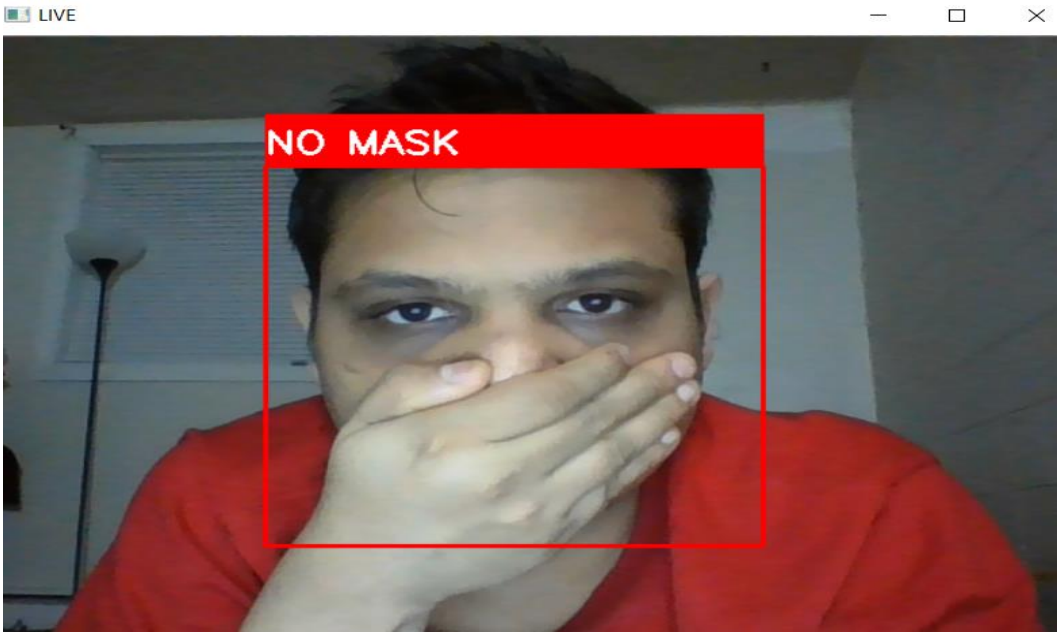
We ran our neural network for some time with different combinations and results are shown below:

Case 1) In this case there were two people, one without mask and other with partially covered mask. It was critical for the neural network to distinguish between the partially covered and fully covered face condition. Our results shown that the neural network was able to discern such condition. It gave accurate result as shown below:



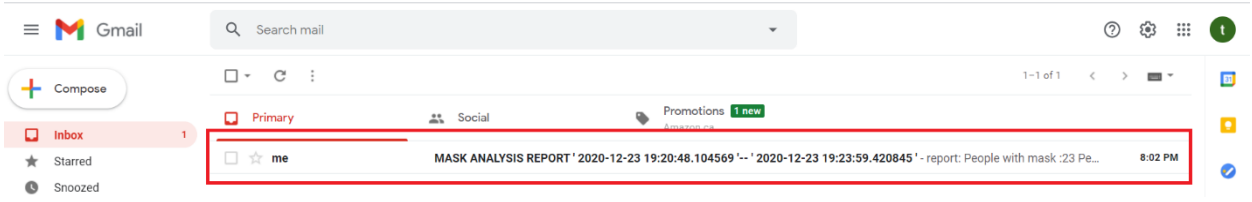
Case 2) In this case once again there were two people, one without mask looking into mobile and other with mask. The neural network gave accurate output for both the people and the result is shown below:

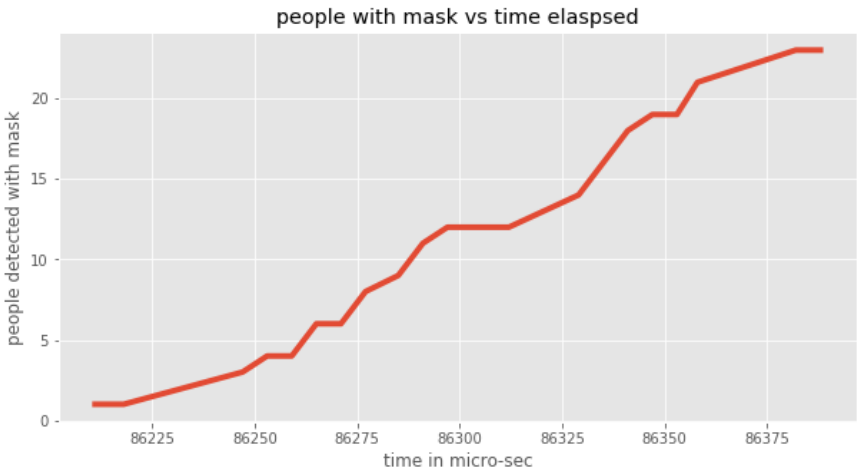
Case3) In this case the person was covering his face using his hands and neural network output was shown to be as a person without mask.



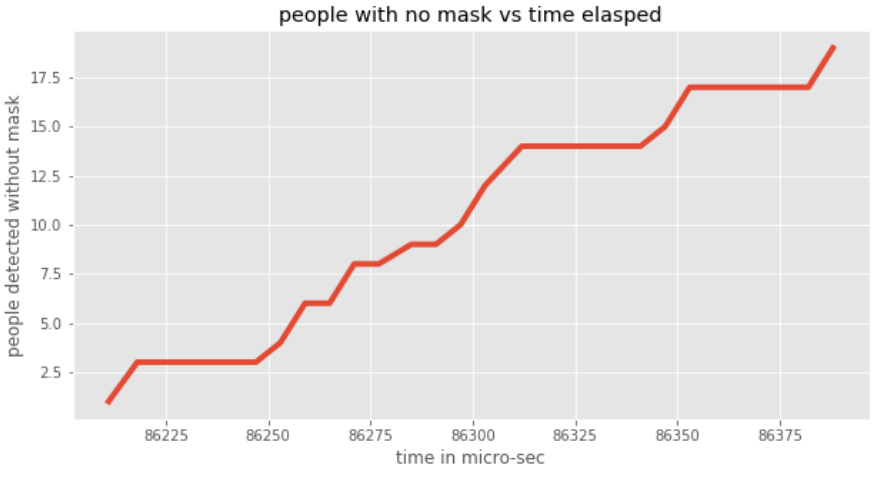
Data analytics on the data gathered from predictions

An auto-generated report is e-mailed to the concerned authority at the end of the session for mask detections code. Report comprises of the following sections.

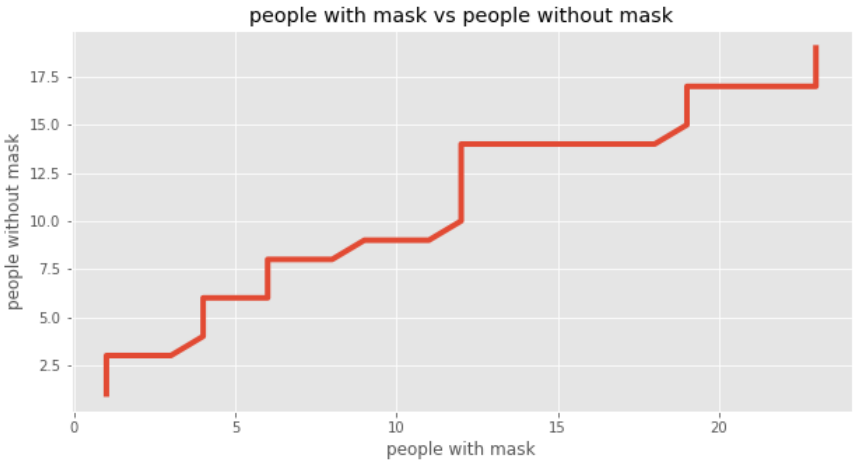




We can infer from graph that how many people were detected with mask at a given time range. Also, total people with mask detected during this session.



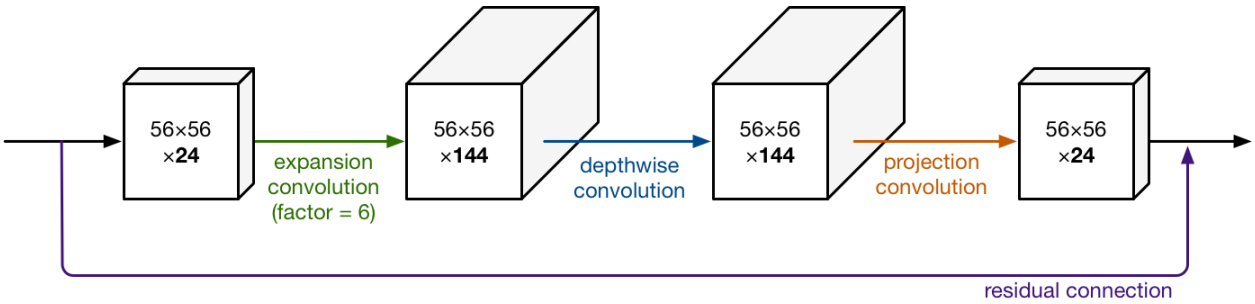
We can inference from graph that how many people were detected without mask at a given time range. This graph will help us to find time slots where security must enforce laws stringently.



In Ideal case if all people are wearing mask this plot should be a horizontal line but if people without mask are detected it will spike. This can be helpful in real time monitoring

Current Benchmark method: MobilenetV2

This model was developed by the Google researchers and they had optimized this model for the mobile devices. This was computationally less expensive as it involved reduction in parameters and mathematical operations. This model - MobileNet uses depthwise separable convolutions, which comprises a depthwise and a pointwise convolution after one another.



Reinforcement learning Code for MobilenetV2

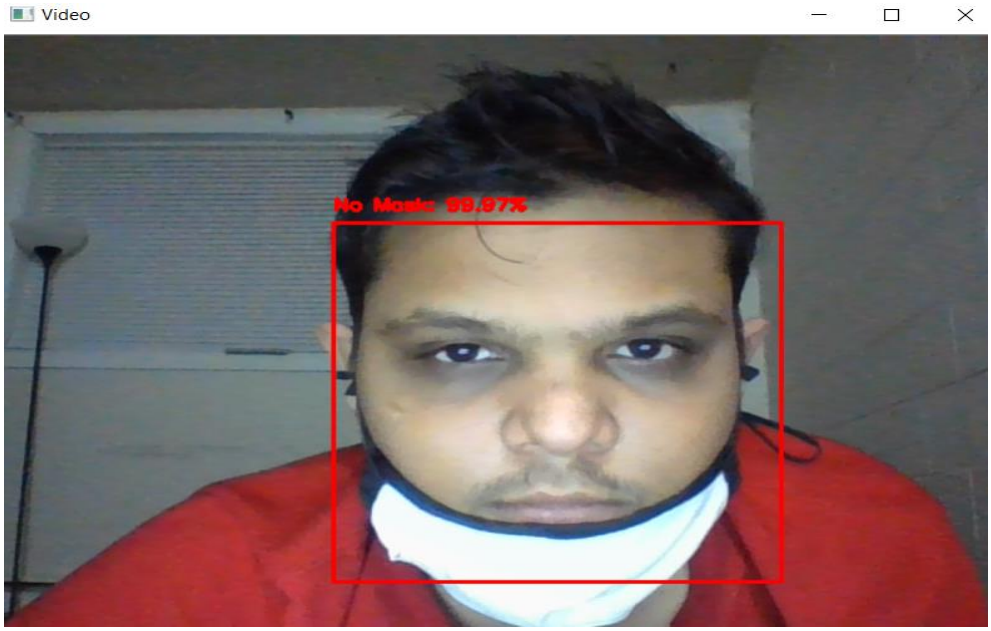
The below snippet modifies the top layer of MobileNetV2 to allow us to solve mask detection problem. We have added 3 layers dense layer(128), Dropout layer(0.5) and Dense layer(2). Here we have changed the same preprocessing code we have used in our custom NN model step 1 to create dataset of 224*224 grayscale images and feed it in NN for training [checkout github link for complete code].

```
baseModel = MobileNetV2(weights="imagenet", include_top=False,
    input_shape=(224, 224, 3))
# construct the head of the model that will be placed on top of the
# the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(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)
for layer in baseModel.layers:
        layer.trainable = False
```

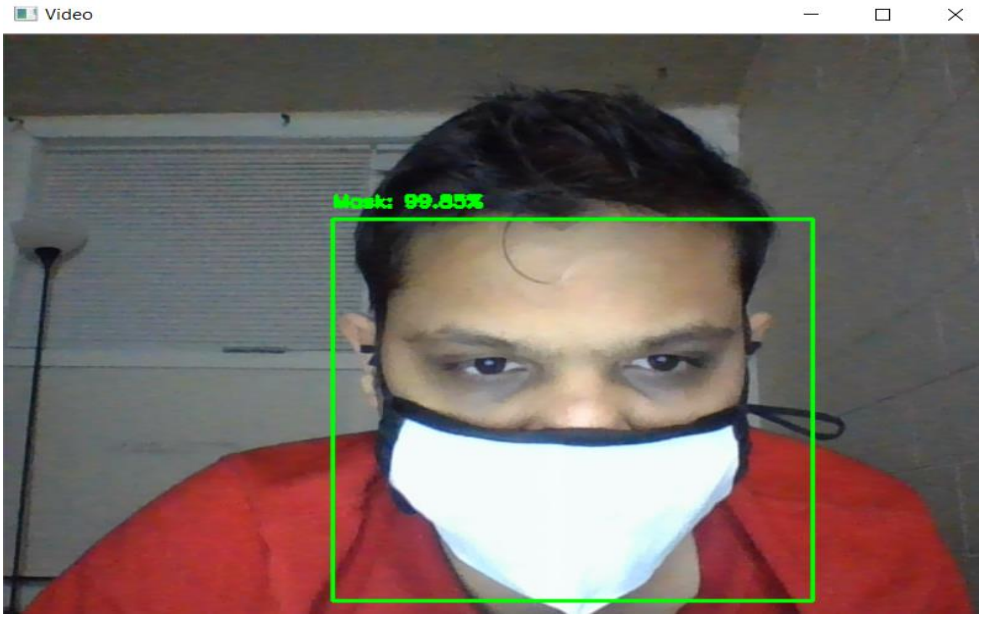
Results of benchmark method – MobilenetV2

We ran MobilenetV2 neural network with different combinations and results are shown below.

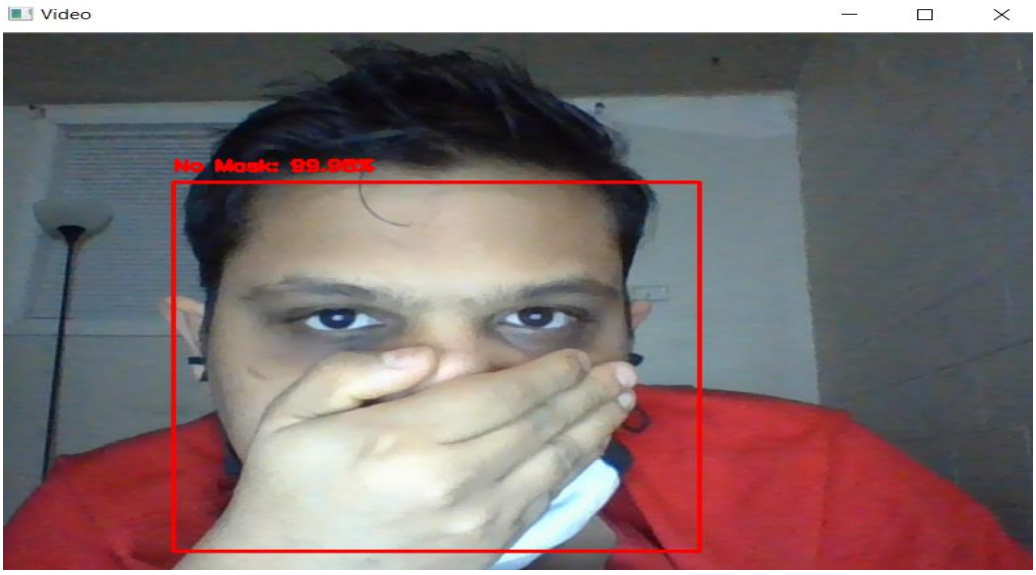
Case 1) In this case there was a person with partially covered mask and MobilenetV2 was able to conclude that it is a no_mask condition with 99.97% accuracy.



Case 2) In this case there was a person with fully covered mask and MobilenetV2 was able to conclude that it is mask condition with 99.85% accuracy.



Case3) In this case the person was covering his face using his hands and MobilenetV2 output was shown to be as a person without mask.



TRAINING PLOTS - REINFORCEMENT LEARNING MOBILENETV2

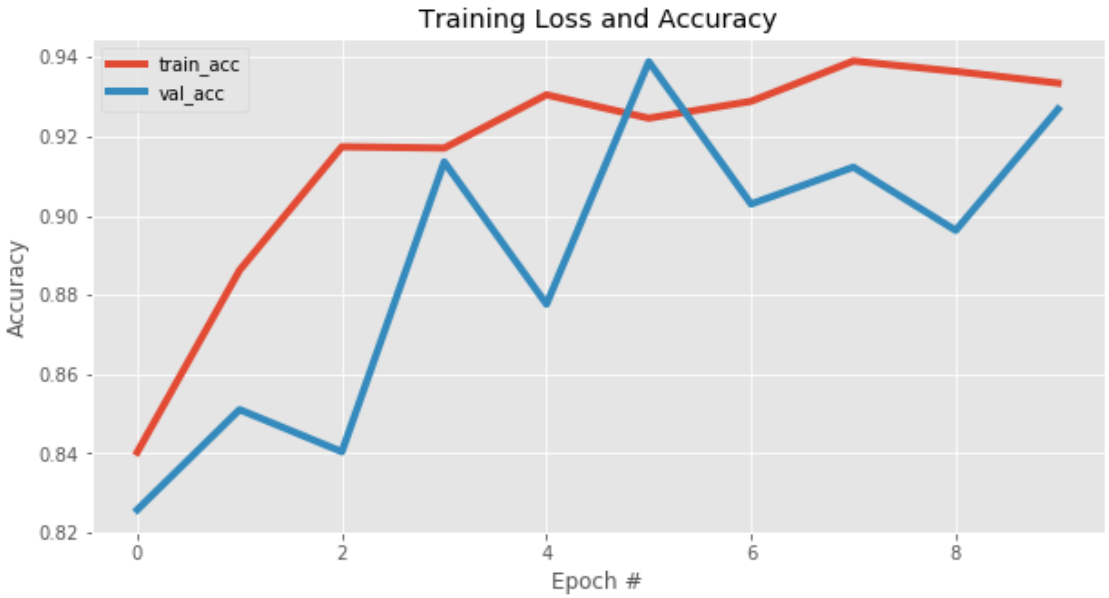


Figure: plot of training accuracy and validation accuracy.

Figure: plot of training loss and validation loss.

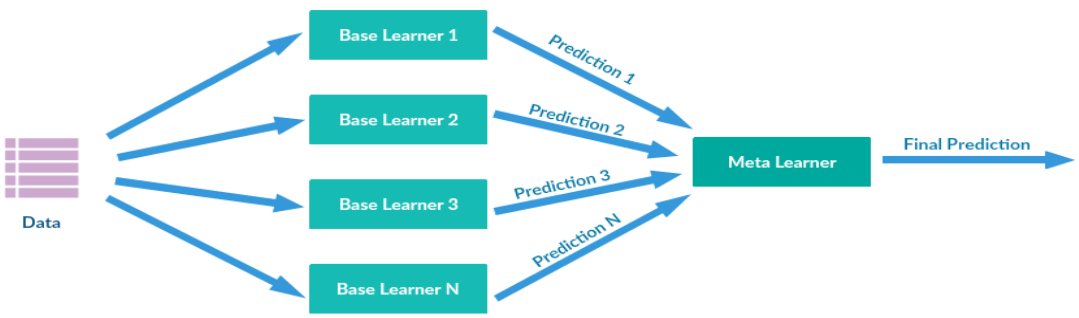
Comparison proposed neural network VS MobilenetV2

The comparison between our proposed neural network and benchmark – MobilenetV2 is tabulated below:

Sr.	Parameter	Proposed neural network	Benchmark – Mobilenetv2
1	**Number of epochs**	10	10
2	**Average training time for each epoch**	***Approximately 43 seconds***	***Approximately 383 seconds***
3	**Best training accuracy**	***97.98%***	***93.34%***
4	**Result for fully covered condition**	Detected the mask	Detected the mask
5	**Result for partially covered condition**	Detected the no mask condition	Detected the no mask condition
6	**Result for face covered with hands**	Detected the no mask condition	Detected the no mask condition

Future Work

We would like to use stacking approach in machine learning for further developing our custom model. Using our custom pretrained model with other model to detect if user has worn mask partially and which part of his face is exposed.



GITHUB LINK

- **https://github.com/Dhaval-B-Patel/AANN**

REFERENCES

1) "Convolutional Neural Networks." Convolutional Neural Networks, www.doc.ic.ac.uk/~jce317/introduction-cnns.html#top.
2) Phung, V.H.; Rhee, E.J. A Deep Learning Approach for Classification of Cloud Image Patches on Small Datasets. J. Inf. Commun. Converg. Eng. 2018, 16, 173–178, doi:10.6109/jicce.2018.16.3.173.
3) "Understanding of Convolutional Neural Network (CNN) — Deep Learning." Understanding of Convolutional Neural Network (CNN) — Deep Learning, medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148.
4) "Convolutional neural network." Convolutional neural network - Wikipedia, en.wikipedia.org/wiki/Convolutional_neural_network.
5) "Provincial and territorial resources for COVID-19." COVID-19: Non-medical masks and face coverings - Canada.ca, www.canada.ca/en/public-health/services/diseases/2019-novel-coronavirus-infection/symptoms/provincial-territorial-resources-covid-19.html.
6) Gurav, Omkar. "Face Mask Detection Dataset." Face Mask Detection Dataset | Kaggle, www.kaggle.com/omkargurav/face-mask-dataset.
7) How to interpret “loss” and “accuracy” for a machine learning, intellipaat.com/community/368/how-to-interpret-loss-and-accuracy-for-a-machine-learning-model.
8) MobileNetV2: Inverted Residuals and Linear Bottlenecks, towardsdatascience.com/mobilenetv2-inverted-residuals-and-linear-bottlenecks-8a4362f4ffd5.
9) Hollemans, Matthijs. https://machinethink.net/blog/mobilenet-v2/.
10) Vinitha and Velantina . "COVID-19 FACEMASK DETECTION WITH DEEP LEARNING AND COMPUTER VISION." International Research Journal of Engineering and Technology (IRJET).
11) Setunga, Supun. Stacking in Machine Learning, medium.com/@supun.setunga/stacking-in-machine-learning-357db1cfc3a