Face Mask Detection

With mobile net v2 deployed using streamlit. Link: https://praathapj-facemaskdetection-app-b5c2wm.streamlit.app/

Summary

- Used 1100 images for each class out of 3000 images due to computation constraints.
- standardize the data by dividing with 255.
- Use MobileNet since model faster that VGG Net and can be deployed with low computation requirement.
- Achived 99% accuracy with balanced class.
- Deployed using streamlit.

Business Problem

In recent trend in world wide Lockdowns due to COVID19 outbreak, as Face Mask is became mandatory for everyone while roaming outside. Masks play a crucial role in protecting the health of individuals against respiratory diseases, as is one of the few precautions available for COVID-19 in the absence of immunization.

Is it possible to create a model to detect people wearing masks, not wearing them.

Business objective and constraints

- Interpretablity is partially important.
- Low latency requirement(couple of seconds)
- Errors -> classification of both class is equally important.

Mapping to an ML problem

Data aquisation from kaggle

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
! pip install -q kaggle
```

```
! mkdir ~/.kaggle
```

```
! cp /content/drive/MyDrive/DS_DL_ML_AI_project/kaggle.json ~/.kaggle/
```

```
! chmod 600 ~/.kaggle/kaggle.json
```

```
! kaggle datasets download -d omkargurav/face-mask-dataset
```

```
Downloading face-mask-dataset.zip to /content
 98% 160M/163M [00:04<00:00, 35.4MB/s]
100% 163M/163M [00:04<00:00, 34.3MB/s]
```

```
# All the data in the ZIP will be unzipped in train folder.
! unzip /content/face-mask-dataset.zip
```

```
Streaming output truncated to the last 5000 lines.
  inflating: data/with_mask/with_mask_3297.jpg
  inflating: data/with_mask/with_mask_3298.jpg
  inflating: data/with_mask/with_mask_3299.jpg
  inflating: data/with_mask/with_mask_33.jpg
  inflating: data/with_mask/with_mask_330.jpg
  inflating: data/with_mask/with_mask_3300.jpg
  inflating: data/with_mask/with_mask_3301.jpg
  inflating: data/with_mask/with_mask_3302.jpg
  inflating: data/with_mask/with_mask_3303.jpg
  inflating: data/with_mask/with_mask_3304.jpg
  inflating: data/with_mask/with_mask_3305.jpg
  inflating: data/with_mask/with_mask_3306.jpg
  inflating: data/with_mask/with_mask_3307.jpg
  inflating: data/with_mask/with_mask_3308.jpg
  inflating: data/with_mask/with_mask_3309.jpg
  inflating: data/with_mask/with_mask_331.jpg
  inflating: data/with_mask/with_mask_3310.jpg
  inflating: data/with_mask/with_mask_3311.jpg
  inflating: data/with_mask/with_mask_3312.jpg
  inflating: data/with_mask/with_mask_3313.jpg
  inflating: data/with_mask/with_mask_3314.jpg
  inflating: data/with_mask/with_mask_3315.jpg
  inflating: data/with_mask/with_mask_3316.jpg
  inflating: data/with_mask/with_mask_3317.jpg
  inflating: data/with_mask/with_mask_3318.jpg
  inflating: data/with_mask/with_mask_3319.jpg
  inflating: data/with_mask/with_mask_332.jpg
  inflating: data/with_mask/with_mask_3320.jpg
  inflating: data/with_mask/with_mask_3321.jpg
  inflating: data/with_mask/with_mask_3322.jpg
  inflating: data/with_mask/with_mask_3323.jpg
  inflating: data/with_mask/with_mask_3324.jpg
  inflating: data/with_mask/with_mask_3325.jpg
  inflating: data/with_mask/with_mask_3326.jpg
  inflating: data/with_mask/with_mask_3327.jpg
  inflating: data/with_mask/with_mask_3328.jpg
  inflating: data/with_mask/with_mask_3329.jpg
  inflating: data/with_mask/with_mask_333.jpg
  inflating: data/with_mask/with_mask_3330.jpg
  inflating: data/with_mask/with_mask_3331.jpg
  inflating: data/with_mask/with_mask_3332.jpg
  inflating: data/with_mask/with_mask_3333.jpg
  inflating: data/with_mask/with_mask_3334.jpg
  inflating: data/with_mask/with_mask_3335.jpg
  inflating: data/with_mask/with_mask_3336.jpg
  inflating: data/with_mask/with_mask_3337.jpg
  inflating: data/with_mask/with_mask_3338.jpg
  inflating: data/with_mask/with_mask_3339.jpg
  inflating: data/with_mask/with_mask_334.jpg
  inflating: data/with_mask/with_mask_3340.jpg
  inflating: data/with_mask/with_mask_3341.jpg
  inflating: data/with_mask/with_mask_3342.jpg
  inflating: data/with_mask/with_mask_3343.jpg
  inflating: data/with_mask/with_mask_3344.jpg
  inflating: data/with_mask/with_mask_3345.jpg
  inflating: data/with_mask/with_mask_3346.jpg
  inflating: data/with_mask/with_mask_3347.jpg
```

Data Files Overview

Data set consists of 7553 RGB images in 2 folders as with_mask and without_mask. Images are named as label with_mask and without_mask. Images of faces with mask are 3725 and images of faces without mask are 3828.

But used 1100 images for each class due to computation constraints.

ML Problem

- Binary classification problem.

Performance Metric

- Binary Crossentropy.

Train Test Split

- Random split as there is no time stamps in the data.
- Split with stratify since classification problem.

Data Loading, Cleaning and Prepration

Import Libraries

```
import os
import cv2
import random
import numpy as np
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
from keras.applications import MobileNet
from keras import Sequential
from keras.layers import Dense


# Libraries for capturing web cam on google colab
# source: https://python.plainenglish.io/how-to-get-your-webcam-stream-in-colab-and-use-it-with-python-1
from IPython.display import display, Javascript, Image
from google.colab.output import eval_js
import html
from base64 import b64decode, b64encode
import cv2
import numpy as np
import PIL
import io
```

Read data

```
# Creating categories of data

img_cat = ["with_mask","without_mask"]

# Read all the images
data = []

for cat in img_cat:
  path = os.path.join("/content/data",cat)

  img_label = img_cat.index(cat) # Index of current category

  img_count = 0 #to count the number of images in each category

  for file in os.listdir(path):

    if img_count == 1100:
      break

    # Read file
    img_path = os.path.join(path,file)
    img  = cv2.imread(img_path)
    img = cv2.resize(img,(224,224)) ## VGG net accepts only image of this shape

    # crete 2 feature column 1 -> to store array and 2-> image class
    data.append([img,img_label])

    img_count += 1
```

```
# Since image_category is in order we need to shuffle for a generalized model
random.shuffle(data)
```

```
# Seperate dependent and independent variable

x = []
y = []

for feature,label in data:
  x.append(feature)
  y.append(label)
```

```
# Check the shape of input array

x = np.array(x)
print("Shape of x:",x.shape)
y = np.array(y)
print("Shape of y:",y.shape)
```

```
Shape of x: (2200, 224, 224, 3)
Shape of y: (2200,)
```

```
# standardize the data
x = x/255
```

```
# Train test split

x_train,x_test,y_train,y_test = train_test_split(x,y, test_size=0.2, random_state=35, stratify=y)

print("shape of train:",x_train.shape, y_train.shape)
print("shape of test:",x_test.shape, y_test.shape)
```

```
shape of train: (1760, 224, 224, 3) (1760,)
shape of test: (440, 224, 224, 3) (440,)
```

Predictive modelling

```
# Instantiates the MobileNet architecture.

mob_net = MobileNet()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet/mobil
17225924/17225924 [==============================] - 1s 0us/step
```

```
# Summary of existing VGG model
mob_net.summary()
```

```
Model: "mobilenet_1.00_224"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 input_1 (InputLayer)         [(None, 224, 224, 3)]     0

 conv1 (Conv2D)               (None, 112, 112, 32)      864

 conv1_bn (BatchNormalizatio  (None, 112, 112, 32)     128
 n)

 conv1_relu (ReLU)            (None, 112, 112, 32)      0

 conv_dw_1 (DepthwiseConv2D)  (None, 112, 112, 32)     288

 conv_dw_1_bn (BatchNormaliz  (None, 112, 112, 32)     128
 ation)

 conv_dw_1_relu (ReLU)        (None, 112, 112, 32)      0

 conv_pw_1 (Conv2D)           (None, 112, 112, 64)      2048

 conv_pw_1_bn (BatchNormaliz  (None, 112, 112, 64)     256
 ation)

 conv_pw_1_relu (ReLU)        (None, 112, 112, 64)      0

 conv_pad_2 (ZeroPadding2D)   (None, 113, 113, 64)      0

 conv_dw_2 (DepthwiseConv2D)  (None, 56, 56, 64)       576

 conv_dw_2_bn (BatchNormaliz  (None, 56, 56, 64)       256
 ation)

 conv_dw_2_relu (ReLU)        (None, 56, 56, 64)        0

 conv_pw_2 (Conv2D)           (None, 56, 56, 128)       8192

 conv_pw_2_bn (BatchNormaliz  (None, 56, 56, 128)      512
 ation)
```

```
conv_pw_2_relu (ReLU)       (None, 56, 56, 128)       0

conv_dw_3 (DepthwiseConv2D)  (None, 56, 56, 128)      1152

conv_dw_3_bn (BatchNormaliz  (None, 56, 56, 128)      512
ation)

conv_dw_3_relu (ReLU)       (None, 56, 56, 128)       0

conv_pw_3 (Conv2D)          (None, 56, 56, 128)       16384

conv_pw_3_bn (BatchNormaliz  (None, 56, 56, 128)      512
ation)

conv_pw_3_relu (ReLU)       (None, 56, 56, 128)       0
```

```
# We will perform transfer learning
# last layer will predict 2 classes
# wont train any parameters except last layer

model = Sequential()

for layer in mob_net.layers[:-1]: #all layer except last one.
    layer.trainable=False # Making all the layers non trainable except ast one.
    model.add(layer)

# Summary of new model
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1 (Conv2D)               (None, 112, 112, 32)      864

conv1_bn (BatchNormalizatio  (None, 112, 112, 32)     128
n)

conv1_relu (ReLU)            (None, 112, 112, 32)      0

conv_dw_1 (DepthwiseConv2D)  (None, 112, 112, 32)     288

conv_dw_1_bn (BatchNormaliz  (None, 112, 112, 32)     128
ation)

conv_dw_1_relu (ReLU)        (None, 112, 112, 32)      0

conv_pw_1 (Conv2D)           (None, 112, 112, 64)      2048

conv_pw_1_bn (BatchNormaliz  (None, 112, 112, 64)     256
ation)

conv_pw_1_relu (ReLU)        (None, 112, 112, 64)      0

conv_pad_2 (ZeroPadding2D)   (None, 113, 113, 64)      0

conv_dw_2 (DepthwiseConv2D)  (None, 56, 56, 64)       576

conv_dw_2_bn (BatchNormaliz  (None, 56, 56, 64)       256
ation)

conv_dw_2_relu (ReLU)        (None, 56, 56, 64)        0

conv_pw_2 (Conv2D)           (None, 56, 56, 128)       8192
```

```
conv_pw_2_bn (BatchNormaliz  (None, 56, 56, 128)      512
ation)

conv_pw_2_relu (ReLU)       (None, 56, 56, 128)       0

conv_dw_3 (DepthwiseConv2D)  (None, 56, 56, 128)      1152

conv_dw_3_bn (BatchNormaliz  (None, 56, 56, 128)      512
ation)

conv_dw_3_relu (ReLU)       (None, 56, 56, 128)       0

conv_pw_3 (Conv2D)          (None, 56, 56, 128)       16384

conv_pw_3_bn (BatchNormaliz  (None, 56, 56, 128)      512
ation)

conv_pw_3_relu (ReLU)       (None, 56, 56, 128)       0

conv_pad_4 (ZeroPadding2D)  (None, 57, 57, 128)       0

conv_dw_4 (DepthwiseConv2D)  (None, 28, 28, 128)      1152
```

```
# Add last classification layer with sigmoid activation

model.add(Dense(1,activation="sigmoid"))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1 (Conv2D)              (None, 112, 112, 32)      864

 conv1_bn (BatchNormalizatio  (None, 112, 112, 32)     128
 n)

 conv1_relu (ReLU)           (None, 112, 112, 32)      0

 conv_dw_1 (DepthwiseConv2D)  (None, 112, 112, 32)     288

 conv_dw_1_bn (BatchNormaliz  (None, 112, 112, 32)     128
 ation)

 conv_dw_1_relu (ReLU)       (None, 112, 112, 32)      0

 conv_pw_1 (Conv2D)          (None, 112, 112, 64)      2048

 conv_pw_1_bn (BatchNormaliz  (None, 112, 112, 64)     256
 ation)

 conv_pw_1_relu (ReLU)       (None, 112, 112, 64)      0

 conv_pad_2 (ZeroPadding2D)  (None, 113, 113, 64)      0

 conv_dw_2 (DepthwiseConv2D)  (None, 56, 56, 64)       576

 conv_dw_2_bn (BatchNormaliz  (None, 56, 56, 64)       256
 ation)

 conv_dw_2_relu (ReLU)       (None, 56, 56, 64)        0

 conv_pw_2 (Conv2D)          (None, 56, 56, 128)       8192

 conv_pw_2_bn (BatchNormaliz  (None, 56, 56, 128)      512
 ation)

 conv_pw_2_relu (ReLU)       (None, 56, 56, 128)       0
```

```
conv_dw_3 (DepthwiseConv2D)  (None, 56, 56, 128)      1152

conv_dw_3_bn (BatchNormaliz  (None, 56, 56, 128)      512
ation)

conv_dw_3_relu (ReLU)        (None, 56, 56, 128)      0

conv_pw_3 (Conv2D)           (None, 56, 56, 128)      16384

conv_pw_3_bn (BatchNormaliz  (None, 56, 56, 128)      512
ation)

conv_pw_3_relu (ReLU)        (None, 56, 56, 128)      0

conv_pad_4 (ZeroPadding2D)   (None, 57, 57, 128)      0

conv_dw_4 (DepthwiseConv2D)  (None, 28, 28, 128)      1152
```

```
model.compile(optimizer='Adam',loss='binary_crossentropy', metrics=['accuracy'])
model.fit(x_train,y_train, epochs=7, validation_data = (x_test,y_test))
```

```
============] - 16s 87ms/step - loss: 0.2598 - accuracy: 0.8983 - val_loss: 0.1088 - val_accuracy:

============] - 3s 50ms/step - loss: 0.0685 - accuracy: 0.9739 - val_loss: 0.0474 - val_accuracy: 0

============] - 3s 51ms/step - loss: 0.0412 - accuracy: 0.9835 - val_loss: 0.0407 - val_accuracy: 0

============] - 3s 50ms/step - loss: 0.0271 - accuracy: 0.9920 - val_loss: 0.0368 - val_accuracy: 0

============] - 3s 48ms/step - loss: 0.0197 - accuracy: 0.9949 - val_loss: 0.0363 - val_accuracy: 0

============] - 3s 50ms/step - loss: 0.0152 - accuracy: 0.9972 - val_loss: 0.0265 - val_accuracy: 0

============] - 3s 50ms/step - loss: 0.0121 - accuracy: 0.9989 - val_loss: 0.0257 - val_accuracy: 0
at 0x7fef8019ed90>
```

```
# Stream local webcam to google colab
# source: https://python.plainenglish.io/how-to-get-your-webcam-stream-in-colab-and-use-it-with-python-1

# create a function to convert this image object from JS to an OpenCV image to be able to use it in your

def jsob_to_image(js_object):
  # decode base64 image
  image_bytes = b64decode(js_object.split(',')[1])
  # convert bytes to numpy array
  img_array = np.frombuffer(image_bytes, dtype=np.uint8)
  # convert numpy array into OpenCV BGR
  frame = cv2.imdecode(img_array, flags=1)

  return frame
```

```
# to stream the image

def video_stream():
  js = Javascript('''
    var video;
    var div = null;
    var stream;
    var captureCanvas;
    var imgElement;
    var labelElement;
```

```
var pendingResolve = null;
var shutdown = false;

function removeDom() {
   stream.getVideoTracks()[0].stop();
   video.remove();
   div.remove();
   video = null;
   div = null;
   stream = null;
   imgElement = null;
   captureCanvas = null;
   labelElement = null;
}

function onAnimationFrame() {
  if (!shutdown) {
    window.requestAnimationFrame(onAnimationFrame);
  }
  if (pendingResolve) {
    var result = "";
    if (!shutdown) {
      captureCanvas.getContext('2d').drawImage(video, 0, 0, 640, 480);
      result = captureCanvas.toDataURL('image/jpeg', 0.8)
    }
    var lp = pendingResolve;
    pendingResolve = null;
    lp(result);
  }
}

async function createDom() {
  if (div !== null) {
    return stream;
  }
  div = document.createElement('div');
  div.style.border = '2px solid black';
  div.style.padding = '3px';
  div.style.width = '100%';
  div.style.maxWidth = '600px';
  document.body.appendChild(div);


  video = document.createElement('video');
  video.style.display = 'block';
  video.width = div.clientWidth - 6;
  video.setAttribute('playsinline', '');
  video.onclick = () => { shutdown = true; };
  stream = await navigator.mediaDevices.getUserMedia(
      {video: { facingMode: "environment"}});
  div.appendChild(video);
  imgElement = document.createElement('img');
  imgElement.style.position = 'absolute';
  imgElement.style.zIndex = 1;
  imgElement.onclick = () => { shutdown = true; };
  div.appendChild(imgElement);

  const instruction = document.createElement('div');
  instruction.innerHTML =
      '<span style="blue: red; font-weight: bold;">' +
      'click here to stop the video</span>';
  div.appendChild(instruction);
```

```
      instruction.onclick = () => { shutdown = true; };

      video.srcObject = stream;
      await video.play();
      captureCanvas = document.createElement('canvas');
      captureCanvas.width = 640;
      captureCanvas.height = 480;
      window.requestAnimationFrame(onAnimationFrame);

      return stream;
    }
    async function stream_frame() {
      if (shutdown) {
        removeDom();
        shutdown = false;
        return '';
      }
      var preCreate = Date.now();
      stream = await createDom();

      var preShow = Date.now();



      var preCapture = Date.now();
      var result = await new Promise(function(resolve, reject) {
        pendingResolve = resolve;
      });
      shutdown = false;

      return {'create': preShow - preCreate,
              'show': preCapture - preShow,
              'capture': Date.now() - preCapture,
              'img': result};
    }
    ''')

  display(js)

def video_frame():
  data = eval_js('stream_frame()')
  return data
```

```
# detection functin
def detect_face_mask(img):

  img = cv2.resize(img,(224,224))
  img = img/255 # standardize img
  pred = (model.predict(img.reshape(1,224,224,3)) > 0.5).astype("int32")

  return pred[0][0]
```

```
# Get the webcam stream and forward it to python
video_stream()

while True:
    frame_js = video_frame()
    if not frame_js:
        break
    img = jsob_to_image(frame_js["img"])
```

```
    y_pred = detect_face_mask(img)

    if y_pred == 0:
      print("Mask")
    if y_pred == 1:
      print("No Mask")
```

```
 1/1 [==============================] - 1s 1s/step
 No Mask
 1/1 [==============================] - 0s 22ms/step
 No Mask
 1/1 [==============================] - 0s 25ms/step
 No Mask
 1/1 [==============================] - 0s 24ms/step
 No Mask
 1/1 [==============================] - 0s 32ms/step
 No Mask
 1/1 [==============================] - 0s 23ms/step
 No Mask
 1/1 [==============================] - 0s 23ms/step
 No Mask
 1/1 [==============================] - 0s 26ms/step
 No Mask
 1/1 [==============================] - 0s 22ms/step
 Mask
 1/1 [==============================] - 0s 30ms/step
 No Mask
 1/1 [==============================] - 0s 35ms/step
 No Mask
 1/1 [==============================] - 0s 21ms/step
 No Mask
 1/1 [==============================] - 0s 24ms/step
 No Mask
 1/1 [==============================] - 0s 23ms/step
 No Mask
```

Modelling Observation

- Achieved 99% accuracy in both train and test data using MobileNet.

Deployment

- Save required models,variables for deployment.

Save model

```
# To save in regular tensor flow format
#model.save('/content/drive/MyDrive/DS_DL_ML_AI_project/FaceMaskDetection/lite_model')

# To save in old keras format
model.save("/content/drive/MyDrive/DS_DL_ML_AI_project/FaceMaskDetection/faceMaskDetect.hdf5")
```

Conclusion

- The above model has 99% accuracy.
- To continue the project:
- Obtain more data set with has images of 'mask not worn properly'.
- Also include images which has face covered without mask as 'No Mask'