Importing all the required libraries

```
import csv
import os
from random import seed
from random import random
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot
from PIL import Image
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, MaxPooling2D, Flatten, Conv2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.activations import softmax, sigmoid
from tensorflow.python.keras.optimizers import SGD
import tensorflow as tf
```

Rotating the images to increase the size of the dataset

```
# import PIL
# from PIL import Image
# import cv2
# image_paths = ["/content/drive/My Drive/data/with_mask", "/content/drive/My Drive/data/w
# for path in image_paths:
#   i = 0
#   for image_path in os.listdir(path):
#         input_path = os.path.join(path, image_path)
#         image_to_rotate = Image.open(input_path)
#         fullpath = os.path.join(path, 'rotated_'+ str(i) + '_' +image_path)
#         rotated = image_to_rotate.rotate(90).save(fullpath)
#         i = i + 1
```

Flipping the images to increase the size of the dataset

```
# image_paths = ["/content/drive/My Drive/data/with_mask", "/content/drive/My Drive/data/w
# for path in image_paths:
#   i = 0
#   for image_path in os.listdir(path):
#         input_path = os.path.join(path, image_path)
#         image_to_flip = Image.open(input_path)
#         fullpath = os.path.join(path, 'flipped_'+ str(i) + '_' +image_path)
#         out = image_to_flip.transpose(PIL.Image.FLIP_LEFT_RIGHT).save(fullpath)
#         i = i + 1
```

Making thee directory structure required by the image data generator keras

```
# subdirs = ['train/', 'test/']
# for dir in subdirs:
#     labeldirs = ['with_mask/', 'without_mask/']
```

```
#     for labeldir in labeldirs:
#         newdir = '/content/drive/My Drive/train_data/' + dir + labeldir
#         os.makedirs(newdir, exist_ok=True)
```

Splitting the data (80-20 split) and putting them in the required directories under their class labels

```
# seed(1)
# test_ratio = 0.20
# def category_split():
#   image_paths = ["with_mask", "without_mask"]
#   for directory in image_paths:
#     path_dir = '/content/drive/My Drive/data/' + directory
#     for file in os.listdir(path_dir):
#         type_dir = "train/"
#         main_dir = ''
#         if (random() < 0.20):
#             type_dir = "test/"
#         if (directory == "with_mask"):
#             main_dir = '/content/drive/My Drive/train_data/' + type_dir + 'with_mask/'
#         if (directory == "without_mask"):
#             main_dir = '/content/drive/My Drive/train_data/' + type_dir + 'without_mask/
#         os.replace(path_dir + '/' + file, main_dir + file)

# category_split()
```

Creating a simple convolutional network using 2 convolution layers with 3x3 filters

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(100, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(100, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(50, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
```

Passing the data from the directory using the generator object to reduce RAM usage

```
datagen = ImageDataGenerator(rescale=1.0/255.0)
train_it = datagen.flow_from_directory('/content/drive/My Drive/train_data/train', target_
test_it = datagen.flow_from_directory('/content/drive/My Drive/train_data/test', target_si
```

```
Found 4408 images belonging to 2 classes.
Found 1096 images belonging to 2 classes.
```

Fitting the model

```
# Fitting the model
print("Fitting the model....")
history = model.fit_generator(train_it, steps_per_epoch = len(train_it), validation_data=
```

```
Fitting the model....
WARNING:tensorflow:From <ipython-input-8-95539f1f76b3>:3: Model.fit_generator (from t
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/30
69/69 - 27s - loss: 0.4259 - acc: 0.8101 - val_loss: 0.1320 - val_acc: 0.9553
Epoch 2/30
69/69 - 27s - loss: 0.1182 - acc: 0.9571 - val_loss: 0.1024 - val_acc: 0.9662
Epoch 3/30
69/69 - 26s - loss: 0.0685 - acc: 0.9766 - val_loss: 0.0739 - val_acc: 0.9781
Epoch 4/30
69/69 - 27s - loss: 0.0474 - acc: 0.9848 - val_loss: 0.0613 - val_acc: 0.9827
Epoch 5/30
```

Saving the weights

```
print("Saving the model")
model.save_weights('/content/drive/My Drive/saved_model/model_wieghts_150.h5')
model.save('/content/drive/My Drive/saved_model/model_keras_150.h5')
```

```
Saving the model
69/69 - 26s - loss: 0.0338 - acc: 0.9905 - val loss: 0.6940 - val acc: 0.8467
```

TO load the local webcam in google colab

```
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode

def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
```

```
        canvas.getContext('2d').drawImage(video, 0, 0);
        stream.getVideoTracks()[0].stop();
        div.remove();
        return canvas.toDataURL('image/jpeg', quality);
      }
      ''')
    display(js)
    data = eval_js('takePhoto({})'.format(quality))
    binary = b64decode(data.split(',')[1])
    with open(filename, 'wb') as f:
      f.write(binary)
    return filename


from IPython.display import Image
# try:
#   filename = take_photo()
#   print('Saved to {}'.format(filename))
#   # Show the image which was just taken.
#   display(Image(filename))
# except Exception as err:
#   # Errors will be thrown if the user does not have a webcam or if they do not
#   # grant the page permission to access it.
#   print(str(err))
```

Predicting the images in the test set using the openCV library

```
import cv2
from google.colab.patches import cv2_imshow
import numpy as np
labels_dict={0:'without_mask',1:'with_mask'}
color_dict={0:(0,0,255),1:(0,255,0)}
predict_path = "/content/drive/My Drive/prediction_images/all_classes/"
model = tf.keras.models.load_model("/content/drive/My Drive/saved_model/model_keras_150.h5
size = 4
classifier = cv2.CascadeClassifier('/content/drive/My Drive/haarcascade_frontalface_defaul

for file in os.listdir(predict_path):
    im = cv2.imread(predict_path + file)
    # im=cv2.flip(im,1,1) #Flip to act as a mirror

    mini = cv2.resize(im, (im.shape[1] // size, im.shape[0] // size))

    faces = classifier.detectMultiScale(mini)

    for f in faces:
        (x, y, w, h) = [v * size for v in f]
        face_img = im[y:y+h, x:x+w]
        resized=cv2.resize(face_img,(150,150))
        normalized=resized/255.0
        reshaped=np.reshape(normalized,(1,150,150,3))
        reshaped = np.vstack([reshaped])
        result=model.predict(reshaped)
        #print(result)
```

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

            cv2.rectangle(im,(x,y),(x+w,y+h),color_dict[label],2)
            cv2.rectangle(im,(x,y-40),(x+w,y),color_dict[label],-1)
            cv2.putText(im, labels_dict[label], (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,25

        # cv2_imshow(im)
        key = cv2.waitKey(10)
        if key == 27:
            break

    cv2.destroyAllWindows()
```