Real-Time Face Mask Detection

In this task, I bill build a real-time face mask detection to detect whether the person on the webcam is wearing a mask or not.

The dataset use in this task consists of 1376 images with 690 images containing images of people wearing masks and 686 images with people without masks.

```
!unzip /content/drive/MyDrive/Dataset/test.zip
```

```
Archive:  /content/drive/MyDrive/Dataset/test.zip
replace test/with_mask/1-with-mask.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename:
```

```
!unzip /content/drive/MyDrive/Dataset/train.zip
```

```
  inflating: train/without_mask/gettyimages-915227284-1024x1024.jpg
  inflating: train/without_mask/image-asset.jpeg
  inflating: train/without_mask/image.jpg
 extracting: train/without_mask/indian-ethnicity-cheerful-confident-studio-260nw-552
  inflating: train/without_mask/indian-face-series-1536016.jpg
  inflating: train/without_mask/indian-faces-3548.JPG
  inflating: train/without_mask/maxresdefault.jpg
  inflating: train/without_mask/offset-322695.jpg
  inflating: train/without_mask/offset-870274.jpg
  inflating: train/without_mask/photo-1493106819501-66d381c466f1.jpeg
  inflating: train/without_mask/pm.jpg
  inflating: train/without_mask/pm1.jpg
  inflating: train/without_mask/pm2.jpg
  inflating: train/without_mask/pm3.jpg
  inflating: train/without_mask/short-hairstyles-for-indian-faces-406016-50-indian-h
  inflating: train/without_mask/short-hairstyles-for-indian-faces-406016-indian-hair
  inflating: train/without_mask/top-50-best-faces.jpg
  inflating: train/without_mask/want-to-see-more-indian-faces-in-us-government-raj-m
```

1. **Import libraries:**

1. **Keras :**used for distributed training of deep learning models
2. **Sklearn :**provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python
3. **Imutils :** A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges
4. **Numpy :** offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more

```
from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
import cv2
from keras.models import Sequential
from keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, Activation, MaxPo
from keras.models import Model, load_model
from keras.callbacks import TensorBoard, ModelCheckpoint
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.utils import shuffle
import imutils
import numpy as np
```

2.**Build the neural network**

This convolution network consists of two pairs of Conv and MaxPool layers to extract features from the dataset. Which is then followed by a Flatten and Dropout layer to convert the data in 1D and ensure overfitting.

```
model = Sequential([
    Conv2D(100, (3,3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D(2,2),

    Conv2D(100, (3,3), activation='relu'),
    MaxPooling2D(2,2),

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

3. **Image data Generation**

```
TRAINING_DIR = "/content/train"
train_datagen = ImageDataGenerator(rescale=1.0/255,
                                   rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   fill_mode='nearest')

train_generator = train_datagen.flow_from_directory(TRAINING_DIR,
                                                    batch_size=10,
                                                    target_size=(150, 150))
VALIDATION_DIR = "/content/test"
validation_datagen = ImageDataGenerator(rescale=1.0/255)

validation_generator = validation_datagen.flow_from_directory(VALIDATION_DIR,
                                                              batch_size=10,
                                                              target_size=(150, 150))
```

```
Found 1315 images belonging to 2 classes.
Found 194 images belonging to 2 classes.
```

4. Initialize a callback checkpoint to keep saving best model after each epoch while training

```
checkpoint = ModelCheckpoint('model2-{epoch:03d}.model',monitor='val_loss',verbose=0,save_be
```

5.Train the model

this process take 32 minutes to complete

```
history = model.fit_generator(train_generator,
                              epochs=10,
                              validation_data=validation_generator,
                              callbacks=[checkpoint])
```

```
<ipython-input-10-6272b23e5a0b>:1: UserWarning: `Model.fit_generator` is deprecated and
  history = model.fit_generator(train_generator,
Epoch 1/10
132/132 [==============================] - ETA: 0s - loss: 0.7012 - acc: 0.5270WARNING:
132/132 [==============================] - 195s 1s/step - loss: 0.7012 - acc: 0.5270 -
Epoch 2/10
132/132 [==============================] - ETA: 0s - loss: 0.4516 - acc: 0.8091WARNING:
132/132 [==============================] - 186s 1s/step - loss: 0.4516 - acc: 0.8091 -
Epoch 3/10
132/132 [==============================] - ETA: 0s - loss: 0.3281 - acc: 0.8806WARNING:
132/132 [==============================] - 179s 1s/step - loss: 0.3281 - acc: 0.8806 -
Epoch 4/10
132/132 [==============================] - 182s 1s/step - loss: 0.2946 - acc: 0.8837 -
Epoch 5/10
132/132 [==============================] - ETA: 0s - loss: 0.2568 - acc: 0.9027WARNING:
132/132 [==============================] - 177s 1s/step - loss: 0.2568 - acc: 0.9027 -
Epoch 6/10
132/132 [==============================] - ETA: 0s - loss: 0.2003 - acc: 0.9270WARNING:
132/132 [==============================] - 179s 1s/step - loss: 0.2003 - acc: 0.9270 -
Epoch 7/10
132/132 [==============================] - 194s 1s/step - loss: 0.2111 - acc: 0.9240 -
Epoch 8/10
132/132 [==============================] - 176s 1s/step - loss: 0.2037 - acc: 0.9262 -
Epoch 9/10
132/132 [==============================] - 180s 1s/step - loss: 0.1925 - acc: 0.9300 -
Epoch 10/10
132/132 [==============================] - ETA: 0s - loss: 0.1805 - acc: 0.9308WARNING:
132/132 [==============================] - 187s 1s/step - loss: 0.1805 - acc: 0.9308 -
```

Now, we save model

```
model.save('model.h5',history)
```

✓ 0s completed at 11:08 AM