Machine Learning with Python
Major Project

BHUPENDRA TIWARI

BT21ECE027

ABSTRACT :- Face Mask Detection Using Deep Learning with Python

THEORY :- Face mask detection is a classification problem where an image is analyzed to determine whether a person is wearing a mask or not. It has gained significant importance in ensuring compliance with public safety measures. This task leverages machine learning techniques, particularly convolutional neural networks (CNNs), to extract features from images and classify them into two categories where a person is wearing mask or not.

MobileNetV2 is an efficient deep learning architecture designed for mobile and embedded applications. It uses **Depthwise separable convolutions** to reduce computational cost and parameters, making it lightweight and suitable for resource-constrained devices.

Key features include:

- **Inverted Residuals**: Expands features to higher dimensions, applies depthwise convolutions, and projects back to low dimensions.
- **Linear Bottlenecks**: Avoids non-linearities in low-dimensional feature spaces to retain information.
- **Scalability**: Allows adjusting the model size with width and resolution multipliers for performance-efficiency trade-offs.

.

MODEL ARCHITECTURE :-

```
Input (128x128x3)
↓
Pretrained MobileNetV2 (Feature Extractor, Frozen)
↓
GlobalAveragePooling2D
↓
Dense (128, ReLU)
↓
Dropout (50%)
↓
Dense (2, Softmax)
```

RESULT :-

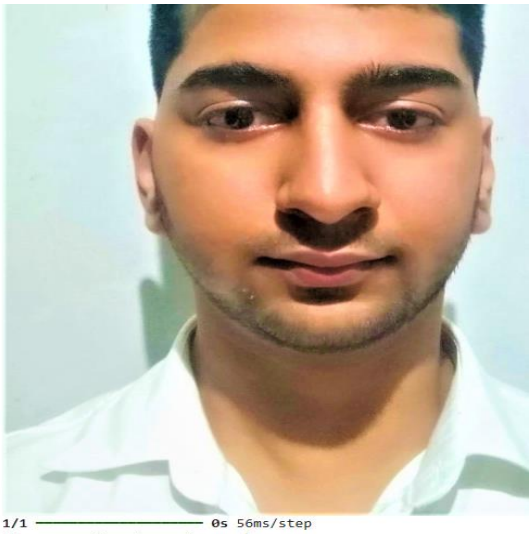
Test Accuracy is coming out to be 98.477%

```
path of image to be predicted: /content/image12.jpeg
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step
The person is wearing mask
```

path of image to be predicted: /content/imag

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
The person is wearing mask



1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step
The person is not wearing mask

path of image to be predicted: /conter

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
The person is not wearing mask

APPENDIX :-

```
[4]  !mkdir -p ~/.kaggle
     !cp kaggle.json ~/.kaggle/
     !chmod 600 ~/.kaggle/kaggle.json
```

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

```
Dataset URL: https://www.kaggle.com/datasets/omkargurav/face-mask-da
License(s): unknown
Downloading face-mask-dataset.zip to /content
 98% 160M/163M [00:01<00:00, 127MB/s]
100% 163M/163M [00:01<00:00, 119MB/s]
```

```
[6]  from zipfile import ZipFile
     dataset ='/content/face-mask-dataset.zip'

     with ZipFile(dataset,'r') as zip:
       zip.extractall()
       print('The dataset is extracted')
```

```
The dataset is extracted
```

```
     !ls
```

```
data  face-mask-dataset.zip  kaggle.json  sample_data
```

```
[8]  import os
```

```
[9]  filenames_with_mask=os.listdir('/content/data/with_mask')
     print(filenames_with_mask)
```

```
['with_mask_2212.jpg', 'with_mask_3343.jpg', 'with_mask_1834.jpg', '
```

```
filenames_without_mask=os.listdir('/content/data/without_mask')
print(filenames_with_mask)
```

```
['with_mask_2212.jpg', 'with_mask_3343.jpg', 'with_mask_1834.jpg'
```

```
num_of_with_mask=len(filenames_with_mask)
print("Number of images with mask: ",num_of_with_mask)
num_of_without_mask=len(filenames_without_mask)
print("Number of images without mask: ",num_of_without_mask)
```

```
Number of images with mask:  3725
Number of images without mask:  3828
```

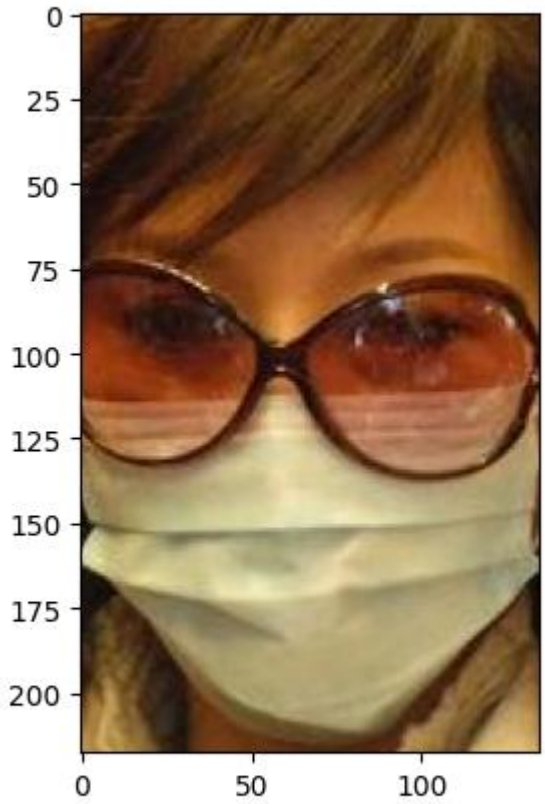
```
#importing the depency
import numpy as np
from PIL import Image
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

ting lables for two class ofd imagess

```
with_mask_labels=[1]*num_of_with_mask
print(with_mask_labels)
without_mask_labels=[0]*num_of_without_mask
print(without_mask_labels)
```

```
#displaying with_mask image
img=mpimg.imread('/content/data/with_mask/with_mask_10.jpg')
plt.imshow(img)
```

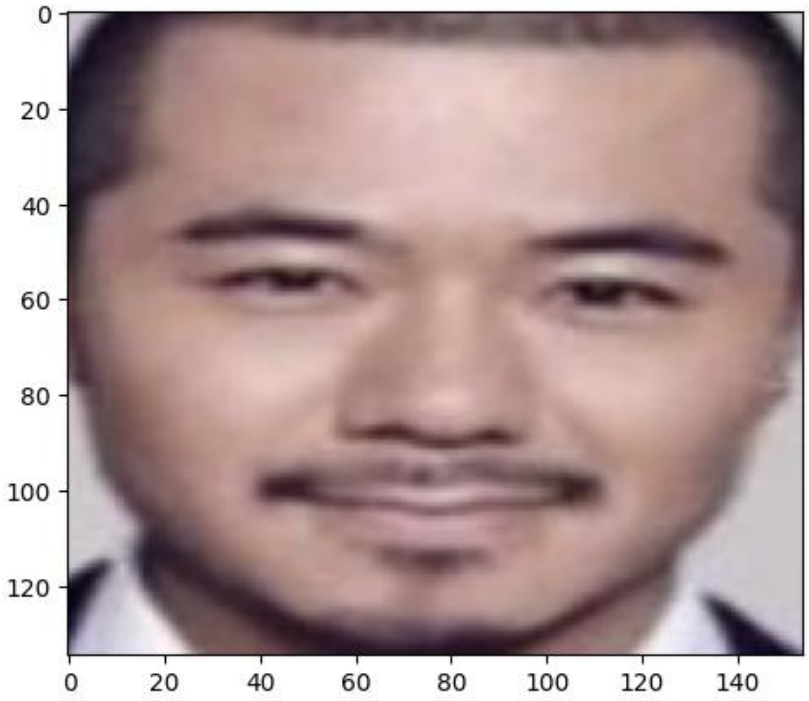
```
<matplotlib.image.AxesImage at 0x78e149dc17b0>
```




```
img=mpimg.imread('/content/data/without_mask/without_mask_10.jpg')
plt.imshow(img)
```

```
[18] img=mpimg.imread('/content/data/without_mask/without_mask_10.jpg')
     plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x78e149e59ab0>
```




```
#Resizing All images and saving image in different folders
#creating directory for resized image
os.mkdir('with_mask_resized')
os.mkdir('without_mask_resized')
```

[20]
```
#resize with mask image
with_mask_folder='/content/data/with_mask/'
with_mask_resized_folder='/content/with_mask_resized/'

for filename in os.listdir(with_mask_folder):
  img_path=with_mask_folder+filename
  img = Image.open(img_path)
  img = img.resize((128,128))
  img = img.convert('RGB')
  img.save(with_mask_resized_folder+filename)
```

```
/usr/local/lib/python3.10/dist-packages/PIL/Image.py:1054: UserWarning: Pale
  warnings.warn(
```

[21]
```
#resize with mask image
without_mask_folder='/content/data/without_mask/'
without_mask_resized_folder='/content/without_mask_resized/'

for filename in os.listdir(without_mask_folder):
  img_path=without_mask_folder+filename
  img = Image.open(img_path)
  img = img.resize((128,128))
  img = img.convert('RGB')
  img.save(without_mask_resized_folder+filename)
```

```
img=mpimg.imread('/content/without_mask_resized/without_mask_10.jpg')
plt.imshow(img)
```

```
[23] #converting image to numpy array
     import cv2
     import glob
```

```
[24] imdir = '/content/with_mask_resized/'
     ext = ['png','jpg']

     files = []
     [files.extend(glob.glob(imdir + '*.' + e)) for e in ext]

     with_mask_images = np.asarray([cv2.imread(file) for file in files])
```

```
[25] imdir = '/content/without_mask_resized/'
     ext = ['png','jpg']

     files = []
     [files.extend(glob.glob(imdir + '*.' + e)) for e in ext]

     without_mask_images = np.asarray([cv2.imread(file) for file in files])
```

```
     print(with_mask_images[0])
     print(without_mask_images[0])
```

```
combined_images=np.concatenate((with_mask_images,without_mask_images))
```

```
print(combined_images.shape)
```

```
(7553, 128, 128, 3)
```

```
X=combined_images
Y=np.asarray(labels)
```

```
#Train & Test Split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state
```

```
print(X.shape,X_train.shape,Y_train.shape,X_test.shape,Y_test.shape)
```

```
(7553, 128, 128, 3) (6042, 128, 128, 3) (6042,) (1511, 128, 128, 3) (1511,)
```

```
#standardinzing the data
X_train_std=X_train/255
X_test_std=X_test/255
```

```
import tensorflow as tf
import keras
```

```
from tensorflow.keras import layers, models
from tensorflow.keras.applications import MobileNetV2

# Load a pretrained MobileNetV2 model
base_model = MobileNetV2(input_shape=(128, 128, 3), include_top=False, weights='imagenet')
base_model.trainable = False  # Freeze the base model layers

# Build the model
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),  # Reduce feature maps to a single vector
    layers.Dense(128, activation='relu'),  # Dense layer for classification
    layers.Dropout(0.5),  # Prevent overfitting
    layers.Dense(2, activation='softmax')  # Two output classes (mask, no mask)
])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

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

```
model.fit(X_train_std,Y_train,epochs=10)
```

```
38] 189/189 ━━━━━━━━━━━━━━━━━━━━ 86s 431ms/step - accuracy: 0.9844 - loss: 0.0414
Epoch 4/10
189/189 ━━━━━━━━━━━━━━━━━━━━ 77s 410ms/step - accuracy: 0.9885 - loss: 0.0352
Epoch 5/10
189/189 ━━━━━━━━━━━━━━━━━━━━ 86s 429ms/step - accuracy: 0.9909 - loss: 0.0279
Epoch 6/10
189/189 ━━━━━━━━━━━━━━━━━━━━ 80s 418ms/step - accuracy: 0.9884 - loss: 0.0330
Epoch 7/10
189/189 ━━━━━━━━━━━━━━━━━━━━ 86s 437ms/step - accuracy: 0.9916 - loss: 0.0257
Epoch 8/10
189/189 ━━━━━━━━━━━━━━━━━━━━ 79s 418ms/step - accuracy: 0.9942 - loss: 0.0151
Epoch 9/10
189/189 ━━━━━━━━━━━━━━━━━━━━ 82s 419ms/step - accuracy: 0.9916 - loss: 0.0201
Epoch 10/10
189/189 ━━━━━━━━━━━━━━━━━━━━ 81s 417ms/step - accuracy: 0.9969 - loss: 0.0106
<keras.src.callbacks.history.History at 0x78dff998b5e0>
```

```
score,acc=model.evaluate(X_test_std,Y_test)
print('Test data loss:',score)
print('Test data accuracy:',acc)
```

```
48/48 ━━━━━━━━━━━━━━━━━━━━ 19s 388ms/step - accuracy: 0.9813 - loss: 0.0798
Test data loss: 0.054390110075473785
Test data accuracy: 0.9847782850265503
```

BUILDING A PREDICTIVE SYSTEM

```
41] from google.colab.patches import cv2_imshow
```

```python
input_image_path=input('path of image to be predicted: '
input_image=cv2.imread(input_image_path)
cv2_imshow(input_image)
input_image=cv2.resize(input_image,(128,128))
input_image=input_image/255
input_image=np.reshape(input_image,[1,128,128,3])
prediction=model.predict(input_image)
input_pred_label=np.argmax(prediction)
if input_pred_label==1:
  print('The person is wearing mask')
else:
  print('The person is not wearing mask')
```

```
path of image to be predicted: /content/image7.jpeg
```


```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
The person is not wearing mask
```