MINI PROJECT

Face Mask Detection using Convolutional Neural Network (CNN)

PROBLEM STATEMENT:

The COVID-19 pandemic has emphasized the importance of wearing face masks to reduce the spread of the virus. Mask compliance has become a critical aspect of public health measures, but manual monitoring in crowded places is not only labor-intensive but also prone to errors. As a solution, deep learning techniques provide a robust and automated way to detect mask usage in real-time, addressing the challenges of manual enforcement. This project focuses on developing a face mask detection system using a custom Convolutional Neural Network (CNN) to automate the process and ensure efficient compliance monitoring.

The proposed system will analyze facial images to classify them into two categories: with mask and without mask. By training the model on a labeled dataset of images, it aims to achieve high accuracy in identifying masked and unmasked faces. Such a system can be integrated with surveillance cameras or other real-time monitoring systems to ensure safety measures are followed. The project's outcomes are expected to enhance public safety in high-traffic areas such as airports, malls, and public transportation while reducing the burden on manual monitoring processes. This technology also has potential applications in healthcare, workplaces, and educational institutions to maintain adherence to safety protocols.

DATASET USED:

In recent trend in world wide Lockdowns due to COVID19 outbreak, as Face Mask is became mandatory for everyone while roaming outside, approach of Deep Learning for Detecting Faces With and Without mask were a good trendy practice. Here I have created a model that detects face mask trained on 7553 images with 3 color channels (RGB). 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.

IMPLEMENTATION AND CODE:

1. Importing the necessary libraries

```
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
from google.colab.patches import cv2_imshow
from PIL import Image
from sklearn.model_selection import train_test_split
with_mask_files = os.listdir('/content/data/with_mask')
print(with_mask_files[0:5])
print(with_mask_files[-5:])
without_mask_files = os.listdir('/content/data/without_mask')
print(without_mask_files[0:5])
print(without_mask_files[-5:])
```

2. Creating Labels for the two class of Images

```
with_mask_labels = [1]*3725
without_mask_labels = [0]*3828
print(with_mask_labels[0:5])
print(without_mask_labels[0:5])
img = mpimg.imread('/content/data/with_mask/with_mask_1545.jpg')
imgplot = plt.imshow(img)
plt.show()
```

3. Image Preprocessing

```
with_mask_path = '/content/data/with_mask/'

data = []

for img_file in with_mask_files:

  image = Image.open(with_mask_path + img_file)
  image = image.resize((128,128))
  image = image.convert('RGB')
  image = np.array(image)
  data.append(image)
```

```
without_mask_path = '/content/data/without_mask/'


for img_file in without_mask_files:

  image = Image.open(without_mask_path + img_file)
  image = image.resize((128,128))
  image = image.convert('RGB')
  image = np.array(image)
  data.append(image)
```

4. Splitting of Training and Testing dataset

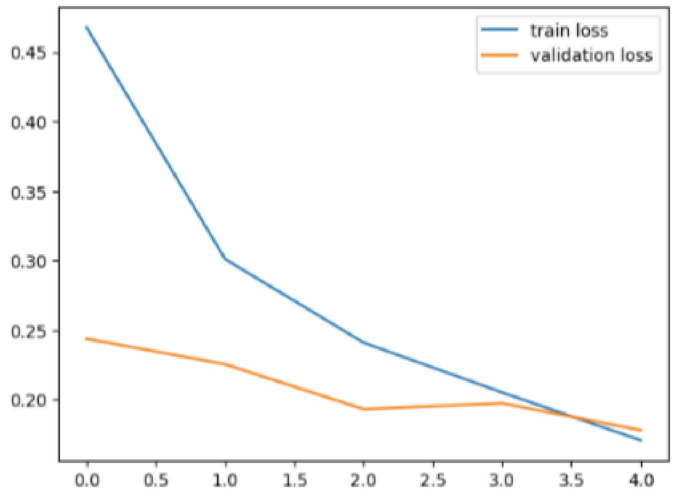
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=2)
# scaling the data
X_train_scaled = X_train/255
X_test_scaled = X_test/255
X_train[0]
X_train_scaled[0]
```

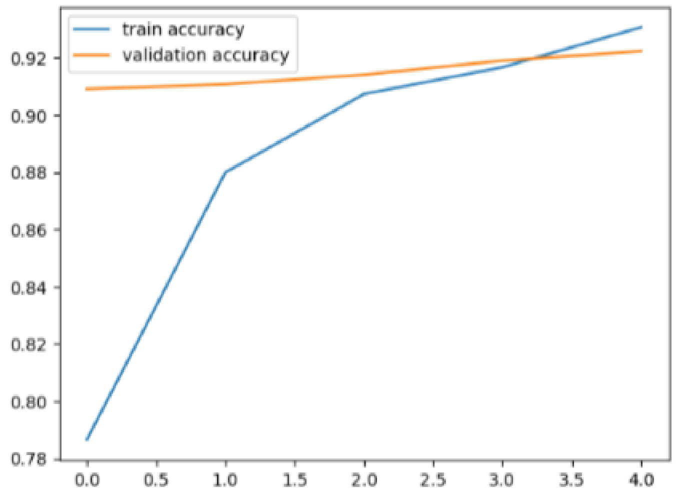
5. Building CNN model

```
# Build CNN model
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(128,
128, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(train_data.num_classes, activation='softmax')])
# Compile the model
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
import tensorflow as tf
from tensorflow import keras
num_of_classes = 2

model = keras.Sequential()

model.add(keras.layers.Conv2D(32, kernel_size=(3,3),
activation='relu', input_shape=(128,128,3)))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))


model.add(keras.layers.Conv2D(64, kernel_size=(3,3),
activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))

model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(128, activation='relu'))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dropout(0.5))


model.add(keras.layers.Dense(num_of_classes, activation='sigmoid'))
# compile the neural network
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['acc'])
# training the neural network
history = model.fit(X_train_scaled, Y_train, validation_split=0.1,
epochs=5)
```

6. Model evaluation

```
loss, accuracy = model.evaluate(X_test_scaled, Y_test)
print('Test Accuracy =', accuracy)
h = history
# plot the loss value
plt.plot(h.history['loss'], label='train loss')
plt.plot(h.history['val_loss'], label='validation loss')
plt.legend()
plt.show()

# plot the accuracy value
plt.plot(h.history['acc'], label='train accuracy')
plt.plot(h.history['val_acc'], label='validation accuracy')
plt.legend()
plt.show()
```





7. Model prediction and Output

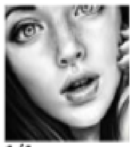
```
input_image_path = input('Path of the image to be predicted: ')
input_image = cv2.imread(input_image_path)
cv2_imshow(input_image)
input_image_resized = cv2.resize(input_image, (128,128))
input_image_scaled = input_image_resized/255
input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])

input_prediction = model.predict(input_image_reshaped)
```

```
print(input_prediction)
input_pred_label = np.argmax(input_prediction)
print(input_pred_label)
if input_pred_label == 1:
  print('The person in the image is wearing a mask')
else:
  print('The person in the image is not wearing a mask')
```

```
Path of the image to be predicted: /content/data/with_mask/with_mask_1.jpg
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 705ms/step
[[0.3104725  0.75629765]]
1
The person in the image is wearing a mask
```

```
Path of the image to be predicted: /content/data/without_mask/without_mask_1027.jpg
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
[[0.5376094  0.44479907]]
0
The person in the image is not wearing a mask
```

RESULT:

Thus the Face Mask Detection using CNN is built successfully and the output is verified.