Face Mask Classification using Transfer Learning (MobileNetV2)

Rushikesh Musale (Roll No: 240670)

Objective

The goal of this project is to build a deep learning model that classifies face images into three categories:

- **with_mask**
- **without_mask**
- **mask_weared_incorrect**

We use **transfer learning** with the MobileNetV2 architecture, which allows efficient training on small datasets.

Dataset

The dataset is organized into the following folders:

- `faces/train/` - for training
- `faces/val/` - for validation
- `faces/test/` - for testing

Each folder contains three subfolders corresponding to the mask classes.

Preprocessing

We use Keras `ImageDataGenerator` to apply preprocessing and data augmentation such as:

- Rescaling pixel values to [0, 1]
- Random rotations, zooms, shifts, shears, and horizontal flips

Model Architecture

We use MobileNetV2 as a feature extractor:

- Pre-trained weights from ImageNet
- Top layers (classification head) replaced with:
  - Global Average Pooling
  - Dropout (0.5)
  - Dense layer with `softmax` activation

Training

- Optimizer: Adam
- Loss: Categorical Crossentropy
- Epochs: 15 (with EarlyStopping based on validation loss)

Evaluation and Inference

After training:

- The model is saved as `mask_model_mobilenetv2.h5`
- Each image in the test set is predicted individually
- A label is drawn directly on the image using OpenCV
- Resulting images are saved in `labeled_test_images/`

Sample Output

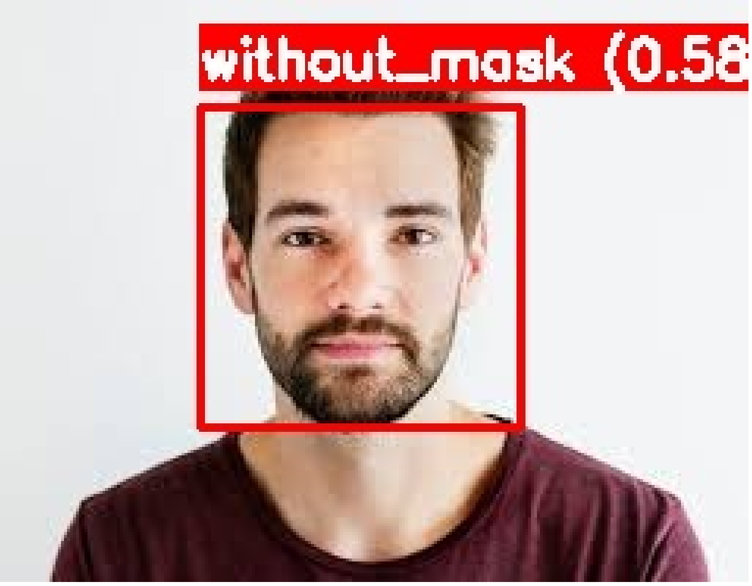

Figure 1: Annotated face mask classification output

Inference on New Images

To test the model on unseen images, we perform the following steps:

1. Load the trained model `mask_model_mobilenetv2.h5`
2. Load the Haar cascade face detector from OpenCV
3. Loop through images in the `test_images` folder
4. For each face detected:
    - Crop the face from the image
    - Resize to 128x128 pixels
    - Normalize and preprocess the image
    - Use the model to predict mask status
5. Draw a bounding box around the face with the label and confidence
6. Save the annotated image to the `output` folder

Label Colors:

- Green box - With Mask
- Red box - Without Mask or Incorrect

Sample code snippet:

```
face = image[y:y + h, x:x + w]
face_resized = cv2.resize(face, (128, 128))
face_normalized = face_resized.astype("float32") / 255.0
face_input = np.expand_dims(face_normalized, axis=0)
pred = model.predict(face_input)[0]
label = class_names[np.argmax(pred)]
```

Output images with bounding boxes and class labels are saved automatically for review.

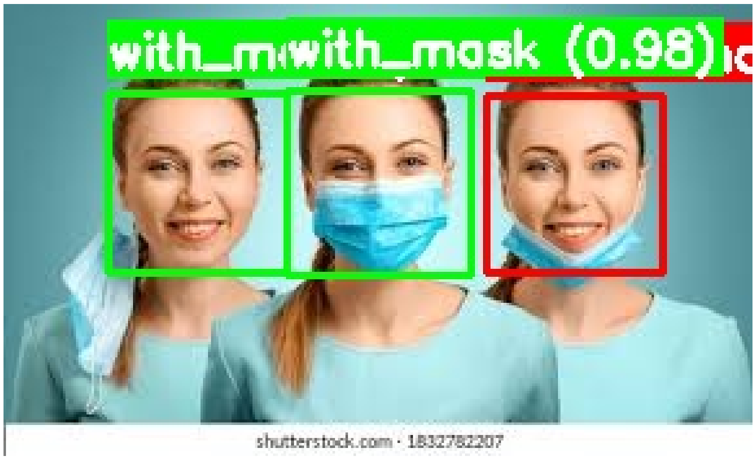

Figure 2: Labeled result using face detection and mask classification

This completes the full pipeline from training to real-world inference.