**Model :**

import os

import cv2

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import time

from keras import Sequential

import seaborn as sns

import random

from sklearn.model\_selection import train\_test\_split

import tensorflow as tf

from tensorflow import keras

from keras.applications.vgg16 import VGG16

from tensorflow.keras.models import load\_model

categories = ['with\_mask','without\_mask']

data = []

for category in categories:

path = os.path.join('train',category)

label = categories.index(category)

for file in os.listdir(path):

img\_path = os.path.join(path,file)

img = cv2.imread(img\_path)

img = cv2.resize(img,(224,224))

data.append([img,label])

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.2)

vgg = VGG16()

model = Sequential()

for layer in vgg.layers[:-1]:

model.add(layer)

for layer in model.layers:

layer.trainable = False

model.add(Dense(1,activation='sigmoid'))

model.compile(optimizer='adam',loss='binary\_crossentropy',metrics=['accuracy'])

h1 = model.fit(X\_train,y\_train,epochs=5,validation\_data=(X\_test,y\_test))

import cv2

import numpy as np

from tensorflow.keras.applications.vgg16 import preprocess\_input

def detect\_mask\_from\_frame(frame, model):

img = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB) # Convert BGR to RGB

img\_resized = cv2.resize(img, (224, 224)) # Resize to VGG16 input size

img\_resized = np.array(img\_resized, dtype=np.float32) # Convert to NumPy array

# Ensure 3 channels (handle grayscale images)

if len(img\_resized.shape) == 2:

img\_resized = cv2.cvtColor(img\_resized, cv2.COLOR\_GRAY2RGB)

img\_resized = preprocess\_input(img\_resized) # Apply VGG16 preprocessing

img\_resized = img\_resized.reshape(1, 224, 224, 3) # Reshape for model input

y\_pred = model.predict(img\_resized)

y\_pred1 = model.predict(img\_resized)

y\_pred = (y\_pred > 0.5).astype(int) # Convert probability to 0 or 1

y\_pred = int(y\_pred[0][0]) # Convert NumPy array to integer

class\_mapping = {0: "With Mask", 1: "Without Mask"}

return class\_mapping[y\_pred],y\_pred1

**Flask :**

from flask import Flask, request, render\_template  
import numpy as np  
import cv2  
from keras.models import load\_model  
from keras.applications.vgg16 import preprocess\_input  
from PIL import Image  
import os  
  
app = Flask(\_\_name\_\_)  
model = load\_model('model1.h5') # Path to your .h5 model  
  
UPLOAD\_FOLDER = 'static/uploads'  
os.makedirs(UPLOAD\_FOLDER, exist\_ok=True)  
app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER  
  
def detect\_mask\_from\_frame(frame, model):  
 img = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)  
 img\_resized = cv2.resize(img, (224, 224)).astype(np.float32)  
  
 if len(img\_resized.shape) == 2:  
 img\_resized = cv2.cvtColor(img\_resized, cv2.COLOR\_GRAY2RGB)  
  
 img\_resized = preprocess\_input(img\_resized)  
 img\_resized = img\_resized.reshape(1, 224, 224, 3)  
  
 y\_pred\_prob = model.predict(img\_resized)[0][0]  
 y\_pred = 0 if y\_pred\_prob <= 0.5 else 1  
  
 class\_mapping = {0: "With Mask", 1: "Without Mask"}  
 confidence = y\_pred\_prob \* 100 if y\_pred == 1 else (1 - y\_pred\_prob) \* 100  
  
 return class\_mapping[y\_pred], round(confidence, 2)  
  
@app.route('/', methods=['GET', 'POST'])  
def index():  
 prediction = None  
 confidence = None  
 filename = None  
  
 if request.method == 'POST':  
 file = request.files['image']  
 if file:  
 filename = os.path.join(app.config['UPLOAD\_FOLDER'], file.filename)  
 file.save(filename)  
  
 image\_pil = Image.open(filename).convert("RGB")  
 frame = np.array(image\_pil)  
 frame = cv2.cvtColor(frame, cv2.COLOR\_RGB2BGR)  
  
 prediction, confidence = detect\_mask\_from\_frame(frame, model)  
  
 return render\_template('index.html',  
 prediction=prediction,  
 confidence=confidence,  
 filename=filename)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

**Webpage :**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Mask Detection App</title>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 }  
  
 html, body {  
 height: 100%;  
 width: 100%;  
 font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
 background: linear-gradient(135deg, #74ebd5 0%, #ACB6E5 100%);  
 display: flex;  
 justify-content: center;  
 align-items: center;  
 overflow: hidden;  
 }  
  
 .container {  
 background: rgba(255, 255, 255, 0.95);  
 border-radius: 20px;  
 padding: 30px 40px;  
 box-shadow: 0 10px 25px rgba(0, 0, 0, 0.2);  
 max-width: 1000px;  
 width: 90%;  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
 gap: 20px;  
 }  
  
 h1 {  
 color: #2c3e50;  
 font-size: 32px;  
 margin-bottom: 10px;  
 }  
  
 form {  
 display: flex;  
 flex-direction: column;  
 align-items: center;  
 gap: 15px;  
 }  
  
 input[type="file"] {  
 font-size: 16px;  
 }  
  
 button {  
 background-color: #007BFF;  
 color: white;  
 border: none;  
 padding: 10px 25px;  
 font-size: 16px;  
 border-radius: 8px;  
 cursor: pointer;  
 transition: background 0.3s ease;  
 }  
  
 button:hover {  
 background-color: #0056b3;  
 }  
  
 .preview-img,  
 .result {  
 background: #f1f1f1;  
 padding: 15px;  
 border-radius: 12px;  
 width: 100%;  
 text-align: center;  
 box-shadow: 0 4px 10px rgba(0, 0, 0, 0.08);  
 }  
  
 .preview-img img {  
 max-height: 200px;  
 border-radius: 8px;  
 margin-top: 10px;  
 }  
  
 .result p {  
 margin: 5px 0;  
 font-size: 18px;  
 }  
  
 .prediction {  
 color: #ff6b6b;  
 font-weight: bold;  
 }  
  
 .confidence {  
 color: #28a745;  
 }  
  
 .footer {  
 margin-top: 10px;  
 font-size: 14px;  
 color: #555;  
 }  
 </style>  
</head>  
<body>  
 <div class="container">  
 <h1>😷 MaskSetu - Mask Detection App</h1>  
  
 <form method="POST" enctype="multipart/form-data">  
 <input type="file" name="image" required>  
 <button type="submit">Predict</button>  
 </form>  
  
 {% if filename %}  
 <div class="preview-img">  
 <p>Uploaded Image:</p>  
 <img src="{{ filename }}" alt="Uploaded Image">  
 </div>  
 {% endif %}  
  
 {% if prediction %}  
 <div class="result">  
 <p class="prediction">Prediction: {{ prediction }}</p>  
 <p class="confidence">Confidence: {{ confidence }}%</p>  
 </div>  
 {% endif %}  
  
 </div>  
</body>  
</html>