```
import numpy as np
import os
import math
import cv2
from fer import FER
import pyttsx3
from keras.models import model_from_json
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
from keras.models import load_model
from flask import Flask, render_template, Response, request
import  tensorflow as tf
from cvzone.HandTrackingModule import HandDetector
from skimage.transform import resize
model=load_model('model.h5')
font = cv2.FONT_HERSHEY_SIMPLEX
vals=['A','B','C','D','E','F','G','H','I']
emotion_detector = FER(mtcnn=True)
app=Flask(__name__,template_folder="template")
print("Accessing video stream")
app.static_folder = 'static'
vs=cv2.VideoCapture(0)
detector=HandDetector(maxHands=1)
pred=""
def SpeakText(command):
    engine = pyttsx3.init()
    engine.say(command)
    engine.runAndWait()
def generate_frames():
    while (vs.isOpened()):
        success, frame = vs.read()
        hands, frame=detector.findHands(frame)
        dominant_emotion, emotion_score = emotion_detector.top_emotion(frame)
```

```
if not success:
    break
else:
    if hands:
        hand=hands[0]
        x,y,w,h=hand['bbox']
        imgCrop=frame[y-20:y+h+20,x-20:x+w+20]
        black=np.ones((300,300,3), np.uint8)*0
        ishape=imgCrop.shape
        if h/w>1:
            k=300/h
            wcal=math.ceil(k*w)
            imgresize=cv2.resize(imgCrop,(wcal,300))
            irshape=imgresize.shape
            wgap=math.ceil((300-wcal)/2)
            black[:,wgap:wcal+wgap]=imgresize
        else:
            k=300/w
            hcal=math.ceil(k*h)
            imgresize=cv2.resize(imgCrop,(300,hcal))
            irshape=imgresize.shape
            hgap=math.ceil((300-hcal)/2)
            black[hgap:hcal+hgap,:]=imgresize
        img=resize(black,(64,64,1))
        img=np.expand_dims(img,axis=0)
        if(np.max(img)>1):
            img = img/255.0
        predict_x=model.predict(img)
        classes_x=np.argmax(predict_x,axis=1)
        x=classes_x[0]
        SpeakText(vals[x])
        value=vals[x] +" "+ dominant_emotion
```

```
            cv2.putText(frame,value,(x+20,y+20),cv2.FONT_HERSHEY_SIMPLEX, 1,(255, 255, 150),2,cv2.LINE_AA)
        ret, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')


@app.route('/')
def index():
    return render_template('index.html')
@app.route('/sign_to_speech')
def sign_to_speech():
    return render_template('sign_to_speech.html')
@app.route('/speech_to_sign')
def speech_to_sign():
    return render_template('speech_to_sign.html')



@app.route('/video',methods=['GET', 'POST'])
def video():
    return Response(generate_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')
if (__name__ == "__main__"):
    app.run(debug=True)
```