AI Module

22 September 2020 17:20

© RAJAT KUMAR
https://www.linkedin.com/in/imRajat/
https://github.com/im-Rajat

AI module

The AI Module includes :

- Detect Face : Detecting Human face using a given image and returning true or false on the basis of human presence.
- Detect Mask : Detecting Mask on Human Face using a given image and returning true or false on the basis of Mask presence.
- Coordinates for center of forehead : Given an image which contains a human face, Getting Coordinates for center of forehead and returning it's value.

For Face Detection:
We require only one library OpenCV. We also require a required trained XML front face classifier which will be applied on the image given as input after converting it to greyscale.
We are using Haar Cascade algorithm, also known as Voila-Jones algorithm to detect faces. It is basically a machine learning object detection algorithm which is used to identify objects in an image or video. In OpenCV, we have several trained Haar Cascade models which are saved as XML files. Instead of creating and training the model from scratch, we use this file. We are going to use "haarcascade_frontalface_alt2.xml" file in this project.

Library Used: OpenCV
Function : def face_detection(image_path):
It require a single input, path of the image and return true and false on the basis of human presence

For Mask Detection:
It require libraries like TensorFlow, keras. First we are detecting face and after that we are using a pre-trained model for recognizing faces wearing a mask. The model is trained on RGB images so we convert the image into RGB here. The model identify mask and return probability of images with masks and without masks. On comparing these probability we can return true if with mask probability is greater than without mask probability and vice versa.

Library Used : OpenCV , tensorflow.keras.preprocessing.image, tensorflow.keras.models, tensorflow.keras.applications.mobilenet_v2, numpy
Function : def mask_detection(image_path):
It require a single input, path of the image and return true and false on the basis of presence of mask on human face.

Coordinates for center of forehead:
Firstly we do the same process of face detection. After detecting the face we will have coordinates of the same. So now we are dividing the face horizontally in 3 parts. The upper part are get the forehead of face. Now we are finding the middle of that part, it will give us the x, y coordinates of the center of forehead

Library Used: OpenCV
Function : def forehead_coordinates(image_path):
It require a single input, path of the image and x, y coordinates of center of forehead. If face not detected in the given image then it will return "face not detected".

References:
https://www.mygreatlearning.com/blog/real-time-face-detection/
https://github.com/opencv/opencv/tree/master/data/haarcascades
https://www.youtube.com/watch?v=Ax6P93r32KU&ab_channel=BalajiSrinivasan
https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/
https://towardsdatascience.com/a-guide-to-face-detection-in-python-3eab0f6b9fc1
https://qengineering.eu/install-tensorflow-2.1.0-on-raspberry-pi-4.html

Face Detection using OpenCV:-

We will be using Haar Cascade algorithm, also known as Voila-Jones algorithm to detect faces. It is basically a machine learning object detection algorithm which is used to identify objects in an image or video. In OpenCV, we have several trained Haar Cascade models which are saved as XML files. Instead of creating and training the model from scratch, we use this file. We are going to use "haarcascade_frontalface_alt2.xml" file.
Its taking about 4.5 second to detecting faces on 30 images.

Face Detection using MTCNN:-

we are going to see how we can detect faces by using an Image-based approach. **MTCNN or Multi-Task Cascaded Convolutional Neural Network** is unquestionably one of the most popular and most accurate face detection tools that work this principle. As such, it is based on a **deep learning architecture**, it specifically consists of 3 neural networks (P-Net, R-Net, and O-Net) connected in a cascade. We need to install MTCNN library which contains a trained model that can detect faces.
Its taking about 35.7 seconds to detecting faces on 30 images.

Histogram of Oriented Gradients (HOG) in Dlib
The second most popular implement for face detection is offered by Dlib and uses a concept called Histogram of Oriented Gradients (HOG).
The idea behind HOG is to extract features into a vector, and feed it into a classification algorithm like a Support Vector Machine for example that will assess whether a face (or any object you train it to recognize actually) is present in a region or not. The features extracted are the distribution (histograms) of directions of gradients (oriented gradients) of the image. Gradients are typically large around edges and corners and allow us to detect those regions.
Its taking about 20 seconds to detecting faces on 30 images.

Observation : According to Me, Face Detection using OpenCV is fastest. I takes only 4.5 seconds to detect faces on 30 images. But success rate of face detection using HOG is better then opencv. The success rate of HOG is almost 95%+ meanwhile success rate of opencv is around 80%. Need to test on more dataset for accurate results.