Hackathon – III

Table of Contents

This hackathon has been designed to help you practice, reinforce and apply various concepts learned in Module - 3.

Objective:

Upon successful completion of this Hackathon, you will integrate a system accessible through a mobile app, which can recognize expressions and identify the person in it.

Datasets:

- **For Face Recognition:**
  IMFDB Data is provided. (Data is provided in the shared folder)
  For more details, click on the link: http://cvit.iiit.ac.in/projects/IMFDB/

- **For Expression Recognition:**
  IMFDB Data Segregated by expressions is provided.
  Note: The data is not uniformly spread across all the classes.

  The Expressions available are as follows:
  - ANGER
  - DISGUST
  - FEAR
  - HAPPINESS
  - NEUTRAL
  - SADNESS
  - SURPRISE

Below is the list of files provided:

1. exp_recognition_model.py
2. exp_recognition.py
3. face_recognition.py
4. face_recognition_model.py
5. haarcascade_frontalface_default.xml
6. lbpcascade_frontalface.xml

exp_recognition.py:

- **detected_face:** (To detect faces in the image, upon which expression recognition model)
    - It uses Viola-jones face detector
    - It returns only one image which has maximum area out of all the detected faces in the image
    - If no face is detected, it returns zero (0)
- **get_expression:** (To return the detected Expression from the app)
    - Captured image from mobile is passed as parameter in base64 encoding in the API call.
    - The code to decode the image in base64 encoding is provided within the function.
    - Load the trained model and use it for expression recognition
    - This function should return the Expression in string form ex: "Anger"
    - **Caution: Don't change the definition or function name; for loading the model use the *current_path* variable*(It gives the path of the directory where the python file is getting executed from)*. Example is provided in comments in the file**

face_recognition.py:

- **get_similarity:** (To return the similarity between two faces from the app)
    - Captured image from mobile is passed as parameter in base64 encoding in the API call.
    - The code to decode the image in base64 encoding is provided within the function.
    - Load the trained Siamese model
    - Get the features for both the faces from the model and return the relevant similarity measure such as euclidean, cosine etc.
    - **Caution: Don't change the definition or function name; for loading the model use the *current_path* variable*(It gives the path of the directory where the python file is getting executed from)* . Example is provided in comments in the file.**

- **get_face_class:** (To return the face class from the app)

    - Captured image from mobile is passed as parameter in base64 encoding in the API call.

- The code to decode the image in base64 encoding is provided within the function.
- Load the trained Siamese model
- This should return the Face Class in string form ex: "AnilKapoor"
- Along with the Siamese, you need the classifier as well, which is to be fine-tuned with the classes that you want to recognize
- **Caution: Don't change the definition or function name; for referring to any path use the *current_path(It gives the path of the directory where the python file is getting executed from)* variable. Example is provided in comments.**

exp_recognition_model.py, face_recognition_model.py:

- Define your models, transformation and all necessary helper functions here respectively for Expression Recognition and face Recognition model
- You can 'import' these into the files **exp_recognition.py, face_recognition.py**
- **READ ALL CODE COMMENTS CAREFULLY**

Haarcascade_frontalface_default.xml, lbpcascade_frontalface.xml:

- A Cascade is basically a classifier which is used to detect particular objects from the source. The haarcascade_frontalface_default.xml and lbpcascade_frontalface.xml are cascades designed by OpenCV to detect the frontal face
- Place these files in the same directory as capture_face_imagesv1.py, capture_expression_imagesv1.py, exp_recognition.py, face_recognition.py

Steps to upload your code to the server to access it from the Mobile app:

- Upload your files to the given ftp server and test your model on the mobile app
- Steps to upload the updated code files to the server for the mobile app is present in the document "FileZilla installation"

Mobile App URL:

<Will be made available here a few days before Hackathon>

- Open this link in your Mobile phone and join as a tester by clicking on the button “Become a tester”
- In the page redirected, click on “download it on google play”
- Install the app in your android phone
- For App usage documentation refer to “Mobile_APP_Documentation”

Tasks:

For tasks refer the starter code at the following link
https://drive.google.com/file/d/17QCNvUNHvNirQWo06uTre7JCtx88kB6U/view?pli=1