Deployment Related Files

This document contains the explanation/details of files that will help you deploy/unpack your machine-learning model on the remote server. It's very important that you use PyTorch 'state dictionaries' to load and save your model. It is strictly advised to not save the entire model itself. Related links (for loading/saving models in PyTorch) were already shared as part of reading materials.

Below is the list of files provided for Deployment:

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 a 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 a 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:

1. **Don't change the definition or function name**
2. **For loading the model, use the *current_path* variable *(It gives the path of the directory where the python file is getting executed from)*. An 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 a 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:

1. **Don't change the definition or function name**
2. **For loading the model, use the *current_path* variable *(It gives the path of the directory where the python file is getting executed from)*. An example is provided in the comments in the file.**

- **get_face_class:** (To return the face class from the app)
  - Captured image from mobile is passed as a 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:

1. **Don't change the definition or function name.**
2. **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. An example is provided in the comments.**

exp_recognition_model.py, face_recognition_model.py:

- Define your models, transformation, and all necessary helper functions here respectively for the 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 that 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
  - exp_recognition.py, face_recognition.py, exp_recognition_model.py, face_recognition_model.py