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.

Required packages:
Pytorch 0.4.1, base64, torchvision 0.2.1, cv2 3.4.2, io, os, PIL 1.1.7(Version compatible with your Pytorch version), numpy 1.15.0, pandas 0.23.4, scipy 1.1.0

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. capture_face_imagesv1.py
2. capture_expression_imagesv1.py
3. exp_recognition_model.py
4. exp_recognition.py
5. face_recognition.py
6. face_recognition_model.py
7. Functionality_check.py
8. haarcascade_frontalface_default.xml
9. lbpcascade_frontalface.xml

capture_face_imagesv1.py:

(To collect face images to fine tune the face recognition network with your data)

- When you run it, it opens up a video capturing window with a prompt for a particular class.
- When space is pressed, it captures the image and saves the largest face present in the image into the folder "captured_face_images"
- Please make sure that the area of faces saved is comparable with the areas of images used to train the network by adjusting *area_threshold* parameter within the file under discussion.
- You can set the number of images required per class by changing *Images_required* within the file under discussion.
- The naming convention for the saved images is <class_name>_<unique_id>.png

capture_expression_imagesv1.py:

(To collect expression images to fine tune the expression recognition network with your data)

- When you run it, it opens up a video capturing window with a prompt for a particular class.
- When space is pressed, it captures the image and saves the largest face present in the image into the folder "captured_images_with_Expression"
- Please make sure that the area of faces saved is comparable with the areas of images used to train the network by adjusting *area_threshold* parameter within the file under discussion
- You can set the number of images required by changing *Images_required* parameter within the file under discussion.

- The naming convention for the saved images is <expression>_<unique_id>.png

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**

Functionality_check.py:

- This is to mimic the mobile app camera and functionality on your laptop

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:

https://play.google.com/apps/testing/com.talentsprint.android.expressionrecognition

- 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:

Stage 1 (Face Recognition): (10 marks) (Hint: We had an experiment on it)

- From the provided IMFDB data, select any ten celebrity face images (100 classes is too big) and add one of your team member's data
  (11th celebrity :p) using the "capture_face_imagesv1.py"
- Define and train a Siamese network
- *Save the state dictionary of the Siamese network (use pytorch only), It will be useful in integrating to the mobile app*
- Define and train a classifier which takes output from the above trained Siamese network (*feature extraction network)* as input to classify the above 11 classes
- *Save the state dictionary of the classifier (use pytorch only), It will be useful in integrating to the mobile app*
- "face_recognition.py" routine contains the necessary skeleton, integrate your model in this to predict the face and to get similarity measure.
- Upload your files to the given ftp server and test your model on the mobile app
- **Grading Scheme:**
  - Face Similarity (5M): The face similarity should close similar faces and should be distant for dissimilar faces using the mobile app's "Face Similarity" functionality
  - Face Recognition (5M): Recognize the person correctly using the mobile app's "Face Recognition" functionality

Stage 2 (Expression Recognition): (15 marks)

- *Define and train a CNN for expression recognition with the given IMFDB data segregated on expression basis.*
- Collect your data using "capture_expression_imagesv1" and fine-tune the CNN for expression data on your team
- "exp_recognition.py" routine contains the necessary skeleton, integrate your model to predict the expression on the face
- Upload your files to the given ftp server and test your model on the mobile app
- **Grading Scheme:**
  - Expression Recognition (5M): If the functionality of giving back an expression class for the face using the mobile app's "Expression Recognition" functionality
  - Sequence Expression (10M): Get three consecutive correct Expressions using the mobile app's "Sequence Expressions" functionality

Stage 3 (Anti Face Spoofing): (10 marks)

The objective of anti-spoofing, is to be able to unlock (say) a screen not just by your image (which can be easily be spoofed with a photograph of yours) but by a switch in the expression demanded by the Mobile App (which is much less probable to mimic)

- **Grading scheme:**
    - Anti Face Spoofing: (10M Only if both the cases mentioned below are achieved)
        - Unlock: Correct face + Correct Demanded Expression
        - Stay Locked: Correct face + Incorrect Demanded Expression (as you might imagine there are multiple other such possibilities, which you are free to explore)

BONUS: (Functionality_check.py) 5M:

recognize face and expression for multiple users on the laptop camera screen which was opened by running Functionality_check.py