CAP 4628/5627 Project 2 – Emotion Recognition
Due 04/06 by 11:59pm

Project Description

The US Armed Forces has over 2 million soldiers when reserve components are included. It includes the Army, Marines, Air Force, and Coast Guard. New ways to manage soldiers pain, in fight against opioid abuse, are being investigated.

One way of managing soldier's pain is identifying and treating pain as soon as it occurs. Considering this, they are interested in ways to automatically determine if soldiers are in pain (e.g. wounded), in real-time.

As you have shown them that it is possible to recognize pain with physiological data, they are interested in the development of a new system that can identify pain from images (e.g. drone data), using state-of-the-art deep learning. You will have access to the type of data that will be collected to train and test your system. See below for details.

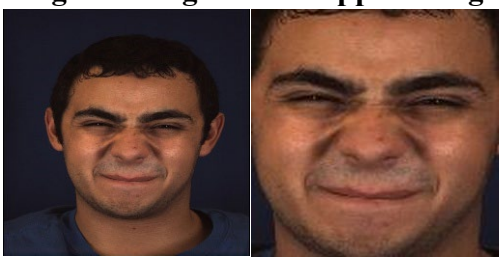
Instructions

- The project must be written in Python. You are required to write a script called Project2.py, however, you are free to split up your program into multiple files if you can run python Project2.py W H dataDirectory.
    - W is the width of an image (integer)
    - H is the height of an image (integer)
    - dataDirectory is directory where data is located. This should be an **absolute** directory. Don't make it relative to your script.
    - More details on this are given below.
- The data for this project is available on Canvas.
- The data has already been split into train, test, and validation sets. **DO NOT** do any splits of the data. The data is ~320 MB in size. If you have an issue with this much data, let me know as soon as possible.
- Data hierarchy is as follows:
    - Project2Data
        - Testing
            - Pain
                - Image data
            - No_pain
                - Image data
        - Training
            - Pain
                - Image data
            - No_pain
                - Image data

- Validation
  - Pain
    - Image data
  - No_pain
    - Image data

- You will use deep learning to classify the images as either pain or no pain. You can use any network architecture that you choose for this including the architecture in testCNN.py that was already given to you.
- You need to read in the training, validation, and testing splits into memory. You can use whatever you want for this. Keras and OpenCV have great functions to do this. The data will need to be read in then sent to the deep network for training and testing (see testCNN.py).
- You must crop (only show the face) the face images to a size of WxH, where W is the width of the cropped image and H is the height of the cropped image. OpenCV works great for this. You can use OpenCV to read in the images, detect the face (there are functions for this) and then set the region of interest to the face. You can find a lot of examples online how to do this as it is a common problem in computer vision. In doing this, the size of the images being sent into your network must be WxH.
  - You have to test/report 2 sizes. 128x128 and 64x64.
  - An example test run: python Project2.py 128 128 ./Project2Data

  Original Image **Cropped Image**

- The output of your script must print the confusion matrix, classification accuracy, precision, recall, and binary F1 score (*hint: look up sklearn.metrics.f1_score)* for the test set.

Useful Links

- **Keras -** https://keras.io/
- **Tensorflow -** https://www.tensorflow.org/tutorials
- **Scikit-learn** - https://scikit-learn.org/stable/
- **Python** - https://www.python.org/
- **OpenCV -** https://opencv.org/

Questions

1. What class (pain or no pain) did your architecture classify the most on the testing set for 128x128 and 64x64? In other words, were more classification pain or no pain. Hint: You can look at the confusion matrix to see this. Where they the same? Why do you think this was? Why do you think you got the accuracy that you did?
2. When looking at the accuracy, precision, recall, and f1-scores, do they make sense? In other words, if you have a high accuracy, do you have a high precision, recall, and f1-score as well (or vice-versa – low scores)? Why do we report multiple metrics (e.g. accuracy) when talking about recognition systems?
3. Based on the highest accuracy you got (either 128x128 or 64x64), do you think it makes sense to classify pain based on images alone? Why/Why not? If not, what else do you think we need to use?
4. Did you get a higher accuracy on the physiological data from project 1, or the images from this project (Use the highest accuracy you achieved)? Why? **(CAP 5627 only)** Is this a fair comparison to make? Why/Why not?
5. For a pain recognition system, would you want a higher recall or precision? Why?
6. **(CAP 5627 only)** Given the system you developed; how difficult would it be to change it to a stress recognition system? What would you need/need to do? Do you think it would be more or less difficult to recognize stress, compared to pain, using face images only?

Turn in

- Functioning python script(s).
- PDF of answers to the above questions, as well as, the output from your script for both 128x128 and 64x64 (even if they are the same). This includes confusion matrix, classification accuracy, precision, recall, and F1 score for the test set. You can format this as you want if it is neat and easy to read.
- Turn all of this in a zip file, to Canvas by the due date.