Individual Final Report
Mike Labadie

Overview

- In this project, we explored techniques to perform emotion detection in images.
- My partner found the dataset from Carnegie Mellon and came up with the idea for the project. We worked together to improve the accuracy of our models. My partner was instrumental in finding the best hyperparameters for my initial model. We worked together to create the presentation and final report.

Individual Work

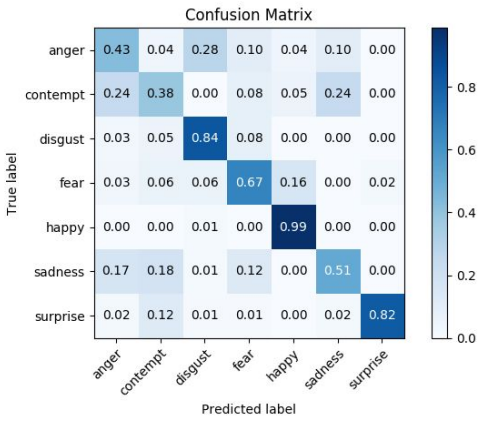
- I took the early lead on our coding efforts. I had only used the Keras framework by the time we started development on the project. As such, I focused on a Keras solution. I had seen during our first exam (in Keras's documentation) the use of training and testing using dataframe ImageDataGenerator. This data generator also provides a handy flow_from_dataframe() functionality. This allows you to identify your data sets with simple file location and class labels. I reference below where I found a good writeup on this. Keras also has good documentation on this practice.
- My first step was to create csv files that inventoried our images that could be read as dataframes for this data generator. This was largely a simple effort in pandas.
- As we discuss in the paper, many images did not have labels. I manually opened the "peak" image for each sequence and documented what I believed the emotion was.
- With dataframes for training and testing, I proceeded to attempt the first model. We've seen in class that CNN appears to be a powerful paradigm for visual recognition. As such, I first searched for documentation and examples on implementing CNN in Keras. Keras has a straightforward example of a model built for image recognition on the cifar10 dataset. This formed the basis of the first CNN model.
- My partner built some very nice parameter tuning code. This code produced hyperparameters that far exceed the original example (i.e. changed the optimizer to Adam). I created a script that trained and saved this model.
- My next step was to create a script to evaluate our models. I wrote the model_evaluation.py script that checks the accuracy on the hold out set. It also produces a confusion matrix and classification report.
- Likely my biggest contribution to the modeling effort was to explore face detection. I read an article (referenced below) on how facial recognition is performed. In this case, they are trying to identify individuals in images. The main takeaway from this article was to first locate and isolate faces in an image. The goal being to reduce the noise of everything else in the image. This post lead me to find the mtcnn library written by Ivan De Paz. This library easily allowed me to create a whole new set of images that contained just faces.
- I updated and trained the best model we had found at that point on the images containing just faces. This was a dramatic step forward in accuracy.

My Individual Scripts

The list below delineates the scripts I wrote for the project.

- Configuration: This file represents a centralized area for settings (i.e. file paths, naming, etc) used throughout the project.
    - configuration.py
- Preprocessing: These scripts were used to create datasets and metadata for training our models.
    - preprocessing_get_sequence_labels.py
    - training_split.py
    - preprocessing_helper_face_detection.py
- Modeling: These scripts represent the progression of our modeling efforts.
    - training_model_building.py
    - training_model_building_best_model.py
    - training_model_building_best_model_just_faces.py
- Evaluation: These scripts were used to evaluate the quality of our trained models.
    - model_evaluation.py
    - model_evaluation_helpers.py

Results

- The example/baseline model had 21% accuracy. The tuned model had 48% accuracy. The tuned model using face detection had 68% accuracy.
- The model had poor accuracy with the anger and contempt emotions. I do not find this surprising. I personally tagged over 200 images. I had a great deal of difficulty delineating the two emotions. I believe the potential inaccuracy of my own tagging compounded the difficulty of distinguishing between two emotions. Further, I struggle to imagine what expression I would make should I be requested to show contempt.
- We had very good results identifying happiness and surprise. I believe these two emotions are very relatable to humans. I believe these subjects probably had far less difficulty expressing these emotions. I certainly found it very easy to assign the happy emotion during manual tagging.
- The matrix below shows our accuracy across classes of emotion for our best model.
    - 

Conclusion

- I was very impressed with the effort Carnegie Mellon undertook. This was an extensive project with a noble goal. The robust effort was boosted by the multidisciplinary team they brought together.
- As the researchers stated, strong models would need a much larger image set. The CMU team speculated they would need tens of thousands of images.
- I found myself often thinking about this project in terms of how humans detect emotion. Humans "train" their own internal models to detect emotion everyday of their lives. As anyone in a relationship can attest, this is actually not always a straightforward (or easy) task.
- Regarding future work, I believe there is interesting information to be gathered in evaluating a sequence of images, not just the peak or near peak images as we did.
- Obviously one large open question is how well our model could predict images "in the wild." The dataset we had very consistent poses and were taken directly from the front. I imagine most images would not come across so cleanly.
- It would also be interesting to understand if our model has particular weakness by gender or race.

Code from Other Sources

- In total, I wrote 12 scripts to perform a variety of functions. A number of these scripts simply allowed me to perform manual tasks or were early testing efforts.
- I borrowed heavily from a variety of sources as listed. Most of the direct code examples I copied were from documentation of the libraries I used, like Keras, OpenCV, mtcnn. I would approximate that ⅔ of my code was taken from such sources. With the other ⅓ being my own code to glue the project together.
- Other sites listed in the References provided inspiration. For example, I gained inspiration from an article about doing facial recognition. This lead me to bring in Ivan De Paz's pretrained face detection library. I didn't really take code from anyone's github. Perhaps, we would have a better model had I done so!

References

- Our Dataset from Carnegie Mellon
  - http://www.consortium.ri.cmu.edu/data/ck/CK+/
- Medium Post from Vijayabhaskar J about flow_from_dataframe in Keras
  - https://medium.com/@vijayabhaskar96/tutorial-on-keras-flow-from-dataframe-1fd4493d237c
- Keras Example using CNN
  - https://keras.io/examples/cifar10_cnn/
- Keras Modeling Documentation
  - https://keras.io/models/sequential/
- Keras Documentation on ImageDataGenerator
  - https://keras.io/preprocessing/image/
- Post from Adrian Rosebrock Describing ImageDataGenerator
  - https://www.pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/
- Post about Facial Recognition
  - https://machinelearningmastery.com/how-to-perform-face-recognition-with-vggface2-convolutional-neural-network-in-keras/
- Github from Ivan De Paz for His Face Detection Library
  - https://github.com/ipazc/mtcnn
- Reading Images as Grayscale and Color from OpenCV's Documentation
  - https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html
- Sklearn's Example of a Confusion Matrix
  - https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html