CS 178 Group 88 SuperFace API Writeup

Group member: Myles Liu, Songheng Zhang, Wenzhuo Wang, Yuexin Chen

Please be notified that this is the first document we have for the report! Just like what we have mentioned in this document's result section, we have another API documentation that should be uploaded along with this document. Please make sure that you also check out that one!

1. Problem Statement

Context

Before starting the development of the SuperFace API, we have looked up a lot of current facial machine learning APIs on the Internet. We have found libraries including TensorFlow, Keras, and even OpenCV. Even though these libraries have large amounts of useful APIs and pre-trained models, they are only specifically designed for developers who are professionals in the area of machine learning, which means if an ordinary developer wants to utilize machine learning, he or she would need to dig into an enormous size of these libraries' documentations.

In addition, although there are already a few ready-to-use facial machine learning APIs, they normally have a lot of limitations. Just take the example of Google ML Kit, which is aimed at solving the difficulty of connecting ordinary applications with facial machine learning techniques. Despite the fact that the kit has partially relieved the issue, itself would still have some problems such as the lack of ability to analyze (instead of just recognize) people's faces. Furthermore, if a developer wants to make use of Google ML Kit, he or she will have to be bound to the corresponding platform, i.e. Google Firebase in this case.

Objectives

We designed the SuperFace API with the following objectives in mind and turned them into great advantages for our API:

- Our API is advanced – Each API is an independent function that could be used frequently in modern computing daily life.
- Our API is ready to use – simply making a HTTP request with a few parameters will return the corresponding data.
- Our API is very efficient – The APIs can respond to the client's requests in just seconds, where consuming only a little bit of resources.
- Our API supports fast deployment – Each API can be deployed to the server easily where the only requirement for the developers is to set up a few basic running environments.
- Our API is easily modifiable – Thanks to the utilization of the standard machine learning libraries, developers who want to make a few tweaks on the APIs could achieve their goals in just a small amount of time compared with building their facial machine learning functionalities from scratch.

- Our API is understandable – With our demo APP, developers could glimpse into our APIs without hassle.

Entry Point

Human's faces have common features that would allow our APIs to capture and determine the person's age, gender, emotion and identity (name). The various colors of an image can badly influence the facial machine training results. Therefore, we make all input images to have the same color, on the level of gray scale. Landmark is an easily recognizable nature of human face; we get the landmarks from the input images and use them to identify the niche of each person's face. For example, we can use landmarks to analyze people's emotions and trace moving faces in the camera. We can also use landmarks for gaze detections.

Future Planning

We plan on continuously adding new useful APIs in order to let more daily applications be able to take advantage of modern and advanced facial machine learning techniques conveniently. Moreover, we will be maintaining our APIs frequently to make sure they are always doing their best jobs, including optimizing the logics of our APIs, upgrading lower-level libraries' functions that we are using.

2. Citation

Citation is provided in the text body of this report at relevant places. Please refer to the numbers in bibliography to find the corresponding sources.

3. Decomposition

For Myles, he was responsible for developing the whole server, the whole Demo App, all the PHP files, and final tweaks of all Python Machine Learning files. For the server, he implemented it on Ubuntu 18.10 with Apache2, where he also set up all other environments needed to run the APIs on the server. For the Demo App, he chose to create an Android App where the developers can easily try all five APIs. Four of them are using pre-trained model, and there is one API (Superface Labeling) that the developers could choose to train and test the model by themselves. For the PHP files, they are all being used as the interfaces of the server, where they could either handle the clients' requests and sending back the results generated by the Python files or do some coordinate works like clearing previous training data and handling new uploaded training data. For tweaking Python Machine Learning files, when these files were finished writing by other group members, it usually would not run properly on the server because of various of issues. Also, the combination of PHP files and Python files would need some additional works. Furthermore, there are some more works done by him to make sure everything is functioning on both the server and the app. For example, since our server is not that powerful, where some Python Machine Learning files were running out of memory at first, he had searched on the Internet and found a method that would not cost a lot of resources to resize the photos uploaded by the clients[1].

For Songheng, he was responsible for developing the Facial Expression Detection API. He implemented the predict_mood.py, which is used to predict the facial expression by following the instruction from the van Gen P's blog. To train the facial emotion detection model, he downloaded the dataset Cohn-Kanade (CK and CK++), which is used for training and testing

the model using OpenCV[2]. He also used OpenCV's built-in haarcascade_frontalface model to get the standard form image from the input image and then created OpenCV internal face recognizer class named fisher faces. He applied cross-validation to randomly split the training image into two parts and use calculate the error rate of the trained model compared with the testing data. The dataset was 1.55 GB so he was not able to upload it. He wrote a helper python script to sorted the original dataset into different files according to their labels. In the next step, he converted each input image to grayscale, crops and save the treated image on the dataset folder which was used to save future added images from the users accessing this API. Even though the emotional classifier was trained with hundreds of photos, the prediction could not give prediction with over 80% confidence. For further improvement, he may apply CNN from TensorFlow to recognize facial expression via camera stream or images.

For Wenzhuo, he was responsible for developing the Gender Detection API and Object Detection API. He implemented the object_detection.py which is used to predict the name of each object in the image and detect_gender.py which is used to predict the gender of people in the image. The basic code of these python modules come from the Github code implemented by Arun Ponnusamy[3]. The detect_gender.py used pretrained model from Keras. The object_detection.py used the pretrained model from YOLO[4] with dataset comes from COCO[5]. He installed the environment for the code to works, including Keras, Opencv, Tensorflow. For the detection of gender, he first tries to import the pretrained model for later usage. Then for each face detected by Opencv, he draws a rectangle to highlight the face. comparing the face with the model, it will output the confidence level and the predicted class. For the detection of object, he uses the model provided by YOLO and the dataset provided by COCO. For the input image, it will check from the left upper side with a scale of 0.0392 that if it could be identified. Once the confidence level is greater than 0.5, it is considered as a predicted object.

For Yuexin, he was responsible for developing the Face Recognizer API. He implemented the FaceRecognizer.py, which is used to predict the name of the person in the test image provided by the user after the model learns and analyzes sample data of two different people (along with the corresponding names of the people) provided by the user. The Python module was based on the code template from Github repository "Face Recognition with OpenCV and Python" created by Ramiz Raja, a passionate Android developer with 5+ years of experience who loves writing Android apps and open source Android libraries[6]. In addition, Yuexin followed the OpenCV tutorial written by Adrian Rosebrock when writing the Python module[7]. The module utilizes OpenCV and Local Binary Patterns Histograms (LBPH) Face Recognizer to train the model with input from the user at real-time. More training data will make the prediction more accurate. Yuexin is also responsible for writing and organizing the final project writeup.

4. Experience

The final project gave us the opportunity to implement what we have learned of machine learning knowledge through logical thinking and programming. For example, in detection facial mood model, we used a dataset of different labeled faces to train our model. Furthermore, we applied cross-validation to randomly split testing data and training data from our dataset to improve the accuracy. Comparing the trained model result with the test data, we recorded both the correct and incorrect answers and use ((100*correct)/(correct + incorrect)) formula to calculate the error rate. Due to limited time, we were not able to use BFGS or steepest gradient

descent method the minimize the error rate, but they are two promising directions that can help improve the accuracy of the model.

5. Results

To take use of our API or want to see what we have done in the API, please refer to another word document called "Superface API Documentation," which should be uploaded along with this word document.

If you would want to access our server, please use the following account.

Use ssh connect to 45.32.67.159.

Account: myles

Password: Fyqlyx12~

After logging into the server, please type "cd /var/www/html/superface" to go to our superface API folder.

This is the server we use to for our demo App. Also, this is a personal server belongs to one of our group member called Myles, if you have any issue with this server, please contact him by email: yuxil11@uci.edu.

Please be notified that this account has sudo right, so do not tell others the account!

Also, if you would like to do any modification to the server, please send a notification to Myles.

6. Conclusions

Even though, the fisherface classier in the Facial Expression Detection model is not with high accuracy, it can be improved in the future with larger training dataset. Facial expression detection can be used in precisely advertisement delivery. Advertisement producers can detect testers' facial expression to confirm if their advertisement successfully makes the viewer move. Also, it has a huge potential in online education industry because facial detectors can automatically determine whether the students are listening or reading or just taking a nap at their desks. According to the Chinese news media, a middle school in Hangzhou is trying to apply to similar technology to oversee the students and give corresponding "Attention rate" to help teachers improve their lesson.

7. Bibliography

[1] Delgado, Carlos., (2016). How to resize an image and reduce quality in php without imagick. <https://ourcodeworld.com/articles/read/197/how-to-resize-an-image-and-reduce-quality-in-php-without-imagick>.

[2] Kanade, T., Cohn, J. F., & Tian, Y. (2000). Comprehensive database for facial expression analysis. Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG'00), Grenoble, France, 46-53. Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., & Matthews, I. (2010). The Extended Cohn-Kanade Dataset (CK+): A complete expression dataset for action unit and emotion-specified expression. Proceedings of the Third International Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB 2010), San Francisco, USA, 94-101. "Cohn-Kanade AU-Coded Expression Database." The Affect Analysis Group at Pittsburgh. University of Pittsburgh. 22 Mar. 2019 <http://www.pitt.edu/~emotion/ck-spread.htm>.

[3] Ponnusamy, Arun. "Arunponnusamy/gender-detection-keras." Gender detection (from scratch) using deep learning with keras and cvlib. 02 Dec. 2018. Github. 22 Mar. 2019 <https://github.com/arunponnusamy/gender-detection-keras>. Ponnusamy, Arun. "Arunponnusamy/cvlib." Cvlib. 02 Dec. 2018. Github. 22 Mar. 2019 <https://github.com/arunponnusamy/cvlib>.

[4] Redmon, Joseph. YOLO: Real-Time Object Detection. 22 Mar. 2019 <https://pjreddie.com/darknet/yolo/>.

[5] "Common Objects in Context." COCO. 22 Mar. 2019 <http://cocodataset.org/#home>.

[6] Raja, Ramiz. "Informramiz/opencv-face-recognition-python." Face Recognition with OpenCV and Python. 20 Oct. 2017. Github. 22 Mar. 2019 <https://github.com/informramiz/opencv-face-recognition-python>.

[7] Rosebrock, Adrian. "OpenCV Tutorial: A Guide to Learn OpenCV." PyImageSearch. 04 Feb. 2019. PyImageSearch. 22 Mar. 2019 <https://www.pyimagesearch.com/2018/07/19/opencv-tutorial-a-guide-to-learn-opencv/>.