Final Exam Summary Report

To: Dr. John Hedengren
From: Group 4
Date: 24 March 2022
Re: Facial Recognition

Summary

Facial recognition has many applications and can be used everyday for simple things such as face ID on smartphones, social media filters, and to help cameras autofocus. The objective of this final project is to implement computer vision for facial recognition applications in a classroom. The facial recognition system should be able to detect when a student enters the classroom, greet them with a personalized message, record attendance with entrance time, and email the students that were absent after collection is finished. The process of making the facial recognition system will follow an Agile software development structure for collaboration and combine elements of data engineering, face detection, classification, and communication.

Agile development

Collaboration for the four main coding projects was coordinated using communication via Discord and uploads to a shared GitHub folder.

Data Engineering

The main goal of the data engineering workflow is to set up retrieval, processing, and storage of the video feed data. Videos for training and validation are read using the cv2 package. The input is resized and then split by frames and written to an associated folder. The frames for each student's video are also split into training and validation categories for later use on the classifier. Upon input to the classifier, an additional feature vector is constructed with the Local Binary Pattern (LBP) for each pixel. As currently written, the cells handling the separation of datasets for each individual and pipeline to train the classifier discussed below must be run once for each student, changing the name each time.

Face Detection

The OpenCV package contains a pre-built Multi-task Cascaded Convolutional Neural Network (MTCNN) for face detection. This pre-trained classifier is stored as an xml file and accessed at http://apmonitor.com/pds/uploads/Main/cascade.xml. The MTCNN is set up to take images and identify all faces by bounding box and the position of features including nose, right eye, left eye, and mouth, which output is seen in **Figure 1** below.

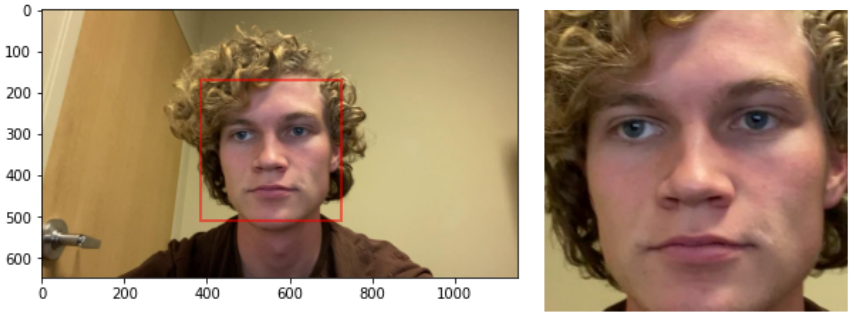

Figure 1. MTCNN Function. (L) Input image. (R) Output restricted to the bounding box.

For our purposes the bounding box returned by the function is used to crop out the faces for input to a classifier with student names. This occurs for the training and validation data as well. Note that just before the classifier training cells, there is a warning that the separated training sets should be manually examined for poor face detection or errant frames.

Classification

Split data sets for training and validation of the student face classifier are organized similarly to the texture classification project. Given the reasonable size of available training data, a Convolutional Neural Network (CNN) is used as the classifier with 3 convolutional layers and an output dense layer. Initial evaluation of the classifier showed an accuracy of 95.2% on the training set and 95.33% on the validation set, split according to **Figure 2-3** below.

Figure 2. Confusion Matrix of Classifier Results on Group Member Training Data.

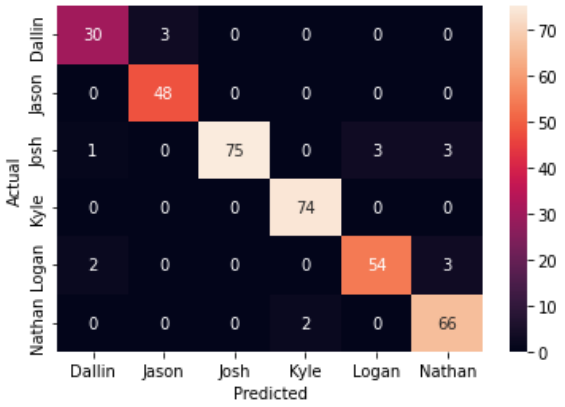

Figure 3. Confusion Matrix of Classifier Results on Group Member Validation Data.

Results shown in the **Figure 2-3** Confusion Matrices indicate minimal errors. This output was expected given the relatively large training dataset provided. No in depth investigation was made into why some errors occurred, but it is likely heavily influenced by the assortment of emotion between training and validation sets. A similar accuracy measure between training and validation also shows that the classifier is not overtrained. The trained classifier was implemented alongside the available cascade classifier previously mentioned under “Face Detection”. Under this configuration the camera connection is opened and faces are separated out from the video feed by their bounding boxes to be fed to the CNN and labeled by student name.

Communication

For the communication aspect of this project, the system was set up to welcome students as they enter the classroom using their name and to email the students that were absent for the day. When the classifier determines that a student has entered a classroom it records that name in a list. The Python package pyttsx3 can be used to speak a customized message, such as “Welcome to class”. This is used for each name that is recorded of the students that were present so that they are welcomed and their name is spoken. The names of the students that were present are then removed from a DataFrame of names and emails. Subsequently, this dataframe of absent students is then used to send an email with a short message about how they were missed and what was done in class for the day. The emails are sent using the Python package smtplib and by connecting to the mail.et.byu.edu SMTP server with a CAEDM username and password. This method is limited by the fact that you need to be on a campus IP address or connect through a VPN to use the SMTP server. Therefore, if this facial recognition system were to be implemented somewhere other than BYU, another SMTP server would be needed.

Conclusions & Recommendations

A functional face-detection system for tracking student attendance complete with personalized greetings and absentee reminders was implemented using a pre-trained MTCNN and CNN classifier. Future iterations of this project should focus on collating attendance information in a more usable format for the professor.

Appendix

Contributions

Dallin: Worked mostly on the communication aspect of the project. Outlined the code that compiled the students' contact information, generated an email with a message that would be sent to the absent students, and welcomed the present students using Python text to speech. Also assisted in evaluation of the trained classifier by generating confusion matrices for the training and test sets.

Jason: Contributed to the overall management of efforts as scrum master. Organized who would work on which element of the project. Gave meaningful feedback on the classification, communication, and face detection code.

Josh: Worked on data engineering, cleaning and prep. Helped take the videos and separate them into individual frames and resize the frames for use in the code.

Kyle: Worked on the face detection and compiled the code together. Also, helped finish up the completed training and test sets.

Logan: Worked mostly on the classification part of the project. Wrote the majority of the classifier used for face detection. Contributed in editing the final report.

Nathan: Contributed a few lines to the attendance time printout and communication section. Wrote summary memo and analysis of classifier results.

Links

https://github.com/17jsweeten/face_recognition_project