Detecting Classroom Engagement Through Facial Expression Analysis

Author: Dan Kercher

Sponsor: Snehasis Mukhopadhyay

Abstract:

The proliferation of online teaching tools has contributed to an environment in which facial recognition and expression tracking is less intrusive than ever before. Prior research in video analysis, facial recognition, and expression detection has paved the way for new applications in engagement analysis. I use these technologies to develop a program to analyze recorded video lectures to try to quantify the engagement of participants using their facial expressions as a metric. The future goal is to combine this raw data with direct survey feedback to train a model to predict engagement. I hope that this part of the engagement analysis puzzle can be combined with other techniques such as text transcriptions and vocal analysis to better measure classroom engagement along several interconnected channels.

Introduction:

Given a recorded classroom lecture, can a program determine if it was a "good" class session or not? The intersection of AI and classroom teaching has never been a more explorable space thanks to the proliferation of digital classroom technologies such as Zoom. Virtual teaching environments often make it difficult to judge student engagement, and this project seeks to make the task easier with an accurate and accessible tool.

Problem:

Detecting "engagement" begins with the question: what does it mean to be engaged? For this project, I focus on the sub-question: what does it look like to be engaged? This is a difficult question to answer as there is no universally agreed-upon empirical measure of "engagement." Previous work in classroom psychology provides insight into how we can begin to quantify engagement. To know what students look like in the first place, existing libraries for computer vision will be used; namely, OpenCV. Several pre-built facial recognition models such as OpenFace, DeepFace, and VGG Face were examined as well, though ultimately not used. A pre-trained open-sources model for facial expression detection is used on the identified faces to catalog the expressions of students. Other technologies used include additional Python libraries such as Keras, Pandas, and Tkinter, as well as the cloud-based survey platform Qualtrics. The first steps in my project were to research the current states of video processing, facial recognition, and facial expression analysis.

Research and Background:

For video processing, I chose to use OpenCV in Python as it is a highly regarded library for this topic. I then began researching facial feature detection analysis. I started with the influential paper "Robust Real-Time Face Detection" written by Paul Viola and Michael Jones and published by the International Journal of Computer Vision in 2004. The paper details the authors' approach to face detection, combining several innovative techniques to improve real-time detection efficiency. These innovations include a mathematical model for images, the adoption of adaptive boosting (AdaBoost), and a progressive filtration system. The system outlined by the authors encompasses what is now known as Haar Cascades. Haar Cascades are a progressive filtration system that the authors call an attentional cascade. Intuitively, the idea is that each successive step in a multistep feature filter should be more complex than the next, and that earlier steps should be made up of the simplest features that filter out the

most nonfaces. Earlier filters mustn't produce false negatives as that will preclude those sub-windows from being evaluated by the more advanced filters later down the line. The goal is to start with a filter that uses the fewest, simplest features that can discard the most nonfaces while detecting 100% of the positive faces. Sub-windows that pass the first feature set filter are then passed to more complex filters that are more accurate but also more computationally expensive. The idea is that the more expensive filters will be used far less often than the simpler initial filters. Mathematically, this feature can be understood as an increasingly expensive calculation set at each successive step of the cascade. Schematically, the cascade approach can be understood as a decision tree in which each previous filter must be passed to move on to the next filter further down the tree. I implemented Haar Cascades through the OpenCV library as an efficient and effective face detection solution.

Next, I researched cutting-edge facial expression recognition models created with the Keras frontend for TensorFlow. Two articles "Facial expression recognition with Keras," and "Real time facial expression recognition on streaming data," by Sefik Serengil got me started on this subject. Serengil outlines a video pipeline for preprocessing images before running them through the Keras prediction model that I adopted for my final project. I also used the same pre-trained weights for my model that Serengil uses which are trained on a famous face image set called Fec2013, introduced in a Kaggle challenge for expression recognition. Serengil's model and weights achieve a rough accuracy of 57%, compared to the Kaggle winner of 34%, which was enough for my project's needs. With video processing, facial recognition, and facial expression tools and models all set, I was ready to begin engineering my solution.

Solution:

Building on these previously researched foundations, my approach is to analyze the facial expressions of students from recorded Zoom lectures. My analyzer takes each frame of a video and runs it through the following pipeline. First, I copy the raw frame image to preserve a backup and resize the copy to 300x300 pixels. I then apply OpenCV's blobFromImage which applies mean subtraction and scaling as preprocessing steps to ready the image for the next stage of the pipeline. Next, I run Haar Cascade detection on the preprocessed image and store the bounds of any found faces in an array. The final stage of the pipeline is to loop through the array of faces, convert each to grayscale, further reduce them to 48x48 pixels, then run them through the Keras expression prediction model. This model returns the likelihood of each of seven emotions which I name: Happy, Sad, Neutral, Angry, Surprise, Thinking, and Disgust. For my program's purposes, I am only interested in the highest likelihood emotion and store that per face per frame. As an interesting note on the expression names, the fec2013 set uses the name "Fear" instead of "Thinking;" however, these names are contextual, and I believe the emotion being shown in the Zoom recordings is a thinking or contemplative state rather than worry or fear.

After resolving the bugs and getting my program working, I created an alternate version that used my webcam as video input and a simple Tkinter UI to allow me to interact with my analyzer. I also wrote some output that displays the original, unprocessed, image with the detected face bounds, and the dominant predicted emotion. This allowed me to further test the speed and optimizations of my code in real-time. It quickly became apparent that while Haar Cascades were very fast, they were not accurate enough for my needs. Reductions in video quality, which I was able to test with my webcam settings degraded the Haar Cascade accuracy sharply to the point where it was not finding clear and obvious faces. When the resolution was tuned up, the Haar Cascades had the opposite program and were returning many background objects that were false-positive faces.

At this point, I was very satisfied with the facial expression predictor, but I needed to upgrade my facial recognition model. I researched several possibilities including Single Shot Multibox Detector (SSD), Histogram of Oriented Gradients (HOG), Max-Margin Object Detection (MMOD), and Multi-task Cascaded Convolutional Networks (MTCNN). After reading about the pluses and deltas of each, I concluded that SSD was right for this project's needs as it was much faster than the others and retained much more accuracy than Haar Cascades. Once I implemented SSD into my program, the accuracy of my facial recognition at all webcam resolutions noticeably improved and I was able to process frames with minimal lag. Looking closely, you can see some lag on the output, but it is close to real-time. I also added a percentage tracker to my debugger output to show the confidence perfect that SSD assigns each face. Using this data, I was able to fine-time the variable in my code to only store faces above a certain confidence threshold to further improve accuracy. With that, my program was finished and produces a text file showing the total count of each of the seven emotions in an uploaded video, the percent for that emotion against the total emotions logged, and the dominant emotion for the entire video.

The final step of my project was to research and create a survey to collect data asking students to rate their engagement in lectures. I learned how to use Qualtrics and read some general advice on creating surveys to capture student engagement. The final version of my Qualtrics survey can be found in my final presentation slides. Once engagement data is collected from students for videos that can also be run through my analyzer, a final, more robust, program can be created to predict engagement based on raw data.

Conclusion, Future of the Project, and Lessons Learned:

I believe a program can be trained to identify "good" class sessions. My subproblem of using emotional state detection to evaluate student engagement can and should be combined with complementary research from other methodologies to create as robust a model as possible. I took every step to ensure that my research is compatible with complementary approaches to solving this problem. The intention is that my findings could be combined with text-chat and audio transcriptions, and audio frequency analysis to create a more rounded predictive model. Throughout this project, I was able to learn about machine learning, AI, biometrics, computer vision, and scientific survey taking. I also hope that my research can contribute to others' research into this topic to improve the quality of virtual classroom education. I am very interested in continuing to improve my program by implementing web integration, as well as looking into some of the other engagement measuring tools such as text and speech.

References:

1. P. Viola and M. Jones, “Robust real-time face detection,” *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*.
2. S. Serengil, “Facial expression recognition with Keras,” *Sefik Ilkin Serengil*, 01-Jan-2018. [Online]. Available: https://sefiks.com/2018/01/01/facial-expression-recognition-with-keras/. [Accessed: 28-Jan-2022].
3. S. Serengil, “Real time facial expression recognition on streaming data,” *Sefik Ilkin Serengil*, 10-Jan-2018. [Online]. Available: https://sefiks.com/2018/01/10/real-time-facial-expression-recognition-on-streaming-data/. [Accessed: 28-Jan-2022].
4. “Top 16 student survey questions to enhance your student feedback,” *QuestionPro*, 04-Oct-2021. [Online]. Available: https://www.questionpro.com/blog/student-survey/. [Accessed: 26-Feb-2022].