CS379FA23 Final Project Report

Title: User Identification via Hybrid Face and Speaker Recognition

Student's Full Name: Warren Wang

Time Taken to Complete: 50 hours

---

1. Introduction

A recorded video demo can be found at this YouTube link: https://youtu.be/T4OJqnAoNrc . Note that some things such as the score calculation for the hybrid attendance identification will have changed since the recording of the demonstration and at the time of writing this report. However, the underlying system remains virtually the same.

The original powerpoint (only modified to correct title) outlining what this project would be a few weeks prior can be found here: CSCI379 Final Project - Proposal Presentation .

This report details the User Authentication web application project that uses both Face and Voice biometrics to identify a user. In this case, the design of the project has been formulated as an attendance system meant to be used in school classroom settings to make sure a student is who they say they are when presenting themselves for attendance. This project builds off of a previous class homework, the facial recognition project, and that has enabled the student to not have to rebuild a new user interface for this project and reuse a lot of code.

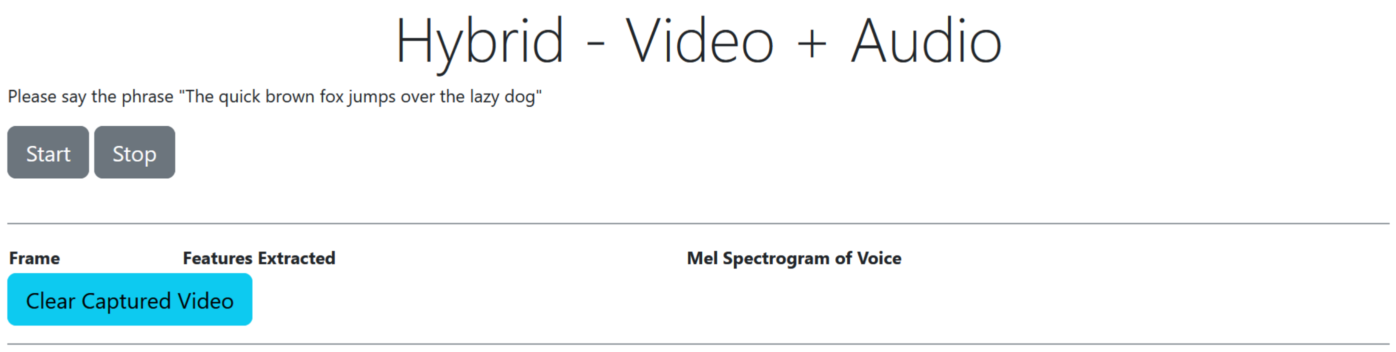
---

2. Methodology

2.1 Enrolling New Users

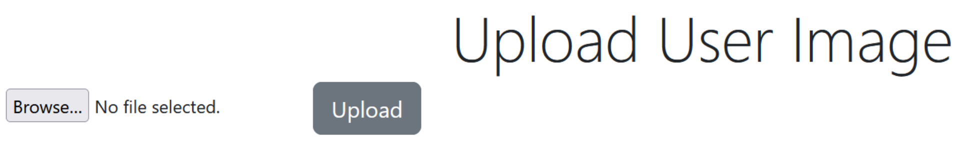
The user interface remains quite easy to navigate and is self explanatory. The new features that make this section different from homework 2 are the following:

1. Hybrid Enrollment (video + audio)
2. Enroll via uploading an image of a face


Hybrid - Video + Audio

Please say the phrase "The quick brown fox jumps over the lazy dog"

Figure 1: Hybrid enrollment section


Upload User Image

Browse... No file selected.

Upload

Figure 2: Upload user image instead of using real-time camera

Hybrid enrollment asks the user to record themself saying a phrase displayed on the screen. The video and audio will then be sent to the backend to be processed by the attendance system. At the moment of writing, the system will take ten random frames from the video to be used for face processing. The entirety of the audio recorded will be used for feature extraction. The audio is first preprocessed with noise reduction to try to remove background sound then normalization to a desired "loudness" of -20 dBFS, using the noisereduce library and of course numpy [1][2]. Then, the audio is transformed into a mel spectrogram image via the librosa and matplotlib libraries [3][4]. The mel spectrogram is then passed into VGG19 to produce embeddings of the image, and it is with these mel spectrogram image embeddings that I use to compute similarity scores between different audio samples of the user for identification.

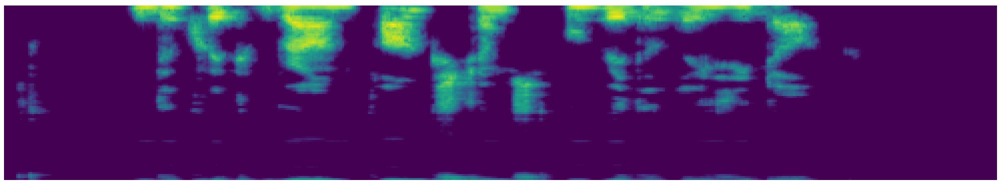
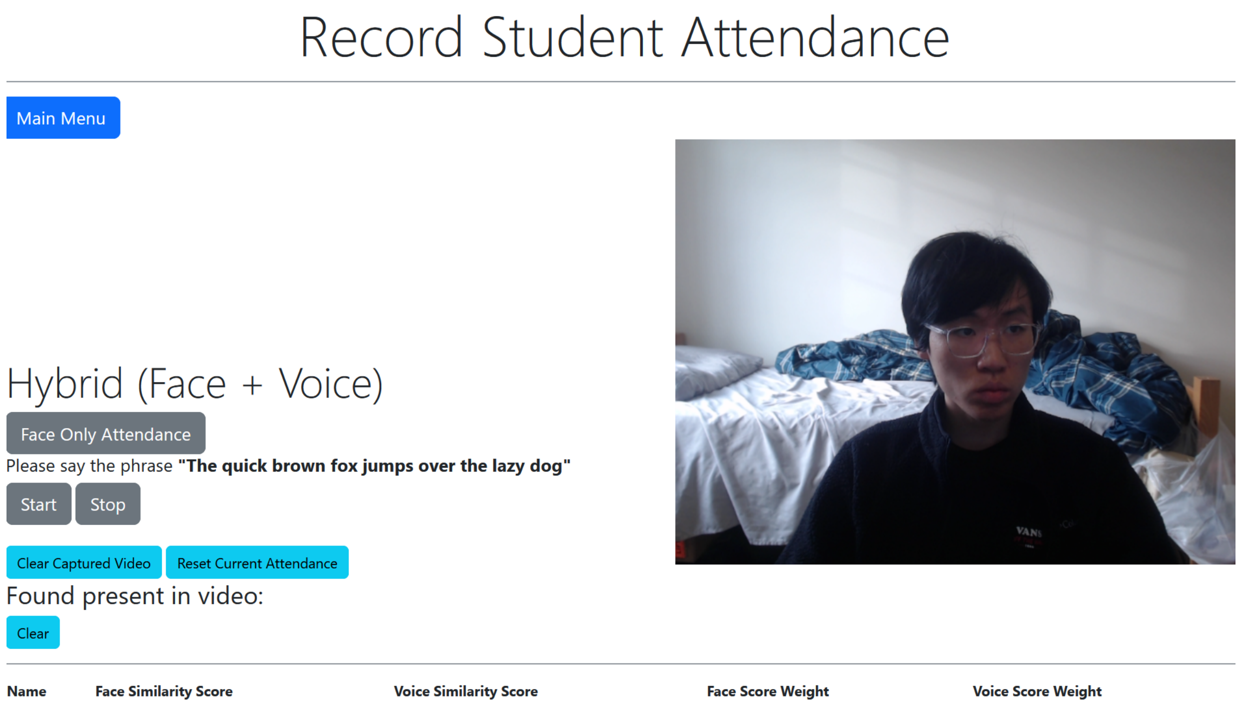


Figure 3: Example mel spectrogram of an audio sample generated by system

I first learned about the idea of recognizing images from audio samples from one of Jeremy Howard's Practical Deep Learning for Coders [5] in which he had mentioned some student had used mel spectrograms of audio samples to train a CNN.

2.2 Taking Attendance

I added a hybrid attendance taking method in which the user essentially does the same exact thing that they did during enrollment, recording themself saying the same phrase.


Record Student Attendance

Main Menu

Hybrid (Face + Voice)

Face Only Attendance

Please say the phrase **"The quick brown fox jumps over the lazy dog"**

Start Stop

Clear Captured Video Reset Current Attendance

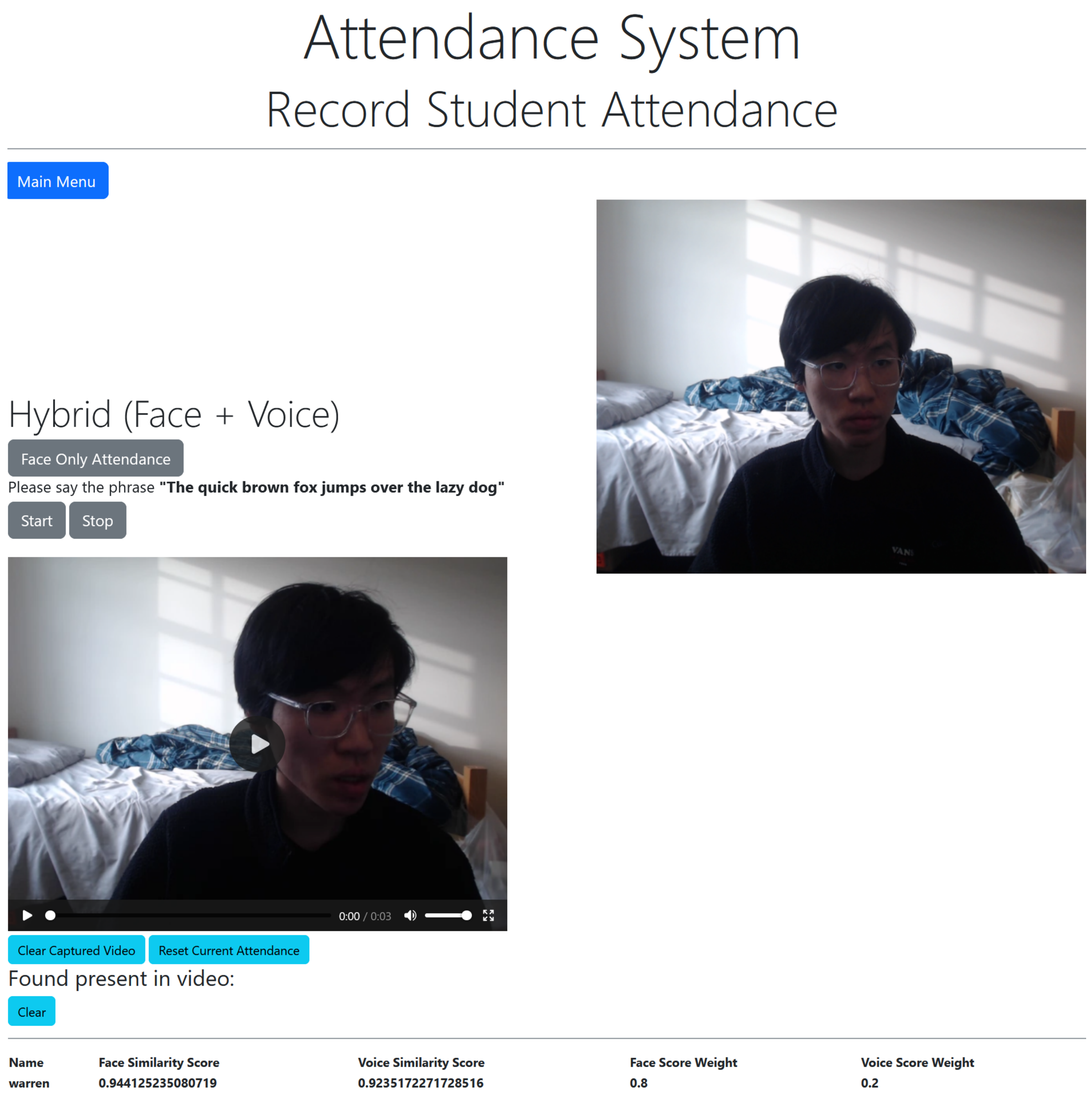
Found present in video:

Clear

Name	Face Similarity Score	Voice Similarity Score	Face Score Weight	Voice Score Weight

Figure 4: Hybrid attendance taking

In the hybrid attendance taking, the user essentially does the same thing they did during enrollment. The decision to allow a student to be marked as present is now done by first computing similarity scores for the face images and the mel spectrogram images for the current time sample against all the samples in the database. Then out of those scores, the system grabs the top face and voice scorers. Then the system checks to see if the top scorers are at least of the same person, if they are not the system immediately rejects the identification. If the names are at least of the same person, then the system will then compute a weighted combination of the face and voice scores with predefined/hardcoded weights (e.g. 0.8 for face and 0.2 for voice). This new averaged score will then need to pass a threshold (e.g. 0.7), and then the system will mark the person as present/identified.



Attendance System

Record Student Attendance

Hybrid (Face + Voice)

Please say the phrase **"The quick brown fox jumps over the lazy dog"**

Found present in video:

Name	**Face Similarity Score**	**Voice Similarity Score**	**Face Score Weight**	**Voice Score Weight**
warren	**0.944125235080719**	**0.9235172271728516**	**0.8**	**0.2**

Figure 5: Marking a person present

2.3 Object-Oriented Design

I made good use of structuring our code following OOP design. For the voice processing code, I organized that into its own class and file, and allowed the AttendanceSystem class to manage an instantiation of the SpeakerRecognition class. While technically the SpeakerRecognition class relies on the usage of VGG19 from the FaceRecognition class, it makes sense not to copy code over from there into SpeakerRecognition because object instantiations of these classes both exist in the AttendanceSystem and User class and therefore can be used in coordination with one another instead of having to duplicate functionality from one class directly into another.

2.4 Code Explanation and Commenting

I made thorough use of in-line comments and docstrings to explain what functions and particularly indecipherable lines of code do. I made use of type annotations for functions in order to keep track of what data types inputs/outputs were supposed to be, even if these are not actually enforced by the default CPython interpreter.

---

3. Experimentation

Through trial and error, I had found that the mel spectrogram images have a lot of pixels that are the same and therefore will have generally high match scores when computing cosine similarity on the embeddings of the images passed through VGG19.

At first, I decided that I would opt to do a slightly complicated score calculation using the top voice and face score, and it did not work in the in-class demonstration and with some testing with my friends. I had to experiment with the voice score threshold manually, and at the time of the in-class demo I had found that a voice score threshold of 0.95 worked best, so that is what I had used. The face score threshold did not matter that much, as anything about an 0.80 threshold would be reasonable and VGG19 was very good. One must consider that the same backgrounds will very likely be used over and over in a classroom or work office environment, and those pixels do matter for different image similarities of the same person. The intraclass variability of different images of the same person on the different days is supposed to be nowhere near as great as the interclass variability of the different images of different students/workers in the same class/company environment.

After the in-class demonstration, I took Professor Kumar’s advice to just use a weighted score average between the face and voice scores, which in hindsight should’ve been the very first thing I should have tried. After implementing that with hard-coded weights that I do not allow the user to adjust, it does appear to work better with different people other than myself.

However, another thing to note is that the quality of the images and voice samples differ based on the lighting conditions and the camera and microphone hardware. Also, it is important to note that the mel spectrograms of the voice samples will look different when comparing the same voice being recorded on two different microphones due to the differences in the hardware of the microphones. Some microphones are of course better than others, and some cameras are of course better than others.

---

4. Conclusion and Further Work

I would consider the project a success. All in all, I have achieved my goal of being able to identify people based on their voice. This project taught me the intricacies of working with audio data in python and the power of image processing horsepower of generic CNNs that have trained on a lot of generic image data.

There is much further work that can be done on this project. For instance, if we had more time, we could have looked into training or finding the model weights to a CNN specifically trained on mel spectrogram images as opposed to using a general VGG19 CNN. I could also look deeper into the standard audio processing techniques to clean up and highlight the user's voice in the recorded audio sample. I asked the user to speak a pangram which contains all the letters of the English alphabet, but in retrospect that is irrelevant to the idea I had in mind – I should have used a phonetic pangram. This may make the mel spectrogram more detailed for each person's voice. The current project also retains raw user data (pictures and audio recordings) as a way to allow the programmer to re-extract features if needed and "reset" the database's extracted features if they would want to make changes to the feature extraction pipeline. I recognize that retaining raw user data comes with the risk of leaking user data. However, we would like to note that this project is mostly made as a proof of concept and not meant to be anywhere near deployed in production due to the many security vulnerabilities that very likely exist at many points in the front-end to back-end connection. With more time and man hours, we can also look into hardening the security of this project. And of course, this includes moving the administration page behind a trusted user login session.

---

5. References

[1] https://pypi.org/project/noisereduce/

[2] https://numpy.org/

[3] https://librosa.org/doc/latest/index.html

[4] https://matplotlib.org/

[5] https://course.fast.ai/
