Distributed Web-Based Facial Recognition for Student Attendance

Kincannon Wilson
University of Wisconsin-Madison
1210 W Dayton St, Madison, WI 53706
kgwilson2@wisc.edu

Abstract

This project involves the creation of a web-based application capable of recognizing the faces of students as they walk in front of a camera in order to record their attendance. Images from a video stream on an instructor's device are sent over HTTP to a server that 1. performs face detection using an MTCNN, 2. detects spoofs with a liveness detection network based on ResNet-18, and 3. if a live face has been detected, generates an embedding of the image using Inception ResNet V1. The embedding is compared to known embeddings using cosine similarity to identify the student in the photo.

1. Summary

I propose an attendance system built upon facial recognition. Such a system would enable students to upload a picture of their face through a website for use by all of their courses. After uploading the image, the server crops the image according to a bounding box generated by Multi-Task Cascaded Convolutional Neural Networks (MTCNN) before passing the cropped image through Inception ResNet V1 (Inception-v1) to obtain an embedding.

At the beginning of each class or examination period, the student will present their face to an instructor's device that is logged into a password protected portion of the site that obtains a video stream from the device's camera. This site will send a frame from the video stream over HTTP to the server every 0.5 seconds. The server receives the image and passes it through an MTCNN-based face detection network. If a face is detected in the image, the face then passes through a liveness detection network. If the face passes the liveness check, the server converts the face to an embedding using the same Inception-v1 model employed in the aforementioned upload process. The embedding is then compared to other embeddings (only for people in the same class) using cosine similarity in order to determine the closest match for the person in the input image. The found person's attendance is then recorded in a database that is easily queried by the course's instructor for input into their gradebook.

2. Literature Review

Kortli et al. provides a comprehensive survey of facial recognition systems.[1] This project follows their basic steps: face detection, feature extraction, and facial recognition. The survey provided an invaluable introduction to the field of face recognition.

In their paper, Kar et al. describe an automated attendance system using facial recognition.[2] The researchers provide implementation details that I found useful while designing my own system. However, the research uses PCA to generate eigenvalues for input images that are then compared to identify the people in the images. Comparative studies show that CNNs yield higher accuracy than PCA.[3]

Finally, Arsenovic et al. propose FaceTime, a deep learning-based face recognition attendance system.[4] The model uses a CNN cascade for face detection and a CNN for generating face embedding. The researchers report a high level of accuracy (95.02%) on a small dataset of images of employees in a realistic environment. This project largely follows the approach of Arsenovic et al. with a few key differences. Namely, my system is distributed (the device capturing the images is different from the device running the various models), implements liveness detection, eliminates the usage of face landmarks, and identifies faces using a distance comparison between embeddings (as in Kar et al.) rather than using an SVM classifier.

3. Method

3.1 - Upload Process

On the frontend of the application, students can upload an image of themselves. Whenever the server receives the image from the frontend, it passes the image through the

Processing	MTCNN	Liveness Detection	Inception-v1	Embedding Search
0.0075	0.05 - 0.10	0.19 - 0.25	0.03	0.001

Figure 1. A feasibility study assessing the time required by each stage of the recognition pipeline on the server side. Measurements shown in seconds.

MTCNN to ensure a face is present in the image. This deep learning-based approach to face detection was chosen because of its proven accuracy and speed.[5] The MTCNN weights come from the original paper by Zhang et al., which resulted from training on the FDDB and WIDER FACES datasets. Then, the image is cropped according to the bounding box found by the MTCNN and resized to the resolution used during training (160x160 pixels). The resized image is passed through the Inception-v1 network to generate an embedding, which gets converted to a numpy array and saved in the .npy file format in order to take advantage of the format's faster load times and reduced storage requirements. I chose Inception-v1 as the network due to its high degree of accuracy (0.99+ accuracy on the Labeled Faces in the Wild Dataset) and very short inference time (Figure 1).[6] The uploaded image is immediately discarded, which mitigates the security and privacy risks that arise from storing this identifying data.

3.2 - Image Acquisition

Instructors have access to many features on the frontend, such as creating/deleting/editing classes, allowing students to join created classes using a unique code, and viewing/downloading their classes' attendance histories. Most importantly, instructors can access the detection page. Here, the application accesses the device's camera and stores a reference to the video stream object. Next, every 0.5 seconds, the video stream object is used to draw an image to an HTML canvas element, effectively taking a screenshot of the video stream. I found that a capture frequency of 2 FPS did well to minimize both network traffic and users' time spent in front of the camera. The image from the canvas gets appended to a new FormData instance. Finally, the frontend sends an HTTP POST request to the server machine with the form data as the body of the request. I used the FormData interface because without the additional formatting provided by the interface, decoding the image server-side was nearly impossible.

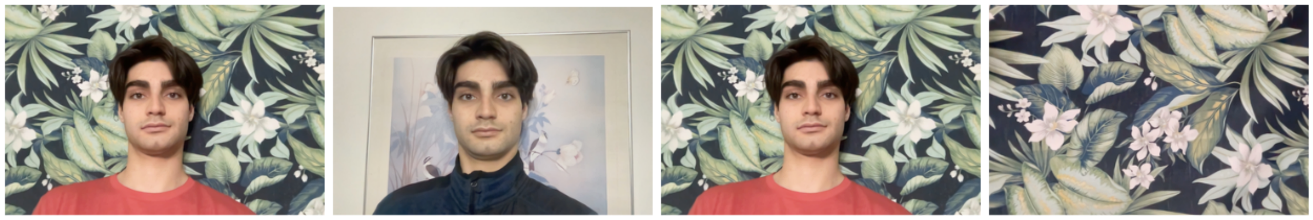
3.2 - Recognition Pipeline

During detection, the server converts the received byte string from the HTTP request body into bytes, decodes those bytes, opens the resulting image data using PIL, and converts the image into RGB format. This image is passed through the MTCNN to find the central face in the image and to generate a crop according to the generated bounding box around that face. Next, the cropped image passes through the liveness detection network. The liveness network was trained on the 625,000-image CelebA-Spoof dataset from Zhang et al.[7] The creators of the CelebA-Spoof dataset also created a network they call Auxiliary Information Embedding Network (AENet). AENet uses a ResNet-18 backbone with three branches for 1) classification as real/fake, 2) semantic analysis for matching images to concepts in annotations such as 'phone' or 'back illumination,' and 3) geometric analysis. The liveness network outputs a score for the 'fakeness' of a face. The value of 0.99 was discovered through experimentation to serve as a reliable threshold. Thus, those faces with a 'fakeness' score over 0.99 are considered fake and are discarded. This functionality helps prevent students from spoofing the application. Without the liveness check, students in the class could hold up a picture of another student (either printed or on a device) to fool the network and count the pictured student as present. Then, the live, cropped image passes through Inception-v1 and generates an embedding.

3	50	200	1000
0.001	0.006	0.025	0.3 - 1

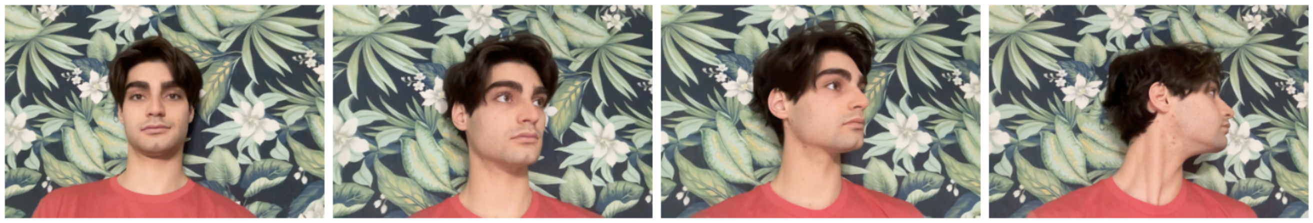
Figure 2. A comparison of runtimes for the embedding search for classes of 3, 5, 200, and 1000 students. Measurements shown in seconds.

Finally, the server loops through the saved face embeddings that belong to students in the same class, computing the cosine similarity between each saved embedding and the generated embedding. Cosine similarity was chosen over Euclidean distance following Qian et al. who found that cosine angle distance works no worse than Euclidean distance while providing other advantages when applied to content based image retrieval.[8] If the maximum computed similarity is less than 0.9, then the server considers the face in the generated embedding to not belong to any student in the



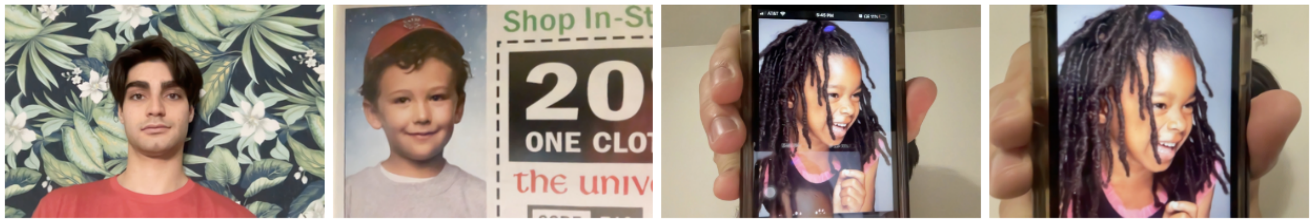
“Not in class” “Upload successful” “Cannon is present” “No face found”

Figure 3. (Left) Live face is detected, but the person is not registered in the class. (Middle Left) This image is uploaded as “Cannon.jpeg” and Cannon is registered for the class. (Middle Right) Cannon is successfully recognized. (Right) Cannon leaves the frame and the MTCNN fails to detect a face.



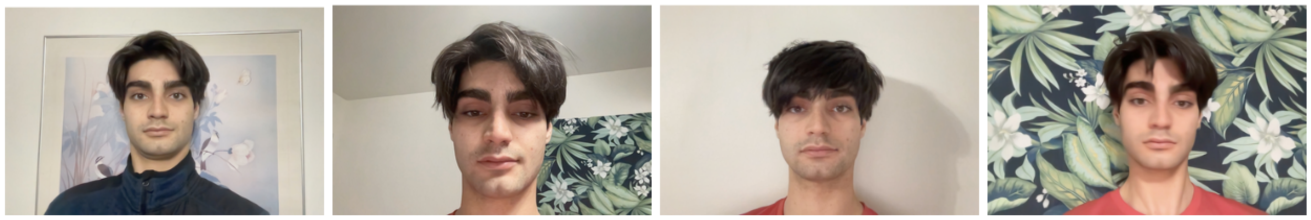
“Cannon is present” “Cannon is present” “Not in class” “Not in class”

Figure 4. When facing the camera at an angle between 0° and 20°, the student is recognized. Beyond 20°, the student is not recognized.



“Cannon is present” “Fake face detected” “Fake face detected” “Girl is present”

Figure 5. (Left) Live person is recognized. (Middle Left) Printed photo of a boy is detected as not live. (Middle Right) Photo of a girl on a phone with reflections is detected as not live. (Right) Photo of a girl on a phone without reflections is recognized.



“Upload successful” “Not in class” “Not in class” “Fake face detected”

Figure 6. (Left) Uploaded image for reference. (Middle Left) Input image with different lighting is not recognized. (Middle Right) Input image with different hairstyle is not recognized. (Right) Input image with slight blur from side-to-side movement is detected as not live.

given class. Otherwise, the name of the found student (names are taken from the known embedding's filename) is used to update the attendance collection in the database.

4. Experiments and Results

4.1 - Feasibility Study of Pipeline Runtimes

The first experiment involved timing the various stages of the recognition pipeline running on the server (Figure 1). In the worst case, the entire pipeline takes 0.3885 seconds. The liveness detection network takes the majority of that time, which suggests that creating an implementation of AENet with a faster inference time (or selecting another liveness detection network altogether) would make the largest improvement to the pipeline's runtime.

4.2 - Effect of Class Size on Embedding Search

Given that the first experiment ran using a class size of just 3 students, the second experiment assesses the impact of class size on the time required by the embedding search (Figure 2). While the runtime for the embedding search remains very low for classes of 3, 5, and 200 students, the embedding search takes between 0.3 and 1 second to run for a class size of 1000 students (0.65 seconds on average). In practice, this causes a significant delay between a user's experience and the incoming HTTP response from the server.

4.3 - Input Tests in Real-Time Environment

The third and final experiment assesses the robustness of the pipeline when applied in a realistic environment and tests the effects of lighting and pose. The pipeline correctly identifies students when only their background and outfit changes, and the pipeline correctly identifies when no face is present in the video stream (Figure 3). The pipeline works well when the uploaded image and the input image are both facing towards the camera at a 0° angle. However, past 20 degrees, the student is no longer recognized (Figure 4). This suggests that the proposed pipeline does not generalize well across shifts in pose.

Next, I tested the liveness detection network. While the network consistently detected fake faces that were printed out, the network struggled to detect fake faces on phones without the presence of artifacts like reflections (Figure 5). Although some photos of devices were present in the CelebA-Spoof dataset, the majority of spoofs were printed pictures of faces or of people wearing printed masks. This fact might explain the network's reliability in detecting printed faces. The photos of devices in the dataset were of early generation iPads and iPhones and appear to contain significant artifacts such as discoloration and reflections. Later generations of devices (such as my iPhone 8) have clearer displays, which makes distinguishing photos of faces on newer devices more difficult for the network.

The final set of tests in the third experiment explore the failure cases of the pipeline (Figure 6). The pipeline proved extremely sensitive to changes in lighting and hairstyle, and changes in these factors often caused the pipeline to fail to recognize a face. Finally, the pipeline incorrectly classified faces as fake whenever the face was slightly blurred due to movement.

5. Conclusion

This paper details the implementation of a distributed web-based facial recognition system. The facial recognition pipeline on the server involves face detection using MTCNN, liveness detection using AENet, embedding generation using Inception-v1, and identify matching using cosine similarity.

The first key finding is that the liveness detection network based on AENet by Zhang et al. has an inference time that is too slow to include the network in the proposed pipeline without significant optimization. In addition, the network appears to have trouble detecting pictures of photos on new phones, which significantly reduces its usefulness.

The second key finding is that the proposed strategy for identifying a student (performing a linear search through known embeddings for the embedding with the highest cosine similarity to the input embedding) does not perform well for larger class sizes.

The last key finding is that the one-shot approach of comparing the embedding of one uploaded image with the embedding of an input image makes the proposed pipeline very sensitive to changes in lighting, hairstyle, etc. As such, any similar attendance system would likely require photos of each student taken from multiple angles in multiple lighting conditions. Future attendance systems based on facial recognition could improve the accuracy of their system by implementing a way to quickly and conveniently gather many images (perhaps out of a video) of each subject from varying angles and in different lighting conditions.

References

[1] Kortli, Yassin, et al. "Face Recognition Systems: A Survey." MDPI, Multidisciplinary Digital Publishing Institute, 7 Jan. 2020, https://www.mdpi.com/1424-8220/20/2/342.

[2] Kar, Nirmalya, et al. "Study of Implementing Automated Attendance System Using Face Recognition Technique." IJCCE, Jul. 2012. http://ijcce.org/papers/28-N010.pdf.

[3] Paul, Sanmoy, and Sameer Kumar Acharya. "A Comparative Study on Facial Recognition Algorithms." First Pan IIT International Management Conference, 21 Dec. 2020, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=

3753064.

[4] Arsenovic, Marko, et al. “FaceTime — Deep learning based face recognition attendance system.” IEEE, 2017, https://ieeexplore.ieee.org/document/8080587.

[5] Zhang, Kaipeng, et al. “Joint Face Detection and Alignment Using Multi-Task Cascaded Convolutional Networks.” IEEE, 11 Apr. 2016, https://arxiv.org/abs/1604.02878.

[6] Sandberg, David. “Davidsandberg/Facenet: Face Recognition Using Tensorflow.” GitHub, https://github.com/davidsandberg/facenet.

[7] Zhang, Yuanhan, et al. “Celeba-Spoof: Large-Scale Face Anti-Spoofing Dataset with Rich Annotations.” ArXiv.org, 1 Aug. 2020, https://arxiv.org/abs/2007.12342.

[8] Qian, Gang, et al. “Similarity between Euclidean and Cosine Angle Distance for Nearest Neighbor Queries.” ACM Conferences, 1 Mar. 2004, https://dl.acm.org/doi/10.1145/967900.968151.