Natural Eye Contact Using Generative Adversarial Networks

Image-to-Image Translation to Alter Camera Position

David Chen
Computer Science
University of Virginia
Charlottesville, VA USA
dzc5ta@virginia.edu

ABSTRACT

Over the course of recent months, COVID-19 has caused a great decrease in the amount of in-person contact. Events are being moved virtual and the use of video conferencing and video calls have increased greatly. Online video conferencing applications including Zoom, Microsoft Teams, and Google Hangouts, have seen a great growth in usage, since the beginning of the COVID-19 pandemic. Zoom in particular grew from averaging 56,000 daily downloads in January of 2020, to averaging 2.13 million daily downloads in March of 2020.[1] These applications, however, are limiting in several ways. One key limiting factor is that eye contact, a key aspect of nonverbal communication, is not accessible to users. This paper will go cover an approach that will use machine learning to help combat this limiting factor. It will involve 4 major steps: data collection, data processing, training, and testing. First, data will be collected in the form of video of the faces of people in conversation that is recorded by multiple devices simultaneously at different angles. Then, the videos will be synced and the frames will be paired to use for training and testing. Then, using Python and TensorFlow, the data will be fed to a convolutional neural network model to train the parameters. Once the model is trained, a second portion of the data will be used to test and evaluate the model.

1 Introduction

The need for eye contact in video conferencing is paramount. Eye contact is a crucial component of nonverbal communication. Since eye contact avoidance is a sign of lying it can lead to a sense of distrust between callers.[2] Up until recently, there have not been any attempts on creating a solution for this problem. One can only imagine how many video calls have been impacted by the lack of eye contact. The problem with current technology is that the camera is not located at the caller's eye that is on the screen. Instead, it is either at the top of the screen or the bottom. Because there is a distance between the camera and the eyes on the screen, the camera, which is the other person's eye, won't provide an accurate view of what the person sees. Eye contact without the assistance of eye contact correction technology, is simply impossible. Let us consider a simple scenario where two people (Person A and Person B) are video calling with both webcams above the screen. If Person A tries to look at Person B's eyes, it'll appear to Person B as if Person A is looking downwards, since Person B's eyes are on the screen below the webcam. In order for Person A to appear like she is looking at Person B's eyes, she has to look directly at the camera. However, by looking directly at the camera, Person A can no longer see if Person B is looking at her eyes.

The ideal goal for natural eye contact should be to perfectly mimic in-person conversation and interaction. One potential solution could be to adapt the physical hardware and insert an invisible centralized camera behind the screen. There have already been attempts to do this on smartphones because of the desire to have notch-less phones.[3] This would be the simplest and most effective solution because it would not require any additional software to alter the video, since the camera can be located exactly at the person's eye. Another potential solution could involve virtual reality, but that technology if even available, would be too costly and take too much computational power to be applicable to the current world. Because most technology users today are limited to a simple laptop/phone screen and non centralized camera, the approach I will propose is to use software to effectively change the location of the camera from its original position to the center of the screen. I will use image to image translation to convert images taken from the top of the screen to appear as if they were taken from the center of the screen.

2 Related Work

Achieving eye contact in video conferencing is not a brand new concept. While there have been several attempts to achieve eye contact in video calls, they all have shortcomings. Apple has recently released a feature that tries to correct eye contact during FaceTime calls. However, one user tweeted a video showing that if he placed a straight object near his eyes, it would cause FaceTime's new feature to bend it, making it look wavy.[4] This is because Apple uses an ARKit depth map to adjust the eyes. This process can be invasive and lead to unwanted altering of other objects.

Intel also published a paper describing their process of correcting eye contact.[5] Their approach was also to only edit the area around the eye. This can be very cumbersome because it required a mechanism to decide when to toggle on or off the eye contact correction software. It had to predict whether or not the user wanted to initiate eye contact adding unnecessary computation to the picture.

Furthermore, DIY Perks, a YouTube Channel released a video that showed viewers how to build an apparatus that involved a second webcam and a two-way mirror.[6] The concept behind this device was to use the mirror to reflect light from the user to a different webcam but also allow light to pass through from the screen to the user's eyes. This physical contraption would be much too complicated for an average user to build, too bulky to carry around, and aesthetically unappealing.

The former methods are not ideal because of the artificial nature of the selective altering of the eyes. The latter method is simply unattractive and bulky. This is while a new approach is needed. Changing the location of the camera seems to be the most natural and effective form of eye contact correction. However, while we are still waiting for the physical hardware to change, we can search for a software approach to simulate the physical change.

3 Methodology

I carried out a relatively small machine learning project in order to see if it would be practical to develop this technology. It will involve using a pair of webcams, QuickTime Player, Google Colab, and Python libraries including NumPy, Pandas, Keras. A Conditional Generative Adversarial Network is a suitable solution because it works in a way that allows two models to train each other to become better at what they do. The generator model tries to replicate an output given the source image while the discriminator tries to determine if the image is a valid transformation of the source image. In this case, the generator model is given an image taken from the upper webcam and tries to replicate an image taken from the lower webcam. It succeeds if it fools the discriminator model, which has the job of determining if the image if received was taken from the lower webcam or if it was created by the generator model. As they are trained, the generator model gets better at creating outputs that are similar to the desired image, while the discriminator model gets better at determining which images are real or fake. When the Conditional GAN is trained, the generative model can be used on new data to perform image-to-image translation on previously unseen data points.

3.1 Data Collection

Data was collected from two Logitech c920s Pro HD webcams, separated vertically by 7.5 cm. These webcams were connected to a 2020 M1 MacBook Pro running 2 instances of QuickTime Player. They were used to record 10 minutes of different facial expressions and angles. These video files were exported at a resolution of 640 by 360 pixels.

3.2 Data Calibration

The two videos were then converted into NumPy arrays using OpenCV. Each image was represented by a Three-dimensional array of shape (640, 360, 3). The images were 640 pixels wide, 360 pixels high, and had three layers - one for each RGB value. The images were then cropped to a 256 by 256 square with the faces at the center. The final data set consisted of 1107 pairs of 256 by 256 images, one image from the upper webcam and one image from the lower webcam.

3.3 Generative Adversarial Network

The Generative Adversarial Network (GAN) model consists of a generator and a discriminator. The generator is responsible for creating images that mimic the output to fool the discriminator that tries to predict whether or not the image is real. The generator makes use of encoder blocks and decoder blocks. An encoder block is a convolution layer of n filters, 4x4 kernels, 2x2 strides, and same padding, followed by a potential batch normalization and a Leaky Rectified Linear Unit (ReLU) activation function. A decoder block, on the other hand, consists of a transposed convolution layer of n filters of 4x4 kernels, 2x2 strides, and same padding, followed by a batch normalization layer, a potential dropout (rate of .5) regularization layer, and a ReLU activation function. The generator begins with 7 encoder blocks with 64, 128, 256, 512, 512, 512, and 512 filters respectively, with only the first encoder block not using batch normalization. Next, there is a convolution layer with 512 filters, 4x4 kernels, 2x2 strides, and same padding, and a ReLU activation function. Then there are 7 decoder blocks with 512, 512, 512, 512, 256, 128, and 64 filters respectively, with only the first 3 utilizing dropout regularization. This connects to the final layers consisting of a transposed convolution layer of 3 filters of 4x4 kernels, 2x2 strides, and same padding and a tanh activation function.

The discriminator model is much simpler and consists of 6 convolution layers of 64, 128, 256, 512, 512, and 1 filter(s) respectively. In between each convolution layer is a batch normalization layer and a Leaky ReLU activation function. After the last convolution layer, there is a sigmoid activation function, which outputs the prediction of whether or not the image was real.

This GAN model was run for 100 epochs and saved every 10 epochs. At each 10 epoch interval, samples of results were graphed to show the progress.

3.4 Index of Performance

Once the model is trained, it will be evaluated to see its usefulness. In order to measure success, we will look at the generated images to see how they compare to the intended output. The generated images should not only look realistic but also appear as if they were taken from a camera that was slightly below (It should appear as if the person is looking a bit higher compared to the input image).

4 Results

Throughout the training process, models were saved every 10 epochs, and samples were generated. Using Google Colab, I was able to train one epoch in 4.5 minutes. It took roughly 7.5 hours to train the 100 epochs. For each model, the performance was visualized by 3 images, input, generated output, and true output. I have included some of the intermediate performances in this report. Initially, after 45 minutes of training, the model could not even generate a realistic image. However, after 7.5 hours, the model is generating almost perfect images.

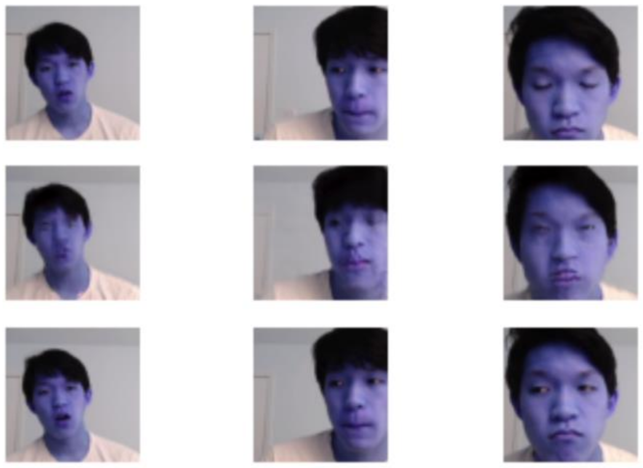

Figure 1: After 10 epochs, in order from top to bottom: images from the upper camera, generated images, and images from the lower camera.

After 10 epochs, the generator is still struggling to produce realistic images. In Figure 1, we see all of the generated images (located in the middle row) are deformed and blurry.

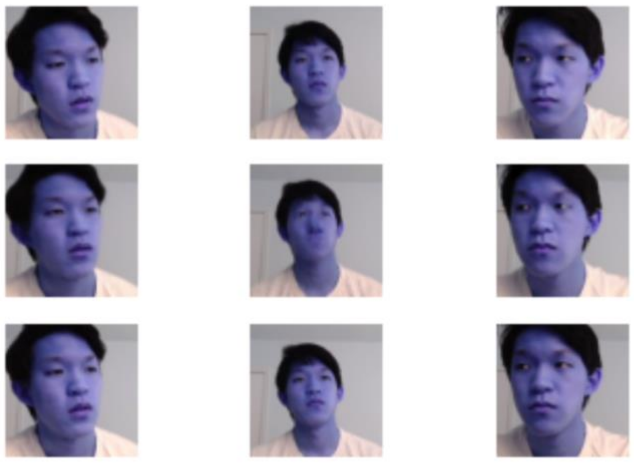

Figure 2: After 40 epochs, in order from top to bottom: images from the upper camera, generated images, and images from the lower camera.

After 40 epochs, the generator already began generating realistic outputs. It can be seen that the middle column of Figure 2 is an angle not yet learned by the GAN network, since the face is disfigured. However, the left and right columns are both examples of successful generations of the new camera angle, since the generated image shows the eyes looking a bit higher than the image taken from the upper camera.

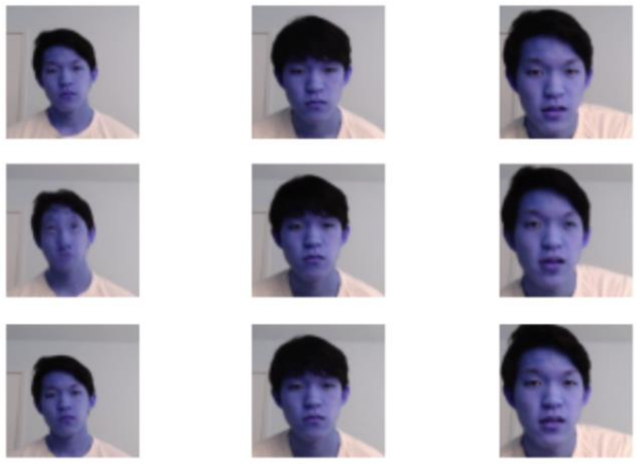

Figure 3: After 80 epochs, in order from top to bottom: image from the upper camera, generated image, image from the lower camera.

We can see in Figure 3 that After 80 epochs, the generator continues to generate realistic outputs but still struggles occasionally. For example, the left column is an undesired result because the face is blurry and disfigured. However, the center and right columns are both examples of successful generations of a new camera angle, since the generated image shows the eyes looking a bit higher than the image taken from the upper camera.

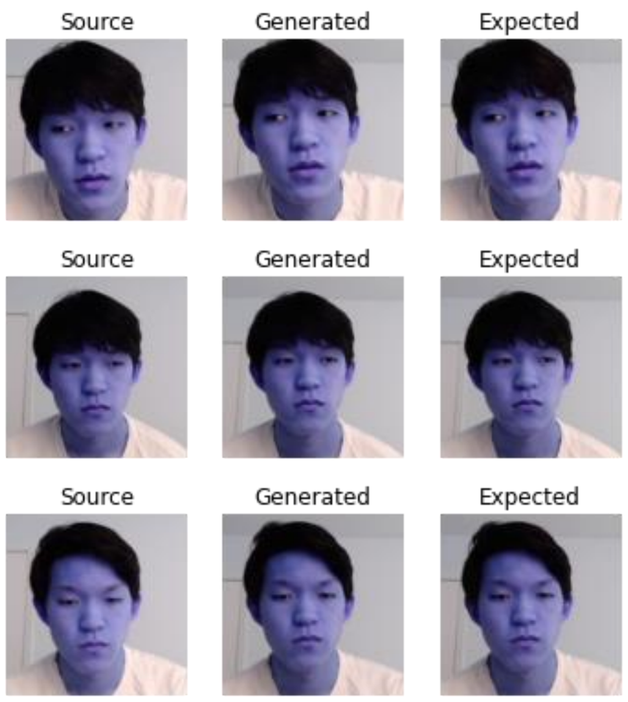

Figure 4: Source, generated, and expected images after 100 epochs of training, without eye contact.

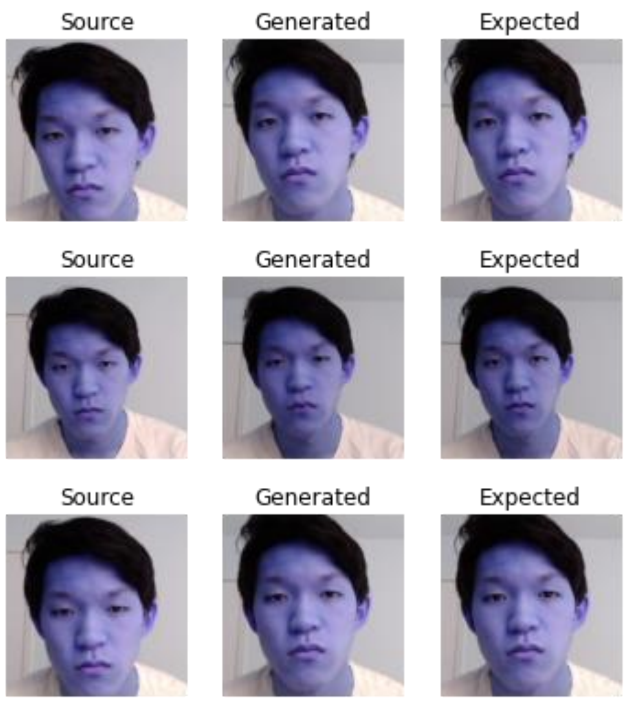

Figure 5: Source, generated, and expected images after 100 epochs of training with eye contact.

We can see in Figures 4 and 5 that after 100 epochs, the results are extremely favorable. In all of the samples, eye contact is only generated if it is expected. In Figure 4, there was no eye contact to begin with so changing the camera angle didn't create any. In Figure 5, the expected images show eye contact and the generator model used the source image to create the eye contact. All the images appear to be natural, and the generated images seem to be the same as the source image but taken from a lower angle.

5 Conclusions

From our results, we can see that a conditional GAN approach to natural eye contact correction is indeed a possible solution to create natural eye contact in video calls. We can see that there were minor hiccups in a few images near the beginning of the training. However, after 100 epochs, the generated images seemed to be much more consistent and natural. With greater optimization and training, this technology can have great potential as a solution to the unnatural nature of video conferencing.

6 Future work

Based on the promising results, this project can be expanded on and further research can be conducted. There are a few next steps that are recommended. One main area to focus on would be data collection. A larger-scale project should include larger images of different people, with different backgrounds, and different lighting. Such a project would require significantly more computational resources, but it can be done. Once such a model is researched, it can be implemented into video calls by feeding each image to the generator at live speed.

REFERENCES

[1] Iqbal, M. (2020, July 20). Zoom revenue and usage statistics (2020). Retrieved September 24, 2020, from https://www.businessofapps.com/data/zoom-statistics/

[2] Murphy, K. (2020, April 29). Why Zoom is terrible. The New York Times. Retrieved from http://www.nytimes.com

[3] Byford, S. (2020, December 21). The world's first under-display selfie camera isn't very good. Retrieved April 25, 2021, from https://www.theverge.com/2020/12/21/22191459/zte-axon-20-5g-under-display-camera-hands-on

[4] Schukin, D. [schukin]. (2019, July 3). How iOS 13 FaceTime Attention Correction works: it simply uses ARKit to grab a depth map/position of your face, and adjusts the eyes accordingly. Notice the warping of the line across both the eyes and nose [Tweet]. Retrieved from https://twitter.com/schukin/status/1146359923158089728?s=21

[5] Fadelli, I. (2019, June 26). Intel researchers develop an eye contact correction system for video chats. Retrieved from https://techxplore.com/news/2019-06-intel-eye-contact-video-chats.html

[6] DIY Perks. (2020, May 27). Weird webcam mod that enables eye-contact conversation [Video]. YouTube. https://www.youtube.com/watch?v=2AecAXinars

[7] Phillip Isola and Jun-Yan Zhu and Tinghui Zhou and Alexei A. Efros (2016). Image-to-Image Translation with Conditional Adversarial Networks. CoRR, abs/1611.07004.