Exposing Face Swap Technology

Yuxuan Zhou(yuxuanz6) Keven Feng(klfeng2)
Electrical and Computer Engineering Department
University of Illinois Urbana-Champaign

ABSTRACT

Recently people have implemented the ability to create synthetic videos of a person doing things they didn't actually do. This is done by changing one person's face into a another person's face. The most well-known of this technology is known as Deepfake. We would like to create an application that can achieve similar results to this technology. In addition, we would also like to implement a detector which can be used to find fake videos to try and protect against malicious use of this software. To achieve this, a machine learning algorithm structured around the GAN learning model will be used. The GAN model has two neural networks constantly competing with each other to optimize their results. The neural network called the generator generates fake images. The neural network called the discriminator determines whether or not an image is fake or real. This type of learning model creates both a generator and a detector at the same time. To swap the faces between two images, we employed an autoencoder as the generator, which will use an encoder to break down images into a latent space and two different decoders to generate faces for each person.

KEYWORDS: DeepFake Detection, Media Forensics, Deep Learning, Face Detector

1 INTRODUCTION

The increasing sophistication of mobile camera technology and the ever-growing reach of social media and media sharing portals have made the creation and propagation of digital videos more convenient than ever before. Until recently, the number of fake videos and their degrees of realism have been limited by the lack of sophisticated editing tools, the high demand on domain expertise, and the complex and time-consuming process involved. However, the time of lubrication and manipulation of videos has decreased significantly in recent years, thanks to the accessibility to large-volume training data and high-throughput computing power, but more to the growth of machine learning and computer vision techniques that eliminate the need for manual editing steps. In particular, a new vein of AI-based fake video generation methods known as Deepfake has attracted a lot of attention recently. It takes as input a video of a specific individual ('target'), and outputs another video with the target's faces replaced with those of another individual ('source'). The backbone of Deepfake are deep neural networks trained on face images to automatically map the facial expressions of the source to the target. With proper post-processing, the resulting videos can achieve a high level of realism.

2 BACKGROUND

2.1 CNN and SVM

In addition to the GAN method, there are other face swap technologies like the python dlib based on CNN and SVM. This methodology is older than the GAN method. The code and testing results for this method on some test data can be found in the final_product/dlib_face_swap folder, which has successfully swapped two person's faces in single photo, no-real time video, and even in real-time videos scenes. The following are the steps to face swap using this method.

1. Face detection – use a bounding box to detect the face and use dlib python face detector.
2. Face points detection – use the dlib shape predictor to get landmark of 68 key points of the face, included eyebrow, eyes, nose, mouth etc.
3. Face detection and track – in the video, person's face will change location, this class helps to detect the face location and update each time.
4. Face swap – included 2D and 3D warp face scenes, use the original face mask and apply it into target face, mainly use the affine transform towards the original face key points/mask.

This feature tag-based algorithm has some success with face swap, but does not work well for tilted faces.

2.2 Autoencoder Face Swap Technology

Another methodology for face swapping is to use an autoencoder. The autoencoder network learns breaks down the real faces of person A with encoder to a latent space and reconstructs the face with a decoder. When a fake face B instead of original input A is fed into the encoder, the encoder will break B down into the same latent space and the decoder will make a reconstructed face that looks the same as A.

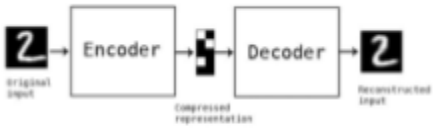


Figure 1. Machine Encoder/Decoder Model [6]

To make sure that the encoder breaks down faces of different people to the same latent space, one encoder is trained on all the different people. By doing this, the encoder learns common features among different people. Then, we have a separate decoder for each face. The decoders are responsible for putting the personal characteristics of each face into the image. Therefore, when you pass a face from person B through the encoder and then use the decoder of person A, you will get a face that matches B's expression, but looks like the face of person A.

From Figure 2, we can see that after going through four convolutional layers, expansion layers, and fully connected layers, we started upscale the entire model. At half our upscale, we cut the encoder and decoder to ensure separation of commonality and personality.

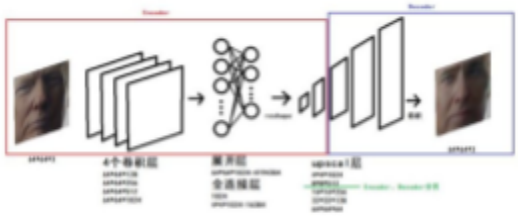
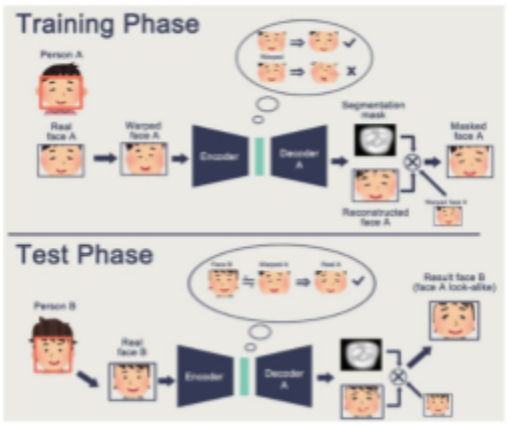


Figure 2. Machine Deep Network Layers [6]

2.3 Autoencoder GAN Face Swap Technology

This technology builds on the autoencoding method mentioned before. The main issue with autoencoders is that they typically only can produce images they have observed before. With a GAN, the model can learn to produce new images not seen before. Also, GANs have usually been observed to produce cleaner outputs. We will describe the GAN autoencoder implemented in [5] that was the foundation for our project.



(来源: shaoanlu/faceswap-GAN)

Figure 3. GAN model train and test phase [5]

As shown in the Figure 3, the network first gets person A's face, transforms it, and then passes it into an encoder and decoder like the one in Figure 1 to generate the reconstructed face A and a mask which represents the facial area to be replaced. The mask is then imposed on the reconstructed face and the resulting face is plastered over the warped A face to get the transformed A face.

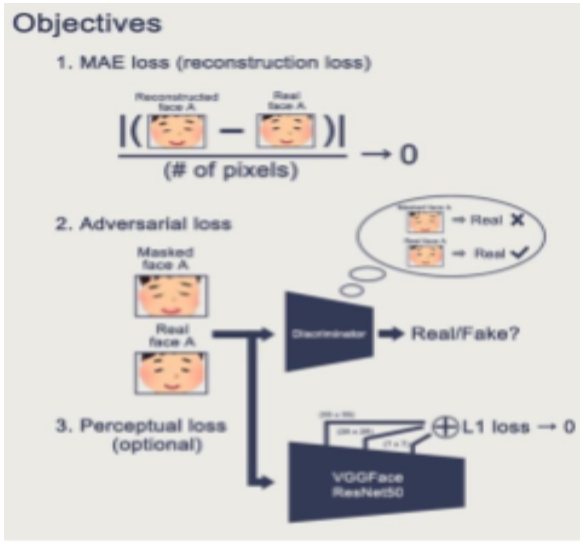


Figure 4. GAN model Objective function [5]

Figure 4 [5] shows the loss functions used for this model. The first loss is the reconstruction loss that is typically found in autoencoders. This loss essentially measures how much data is lost when an image is passed through the encoder and decoder. The second loss is the adversarial loss. This loss measures how well the generator fools the discriminator. In training, both networks train to better understand the images to fool each other. The third loss is the perceptual loss. The perceptual loss

is a loss that takes a well known image classification network like VGG to identify the features of the target and predicted images. The difference between the features then contributes to the loss. This loss is to try and capture how much of a visual difference there is between two images. We see from Figure 4 that L1 loss approaches 0 with VGG Face or ResNet50, which means even the top face detection AI model can't tell the fake face (near 0 accuracy) after GAN model training.

3 METHODOLOGY

In this section we discuss the overall face swap methodology and the GAN autoencoder implementation we used to do perform the face swap.

3.1 Video Processor

In order to swap the face, we first need to upload two videos for person A and person B. The video processor has to handle the transformation of video to images and images to videos process. To do this a face detector will be used to obtain facial images from video frames, which will be used to form our training dataset. Currently we obtain our frame samples using the preprocess_video() function from the preprocess library in [1]. We test sampling at a rate of every frame and every fifth frame. In Figure 5 we illustrate the flow of our design.

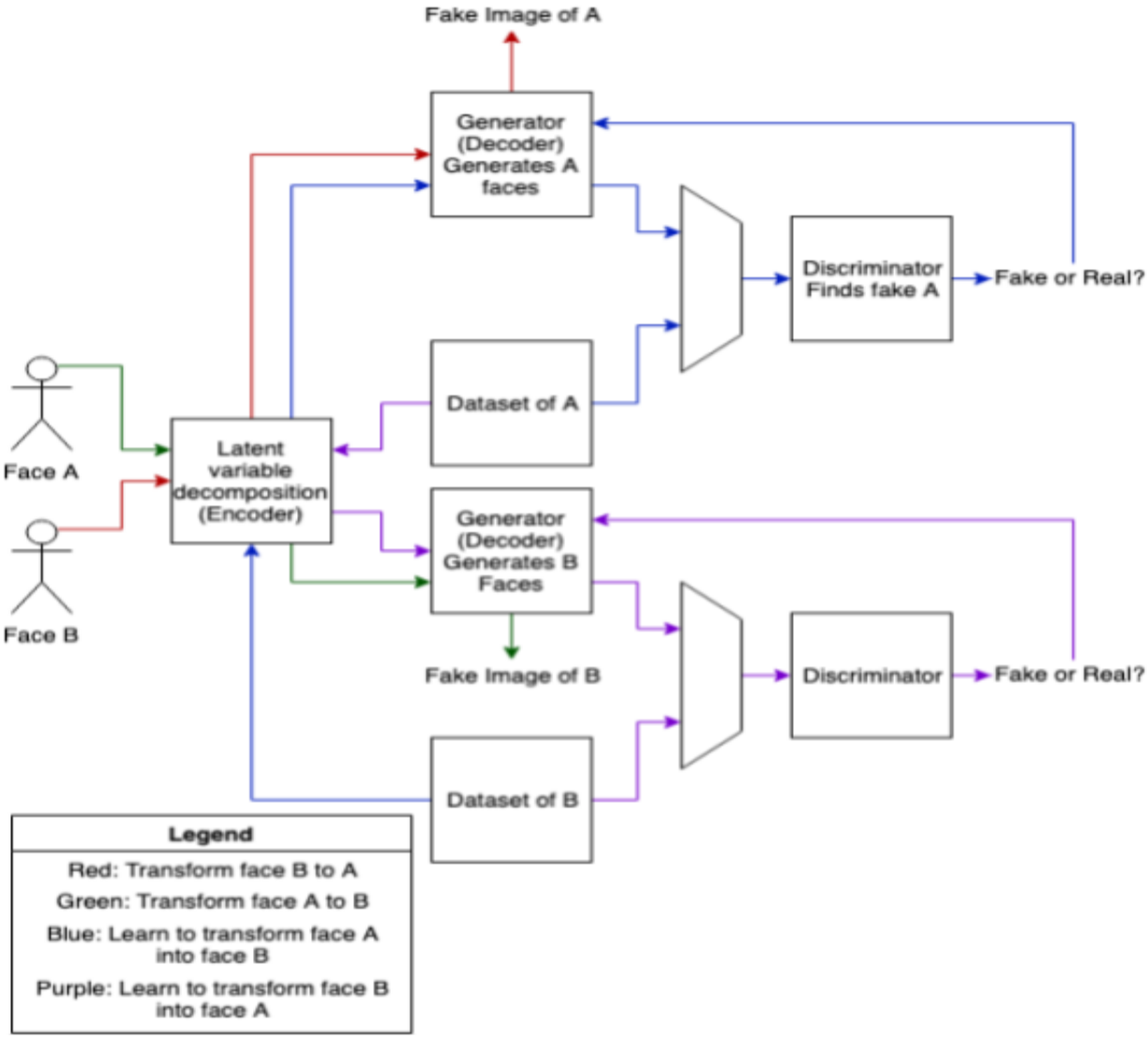


Figure 6. A detail to use GAN model architecture to exchange Face A and Face B

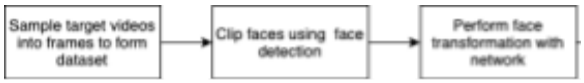


Figure 5. Face data collection from video data

3.2 MTCNN Face Detection

For face detection we choose to use the MTCNN face detector. It's implemented with vgg face [2] network, which has 95% above face detection accuracy. We choose to use 100 facial key points to perform facial detection. This class has many APIs and we could write our own API function there, e.g. cut the video for a specific time period.

3.3 Face transformation

Figure 6 is a diagram which represents the face transformation layout for two distinct people and network modules used to achieve this. Figure 7 breaks down the layers of the network modules used in our architecture.

The encoder and decoder networks are based off the architecture from [6]. They use convolutional layers with LeakyReLU activation functions to downsample the image and then the PixelShuffler layer to upsample the image. The PixelShuffler layer rearranges the shape of the image so that it doubles the height and width, but reduces the number of channels by a quarter. The discriminator is based off of [11] and uses convolutional layers and dropout layers to downsample the image. The dropout layers are to prevent overfitting of the model.

We use three different loss functions for our model's objective. The first is the binary cross entropy loss which is the typical adversarial loss used by GANs. The second is the reconstruction loss mentioned in [5]. The third is another loss introduced in [5] called an edge loss. The edge loss shifts the image by one pixel in the x or y directions and calculates how many edges are consistent over the transformation. This loss hopefully preserves the facial structure after transformation. We weight the reconstruction loss the highest since it is important to have the decoder recover the original image if passed in, while the other losses are involved with the transformation process so they are weighted slightly less so that they don't come into play until later in the learning.

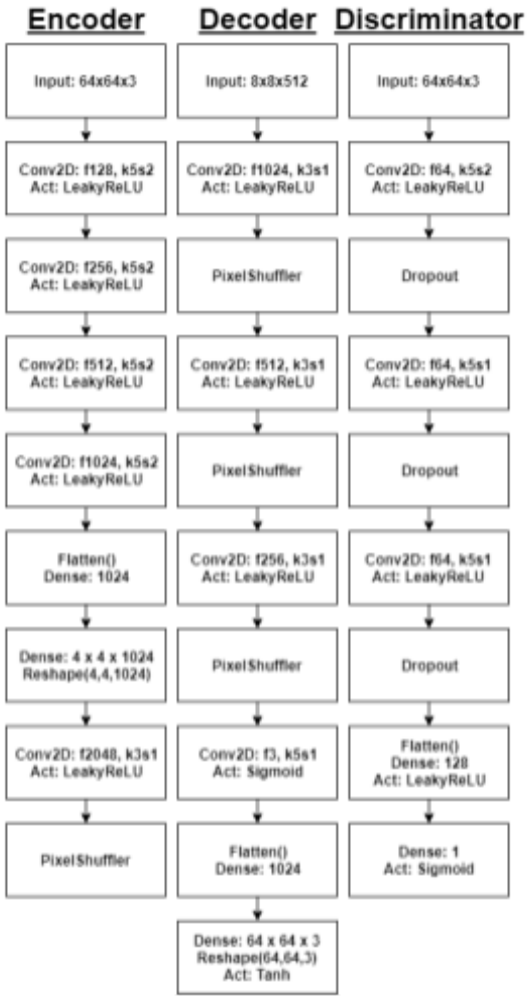


Figure 7: Module Layers

4 RESULTS

For our experiments, we used two different datasets. The first two datasets were sampled from [12] and [13] at a rate of one face per frame to get Donald Trump and Nicolas Cage images respectively. The second set of datasets were sampled from [14] and [15] at a rate of one face per five frames to get John Oliver and Tucker Carlson faces respectively.

In the end, our autoencoder did not produce the results we desired. The individual decoders did manage to learn the transformation to recreate the face of a given person. A Trump decoder would always produce something that resembled a Trump face. Despite that, several problems rose up during the training. First, the decoder only learned to output a single image. We illustrate this in Figure 8 and 9 where we print out 16 images from a batch and the resulting images are all identical. This behavior is

known as mode collapse and is a typical problem found in GANs. This is typically a result of the discriminator learning too well which results in small gradients for the generator. Then the generator only works to optimize a single mode, in this case one facial expression, rather than all other facial expressions.

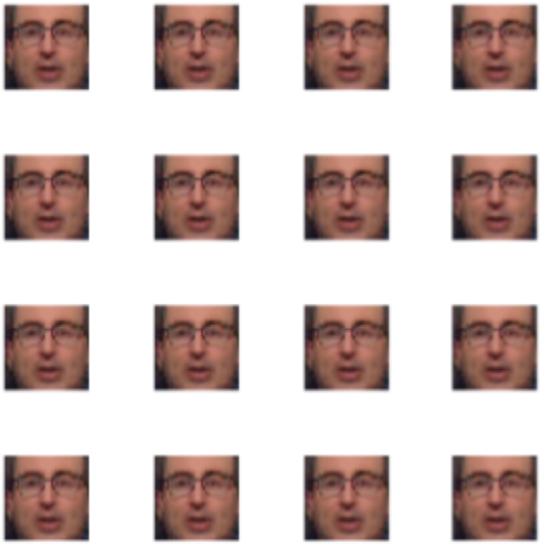


Figure 8: Batch output of Carlson to Oliver 1200 Epochs

Figure 8: Batch output of Oliver to Carlson 1200 Epochs

We tried several things to resolve the mode collapse. First, we tried weighting the losses differently. Second, we tried adding noise to the inputs of the discriminator. Third, we tried adjusting the learning rates. Weighting the losses only produced different mode collapses. Adding noise only made the modes produce noisy images. The learning rates only changed the number of epochs to arrive at a mode. In the end, none of these changes managed to resolve the mode collapse.

Another problem was that in some instances, training longer made the outputs worse as illustrated in Figure 10 and Figure 11 with the Trump and Cage case. As the training went on, the Trump images ended up producing noisier results. The Cage images ended up having a large amount of red bleed into the images later on. We believe one potential cause for this is due to our face detector. We noticed that some of the faces captured our face detector were actually not faces and were red blurred images. The mode collapse of the decoders appeared to be an average over all faces, so these extraneous faces may have started to bleed into our training. This can occur if the discriminator begins to learn that the red blurred image is considered a face and the generator is bad at producing this image. Then the generator will add the blurred image into its output to try and fool the discriminator. This is supported by the fact that when we removed these bad images from the Oliver and Carlson dataset, we managed to produce less noisy images compared to the Trump Cage dataset.

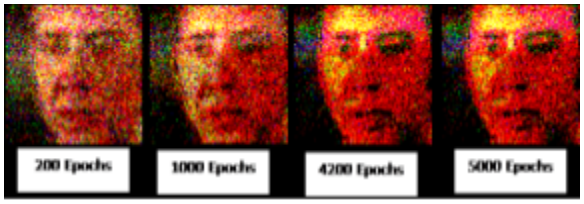


Figure 11: Trump to Cage

5 DISCUSSION

In the end, we did not manage to approach the results from [5]. One large issue was that we did not match the model complexity they used to perform the face swap was much greater than ours. Another issue was that the training time they used to produce their results was longer. We implemented our design using Google Colab, so it was difficult to train for a long time as our machines would disconnect after some time. Also, our training time was quite long. When we added GPU capabilities we estimated the training time to be 3 days for 27k training iterations. Tuning the hyperparameters was hard with such a large training time, so we did not have time to try as many combinations as we would have liked. This was our first time using Keras as well and due to the lack of GAN knowledge such as picking advanced loss, optimizer, layers, we are not able to perform a face swap that captures expression. We believe that

with more time to tune and train and model optimization, GAN may eventually learn to transform into different expressions. One other thing to try would be the addition of perceptual loss. Perceptual loss helps capture differences in features of images which could include facial expressions. We did not implement it as we had trouble integrating an additional neural network to perform perceptual analysis into our model, but it is something to try in the future.

6 CONCLUSIONS

In this final report paper, firstly we could present the principle ideas of the face swap technology from the old time to the most recently. Secondly, we get successfully expected face swap results. Thirdly, we build the optimum GAN model to make fake faces and escape the detection from the most accurate model such as vggface_net [7] and face_net [7]. We are considering proposing the paradigm of "modular learning" which will definitely conducts continuous recursive self-improvement with regards to the utility function (reward system). We can view this as second (and higher) order optimization. In general, we made the contribution to show that a deep CNN [8], without any embellishments but with appropriate training, can achieve results comparable to the state of the art [10]. Again, this is a conclusion that may be applicable to many other tasks.

CONTRIBUTIONS

Yuxuan Zhou collected most initial related resource information for the project. Implemented the initial implementation of the GAN model during the second iteration. Implemented the face detector from an online source to provide dataset data. Worked on the Trump and Cage dataset. Also transfer and test the GAN models we based our project on with successful results of these tests located at this link.

Keven Feng was responsible for the initial GAN implementation, the final model implementation, many code debugging related to the GAN. Did training on the Oliver and Carlson dataset. Also was responsible for investigating research papers

REFERENCES

1. https://github.com/shaoanlu/faceswap-GAN.git, 2018
2. https://github.com/kpzhang93/MTCNN_face_detection_alignment, 2018
3. https://arxiv.org/pdf/1406.2661.pdf, 2017
4. https://github.com/mchablani/deep-learning/blob/master/gan_mnist/Intro_to_GANs_Exercises.ipynb, 2017
5. https://github.com/shaoanlu/faceswap-GAN/blob/master/networks/faceswap_gan_model.py, 2018
6. https://www.jiqizhixin.com/articles/2018-05-04-2, 2018
7. https://new.qq.com/omn/20181231/20181231A0HNKX.html, 2018
8. http://efrosgans.eecs.berkeley.edu/CVPR18_slides/CycleGAN.pdf, 2018
9. https://www.secrss.com/articles/8196, 2017
10. https://arxiv.org/pdf/1811.00656.pdf, 2018
11. https://towardsdatascience.com/implementing-a-generative-adversarial-network-gan-dcgan-to-draw-human-faces-8291616904a, 2017
12. https://www.youtube.com/watch?v=orIDFxMng7w&list=PLZ5TMxXo7Nmx7LOouEbczOmL1PmMmilzo&index=5
13. https://www.youtube.com/watch?v=6RRvDarJsHA&list=PLZ5TMxXo7Nmx7LOouEbczOmL1PmMmilzo&index=2&t=0s
14. https://www.youtube.com/watch?v=7VG_s2PCH_c
15. https://www.youtube.com/watch?v=mSuQ-AyiicA