CS230

Deep Face Swap with GAN

Chi Wang chiwang@stanford.edu **Jinil Jang** jinil@stanford.edu

Abstract

Human image synthetic technology plays an important role in movie editing, in this project, we explored different autoencoder architectures with “DeepFake" algorithm to swap “George Clooney”’s face with ours, and apply GAN to improve the synthetic performance. The generated images show the result is better than the original FaceSwap code.

1 Introduction

Our project - "Deep Face Swap with GAN" aims at producing realistic face-swapped images using Artificial Intelligence (AI), we find AI based image synthesis technology is very interesting since it has broad usage in areas like movie post production, video editing and entertaining.

Manual Image editing requires intense labor work and special training on professional tools, it’s expensive and not scalable. AI based approach gains lots attention nowadays, they are cheap and faster but the synthetic result is constrained with factors such as light conditions, skin tones and head position, we are trying to overcome these limitations in this project.

For code implementation, we start from forking a public GitHub repo “Faceswap” [7] which provides an autoencoder based working solution for swapping images. After setting up in AWS EC2 environment, we explores different autoencoder architectures like adding more intermediate layers and nodes to get better details, and introduce Generative Adversarial Network (GAN) to synthetic result looks more realistic.

2 Related work

Traditional face swap methods - face searching [1] [4] [6] generally takes following three steps to perform face swaping: First, detect all faces in the source image and select candidate face images from the face library that are similar to the input face in appearance and pose. Second, adjust the pose, lighting, and color of the candidate face images to match the appearance of those in the input image, and seamlessly blend in the results. Third, rank the blended candidate replacements by computing a match distance over the overlap region.
The tradition face searching approach could generate good result in certain conditions but it has many limitations, you need to build a large image library and its image synthetic result is very rigid since it requires exact pose matching to get a good replacement.

Deep learning based approaches (DeepFake) [3] [2] [5] start to get very popular in 2016 due to its realistic synthetic result, currently the best perform models in this field are all based from deep learning approach. In general, they leverage two autoencoder models with CNN layers to perform face swap, since the trained autoencoder remember the key features of target faces, it is very adaptive to different conditions from source image to target image. With proper blending, the result could be seen as nearly nature. DeepFake approach also doesn’t require prepare image search library

and implementing image lights, pose, skin color comparison algorithm, it's quit efficient and cheap to write code and prepare training data.
From above comparison, we choose to implement our project base on DeepFake approach.

3 Dataset and Features

We have prepared 500 celebrity - "George Clooney"'s images and 200 our own images (200/Chi or 200/Jinil) for model training and validation. Since we choose to swap "George Clooney"'s face with our faces in "George Clooney"'s image, so all Chi and Jinil's image (200) will be used as training set and we random split 500 George Clooney pictures to 400/50/50 as train, validation and test sets.

3.1 Image collection

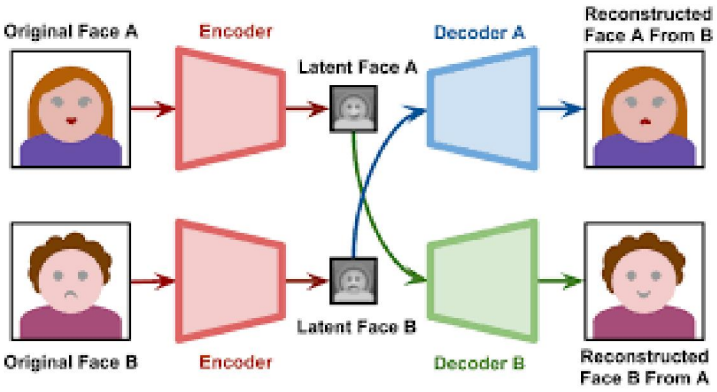
Since George Clooney is a celebrity, we write some scrape code to download his image from Google image search Deep Learning Dataset Collector, and filter out the good quality images, such as pictures has good face orientation, frontal and profile. We random select 200 pictures from our own photos in different pose, daylights and seasons. See example below:

We then run a face extraction script to extract faces from above raw input to 256x256 images and use them as dataset. See example bellow:

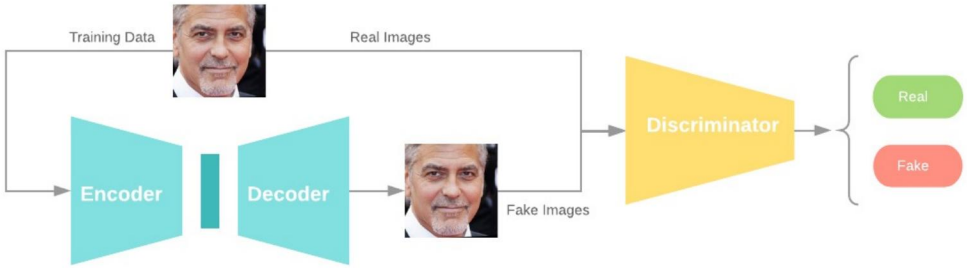
4 Methods

4.1 Deep Face Swap Algorithm

In order to swap person A and B's faces, we trained two autoencoder models (model A and model B), one for each person. During the training, model A and B share a same (common) encoder but have separate decoders.
In this way, both person A and B's face features are encoded at the same shared layers, and both decoder A and B knows how to decode person A and B's face from it.
When we want to make B's face on A's image, we use A's image as input data to perform prediction on model A, the key change is we swap model B's decoder with model A's decoder, so the output images have B's face on A's image. See the conceptual graph below:



Beside exploring different hyper-parameters, we also introduce GAN algorithm to improve autoenoder's training performance. The structure is simple, we choose training data as real image and autoencoder output image as fake image, use them as intput to train a CNN based discriminator, see below graph for concept.

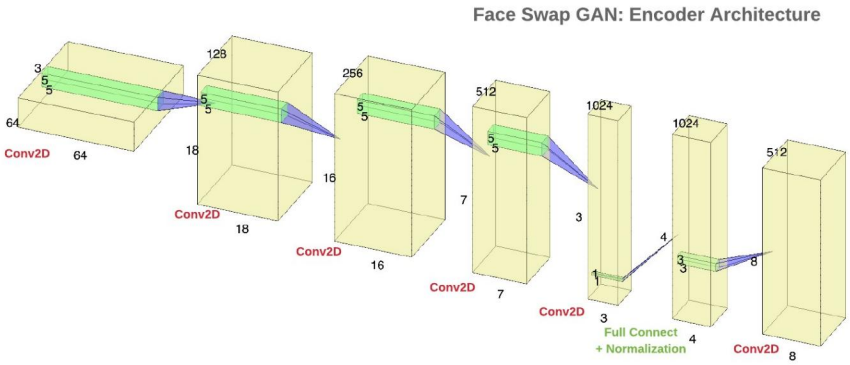


4.2 Code

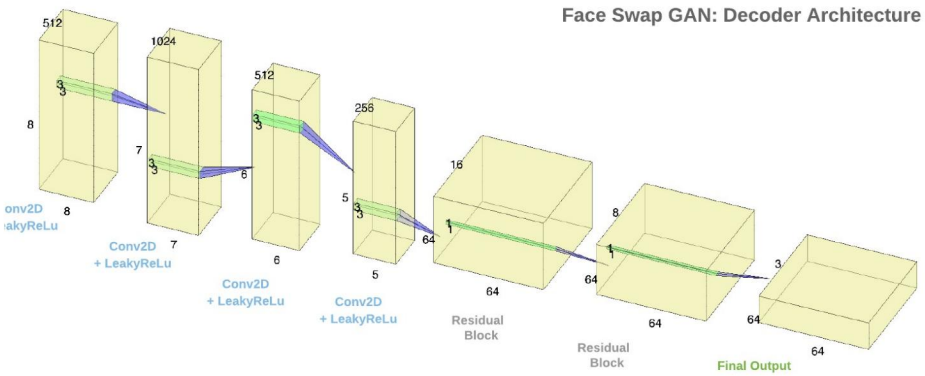
We start our project by forking public FaceSwap Github repository [7], code implementation could be found here: Deep Fake GAN

4.3 Architecture

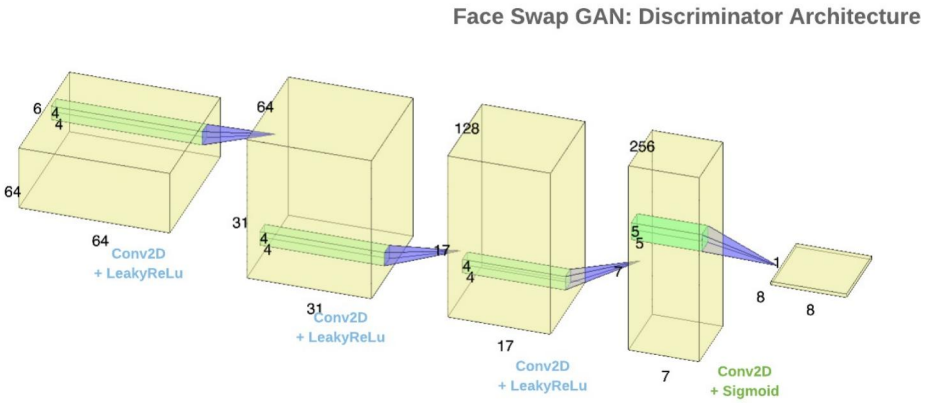
Encoder Architecture:



Decoder Architecture:



GAN Discriminator Architecture:



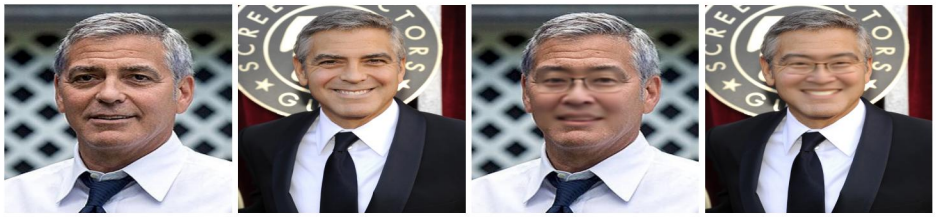
4.4 Loss functions and Optimization

For discriminator, we use Mean Squared Error as the loss function to identify fake image.
For Autoencoder, we use DSSIM as loss function and Adam optimizer, SSIM is used for measuring the similarity between two images. For more detail, please checkout: (Difference of Structural Similarity)

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

5 Experiments/Results/Discussion

To build a baseline for our experimentation, we set up original FaceSwap code [7] on a GPU-enabled EC2 instance, we found it took at least 60 hours to get a acceptable result, it needs more than one week for a non-GPU instance to get similar result. Due to the expensive cost of each run, we decided to adopt the suggest hyper-parameter value and spend limited resource to explore architecture changes, such as adding different layers and testing GAN. The **learning rate** we choose is 0.00005 and **batch size** is 64.
Since our project is an autoencoder based approach, which is essentially unsupervised machine learning, we choose to use visual comparison to evaluate models and we setup tensorboard to monitor the batch loss during training to ensure gradient descent is working. Here are some sample output images from original model (base line output):



From the original Faceswap codebase [7], we add two major improvements, first one is to add two full connect layers and reshape to (4, 4, 1024) before the code layer to capture more details, and add two residual blocks in decoder to help gradient to back propagate. From below sample output, we could see more details are painted, especially the teeth:

The second improvement we did is introducing GAN while training autoencoder, we hope autoencoder could capture more details of person A and B with the extra adversary network. From below sample output, GAN seems didn't provide significant improvement on autoencoder model:

6 Conclusion/Future Work

From our experiments we found best face swap result is the improved autoencoder with GAN. Surprisingly GAN seems only provide a few improvement to our first approach - adding dense layers and residual blocks, so we think it may not worth the efforts to introduce GAN.
See from below bad samples, face edge is not smooth with bear and the outpu image get blurred if the source image is in black and white. To solve these problems, we could do data augmentation to generate more black and white images to improve model performance on dark environment. Also, we could try to apply GAN on the swapped image to make edge of the faces smoother.

7 Contributions

Chi: Data collection script, autoencoder model improvement, report and poster video.
Jinil: Data collection script, autoencoder model improvement, GAN implementation and poster.

References

[1] Dmitri Bitouk, Neeraj Kumar, Samreen Dhillon, Peter Belhumeur, and Shree K Nayar. Face swapping: automatically replacing faces in photographs. In *ACM Transactions on Graphics (TOG)*, volume 27, page 39. ACM, 2008.

[2] Yaroslav Ganin, Daniil Kononenko, Diana Sungatullina, and Victor Lempitsky. Deepwarp: Photorealistic image resynthesis for gaze manipulation. In *European conference on computer vision*, pages 311–326. Springer, 2016.

[3] Hyeongwoo Kim, Pablo Carrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Niessner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep video portraits. *ACM Transactions on Graphics (TOG)*, 37(4):163, 2018.

[4] Yuan Lin, Qian Lin, Feng Tang, and Shengjin Wang. Face replacement with large-pose differences. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 1249–1250. ACM, 2012.

[5] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[6] Brandon M Smith and Li Zhang. Joint face alignment with non-parametric shape models. In *European Conference on Computer Vision*, pages 43–56. Springer, 2012.

[7] Clorr ... torzdf, kvrooman. Face swap. https://github.com/deepfakes/faceswap, 2018.