# BertonGan: a conditional GAN for performing various tasks

Aaron Schindler, Herbert Wright

December 16, 2022

# 1 Introduction

## 1.1 Problem Statement

We wish to construct a network that can, given a few images of an unseen face, construct face images/deepfakes that are similar to that one, even if it has not seen that person before. [1] is an example of face swapping, but without the few-shot learning of new faces.

Deep learning is the traditional method of generating and detecting deepfakes [2]. While some computer graphics methods can be used, they lack the same ability of deep learning architectures to capture and learn complex functions efficiently. Because the images generated by deepfake architectures are very realistic, most humans are not able to distinguish between real and fake images. We can use a different (or same in some cases) models to learn the subtle differences between real and fake images, like differences in noise or counts of pixel colors, to effectively decide if an image is real or not.

## 1.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) were introduced by Ian Goodfellow [3].

These GANs tend to require a lot of data.

# 2 Methods

## 2.1 BertonGan structure

Our approach is to train an encoder-decoder while also simultaneously training the discriminator network. Both the decoder and discriminator(s) will be conditioned on a latent variable that provides all necessary facial information. The discriminator(s) will give two values corresponding to whether or not the reconstructed image is fake and if the image also corresponds to the latent variable it has been conditioned on. The four network components of our project are outlined below:

1. Face encoder network: $f_F : \mathbb{R}^{n \times W \times H} \to \mathbb{R}^{h_f}$
    (a) Input: $n$ images of the same subjects face
    (b) Output: A latent representation of the subjects face
2. Image encoder network: $f_I : \mathbb{R}^{W \times H} \to \mathbb{R}^{h_I}$
    (a) Input: An image of a subjects face
    (b) Output: A latent representation of the image
3. Image decoder network: $f_G : \mathbb{R}^{h_F + h_I} \to \mathbb{R}^{W \times H}$

(a) Input: Latent representations of a face and image

(b) Output: A reconstructed image decoded from the latent features

4. Discriminator network: $f_D : \mathbb{R}^{W \times H} \times \mathbb{R}^{h_F} \to [0,1]^2$

(a) Input: An image and a latent representation of a face

(b) Output: Two probabilities

i. Probability of being a fake image

ii. Probability of being a different person than the faces encoded into the latent vector

A visual encoding of the networks described above is given below:

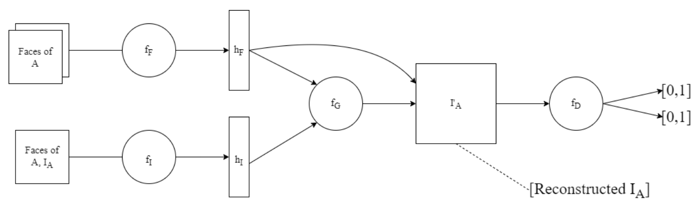


Figure 1: Figure 1 is a visual representation of our total network, comprising each of the four components above

## 2.2 Networks

## 2.3 Training Procedure

We propose using two datasets to train/test. The first is the built in MNIST dataset from the torchvision package. This dataset will allow us to experiment with our networks on a smaller scale, as the MNIST images are $28 \times 28$ images of numbers rather than people. The next step is to use the celebA dataset which is also built in to the torchvision package. Our original plan to use the MS-Celeb-1M dataset [4] has been changed because the dataset is no longer publicly available. The celebA dataset comprises over $200,000$ images of faces, that are $178 \times 218$ in size.

Given a batch $\beta = (F_A, I_A, I_B)$, where $F_A = n$ faces of person $A$, $I_A = N$ other faces of person $A$, and $I_B = N$ faces of people other than person $A$, we will compute the following quantities:

1. $h_F = f_F(F_A) \in \mathbb{R}^{h_F}$

2. $h_I = f_I(F_B) \in \mathbb{R}^{N \times h_I}$

3. $h_B = f_I(F_B) \in \mathbb{R}^{N \times h_I}$

4. $I'_A = f_G(h_F, h_I) \in \mathbb{R}^{N \times W \times H}$

5. $I'_B = f_G(h_F, h_B) \in \mathbb{R}^{N \times W \times H}$
6. $(R_{A'}, C_{A'}) = f_D(I'_A, h_f) \in [0,1]^{N \times 2}$
7. $(R_A, C_A) = f_D(I_A, h_F) \in [0,1]^{N \times 2}$
8. $(R_{B'}, C_{B'}) = f_D(I'_B, h_F) \in [0,1]^{N \times 2}$
9. $(R_B, C_B) = f_D(I_B, h_f) \in [0,1]^{N \times 2}$
10. $D_A = \|I_A - I'_A\|$
11. $D_B = \|I_B - I'_B\|$

We will optimize $f_D$ by maximizing $R_A, R_B, C_A$ and minimizing $R_{A'}, R_{B'}, C_B$. We also optimize $f_F$ by maximizing $C_A, C_{A'}, C_{B'}$ and minimizing $C_B, D_A$. Additionally, we optimize $f_G, f_I$ by maximizing $R_{A'}, R_{B'}, C_{A'}, C_{B'}$ and minimizing $D_A, D_B$. All parameters are optimized by using stochastic gradient descent.

After training, we will be able to use $f_F, f_I,$ and $f_G$ to perform face swaps. We will then use $f_D$ to identify fake/real images generated using face encoding $h_F$. Additionally, we will use $f_F$ and $f_G$ to generate new images of an already learned face.

# 3 Experiments

## 3.1 MNIST Experiments

We first trained on the MNIST dataset with defined networks that were not too far from the GANs we used in the notebook in class. In the first BertonGan we trained ended up with saturated gradients in the discriminator1 network (the one that predicts whether or not the image is fake). After this, the generator was able to easily fool this, leading to fuzzy output images. We trained this network for 50 epochs. We show two different graphics in figures 2 and 3; the first is performing our version of style transfer where the number of the image is the style class it belongs to, and just some generated images from picking random content and style images. All of the style and content images are from the test set and thus not seen before by the network.

Note that in figure 2, columns 1 and 4 are content, 2 and 5 are style (digit number), and 3 and 6 are the generated image. The goal is for the third column's image to be "close" to the first column, with the same number as the second.

The gradients were being saturated because we had a sigmoid in the last layer of the network to output a probability. We removed this activation, added two more layers to the network and retrained. We used a slightly different procedure this time around; we first trained the image encoder and decoder as an autoencoder for 5 epochs, then trained the whole network. Figures 5, 6, 7 and 8 show this network at various epochs. Figures 9 and 10 show, after 50 epochs, our version of style transfer and a sample of generated images similar to figures 2 and 3. Each image used for content and style is from the test set as before

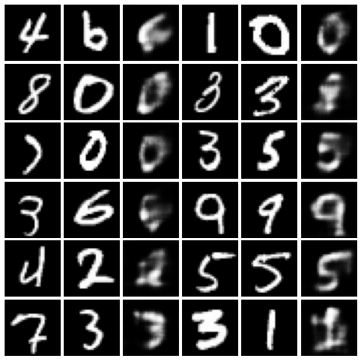

Figure 2: (a)
Style transfer

Figure 3: (b)
Generated images

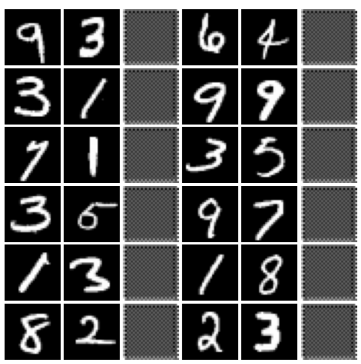

Figure 4: (a)
1st epoch (random noise)

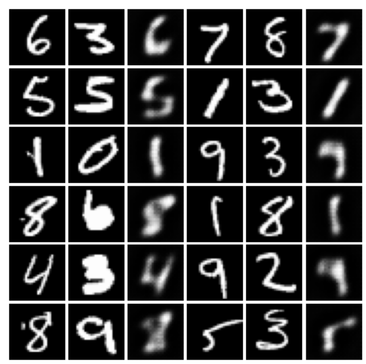

Figure 5: (b)
5th epoch (autoencoder)

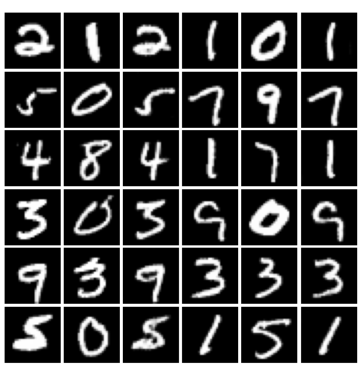

Figure 6: (a)
15th epoch (no style transfer)

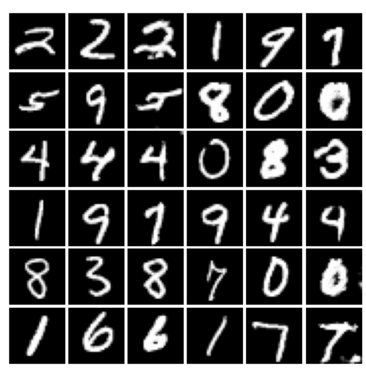

Figure 7: (b)
30th epoch

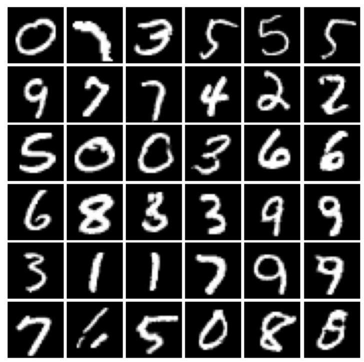

Figure 8: (a)
Style transfer (50th epoch)

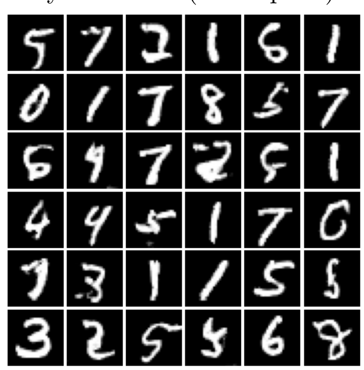

Figure 9: (b)
36 generated images

Because we use latent dimension of 2 for our face/style encodings (encoding of the digit shown), we can display a grid of how this value changes as you move throughout the latent space for a given image. In figure

Figure 10: visualization of latent space

### 3.2 CelebA Experiments

Unfortunately, because we spent more time on building/improving the MNIST dataset there was not enough time to adequately experiment with the celebA dataset. We do have the network partially built, a network that downsamples first in order to be more efficient since the data we are working with in the celebA dataset is much larger than MNIST.

## 4 Conclusion

In conclusion, it is very clear that when generating deepfakes it is possible to combine two images into one using style transfer. One downside to performing this operation was the amount of time and compute resources it takes to train the network. It takes roughly 10 minutes to train using cloud computing resources such as google colab for just the MNIST dataset. Because of this, the celebA dataset presented an entirely new challenge since the dataset size is much larger than MNIST and will require a more efficient network.

We believe the future direction of this work is to build an efficient network for celebA by using downsampling in order to decrease the number of parameters that need to be computed at intermediate steps. An added difficulty of the celebA dataset is that becasue images are larger and more robust than that of MNIST we also believe that a deeper network will be required. The proposed solution will be to model a network like ResNet, which downsamples in intermediate blocks. This not only creates a deeper network, but the size is easily variable since blocks can be added to the model easily.

# References

[1] B.-S. Lin, D.-W. Hsu, C.-H. Shen, and H.-F. Hsiao, "Using fully connected and convolutional net for gan-based face swapping," in *2020 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 185–188, IEEE, 2020.

[2] T. T. Nguyen, Q. V. H. Nguyen, D. T. Nguyen, D. T. Nguyen, T. Huynh-The, S. Nahavandi, T. T. Nguyen, Q.-V. Pham, and C. M. Nguyen, "Deep learning for deepfakes creation and detection: A survey," *Computer Vision and Image Understanding*, vol. 223, p. 103525, 2022.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[4] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition," in *European conference on computer vision*, pp. 87–102, Springer, 2016.