# COP 6610(Machine Learning) Fall 2019
# Project Report

Yefan Tao
UFID: 24055676
ustctao@ufl.edu

October 29, 2019

This report consists of the follwing four parts:

- Introduction and background
- Results of GAN(Generative Adversial Network) on MNIST and face data
- Results of VAE(Variational Auto Encoder) on MNIST and face data
- other advanced topic(ie, generate cat face)

## 1 Introduction and Background

The goal of this project is to apply GAN and VAE on MNIST and given input face data, to generate "fake" handwritten digit and faces. The original idea of GAN was proposed by Goodfellow et al in 2014 [1], and grow rapidly into lots of different variations, including DCGAN(Deep Convolutional GAN) [4], LSGAN(Least Square GAN) [3], Softmax GAN [2] and so on. The key idea of GAN is that, we will have a Generator(**G**) trying to mimic the input data, and a Discriminator(**D**) trying to distinguish "real" data from the "fake" ones. After several epochs of training on **G** and **D**, if converged, **G** will be the same distribution as the input and **D** will go to 1/2, meaning it's unable to distinguish real input from the generation.

# 2 Results of GAN(Generative Adversial Network) on MNIST and face data

The realization of GAN is not hard, there are lots of examples avaible on the Internet. I am taking the one from the GitHub( `https://github.com/eriklindernoren/PyTorch-GAN/blob/master/implementations/gan/gan.py`) to run on MNIST, and use another DCGAN(`https://github.com/eriklindernoren/PyTorch-GAN/blob/master/implementations/dcgan/dcgan.py`) to run on face data.

## 2.1 Data

### 2.1.1 MNIST data

The MNIST data in loaded by using Dataloader in PyTorch directly as below.

```
# Configure data loader
os.makedirs("../../data/mnist", exist_ok=True)
dataloader = torch.utils.data.DataLoader(
    datasets.MNIST(
        "../../data/mnist",
        train=True,
        download=True,
        transform=transforms.Compose(
            [transforms.Resize(opt.img_size), transforms.ToTensor(), transforms.Normalize([0.5], [0.5])]
        ),
    ),
    batch_size=opt.batch_size,
    shuffle=True,
)
```

Each element in "DataLoader" has an input image of 25 handwritten digits, and in each epoch, there are 938 batches. Apparently MNIST has 10 classes representing digits from 0 to 9, and the training/validation split is done automatically. The dataset will be downloaded automatically at run time.

### 2.1.2 Face data

Face data is provided by TA, coming from the VGGFace2 dataset. These data set consists of faces from 500 different people(grouped in different subfolder), and a total number of 170k face images. For the face images, my thought

is to use the subfolder's name to label each person, so we have 500 different classes. And for each loop of the running, an input image with 65 faces will be trained with a DCGAN.

## 2.2 Results

Here I attach only parts of the results, the full set of results are available on my GitHub repository(`https://github.com/yefanTao/CAP6610_GAN`). For MNIST results, output image is stored for every 400 times of training, from 0 to 187200. For face data, output image is stored for every 50 times of training, from 0 to 6596.
Figure 1 shows the output results for MNIST at step=0,800,2000,10000,50000 and 100000. As we can see, the training result is fairly good, the output images look the same as handwritten digits.

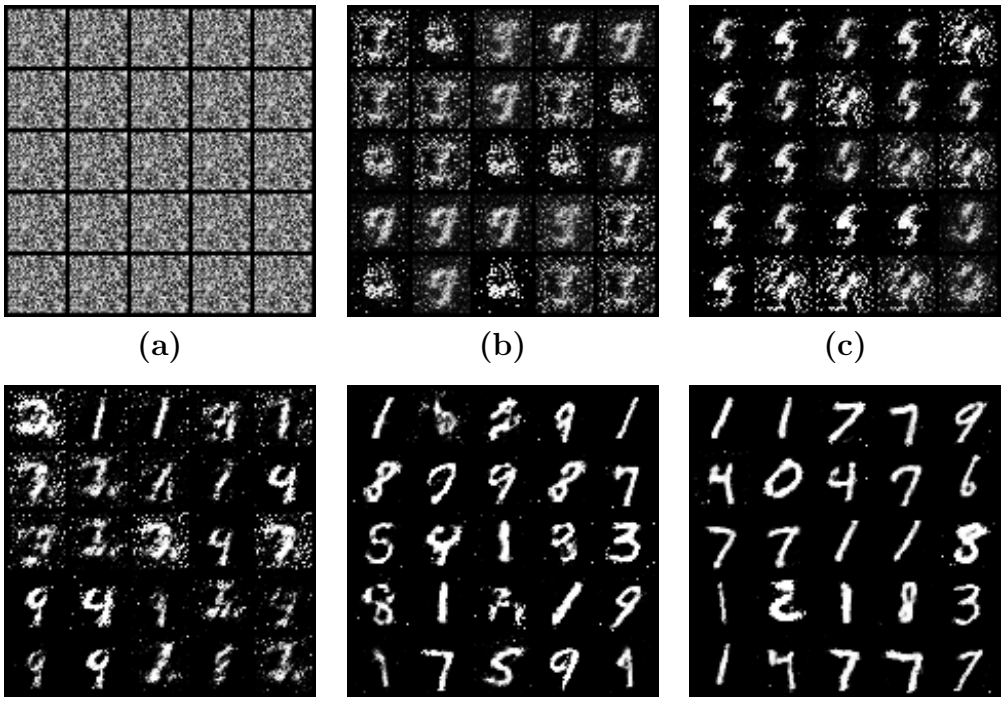
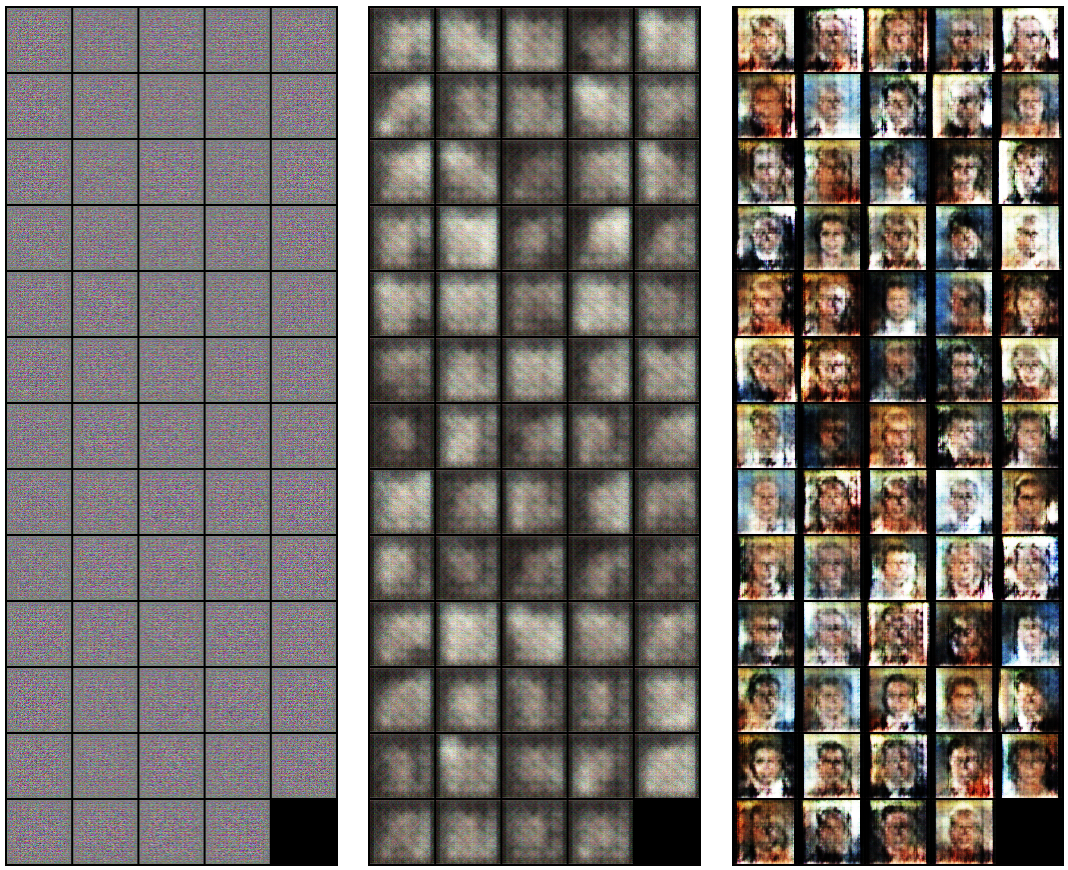



Figure 1: Training result on MNIST at step **(a)** 0 **(b)** 800 **(c)** 2000 **(d)** 10000 **(e)** 50000 **(f)** 100000

Figure 2 shows the output results for face data at step 0, 100, 1000. Similarly, we can see that the output images are getting more and more clear, and we can see the profiles of the faces in the last one. However, even after training for about 10 hours on my laptop, this output is still blurry and not clearly showing the details of the faces. I believe the DCGAN network needs to be tuned and further improved. I will do this work in the future.



**(a)** **(b)** **(c)**

Figure 2: Training result on MNIST at step **(a)** 0 **(b)** 100 **(c)** 200 **(d)** 1000

# 3 Results of VAE(Variational Auto Encoder) on MNIST and face data

Will do.

# 4 other advanced topic(ie, generate cat face)

Will do.

# References


[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[2] Min Lin. Softmax gan, 2017.

[3] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks, 2016.

[4] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.