Lab4-1 - InfoGAN

Wen-Jie Tseng (0556146)
Deep Learning and Practice - 2018 Spring Semester

1. INTRODUCTION

InfoGAN is an information-theoretic extension to the Generative Adversarial Network (GAN) that is able to learn disentangled representations in a completely unsupervised manner. This generative adversarial network that also maximizes the mutual information between a small subset of the latent variables and the observation. We trained this model with 80 epochs on MNIST dataset.

2. EXPERIMENT SETUP

2.1 Implementation of InfoGAN

2.1.1 Adversarial loss

This generator is trained by playing against an adversarial discriminator network D that aims to distinguish between samples from the true data distribution P_{data} and the generator's distribution P_G. For a given generator, the optimal discriminator is $D(x) = P_{data}(x)/(P_{data}(x) + P_G(x))$. Thus, the minimax game is given by the following expression:

$$min_G max_D V(D,G) = E_{x\sim P_{data}}[logD(x)] + E_{z\sim noise}[log(1 - D(G(z)))]$$

Each time we obtain the output from discriminator, we will calculate the loss of this output and label (real and fake). These loss will be send backward to the network. For loss of generator, we passed the input to discriminator and obtain reconstruction loss. Next use the output of Q network to compute loss. Combine two Q loss and reconstruction loss is loss of generator.

2.1.2 Maximizing mutual information

$$\begin{aligned} L_I(G,Q) &= E_{c\sim P(c),x\sim G(z,c)}[\log Q(c|x)] + H(c) \\ &= E_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c) \\ &\leq I(c; G(z,c)) \end{aligned}$$

$L_I(G,Q)$ is easy to approximate with Monte Carlo simulation. In particular, L_I can be maximized with respect to Q directly and with respect to G via the reparametrization trick. This step was made in main.py, setting optimizer, and updating D in the second and the third times.

2.1.3 Generating fixed noise and images

Generate random one hot encoder (size 10) and combine with torch.randn tensor (size 54), which will be z in size 64. Also need to do resize while loading data, otherwise the size would be unequal during training.

2.2 Loss Function of Generator

$$L_G = E_{x\sim p_r}[logD(x)] + E_{x\sim p_g}[log(1 - D(x))] + L_I(G,Q)$$

3. RESULT

3.1 Results of Samples

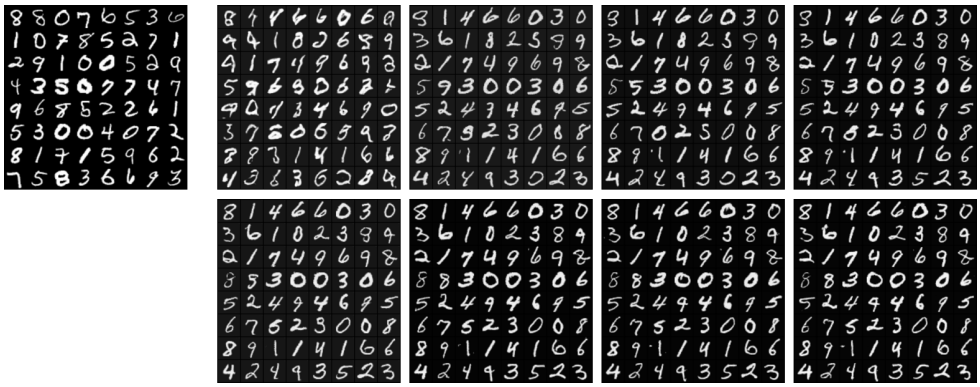

Figure 1. The upper left is a sample from real data. The first row from left to right is the disentangled representation learned by our model in 0, 10, 20, 30 epoch. The second row is 40, 50, 60, and 79 epoch.

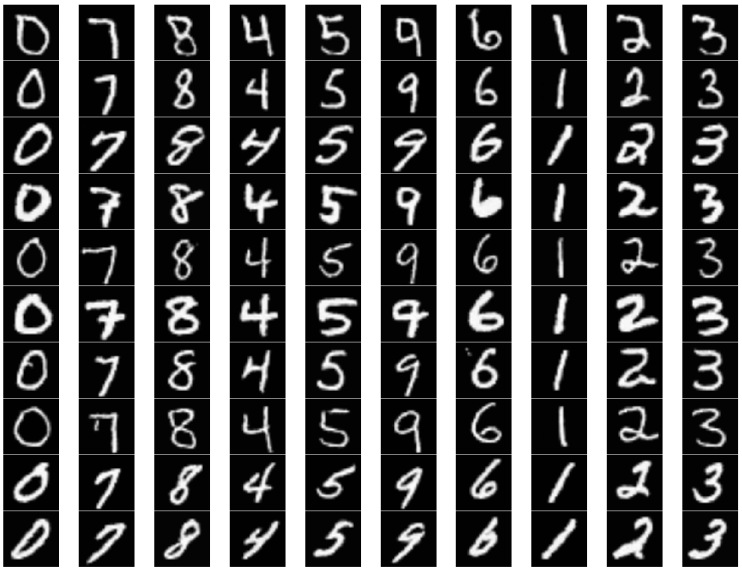

Figure 2. The evaluation result.

3.2 Training Loss and Probabilities Curves

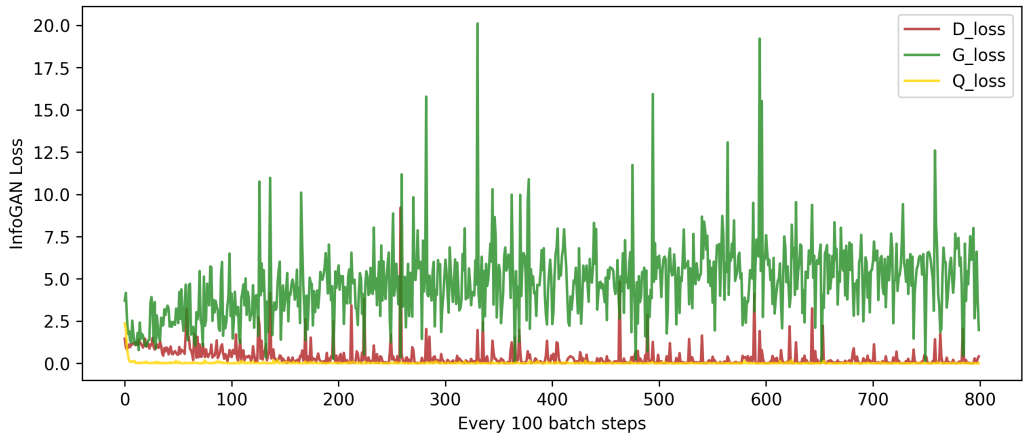

Figure 3. The training loss of decoder, generator and Q. The curves of D and G suffered from instability.

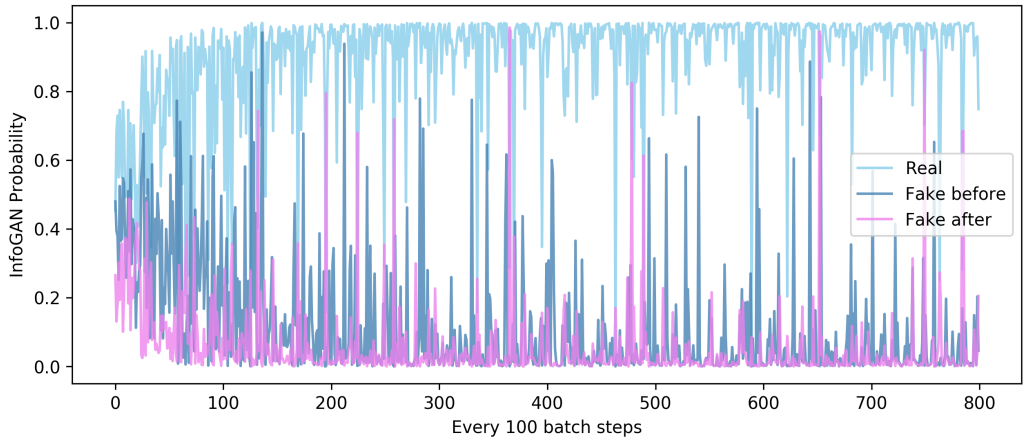

Figure 4. The probabilities of real data, fake data before updating G, and fake data after updating G. Despite the variation is large, one still can observe a trend that real data probability grew up to one and the other two went down to zero.

4. DISCUSSION

Compared to CVAE, InfoGAN produces a more clear output image. But the results somehow suffer from acute style changing.

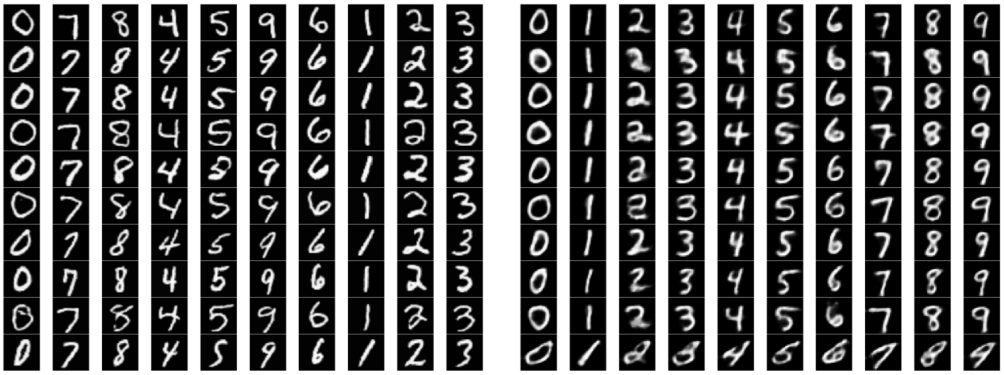

Figure 5. (Left) The disentangled representation learned by InfoGAN, digits vary from answer but clear. (Right) The disentangle representation learned by CVAE, digits are correct but somehow blurred.

REFERENCES

1. DCGAN provided in Lab4.
2. InfoGAN provided in Lab4.