Project 4: Generative modeling

Yufei Hu (404944367)

March 3, 2018

1 Variational Autoencoder

Read sections 2 and 3 in *https://arxiv.org/pdf/1312.6114.pdf* for implementation details. Specifically, I have implemented the reparameterization trick, and used equation (10) as loss term.

After training, show the reconstructed images and sampled images.

The loss function for updating both encoder and decoder models is listed as follows:

$$L = \frac{1}{2}\sum_{j}^{J}(1+\log(\sigma_j^2)-\mu_j^2-\sigma_j^2)+\frac{1}{M}\sum_{i=1}^{M}\log p_\theta(x\,|\,z^{(i)}) \tag{1}$$

where the second term is just the summation of log-likelihood of a Bernoulli distribution.

The architectures for both decoder and encoder are rather simple considering the MNIST dataset is quite simple to fit. The summary for the architectures are listed below:

Table 1: Neural network architecture for VAE encoder

Block	**Parameters**
FC - elu - dropout	hidden unit size = 512, dropout = 0.9
FC - tanh - dropout	hidden unit size = 512, dropout = 0.9
FC	hidden unit size = 20×2

Table 2: Neural network architecture for VAE decoder

Block	**Parameters**
FC - tanh - dropout	hidden unit size = 512, dropout = 0.9
FC - elu - dropout	hidden unit size = 512, dropout = 0.9
FC - elu - dropout	hidden unit size = 512, dropout = 0.9
FC - sigmoid	hidden unit size = 28×28

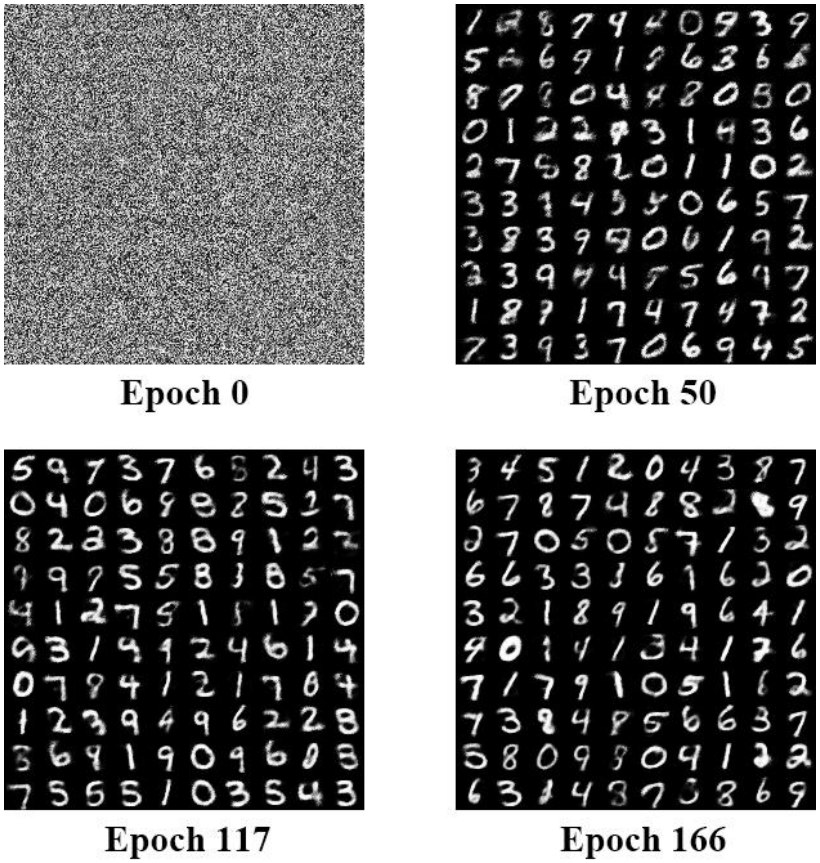

Figure 1: Synthesized fake handwritten numbers at different training epochs

As shown in the figure above, the synthesized fake numbers are getting more and more authentic after a sufficient number of training epochs.

Figure 2: Left column is real sampled data and right column is corresponding synthesized fake data

Since there is reconstruction loss in the total loss function, VAE is trying to mimic the real data. Later it is seen that GAN does not have this effect.

2 Generative Adversarial Network

Read section 3 in *https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf* for implementation details. *https://arxiv.org/pdf/1511.06434.pdf* contains more tricks about how to define my network structure.

After training, the sampled images are shown.

The loss functions for updating both discriminator and generator models are listed as follows:

$$D^* = \arg\max_D E_{p_{data}}[\log D(X)] + E_{h\sim p(h)}[\log(1 - D(G(h)))]$$
$$G^* = \arg\max_G E_{h\sim p(h)}[\log D(G(h))] \quad (2)$$

The architectures for both discriminator and generator are listed below:

Table 3: Neural network architecture for GAN discriminator

Block	Parameters
Conv - Leaky ReLU	filter size = 64×1×5×5, stride = 2
Conv - BN - Leaky ReLU	filter size = 128×64×5×5, stride = 2
Conv - BN - Leaky ReLU	filter size = 256×128×5×5, stride = 2
Conv - BN - Leaky ReLU	filter size = 512×256×5×5, stride = 2
FC - sigmoid	hidden unit size = 1

Table 4: Neural network architecture for GAN generator

Block	Parameters
FC - BN - ReLU	hidden unit size = 512×1×1
Deconv - BN - ReLU	output size = 3×3×256, stride = 2
Deconv - BN - ReLU	output size = 7×7×128, stride = 2
Deconv - BN - ReLU	output size = 14×14×64, stride = 2
Deconv - tanh	output size = 28×28×1, stride = 2

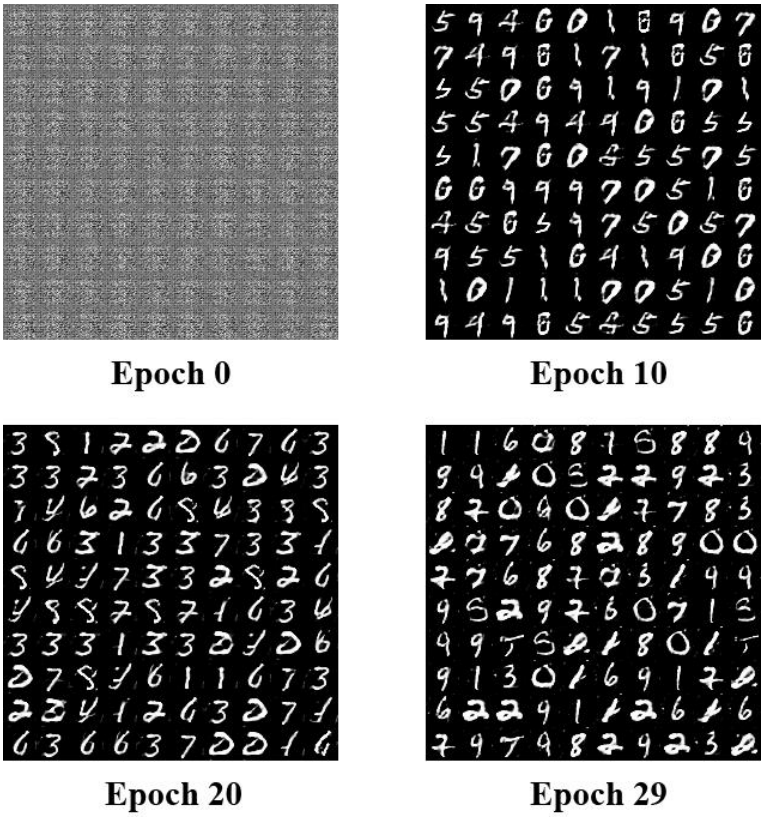

Figure 3: Synthesized fake handwritten numbers at different training epochs

Due to the fact that convolutional layers are applied in GAN model, training is much slower compared to VAE. As shown in the figure above, the synthesized fake numbers are getting more and more authentic after a sufficient number of training epochs. However, there are still some flaws in the synthesized fake data.

Figure 4: Left column is real sampled data and right column is corresponding synthesized fake data

It is seen in the figure above that the synthesized data are generated randomly instead of gradually approaching the real sampled data as shown in VAE. The reason is because there is no reconstruction loss involved and the generated fake data is simply based on the randomly sampled latent variables.