Deep Learning 2020 **(Due: [December, 2020])**

Submission Assignment #[5]

Instructor: Jakub Tomczak *Name:* [Qinghe Gao], *Netid:* [qgo500]

1 Problem statement

In this project, we implement two classic generative models: Variational Auto-Encoder(VAE) and Generative adversarial network(GAN). We firstly build reasonable structures for both models considering computational power. Then we use two datasets: MNIST and Imagenette to train the models. To evaluate the two models we use FID score to compare the generated images and true images. Besides, we also sample from the latent space to see the image transition.

2 Methodology

2.1 VAE

For Variational Auto-Encoder(VAE) two basic generative processes are $z \sim p_z(x)$ and $x \sim p_\theta(x|z)$, and to optimize $log(p(x)) = \int p_\theta(x|z)p_z(x)dz$. It becomes intractable when the transformation is non-linear. In this case we can use variational inference solve the integral:

$$\begin{aligned} log p_\vartheta P(x) &= log \int p_\theta(x|z)p_\lambda(z)dz \\ &= log \int \frac{q_\phi(z|x)}{q_\phi(z|x)} p_\theta(x|z)p_\lambda(z)dz \\ &\geq \int q_\phi(z|x) log \frac{p_\theta(x|z)p_\lambda(z)}{q_\phi(z|x)} dz \\ &= E_{z\sim q_\phi(z|x)}[p_\theta(x|z)] - KL(q_\phi(z|x)||p_\lambda(z)) \end{aligned} \tag{2.1}$$

Equation 2.1 shows the derivation of VAE. To understand this equation we need to combine with the Figure 1. $q_\phi(z|x)$ is encoder which is the variational posterior given x. $p_\theta(x|z)$ is decoder and $p_\lambda(z)$ is prior. Furthermore, the exception part is called reconstruction error and KL part is regularization which makes the distributions returned by the encoder close to a standard normal distribution.

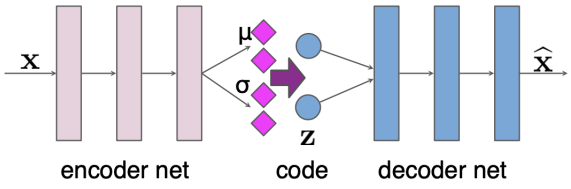


Figure 1: VAE.(VAE)

Furthermore, we need to derive the loss function which is used in our model. Denote $p_\lambda(z) \sim N(\mu_p, \sigma_p^2)$ and $q_\phi(z|x_i) \sim N(\mu_q, \sigma_q^2)$. For KL part:

$$-KL(q_\phi(z|x_i)||p_\lambda(z)) = \int \frac{1}{\sqrt{2\pi\sigma_q^2}} e^{-\frac{(x-\mu_q^2)}{2\sigma_q^2}} log(\frac{\frac{1}{\sqrt{2\pi\sigma_p^2}} e^{-\frac{(x-\mu_p^2)}{2\sigma_p^2}}}{\frac{1}{\sqrt{2\pi\sigma_q^2}} e^{-\frac{(x-\mu_q^2)}{2\sigma_q^2}}})dz$$

$$\textbf{To simplify: } -KL(q_\phi(z|x_i)||p_\lambda(z)) = \frac{1}{\sqrt{2\pi\sigma_q^2}} \int e^{-\frac{(x-\mu_q^2)}{2\sigma_q^2}} (log(\frac{\sigma_q}{\sigma_p}) - \frac{(x-\mu_q)^2}{2\sigma_q^2} + \frac{(x-\mu_p)^2}{2\sigma_p^2})$$

$$\begin{aligned}\textbf{Expectation form: } -KL(q_\phi(z|x_i)||p_\lambda(z)) &= E_{q_\phi(z|x_i)}(log(\frac{\sigma_q}{\sigma_p}) - \frac{(x-\mu_q)^2}{2\sigma_q^2} + \frac{(x-\mu_p)^2}{2\sigma_p^2}) \\ &= log(\frac{\sigma_q}{\sigma_p}) - \frac{1}{2\sigma_p^2} E_{q_\phi(z|x_i)}[(x-u_p)^2] + \frac{\sigma_q^2}{2\sigma_q^2} \\ &= log(\frac{\sigma_q}{\sigma_p}) - \frac{\sigma_q^2 + (\mu_q - \mu_p)^2}{\sigma_p^2} + \frac{\sigma_q^2}{2\sigma_q^2} \\ \sigma_p^2 = 1 \text{ and } \mu_p = 0: \\ &= \frac{1}{2}[1 + log(\sigma_q^2) - \sigma_q^2 + \mu_q^2]\end{aligned} \tag{2.2}$$

Now we can generalize it into a batch of data and the equation 2.1 become:

$$\xi = \sum_{j=1}^{J} \frac{1}{2}[1 + log(\sigma_q^2) - \sigma_q^2 + \mu_q^2] + \frac{1}{L} \sum_l E_{z\sim q_\phi(z|x_i)}[log(x_i|z^{i,l})] \tag{2.3}$$

J is the dimension of latent space and L is the number of random samples using reparametrization trick. Since the loss function in the pytorch is to minimize the loss, while ELBO is needed to be maximize. Thus, we can take negative sign and final equation:

$$L = -\sum_{j=1}^{J} \frac{1}{2}[1 + log(\sigma_q^2) - \sigma_q^2 + \mu_q^2] - \frac{1}{L} \sum_l E_{z\sim q_\phi(z|x_i)}[log(x_i|z^{i,l})] \tag{2.4}$$

Bonus: VAE objective with the Normalizing Flow prior.
We start with th basic loss of VAE:

$$KL(\tilde{p}(x)p(z|x)||q(x|z)q(z)) = \int\int \tilde{p}(x)p(z|x)log(\frac{p(z|x)p(x)}{q(x|z)q(z)}) \tag{2.5}$$

And now p(z—x) is different from the typical VAE, which is complex distribution generated from normalizing flow:

$$p(z|x) = \int \sigma(z - F_x(u))q(u)du \tag{2.6}$$

where u is standard Gaussian distribution and reversible. Then we apply into equation 2.5.

$$\begin{aligned}KL(\tilde{p}(x)p(z|x)||q(x|z)q(z)) &= \int\int\int \tilde{p}(x)q(u)\sigma(z - F_x(u))q(u)log(\frac{\tilde{p}(x)\int \sigma(z - F_x(u'))q(u')du'}{q(x|z)q(z)})dzdudx \\ &= \int\int \tilde{p}(x)q(u)log(\frac{\tilde{p}(x)q(u)}{q(x|F_x(u))q(F_x(u))|\det[\frac{\partial F_x(u)}{\partial u}]|})dudx\end{aligned} \tag{2.7}$$

In this way, we can get F-VAE loss. And sampling process: $u \sim q(u), z = F^{-1}(u), andx = G(z)$.

2.2 GAN

Generative adversarial network(GAN) is an another generative model. The objective function is shown as follow.

$$min_G max_D E_{x\sim P(real)}[log(D(x))] + E_{z\sim P(z)}[log(1 - D(G(z)))] \tag{2.8}$$

To illustrate equation 2.8 we can also combine with figure 2. $G(z)$ is generating part which corresponds to the second part and we want to minimize this part. And $D(-)$ represents the discriminator which is responsible for recognizing fake or true images. This is the normal objective equation. But if we look carefully of equation 2.8 $D(G(z))$ tends to be 0 during traning process and in this way $log(1 - D(G(z)))$ is always 0 which hinders to train. Thus, we can also use 2.9. In this assignment we use equation 2.8 as adversarial loss.

$$min_G max_D E_{x \sim P(real)}[log(D(x))] - E_{z \sim P(z)}[log(D(G(z)))] \tag{2.9}$$

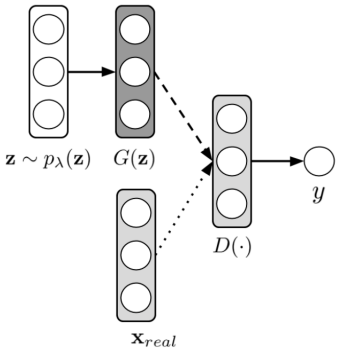


Figure 2: GAN.(GAN)

2.3 FID

Fréchet Inception Distance(FID) is a method to evaluate the generated images and true images. FID calculates the distance between the real picture and the fake picture at the feature level. And the equation:

$$FID = \| \mu_r - \mu_g \|^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2}) \tag{2.10}$$

whereμ_r, μ_g are mean of features for real images and generated images. And Σ_r, Σ_g are covariance matrix of features for real images and generated images

3 Experiments

We separate the whole project into two big parts according to the MNIST and Imagenette dataset.

3.1 MNIST

For MNIST dataset, we use 50000 images for training, 10000 images for validation and 10000 images for test. Besides, the original shape of images is 1*28*28 and we need to reshape into 1*32*32.

The structure of VAE is shown as follow. The image dimension is 1024 and latent dimension is 100. Other parameters: Learning rate:0.0001, Epoch:50 and Batch size: 64.

$$\text{Enconder:} \begin{cases} \text{Linear1(1024,1024), LeakyReLu(0.2)} \\ \text{Linear2(1024,512), LeakyReLu(0.2)} \\ \text{Linear3(512,256), LeakyReLu(0.2)} \\ \text{Linear4(256,100), LeakyReLu(0.2)} \\ \text{nn.Tanh()} \end{cases} \tag{3.1}$$

$$\text{Deconder:} \begin{cases} \text{Linear1(100,256), ReLu()} \\ \text{Linear2(256,512), ReLu()} \\ \text{Linear3(512,1024), ReLu()} \\ \text{Linear4(1024,1024), Sigmoid()} \end{cases} \tag{3.2}$$

For GAN the structure shows as follow. We also use 1024 for image dimension and 100 for latent dimension. Other parameters: Learning rate:0.0002, Epoch:80 and Batch size: 100.

$$\text{Discriminator:} \begin{cases} \text{Linear1(1024,1024), LeakyReLu(0.2)} \\ \text{Linear2(1024,512), LeakyReLu(0.2)} \\ \text{Linear3(512,256), LeakyReLu(0.2)} \\ \text{Linear4(256,100), LeakyReLu(0.2)} \\ \text{nn.Tanh()} \end{cases} \tag{3.3}$$

$$\text{Generator:} \begin{cases} \text{Linear1(100,256), ReLu()} \\ \text{Linear2(256,512), ReLu()} \\ \text{Linear3(512,1024), ReLu()} \\ \text{Linear4(1024,1024), Sigmoid()} \end{cases} \tag{3.4}$$

3.2 Imagenette

There are total 10 calsses for Imagenette dataset. And we use 8000 images for training, 1450 images for validation and 3000 images for test.

For VAE we use Learning rate:0.0001, Epoch:50 and Batch size: 80. Besides, the latent dimension is 100. Since this dataset has three channel, we change the whole structure. The structure of VAE shows as follow:

$$\text{Enconder:} \begin{cases} \text{Conv2d(3,16), LeakyReLu(0.2)} \\ \text{Conv2d(16,32), LeakyReLu(0.2)} \\ \text{Conv2d(32,32), LeakyReLu(0.2)} \\ \text{Conv2d(32,64), LeakyReLu(0.2)} \\ \text{Conv2d(64,100), LeakyReLu(0.2)} \\ \text{nn.Tanh()} \end{cases} \tag{3.5}$$

$$\text{Deconder:} \begin{cases} \text{ConvTranspose2d(100,64), ReLu()} \\ \text{ConvTranspose2d(64,32), ReLu()} \\ \text{ConvTranspose2d(32,32), ReLu()} \\ \text{ConvTranspose2d(32,16), ReLu()} \\ \text{ConvTranspose2d(16,3), ReLu()} \\ \text{nn.Sigmoid()} \end{cases} \tag{3.6}$$

The structure of GAN for imagenette is the same as VAE. And we use Latent dimension 100, Learning rate: 0.00005, Epoch: 150 and Batch size:50.

$$\text{Discriminator:} \begin{cases} \text{Conv2d(3,16), LeakyReLu(0.2)} \\ \text{Conv2d(16,32), LeakyReLu(0.2)} \\ \text{Conv2d(32,32), LeakyReLu(0.2)} \\ \text{Conv2d(32,64), LeakyReLu(0.2)} \\ \text{Conv2d(64,100), LeakyReLu(0.2)} \\ \text{nn.Tanh()} \end{cases} \tag{3.7}$$

$$\text{Generator:} \begin{cases} \text{ConvTranspose2d(100,64), ReLu()} \\ \text{ConvTranspose2d(64,32), ReLu()} \\ \text{ConvTranspose2d(32,32), ReLu()} \\ \text{ConvTranspose2d(32,16), ReLu()} \\ \text{ConvTranspose2d(16,3), ReLu()} \\ \text{nn.Sigmoid()} \end{cases} \tag{3.8}$$

4 Results and discussion

4.1 MNIST

Figure 3 shows the loss and FID plots. Firstly, we can observe that for both training and validation dataset the loss are not fully converge. We stop at 50 epochs because we observe the signal of overfitting which is validation loss is greater than training loss. FID score drops quickly in the beginning and fluctuates around 95.

Figure 3: VAE training loss, validation loss and FID in 50 epochs.

Furthermore, figure 4 shows loss and FID plots of GAN. Loss of Generator drops with increasing number of epochs for both validation and training datasets. And loss of Discriminator increases as epochs increases. FID of GAN is larger than VAE and we can observe that the result are quite unstable, which illustrate that GAN is indeed difficult to train.

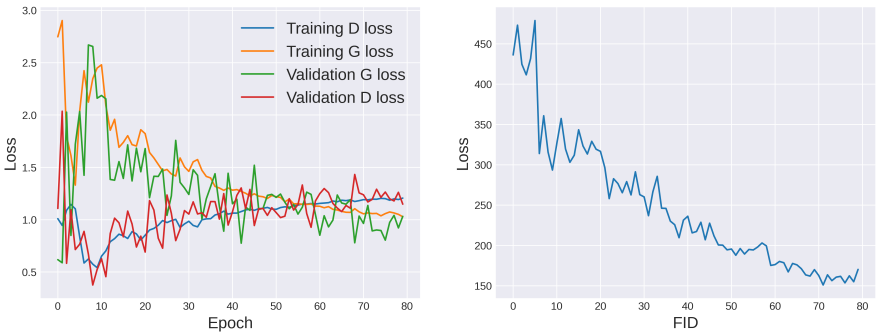
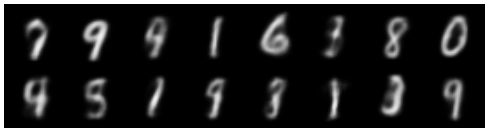
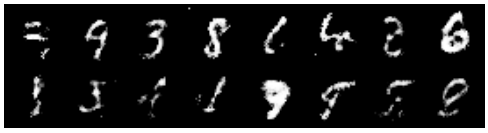


Figure 4: GAN training loss, validation loss and FID in 80 epochs.

Figure 5 shows 16 random samples from VAE and GAN respectively. We can see that the samples from two models are quite different. Samples from VAE have more complete shape while the quality of image is blurry. However, samples from GAN are clearer and sharper but incomplete. The possible explanation is the independence assumption of the samples given latent variables of VAE. Another explanation is because of the number of bottleneck. Increasing the number of latent dimension may help to generate clearer images.



(a) 16 samples from VAE of MNIST



(b) 16 samples from GAN of MNIST

Figure 5: Samples from VAE and GAN

Figure 6 shows 10 interpolations samples from latent space for both VAE and GAN models. We use same 10 points to generate the samples. It is obvious that the samples from VAE are still blurry while samples from GAN are sharp and clear. The transition from VAE is not clear, which is from eight to nine. While the transition of GAN is quite clear. We can see it is from five to nine.

(a) Samples from 10 interpolation points of VAE

(b) Samples from 10 interpolation points of GAN

Figure 6: Samples from VAE and GAN

4.2 Imagenette

Firstly, loss and FID plots of VAE are shown as figure 7. We can see the both training loss and validation loss are quite large. And there is clear overfitting after 20 epochs. The result of FID is not decedent, which is stuck between 420 and 470. The possible explanation is that imagenette has three dimensions and the images are 3*160*160. When we reshape images into 3*299*299 and feed into inspection V3 model it causes many errors. Another explanation is that because of our model and we need to tune all the parameters.

Figure 7: VAE training loss, validation loss and FID in 80 epochs.

Furthermore, figure 8 shows the loss and FID plots. The main trend of loss of Generator and Discriminator is the same as MNIST. But we can see the loss of D are smaller than G in the end for both training dataset and validation dataset. For FID score it shows the same performance as VAE, which is unstable. The scores fluctuate around 490.

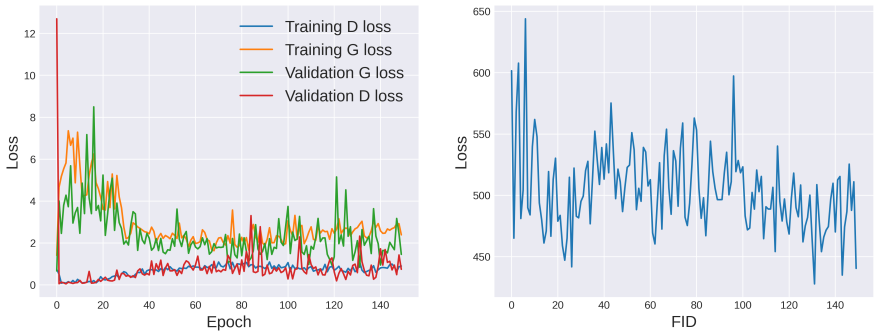
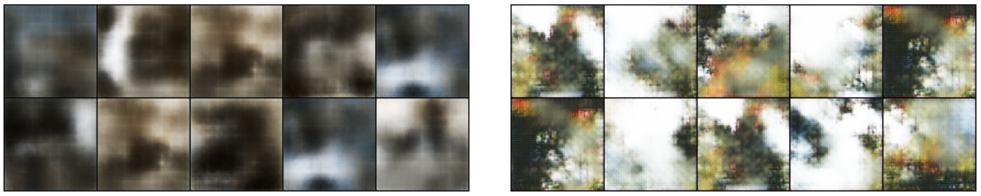


Figure 8: GAN training loss, validation loss and FID in 150 epochs.

Additionally, we generate 10 samples from two models. We find out the samples from both models are not decent. It is obvious that samples from VAE are vague and dark. While samples from GAN are sharp and colorful. But both models can not generate reasonable samples. And we do several improvement and will discuss in the next part.

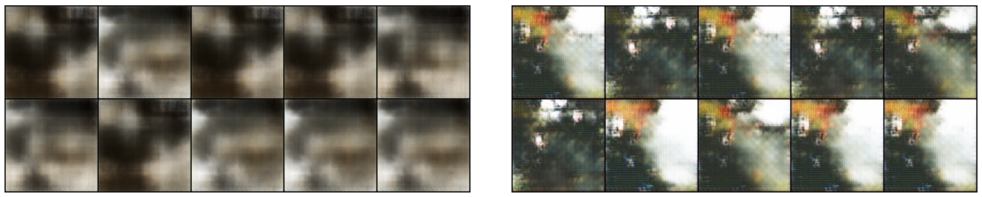


(a) 10 samples from VAE of Imagenette

(b) 10 samples from GAN of Imagenette

Figure 9: Samples from VAE and GAN

Then we interpolate 10 points from latent space and the results show as figure 10. We can see the transition is not clear for both models. The brown shade part in VAE becomes larger. White part in GAN also becomes larger.



(a) Samples from 10 interpolation points of VAE (b) Samples from 10 interpolation points of GAN

Figure 10: Samples from VAE and GAN

4.3 Improvement

Since our results of model are not decent, we need to do some improvement. Firstly we increase the features' dimension for both two models and we also add noises in the target label for GAN.

For the VAE the performance has no improvement. However, we surprisingly observe that the FID drops around 280 as figure 11 shows.

Figure 11: Improved FID for GAN.

Figure 12 shows both 10 random samples and results of interpolation. We can see the samples are quite decent now, which are like Picasso abstract painting.

(a) 10 random samples from improved GAN (b) Samples from 10 interpolation points of improved GAN

Figure 12: Improved GAN

5 Appendix

References

Gan. https://canvas.vu.nl/courses/50054/files/2970589/download?wrap=1.
Vae. https://canvas.vu.nl/courses/50054/files/2970667/download?wrap=1.