ECS795P Deep Learning and Computer Vision, 2022

Course Work 2:
Unsupervised Learning by Generative Adversarial Network

Wenshuo Li **210276415**

The answers are highlighted in red

1. **What is the difference between supervised learning & unsupervised learning in image classification task? (10% of CW2)**

   The label of data in supervised learning is given during training, for the data of unsupervised learning, no label is given during training.

   The goal of supervised learning is to learn a mapping function $f_\theta: x \to y$, the goal of unsupervised learning is to learn some underlying structure of data.

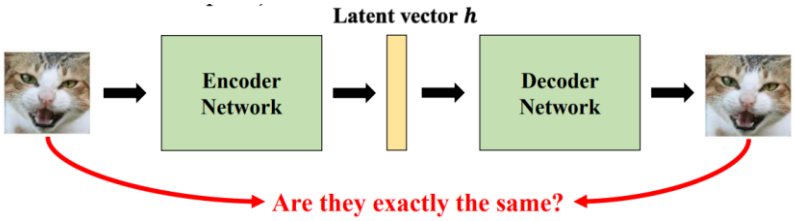
   For image classification, the objective of supervised learning is to learn f_θ to predict image classes by minimizing a cross entropy loss: $L(x, y; \theta) = -\sum y_i \log(p(c_i|x_i))$

   The unsupervised learning can't predict the classes of data, but it can divide data into different classes(clustering).

2. **What is the difference between an auto-encoder and a generative adversarial network considering (1) model structure; (2) optimized objective function; (3) training procedure on different components. (10% of CW2)**

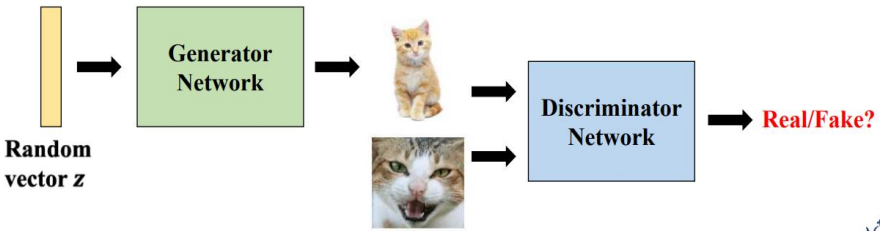
   (1) **model structure**:

   The structure of auto-encoder consists of an encoder network(map input x to latent code h) and a decoder network(map h to reconstructed input r).



   The structure of GAN contains a generator network and a discriminator network, Generator Network: learn to fool the discriminator by generating real-looking images (as real as possible to the real images), Discriminator Network: learn to differentiate between the generated images (fake) and the real images (real).

Auto-Encoder uses an image as an input to generate the corresponding output that resembles the input. GANs aim to generate samples from a simple distribution, i.e. use Gaussian random noise as input z.



(2) optimized objective function:

Auto-Encoder: to minimize the reconstruction mean squared error:

$$L(x, y; \theta) = -\frac{1}{M}\sum_{i=1}^{M} ||x_i - r_i||^2$$

GANs: Two-player minimax game:

$$\min_{\theta_g} \max_{\theta_d} [\mathbb{E}_{x \sim p_{\text{data}}} \log \underbrace{D_{\theta_d}(x)}_{\substack{\text{Discriminator's output} \\ \text{for real data } x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\substack{\text{Discriminator's output} \\ \text{for generated fake data } G_{\theta_g}(z)}})]$$

(3) training procedure on different components:

Auto-Encoder: minimize the error by gradient decent:

1. Set $\Delta W^{(l)} := 0$, $\Delta b^{(l)} := 0$ (matrix/vector of zeros) for all l.
2. For $i = 1$ to m,
   a. Use backpropagation to compute $\nabla_{W^{(l)}} J(W, b; x, y)$ and $\nabla_{b^{(l)}} J(W, b; x, y)$.
   b. Set $\Delta W^{(l)} := \Delta W^{(l)} + \nabla_{W^{(l)}} J(W, b; x, y)$.
   c. Set $\Delta b^{(l)} := \Delta b^{(l)} + \nabla_{b^{(l)}} J(W, b; x, y)$.
3. Update the parameters:

$$W^{(l)} = W^{(l)} - \alpha \left[\left(\frac{1}{m} \Delta W^{(l)} \right) + \lambda W^{(l)} \right]$$

$$b^{(l)} = b^{(l)} - \alpha \left[\frac{1}{m} \Delta b^{(l)} \right]$$

GANs: alternative updates on discriminator and generator, 1. Take the gradient ascent on the discriminator 2. Take the gradient descent on the generator:

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

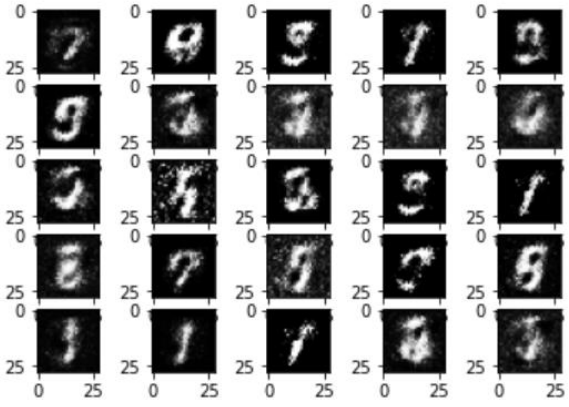
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

3. **How is the distribution $p_g(x)$ learned by the generator compared to the real data distribution $p_{data}(x)$ when the discriminator cannot tell the difference between these two distributions? (10% of CW2)**

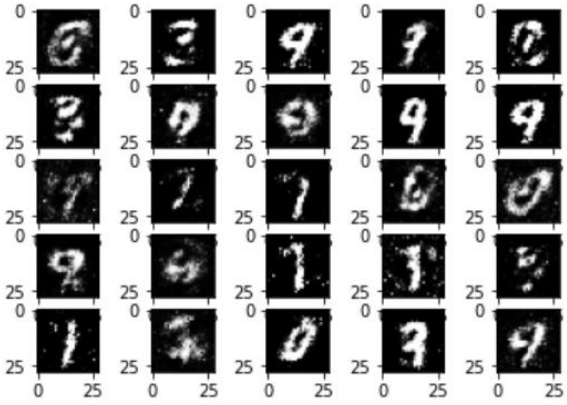
   After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g(x) = p_{data}(x)$. The discriminator is unable to differentiate between the two distributions, i.e. D(x) = $p_{data}(x)/(p_{data}(x) + p_g(x)) = 1/2$.

4. **Show the generated images at the 10th epoch, the 20th epoch, the 50th epoch, the 100th epoch by using the architecture required in Guideline. (10% of CW2)**

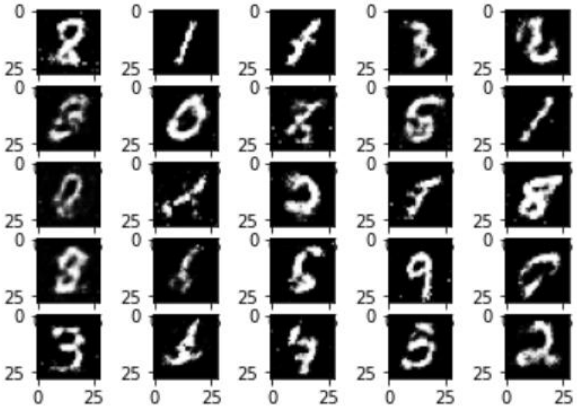
```
Epoch 10 of 100 with 21.12 s
Generator loss: 2.11119426, Discriminator loss: 0.71611321
```



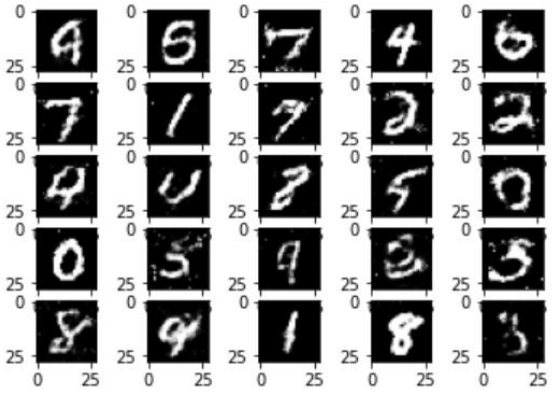
```
Epoch 20 of 100 with 21.29 s
Generator loss: 1.71705687, Discriminator loss: 0.88420451
```



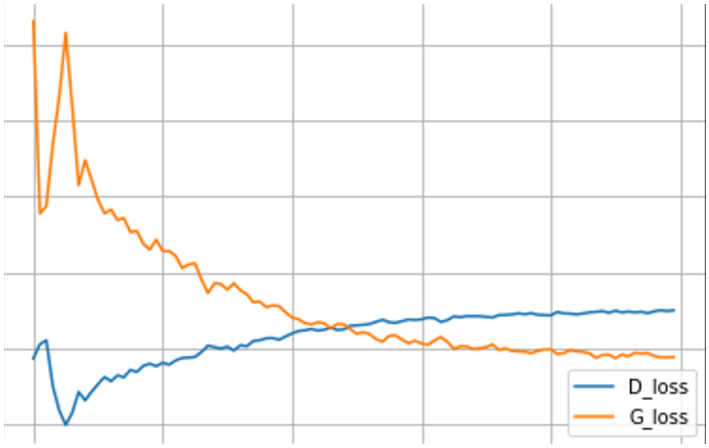
```
Epoch 50 of 100 with 22.94 s
Generator loss: 1.12824205, Discriminator loss: 1.14937483
```



```
Epoch 100 of 100 with 21.38 s
Generator loss: 0.94433404, Discriminator loss: 1.25189549
```



5. **Plot the loss curve during training.** **(10% of CW2)**


D_loss
G_loss