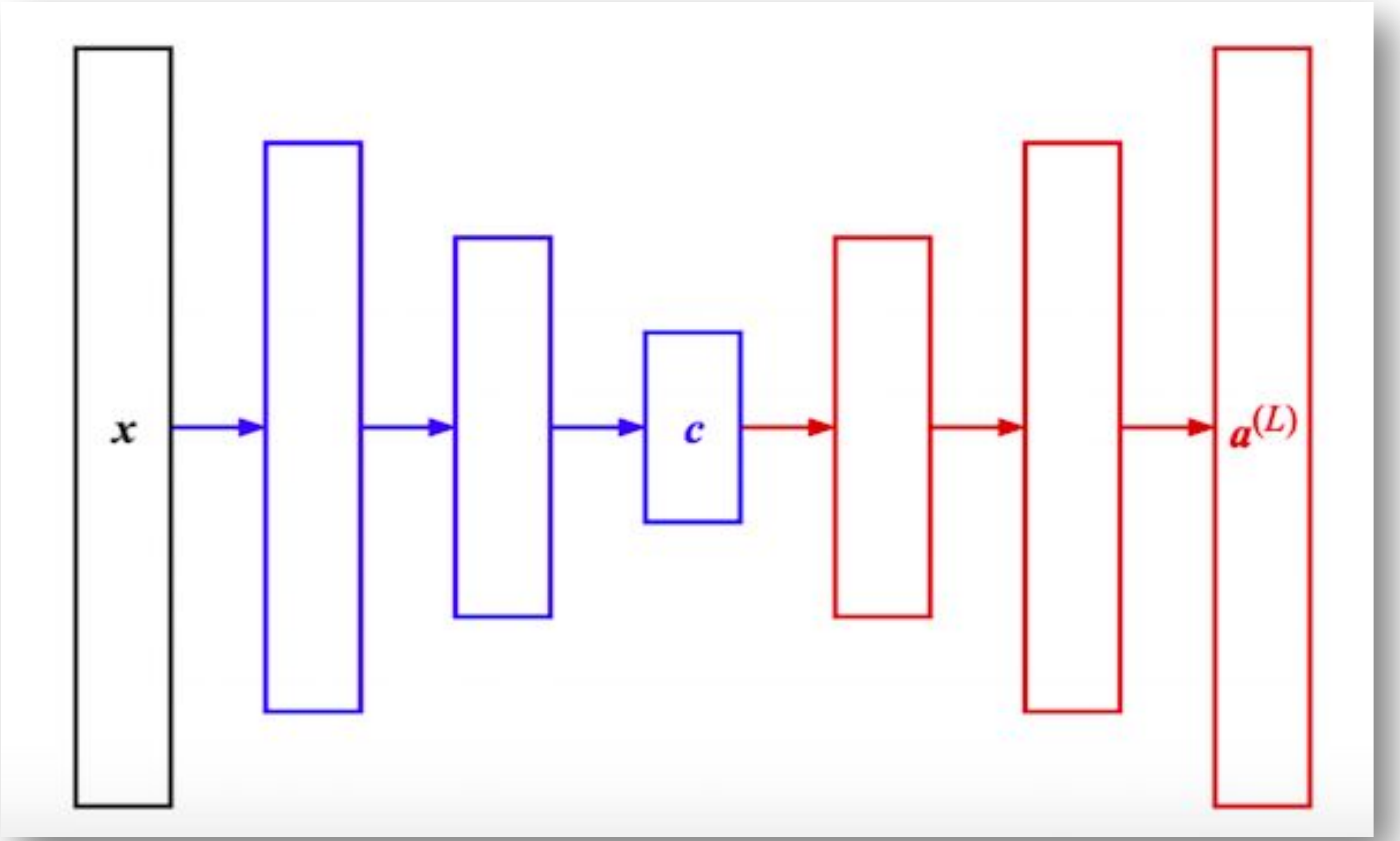
Lab 13
Autoencoder & GANs

DataLab

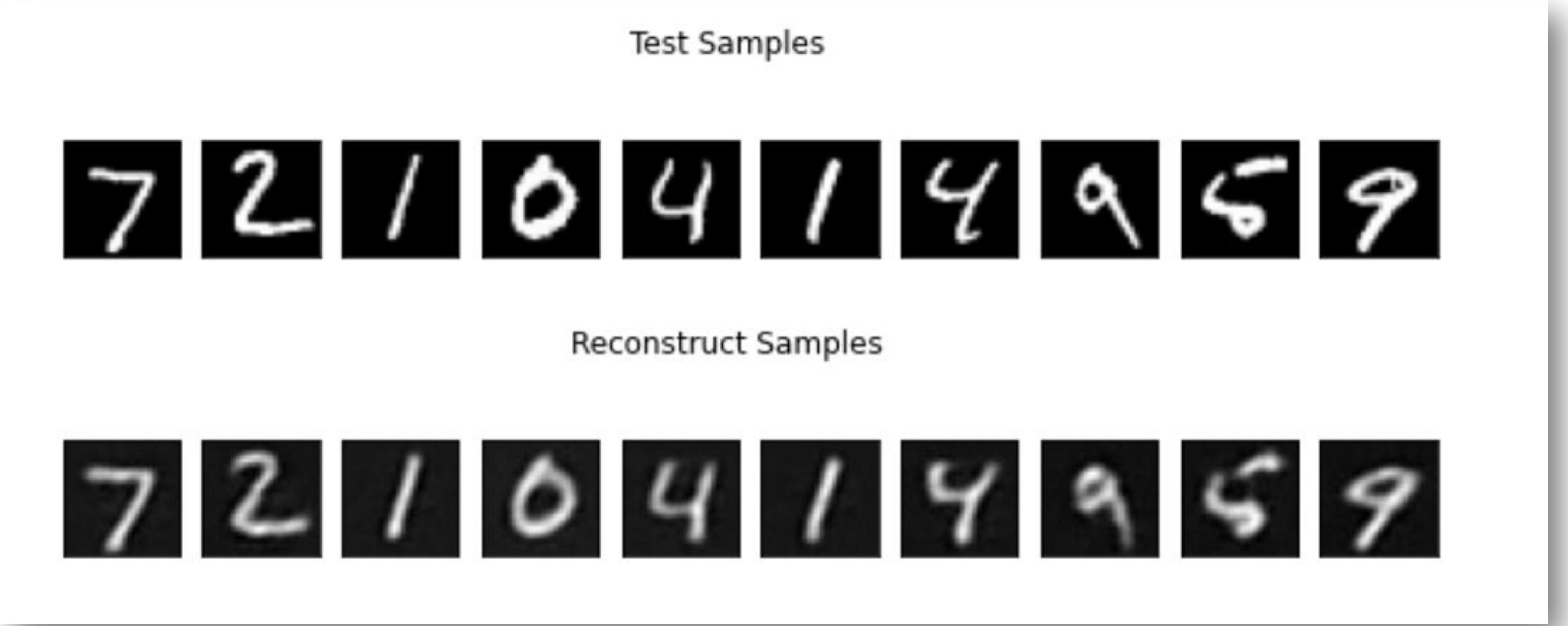
Department of Computer Science,
National Tsing Hua University, Taiwan

13-1 Autoencoder



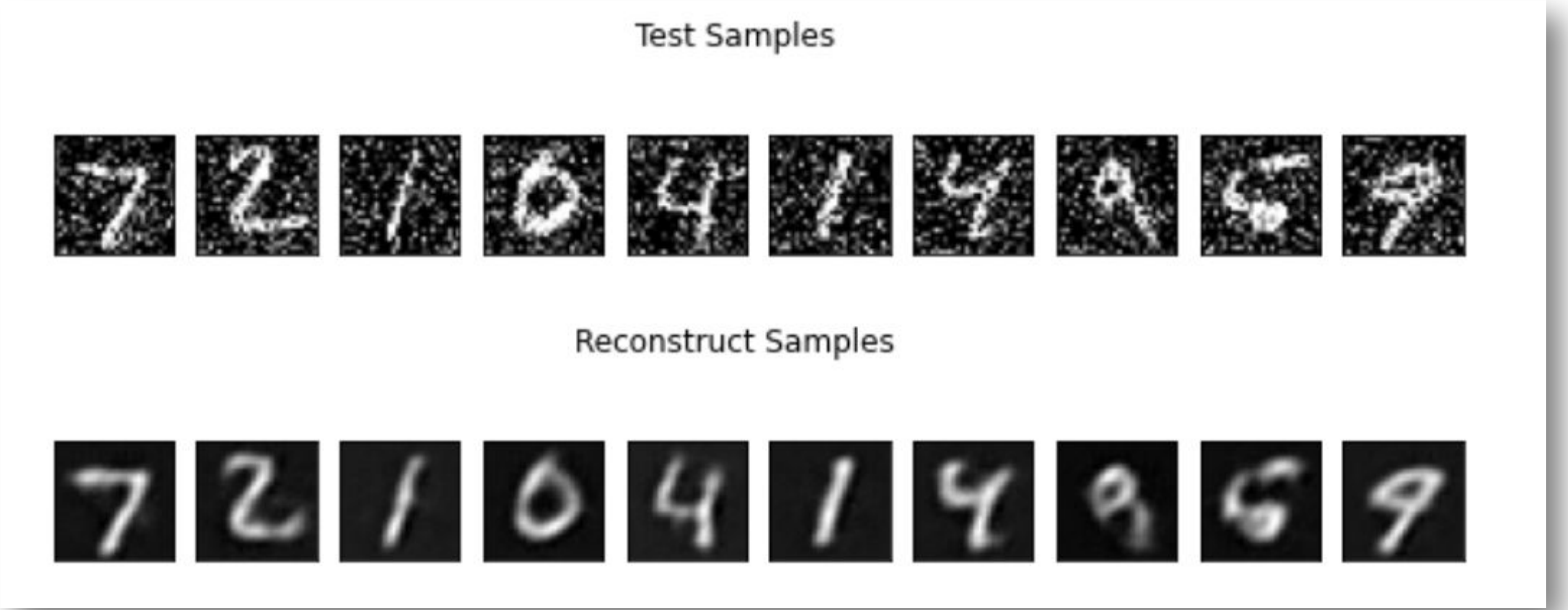
Autoencoder

- Autoencoder without noise



Autoencoder

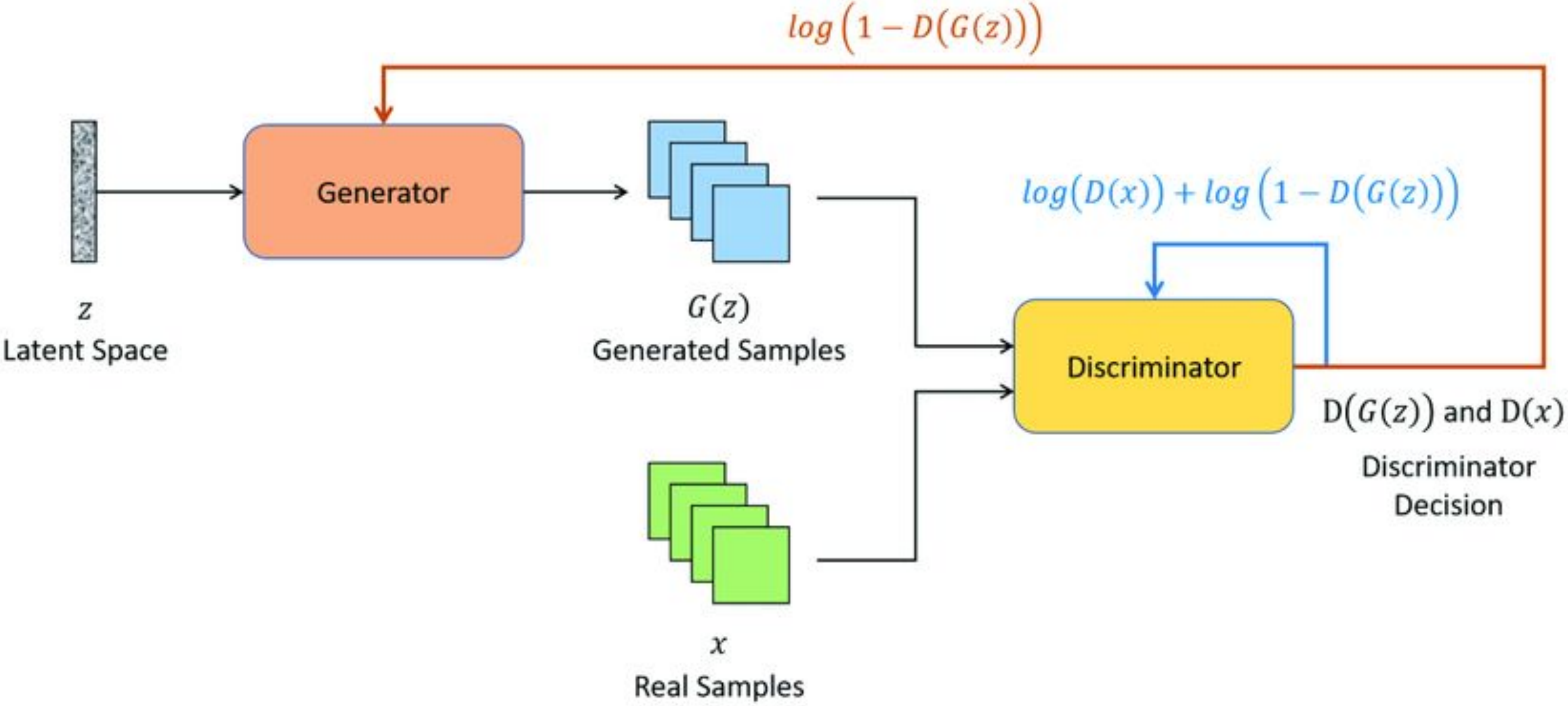
- Autoencoder with noise



13-2 GAN Outline

- Reviewing GAN Structure
- Loss Functions
- WGAN
- WGAN-GP (improved WGAN)

Architecture of Generative Adversarial Network (GAN)



Loss Functions

- Minimax Loss:
  - For D: maximize $\boxed{E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))]}$
  - For G: minimize $\cancel{E_x[\log(D(x))]} + \boxed{E_z[\log(1 - D(G(z)))]}$

GAN's Algorithm

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**

Fixed G, Train D

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ 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

Fixed D, Train G

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

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

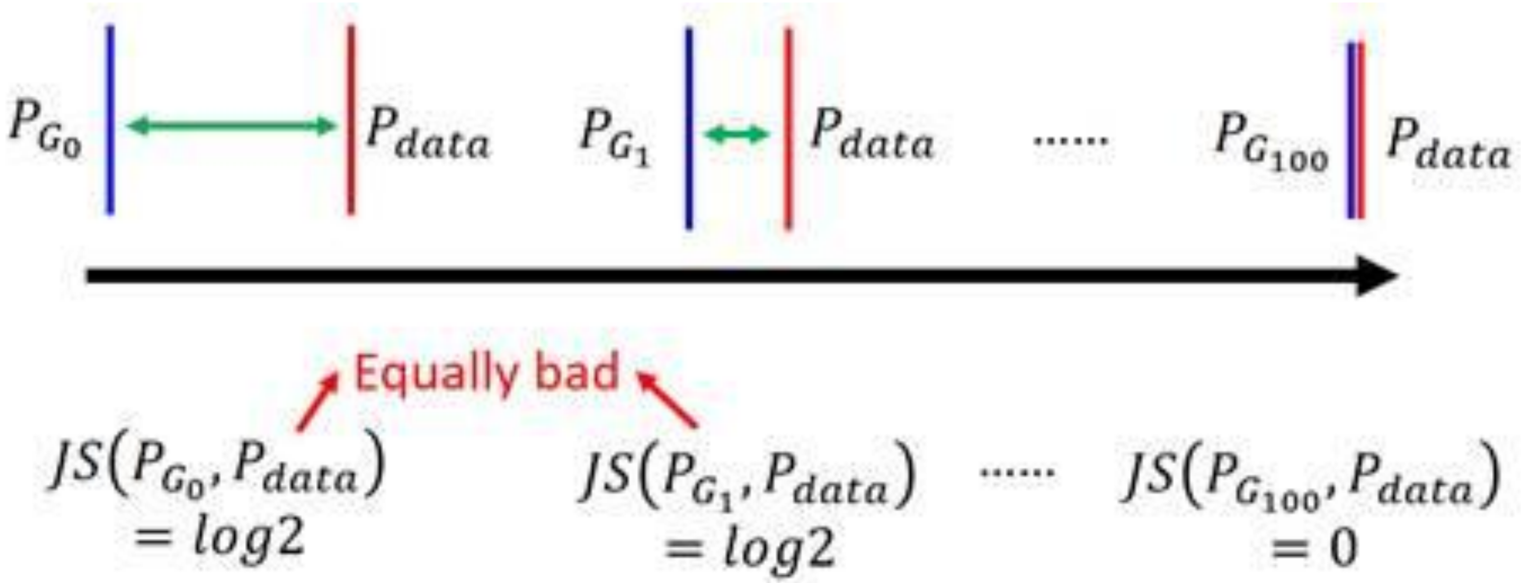
end for

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

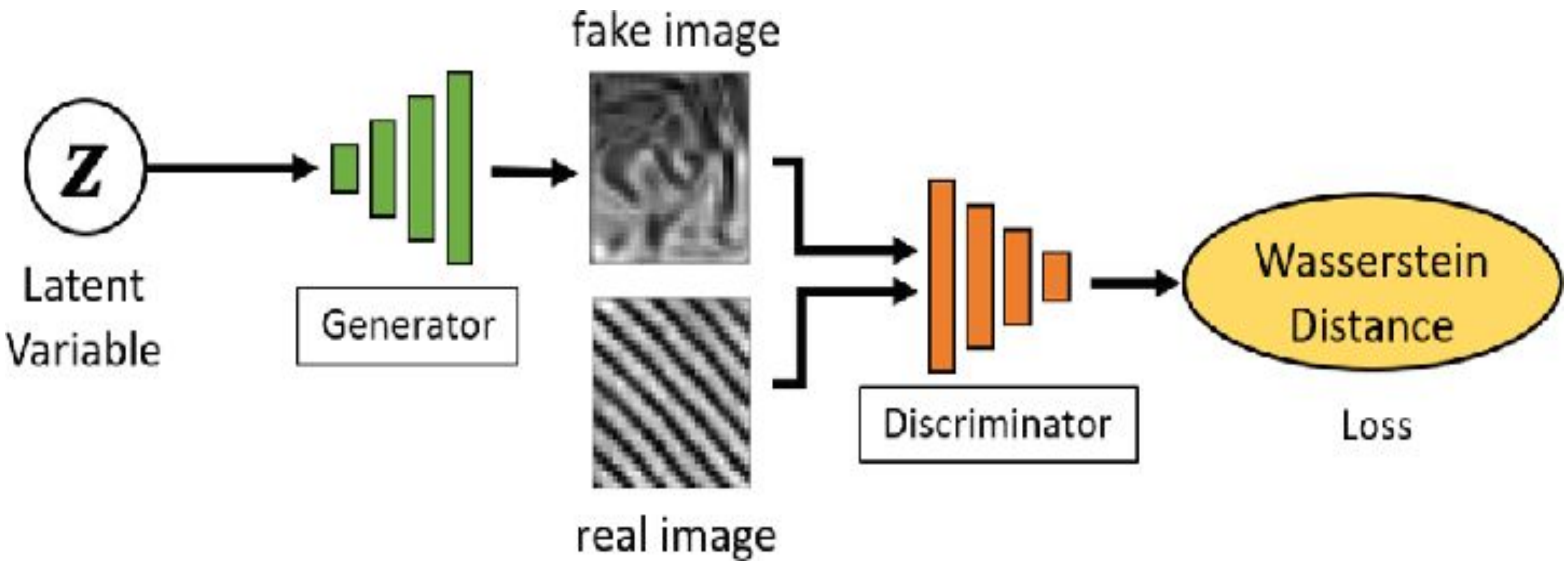
Gradient Vanishing Issue in Generator's Loss

$$loss\ of\ G = \max\left(E_{x\sim P_{data}}[\log(D(x))] + E_{\tilde{x}\sim P_G}[\log(D(\tilde{x}))]\right)$$

$$\cong -2\log(2) + \boxed{2D_{JS}(P_{data}||P_G)}$$



Wasserstein GAN



Loss Function of Wasserstein GAN

- Minimax Loss:
  - For D: maximize $E_x[\boxed{\log(D(x))}] + E_z[\boxed{\log(1 - D(G(z)))}]$
  - For G: minimize $E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))]$
- Wasserstein Loss:
  - For D: maximize $E_{x\sim P_x}[\boxed{f_w(x)}] - E_{z\sim P_z}[\boxed{f_w(G(z))}]$
  - For G: minimize $E_{x\sim P_x}[f_w(x)] - E_{z\sim P_z}[f_w(G(z))]$

$f_w \in K - \mathit{Lipschitz\ functions}$ for some K

Loss Functions

- Lipschitz continuity: a function $f: X \rightarrow Y$ is called **Lipschitz continuous** if there exists a real constant $K \geqq 0$ such that, for all x_1 and x_2 in X

$$d_Y(f(x_1), f(x_2)) \le K d_X(x_1, x_2)$$

- How to make the discriminator Lipschitz continuous?

Weight clipping – clip all weights in f_w into a certain range.

WGAN Algorithm

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**
  for $t = 0, ..., n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \boxed{\nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]}$
    $w \leftarrow w + \alpha \cdot \boxed{\text{RMSProp}(w, g_w)}$
    $w \leftarrow \boxed{\text{clip}(w, -c, c)}$
  end for
  Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while

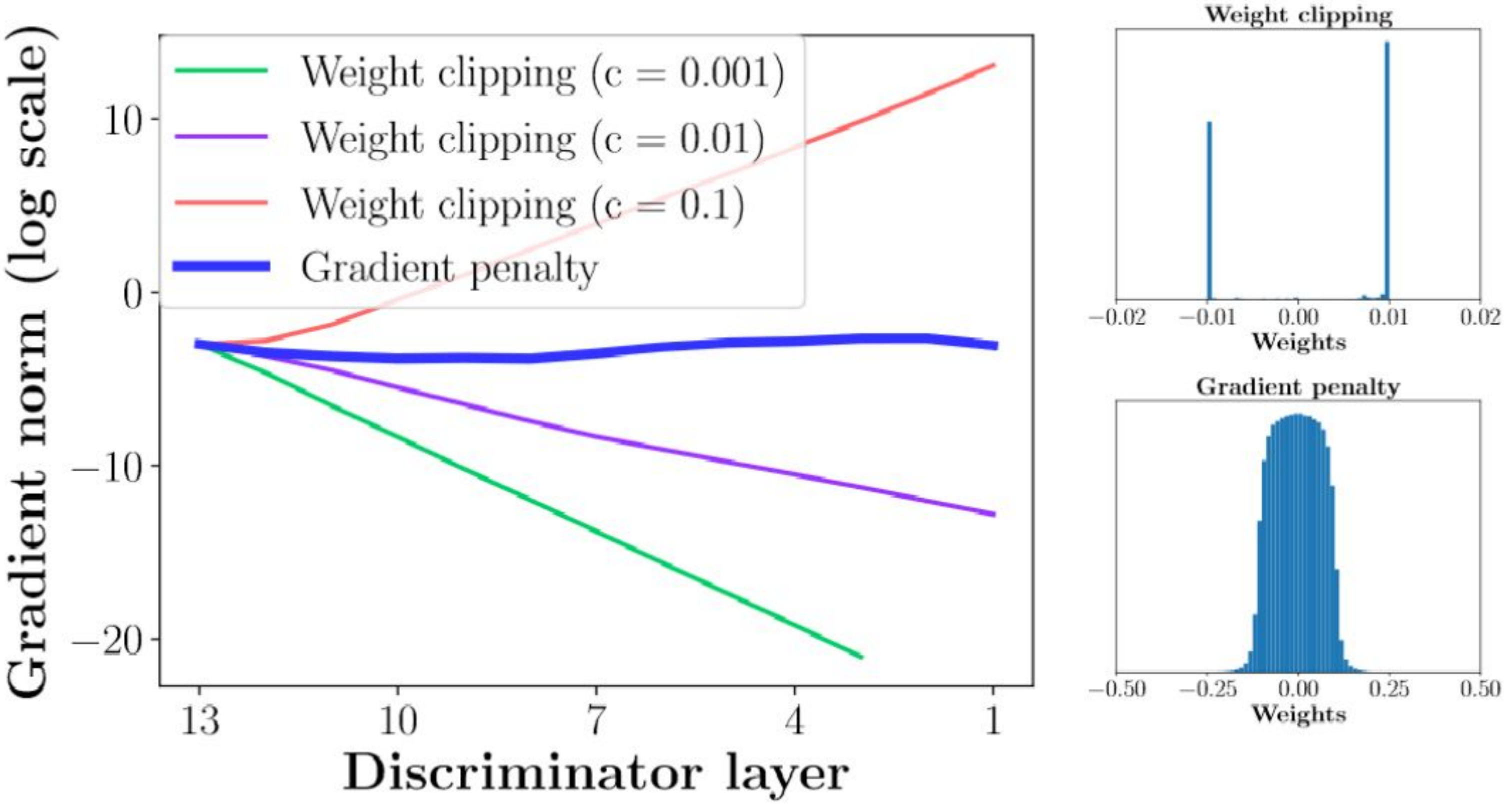
Main Differences Between WGAN and GAN

The WGAN, compared to the first form of the original GAN, only has four changes:

1. The last layer of the discriminator removes the sigmoid.
2. The loss for both the generator and discriminator does not take the logarithm.
3. After updating the parameters of the discriminator, their absolute values are clipped to not exceed a fixed constant c.
4. Do not use momentum-based optimization algorithms (including momentum and Adam); RMSProp is recommended, SGD is also acceptable.

Clipping Issue

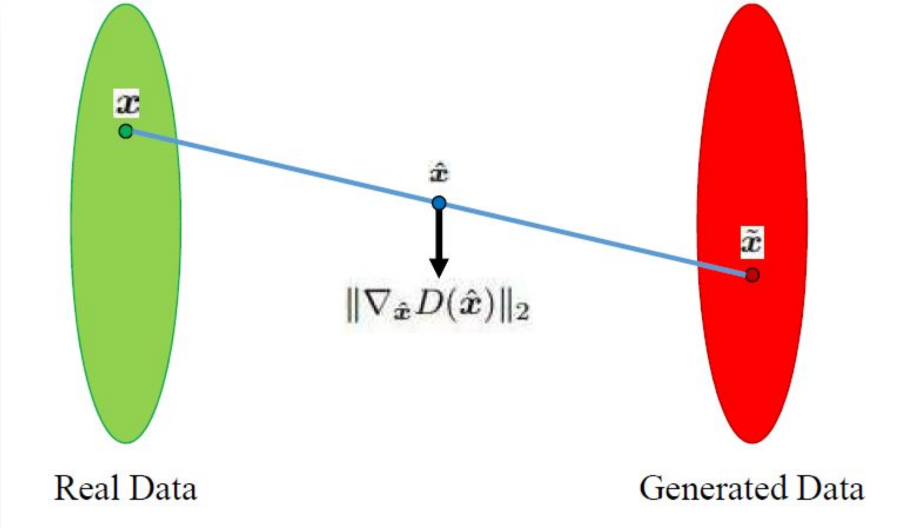
- In comparison with WGAN



WGAN-GP

- Instead of weight clipping, adding gradient penalty can also achieve Lipchitz continuity.

$$L = \underbrace{\underset{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}{\mathbb{E}} \left[D(\tilde{\boldsymbol{x}}) \right] - \underset{\boldsymbol{x} \sim \mathbb{P}_r}{\mathbb{E}} \left[D(\boldsymbol{x}) \right]}_{\text{Original critic loss}} + \underbrace{\lambda \underset{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}{\mathbb{E}} \left[(\| \nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}}) \|_2 - 1)^2 \right]}_{\text{Our gradient penalty}} .$$



WGAN-GP's Algorithm

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

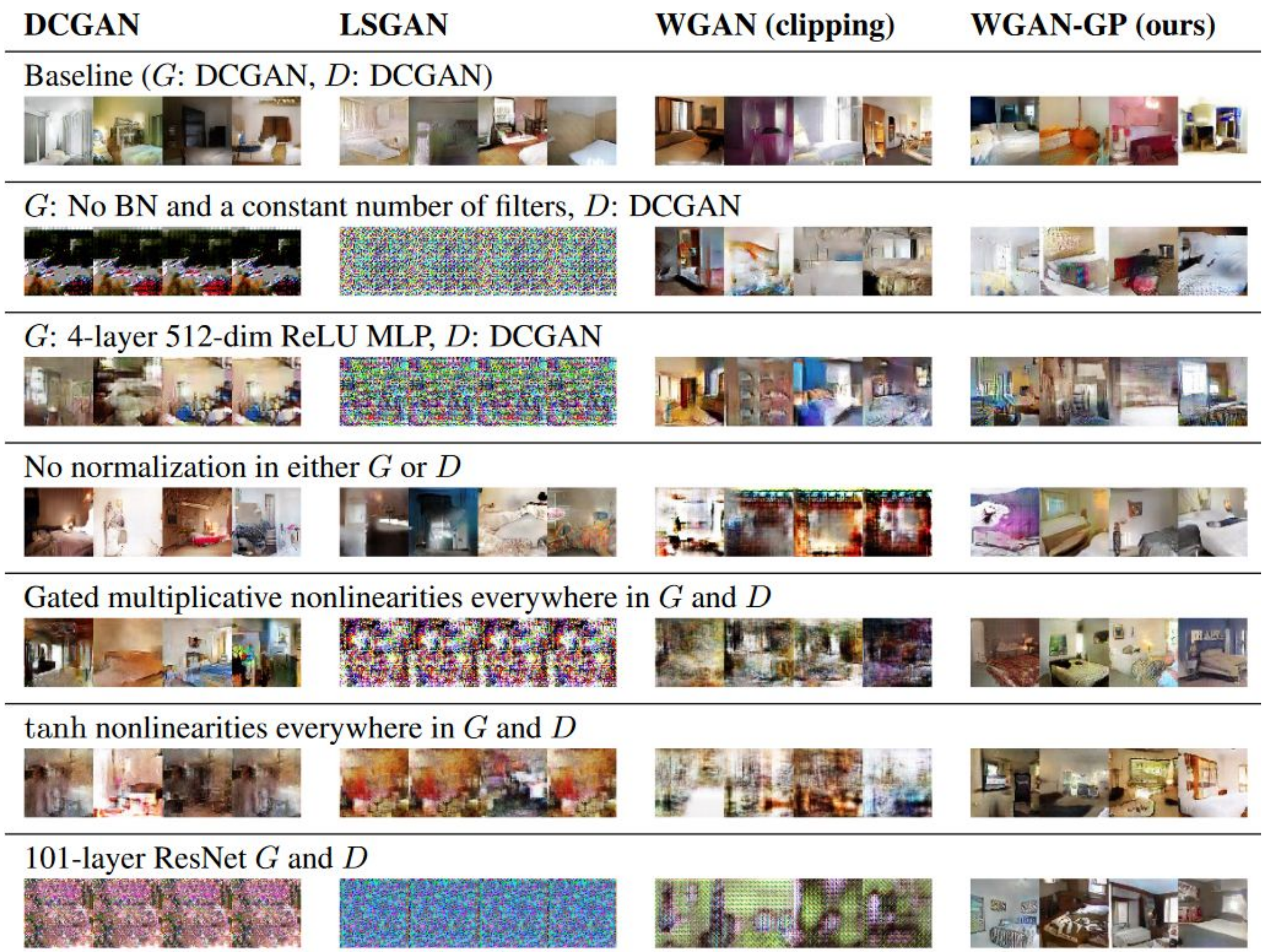
Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

```
while θ has not converged do
    for t = 1, ..., n_critic do
        for i = 1, ..., m do
            Sample real data x ~ P_r, latent variable z ~ p(z), a random number ε ~ U[0, 1].
            x̃ ← G_θ(z)
            x̂ ← εx + (1 − ε)x̃
            L^(i) ← D_w(x̃) − D_w(x) + λ(‖∇_x̂ D_w(x̂)‖_2 − 1)^2
        end for
        w ← Adam(∇_w (1/m) Σ_{i=1}^m L^(i), w, α, β_1, β_2)
    end for
    Sample a batch of latent variables {z^(i)}_{i=1}^m ~ p(z).
    θ ← Adam(∇_θ (1/m) Σ_{i=1}^m −D_w(G_θ(z)), θ, α, β_1, β_2)
end while
```

In math form:

- Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a $\boxed{\text{random number } \epsilon \sim U[0,1]}$.
- $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
- $\boxed{\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1-\epsilon)\tilde{\boldsymbol{x}}}$
- $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \boxed{\lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2}$
- $w \leftarrow \boxed{\text{Adam}(}\nabla_w \frac{1}{m}\sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$
- $\theta \leftarrow \boxed{\text{Adam}(}\nabla_\theta \frac{1}{m}\sum_{i=1}^m -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$

WGAN-GP


DCGAN
LSGAN
WGAN (clipping)
WGAN-GP (ours)
Baseline (G: DCGAN, D: DCGAN)
G: No BN and a constant number of filters, D: DCGAN
G: 4-layer 512-dim ReLU MLP, D: DCGAN
No normalization in either G or D
Gated multiplicative nonlinearities everywhere in G and D
tanh nonlinearities everywhere in G and D
101-layer ResNet G and D

Assignment

- Assignment requirements
  - Implementation of Improved WGAN (WGAN-GP) and train on CelebA.
  - Build dataset to read and resize image to 64×64 for training
  - Training loop(s) / routine(s) for GAN. Pre-trained models are not allowed.
  - Show at least 8×8 animated image of training and some best generated samples.
  - Draw the curve of discriminator loss and generator loss during training process in a single image.
  - Brief report about what you have done.

Assignment

- Submission
  - Upload notebook and attachments to google drive and submit the link to eeclass.
  - Your notebook should be named after “Lab13_{student id}.ipynb”.
  - Deadline : 2022/12/14 23:59