GANs in action

WGAN

한병찬

Contents

1. Problems of GAN
2. WGAN
3. WGAN-GP

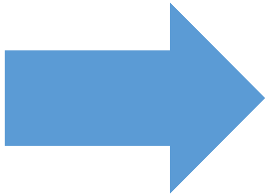
Problems of GAN

Classification이 가능해야한다. → Classifier에 넣었을 때 일정 점수 이상 나와야한다.

생성된 샘플이 원본 데이터의 모든 class를 포함해야한다. → Mode Collapse가 일어나면 안 된다.

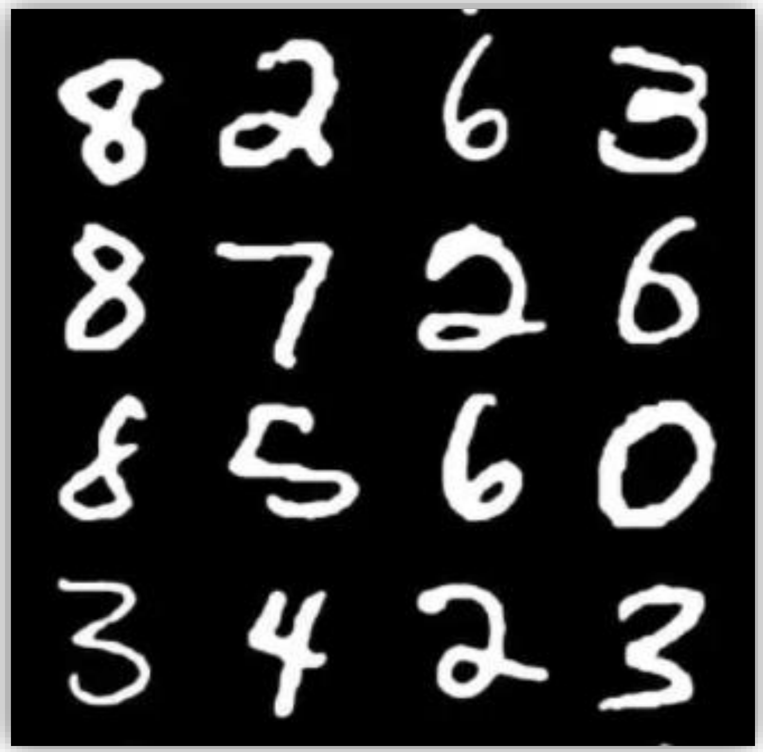
Problems of GAN

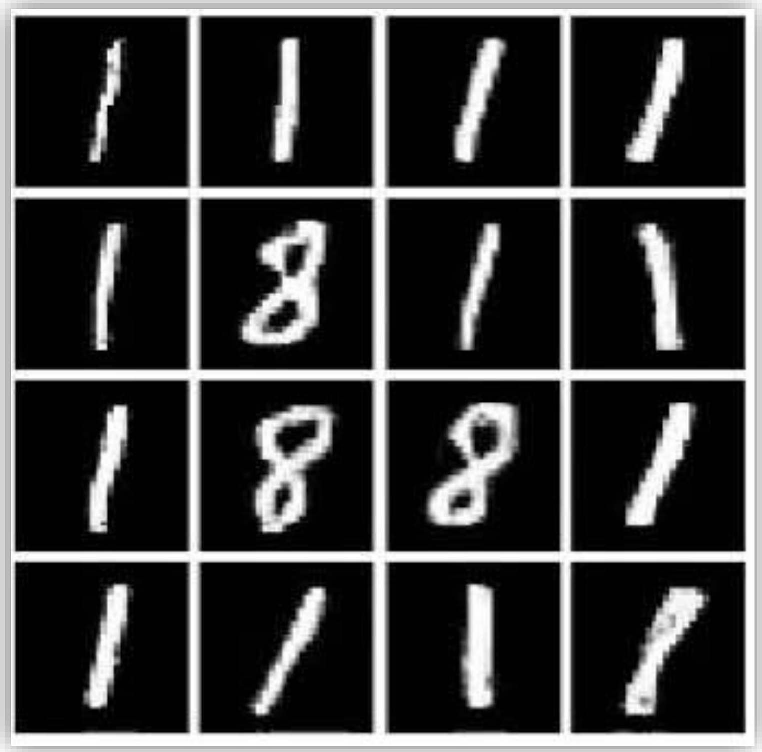
Overgeneralization



Problems of GAN

Mode Collapse

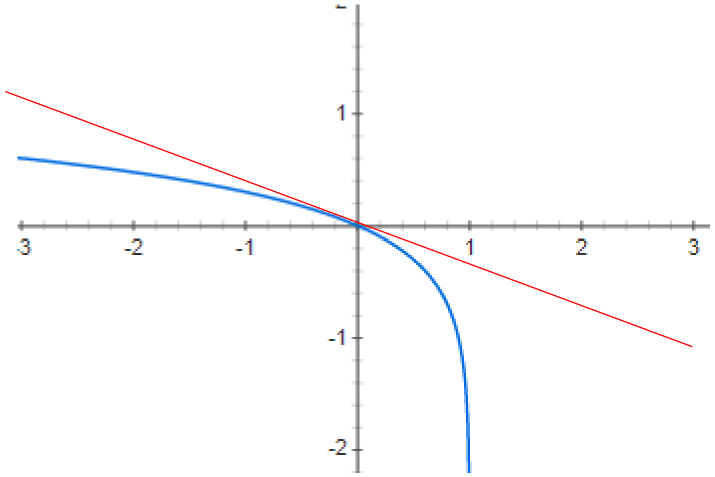


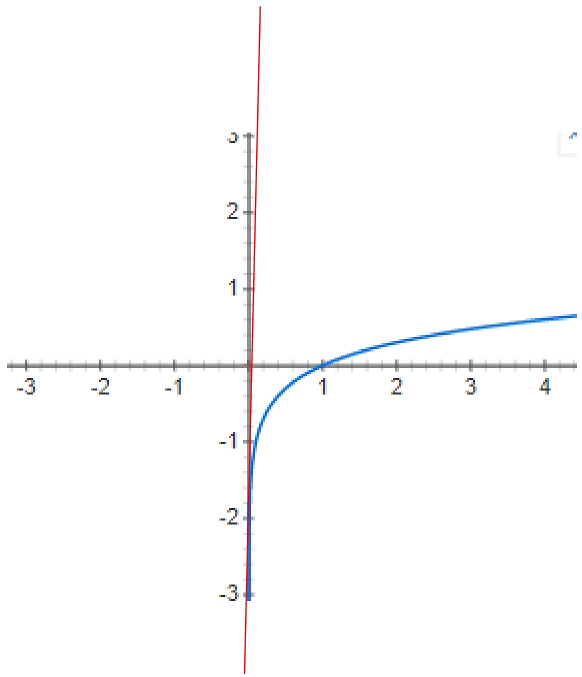


Problems of GAN

Slow Convergence

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \boxed{\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].} \longrightarrow \boxed{\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(D(G(\boldsymbol{z})))].}$$





WGAN

- The *Total Variation* (TV) distance

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)| \ .$$

- The *Kullback-Leibler* (KL) divergence

$$KL(\mathbb{P}_r \| \mathbb{P}_g) = \int \log\left(\frac{P_r(x)}{P_g(x)}\right) P_r(x) d\mu(x) \ ,$$

- The *Jensen-Shannon* (JS) divergence

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \| \mathbb{P}_m) + KL(\mathbb{P}_g \| \mathbb{P}_m) \ , \qquad \mathbb{P}_m = (\mathbb{P}_r + \mathbb{P}_g)/2$$

WGAN

- $\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0 \ , \\ 0 & \text{if } \theta = 0 \ . \end{cases}$
- $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0 \ , \\ 0 & \text{if } \theta = 0 \ , \end{cases}$
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \ , \\ 0 & \text{if } \theta = 0 \ , \end{cases}$

WGAN

- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|,$ → Continuous

- The *Earth-Mover* (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big] \; ,$$

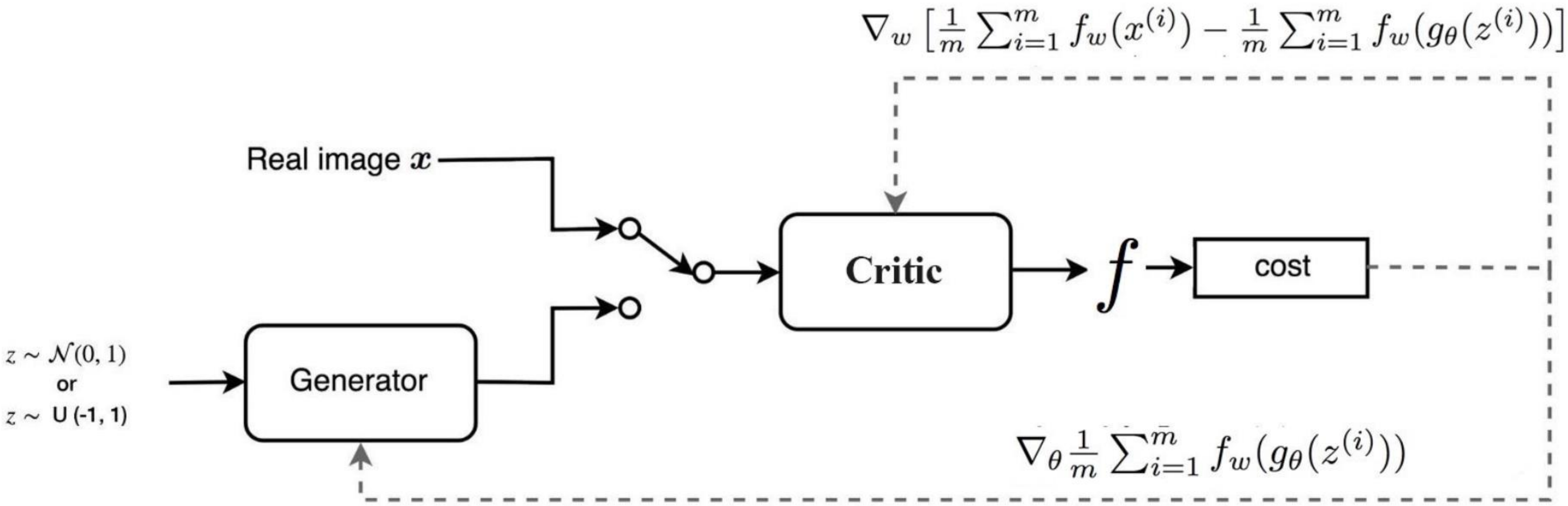
+ Kantorovich-Rubinstein duality

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

$$\frac{|f(x_1) - f(x_2)|}{|x_1 - x_2|} \leq K, \; K \geq 0$$

$$\boxed{\max_{w\in\mathcal{W}} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z\sim p(z)}[f_w(g_\theta(z))]}$$

WGAN


$\nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
Real image x
Critic
f
cost
$z \sim \mathcal{N}(0, 1)$
or
$z \sim$ U (-1, 1)
Generator
$\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$

WGAN

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.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
    $w \leftarrow \text{clip}(w, -c, c)$
  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

WGAN

```
for epoch in range(EPOCHS):
    for batch_idx, (real, _) in enumerate(data_loader):
        real = real.to(DEVICE)
        for _ in range(CRITIC_ITERATIONS):
            z = torch.randn(real.shape[0], Z_DIM, 1, 1).to(DEVICE)
            gen_img = generator(z)

            critic_real = critic(real).reshape(-1)
            critic_fake = critic(gen_img).reshape(-1)

            critic_loss = -(torch.mean(critic_real) - torch.mean(critic_fake))
            critic.zero_grad()
            critic_loss.backward(retain_graph=True)
            critic_optimizer.step()

            for p in critic.parameters():
                p.data.clip_(-WEIGHT_CLIP, WEIGHT_CLIP)

        critic_fake = critic(gen_img).reshape(-1)
        generator_loss = -torch.mean(critic_fake)
        generator.zero_grad()
        generator_loss.backward()
        generator_optimizer.step()
```

$$g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$$

$$w \leftarrow \text{clip}(w, -c, c)$$

$$g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$$

WGAN

WGAN-GP

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, β1, β2)
    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, β1, β2)
end while
```

Where:

$$\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$$

$$\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1-\epsilon)\tilde{\boldsymbol{x}}$$

$$L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$$

$$w \leftarrow \text{Adam}(\nabla_w \frac{1}{m}\sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$$

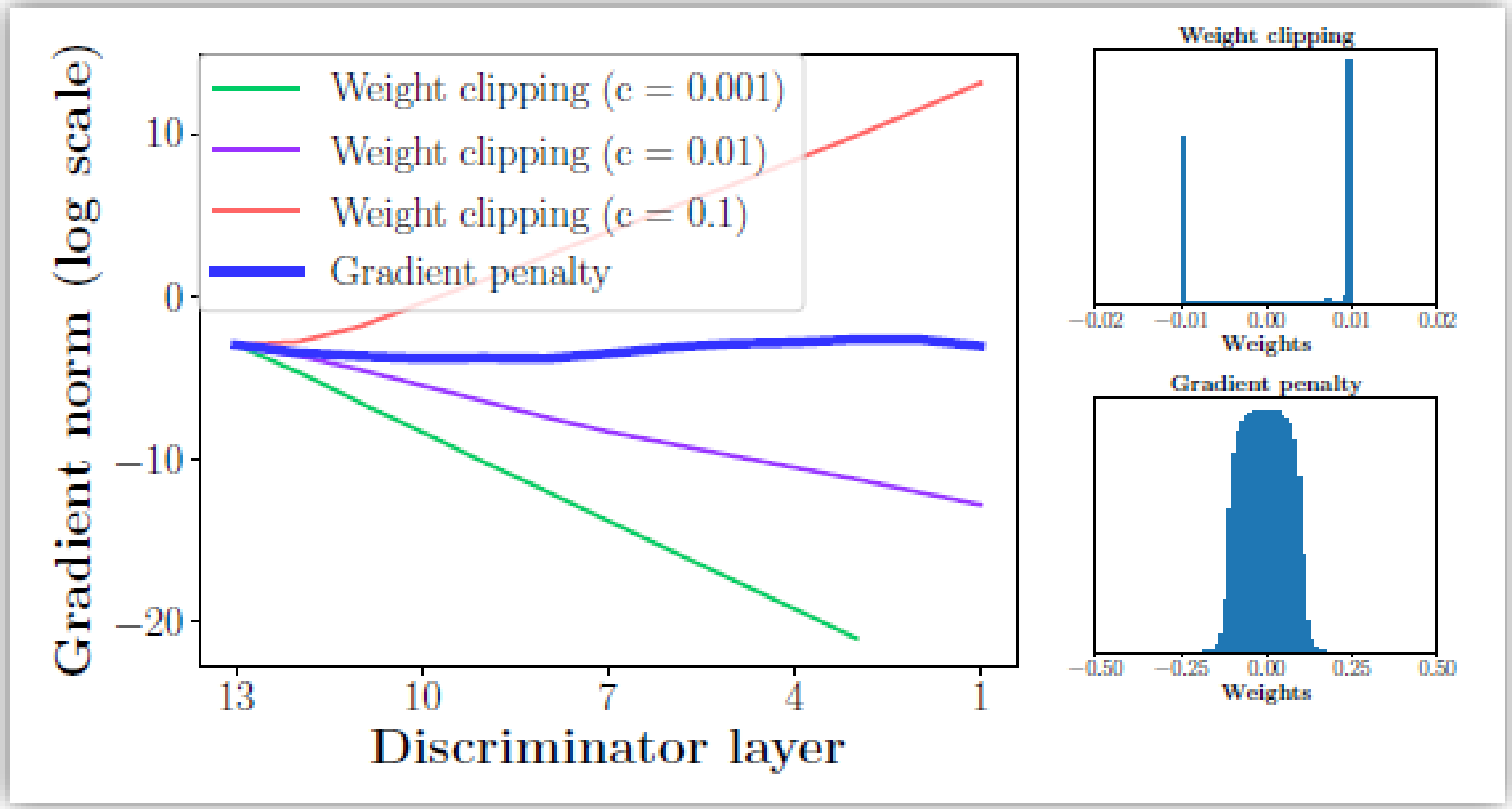
$$\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m}\sum_{i=1}^m -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$$

WGAN-GP

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

```
for epoch in range(EPOCHS):
    for batch_idx, (real, _) in enumerate(data_loader):
        real = real.to(DEVICE)
        for _ in range(CRITIC_ITERATIONS):
            z = torch.randn(real.shape[0], Z_DIM, 1, 1).to(DEVICE)
            gen_img = generator(z)

            critic_real = critic(real).reshape(-1)
            critic_fake = critic(gen_img).reshape(-1)

            gp = gradient_penalty(critic, real, gen_img, device=DEVICE)

            critic_loss = -(torch.mean(critic_real) - torch.mean(critic_fake)) + LAMBDA_GP*gp
            critic.zero_grad()
            critic_loss.backward(retain_graph=True)
            critic_optimizer.step()


        critic_fake = critic(gen_img).reshape(-1)
        generator_loss = -torch.mean(critic_fake)
        generator.zero_grad()
        generator_loss.backward()
        generator_optimizer.step()
```

WGAN-GP



Weight clipping (c = 0.001)
Weight clipping (c = 0.01)
Weight clipping (c = 0.1)
Gradient penalty
Gradient norm (log scale)
10
0
−10
−20
13
10
7
4
1
Discriminator layer
Weight clipping
−0.02
−0.01
0.00
0.01
0.02
Weights
Gradient penalty
−0.50
−0.25
0.00
0.25
0.50
Weights

WGAN-GP