Wasserstein GAN

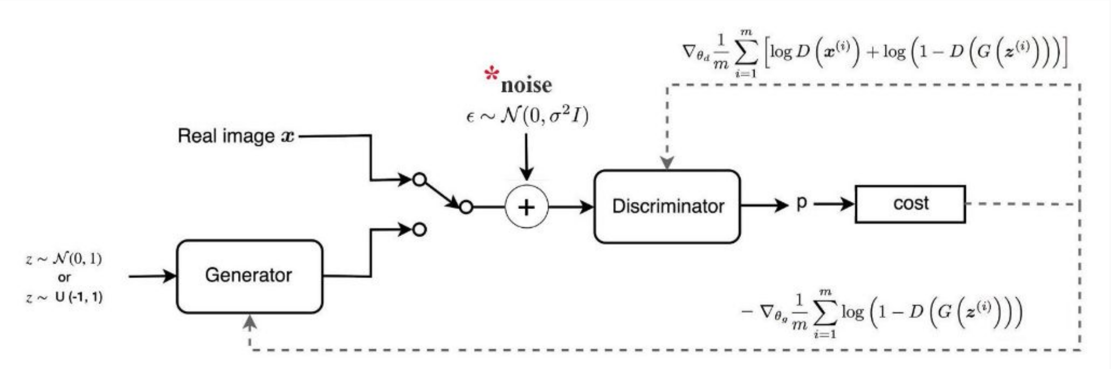
& Gradient Penalty

전설의 GAN Loss

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

$$-\frac{1}{n}\sum_{i=1}^{n}(y_i log(p_i) + (1-y_i)log(1-p_i))$$

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



Problems with GAN Loss

1. Unstable

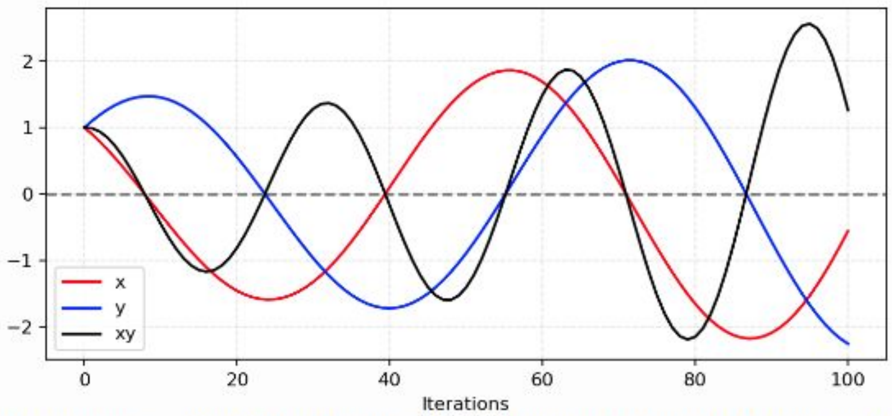


Fig. 3. A simulation of our example for updating x to minimize xy and updating y to minimize $-xy$. The learning rate $\eta = 0.1$. With more iterations, the oscillation grows more and more unstable.

Problems with GAN Loss

2. Vanishing/Exploding Gradient for Generator

$$-\nabla_{\theta_g} \log \left(1 - D\left(G\left(z^{(i)}\right)\right)\right)$$

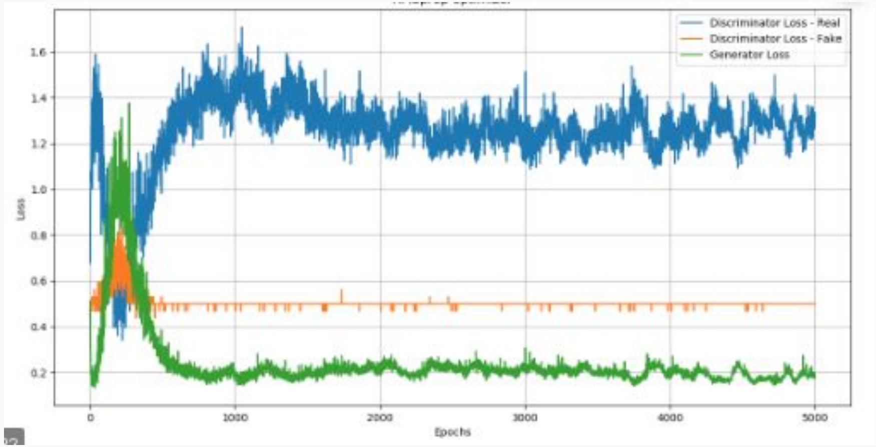
Original proposal

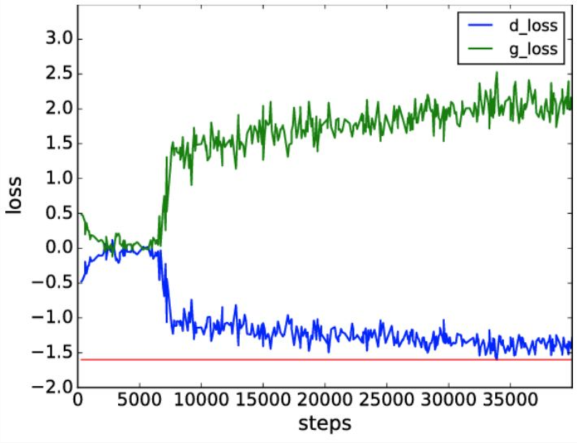
$$\nabla_{\theta_g} \log D\left(G\left(z^{(i)}\right)\right)$$

Alternative proposal

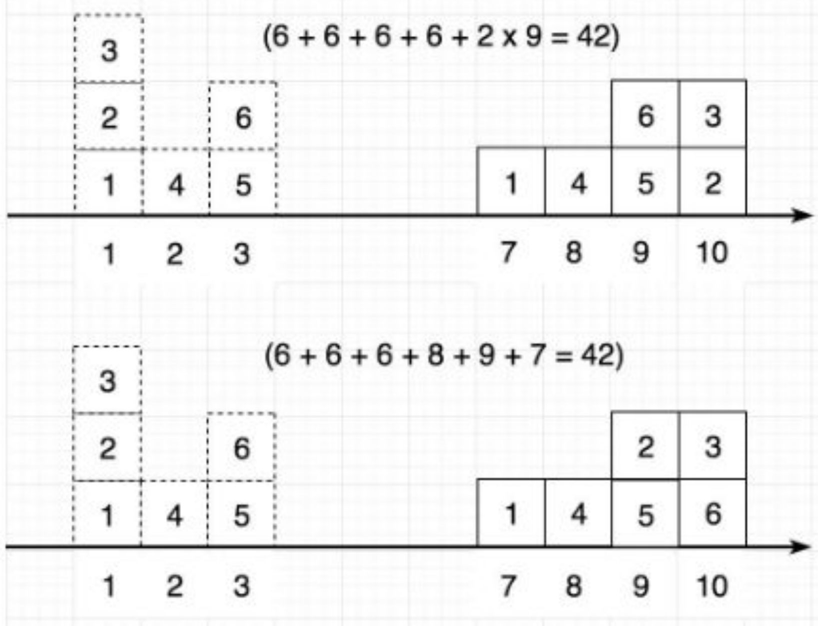
Problems with GAN Loss

3. Loss Not Interpretable





Wasserstein Distance



γ_1

	7	8	9	10
1	1	0	0	2
2	0	1	0	0
3	0	0	2	0

γ_2

	7	8	9	10
1	1	0	1	1
2	0	1	0	0
3	0	0	1	1

Wasserstein Distance

The Wasserstein distance is the minimum cost of transporting mass in converting the data distribution ***q*** to the data distribution ***p***.

- Intractable

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

- Kantorovich-Rubinstein

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

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2|$$

WGAN Loss

$$W_d(P_r, P_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x_r \sim P_r}[f(x_r)] - \mathbb{E}_{x_g \sim P_g}[f(x_g)]$$

Considering all 1-Lipschitz function
(i.e., functions with **bounded derivatives**).

f(x) is a "*critic*".
The critic should give
high value to real samples and
low value to generated samples.

$$-\frac{1}{n}\sum_{i=1}^{n}(y_i p_i)$$

$y_i = \{1, -1\}$

Discriminator output: No sigmoid, Unnormalized

Clipping weights

Train the critic five times per one generator update

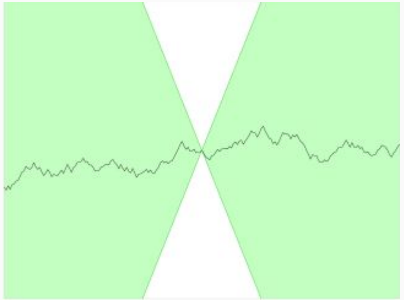
WGAN Loss: 1-Lipschitz continuous

$$\|f\|_L \leq 1$$

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2|$$

$$\frac{|D(x_1) - D(x_2)|}{|x_1 - x_2|} \leq 1$$

```
for l in critic.layers:
        weights = l.get_weights()
        for w in weights:
                w = np.clip(w, -0.01, 0.01)
```



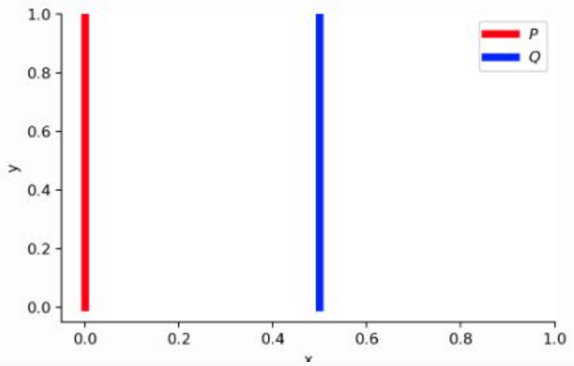
the absolute difference between the critic predictions of any two images

is smaller than

the average pixelwise absolute difference between the images.

*There exists a double cone s.t. the function always remains outside the cone.

Wasserstein > KD & JS



$$D_{KL}(P||Q) = \sum_{x=1}^{N} P(x) \log \frac{P(x)}{Q(x)}$$

$$D_{JS}(p||q) = \frac{1}{2} D_{KL}(p||\frac{p+q}{2}) + \frac{1}{2} D_{KL}(q||\frac{p+q}{2})$$

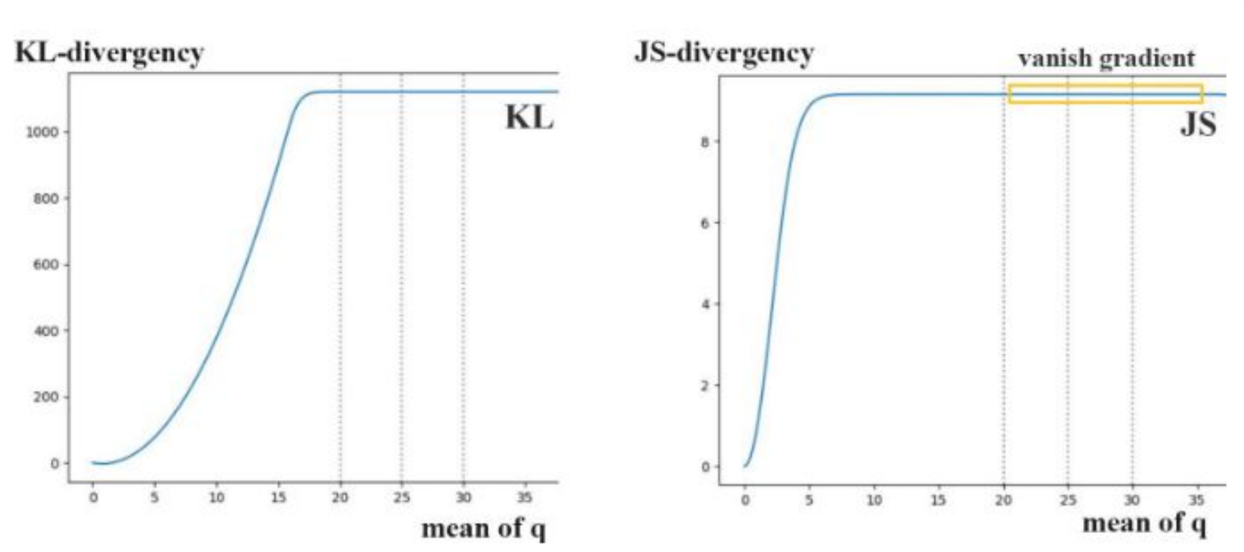
When $\theta \neq 0$:

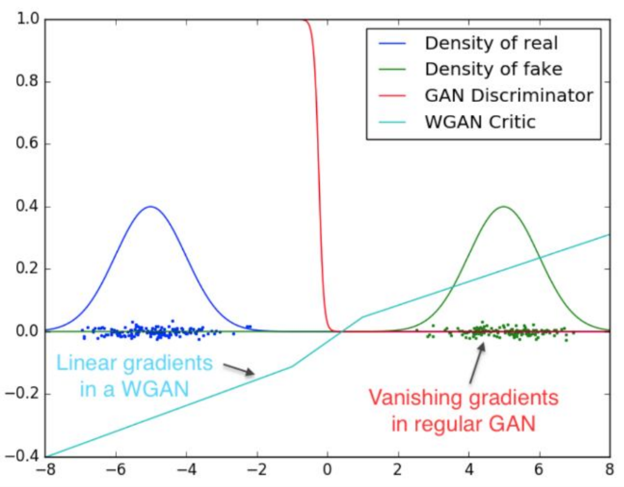
$$D_{KL}(P\|Q) = \sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$
$$D_{KL}(Q\|P) = \sum_{x=\theta, y\sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$
$$D_{JS}(P,Q) = \frac{1}{2}\left(\sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{1/2} + \sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{1/2}\right) = \log 2$$
$$W(P,Q) = |\theta|$$

But when $\theta = 0$, two distributions are fully overlapped:

$$D_{KL}(P\|Q) = D_{KL}(Q\|P) = D_{JS}(P,Q) = 0$$
$$W(P,Q) = 0 = |\theta|$$

Wasserstein > KD & JS





WGAN vs GAN

	Discriminator/Critic	Generator
GAN	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$
WGAN	$\nabla_w \frac{1}{m} \sum_{i=1}^{m} \left[f(x^{(i)}) - f(G(z^{(i)}))\right]$	$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f(G(z^{(i)}))$

$-\frac{1}{n}\sum_{i=1}^{n}(y_i log(p_i) + (1 - y_i) log(1 - p_i))$ → $-\frac{1}{n}\sum_{i=1}^{n}(y_i p_i)$

WGAN vs GAN: Implementation

$$-\frac{1}{n}\sum_{i=1}^{n}(y_i p_i)$$

```python
def get_wgan_losses_fn():
    def d_loss_fn(r_logit, f_logit):
        r_loss = - tf.reduce_mean(r_logit)
        f_loss = tf.reduce_mean(f_logit)
        return r_loss, f_loss

    def g_loss_fn(f_logit):
        f_loss = - tf.reduce_mean(f_logit)
        return f_loss

    return d_loss_fn, g_loss_fn
```

$$-\frac{1}{n}\sum_{i=1}^{n}(y_i log(p_i) + (1 - y_i) log(1 - p_i))$$

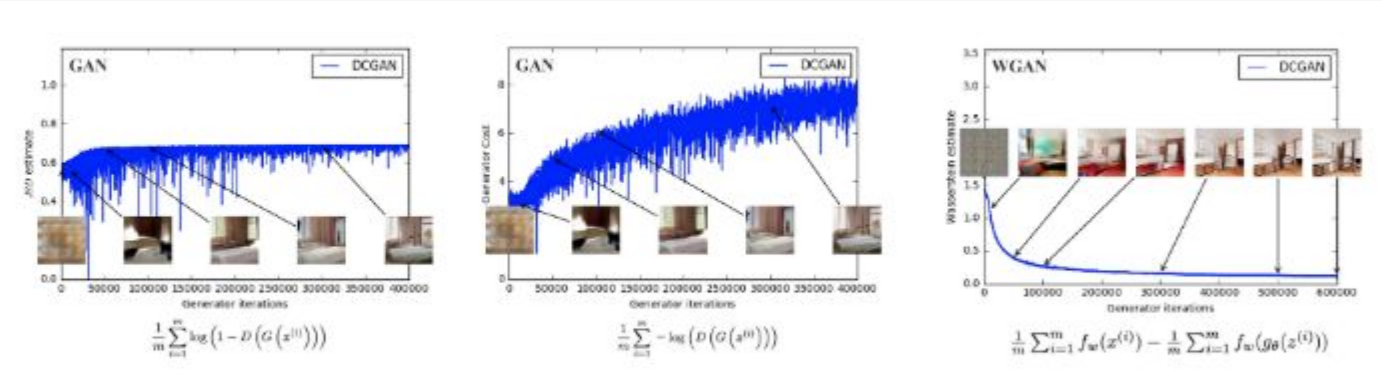
```python
def get_gan_losses_fn():
    bce = tf.losses.BinaryCrossentropy(from_logits=True)

    def d_loss_fn(r_logit, f_logit):
        r_loss = bce(tf.ones_like(r_logit), r_logit)
        f_loss = bce(tf.zeros_like(f_logit), f_logit)
        return r_loss, f_loss

    def g_loss_fn(f_logit):
        f_loss = bce(tf.ones_like(f_logit), f_logit)
        return f_loss

    return d_loss_fn, g_loss_fn
```

WGAN vs GAN

$$-\frac{1}{n}\sum_{i=1}^{n}(y_i p_i)$$

Interpretable

More stable



WGAN vs WGAN GP

$$W_d(P_r, P_g) = \sup_{\|f\|_L \leq 1} \mathop{\mathbb{E}}_{x_r \sim P_r}[f(x_r)] - \mathop{\mathbb{E}}_{x_g \sim P_g}[f(x_g)]$$

Considering all 1-Lipschitz function
(i.e., functions with **bounded derivatives**).

f(x) is a "*critic*".
The critic should give
high value to real samples and
low value to generated samples.

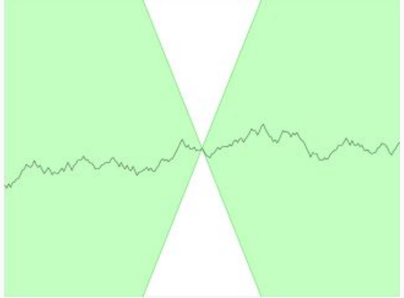
Bounded by:

a) Weight clipping (Wasserstein GAN; "WGAN").
b) Gradient penalty (Improved Training; "WGAN-GP")

How do you determine the clipping value? 0.1 or 0.01?

WGAN GP Loss: 1-Lipschitz continuous

$$\frac{|D(x_1)-D(x_2)|}{|x_1-x_2|} \leq 1$$



A differentiable function f is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere.

f^* has gradient norm $\underline{1}$ almost everywhere under $\mathbb{P}_r$ and $\mathbb{P}_g$.

WGAN GP Loss: 1-Lipschitz continuous

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

where $\hat{\boldsymbol{x}}$ *sampled from* $\tilde{\boldsymbol{x}}$ *and* $\boldsymbol{x}$ *with t uniformly sampled between 0 and 1*

$$\hat{\boldsymbol{x}} = t\tilde{\boldsymbol{x}} + (1-t)\boldsymbol{x} \text{ with } 0 \le t \le 1$$

λ = 10

WGAN GP Implementation

No Batch Normalization in the critic

Freze the weights of the generator

The critic uses thre losses:

- W_loss for the real
- W_loss for the fake
- 10*GP_loss

Adam

```python
def gradient_penalty(f, real, fake, mode):
    def _gradient_penalty(f, real, fake=None):
        def _interpolate(a, b=None):
            if b is None:   # interpolation in DRAGAN
                beta = tf.random.uniform(shape=tf.shape(a), minval=0., maxval=1.)
                b = a + 0.5 * tf.math.reduce_std(a) * beta
            shape = [tf.shape(a)[0]] + [1] * (a.shape.ndims - 1)
            alpha = tf.random.uniform(shape=shape, minval=0., maxval=1.)
            inter = a + alpha * (b - a)
            inter.set_shape(a.shape)
            return inter

        x = _interpolate(real, fake)
        with tf.GradientTape() as t:
            t.watch(x)
            pred = f(x)
        grad = t.gradient(pred, x)
        norm = tf.norm(tf.reshape(grad, [tf.shape(grad)[0], -1]), axis=1)
        gp = tf.reduce_mean((norm - 1.)**2)

        return gp
```