Generative Modeling

Generative adversarial networks

Denis Derkach

Laboratory for methods of big data analysis

HSE-Yandex Autumn School 2021, Moscow

LAMBDA • HSE

November 2021

In this Lecture

- Generative Adversarial Networks
  - algorithm statement;
  - ideal case;
  - shortcomings of vanilla algorithm;
  - proposed shortcuts.

Idea

Reminder: f-divergence

- For convex $f(.)$, P and Q some distributions, we define f-divergence:

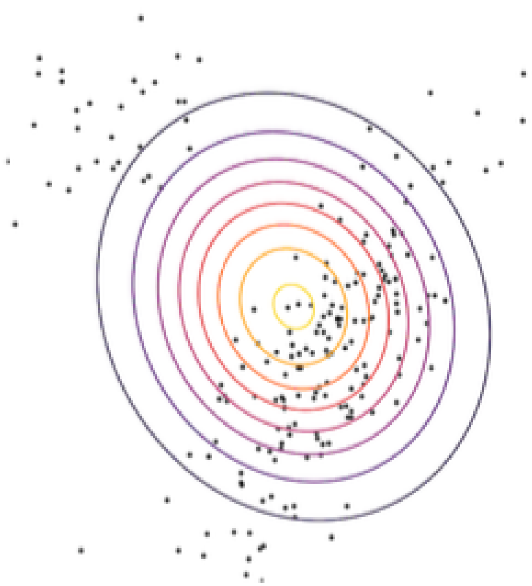
$$\boxed{D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx.}$$

Reminder: f-divergence

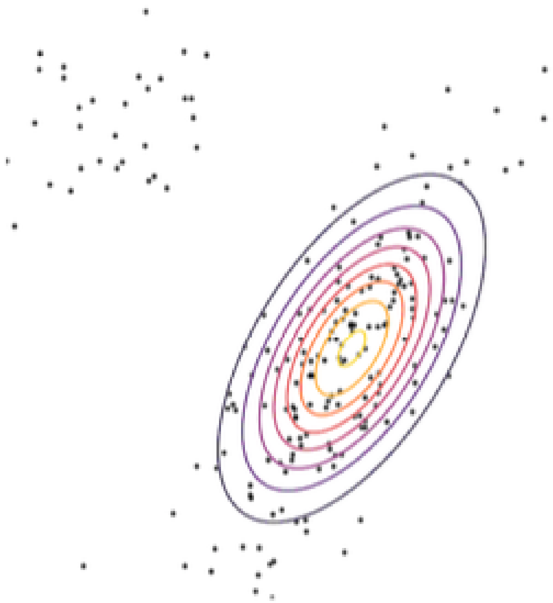
- For convex $f(.)$, P and Q some distributions, we define f-divergence:

$$\boxed{D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx.}$$

$$KL = \int p(X) \log \frac{p(x)}{q_\theta(x)} dx$$



$$rKL = \int q(x) \log \frac{q_\theta(x)}{p(x)} dx$$



Reminder: f-divergence Convergence

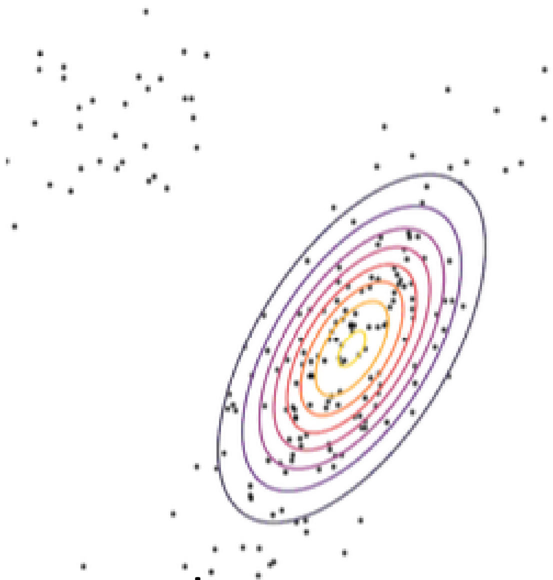
- For convex $f(.)$, P and Q some distributions, we define f-divergence:

$$\boxed{D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx.}$$

$$rKL = \int q(x) \log \frac{q_\theta(x)}{p(x)} dx$$

$$\theta^* = \underset{\theta}{\operatorname{argmin}} KL(q_\theta(x)||p(x))$$

$$= \underset{\theta}{\operatorname{argmax}}(-\mathbb{E}_{\tilde{x}\sim q_\theta}[\log q_\theta(x)] + \mathbb{E}_{\tilde{x}\sim q_\theta}[\log p(x)])$$



To optimize rKL properly we need access to true PDF, **p(x)**.

Reminder: Variational Lower Bound

- For convex $f(.)$, P and Q some distributions, we define f-divergence:

$$\boxed{D_f(P||Q) = \int q(x) f\left(\frac{p(x)}{q(x)}\right) dx.}$$

- Lower bound can be written:

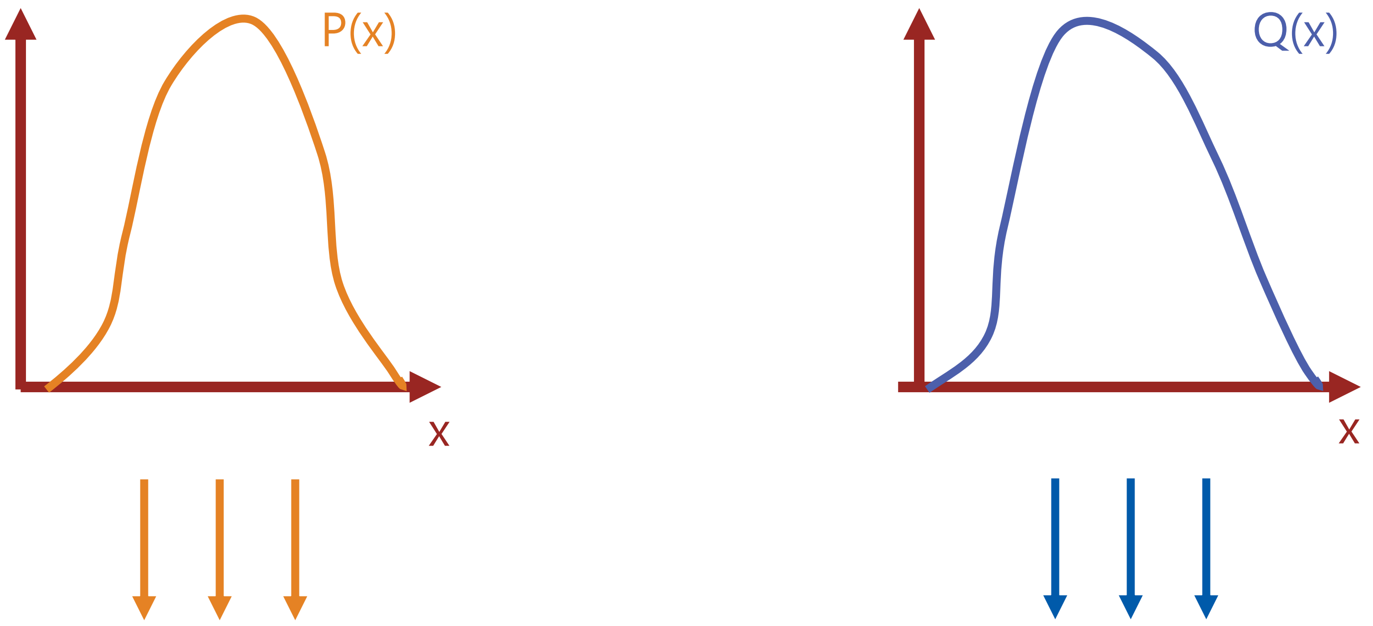
$$D_f(P||Q) \geq \max_{T(x)} \mathbb{E}_{x\sim P} T(x) - \mathbb{E}_{x\sim Q} f^*\big(T(x)\big),$$

$T(x)$ is some random function.

- The tight boundary can be estimated for each f-divergence ($T^*(x)$).
- For JS-divergence $T^*(x) = \log \frac{2p(x)}{p(x)+q(x)}$, $f^*(x) = -\log(2 - \exp(t))$.

Lower Bound for JS

$$JS(p,q) = \frac{1}{2}\left(KL(p(x)||\frac{p(x)+q_\theta(x)}{2}) + KL(q_\theta(x)||\frac{p(x)+q_\theta(x)}{2})\right)$$



$$\mathrm{JS}(P||Q) \geq \mathbb{E}_{x\sim P}\log\frac{2p(x)}{p(x)+q(x)} + \mathbb{E}_{x\sim Q}\log\frac{2q(x)}{p(x)+q(x)}$$

It would be interesting to construct something close.

Adversarial optimization

Rationale

- Need to optimize the model q_θ without the direct access to the $p(x)$.

$$\theta^* = \arg\min_\theta \max_\phi \mathbb{E}_{x\sim p, \tilde{x}\sim q_\theta} V(f_\phi(x), f_\phi(\tilde{x}))$$

- Instead of minimizing over some analytically defined divergence with parameter ϕ, one could minimize over ”learned divergence”.

Generator

- G_θ is a **generator**. It should sample from a random noise:

$$z_j \sim N(0;1);$$

$$x_j = \mathrm{G}_\theta\left(z_j\right).$$

- Our aim is G_θ as a neural network.
- We thus have a sample:

$$\{x_j\} \sim q_\theta(x)$$

- G_θ can be defined in many ways. For example, physics generator.

Borisyak M et al. Adaptive divergence for rapid adversarial optimization. *PeerJ Computer Science* 6:e274 (2020)

Discriminator

- Add a classifying neural network, **discriminator D_ϕ**, to distinguish between the real and generated samples.
- Optimize:

$$\max_\phi \left(\mathbb{E}_{x\sim p(x)}(\log(D_\phi(x)) + \mathbb{E}_{\tilde{x}\sim q_\theta(x)}(1-\log(D_\phi(\tilde{x}))\right)$$

Real samples

Generated samples

G+D recap

We can now put together generator and discriminator.

- objective of discriminator:

$$\max_{\phi} \left(\mathbb{E}_{x\sim p(x)} (\log(D_\phi(x)) + \mathbb{E}_{z\sim\mathcal{N}(0;1)} (1 - \log(D_\phi(G_\theta(z))) \right).$$

- objective of generator:

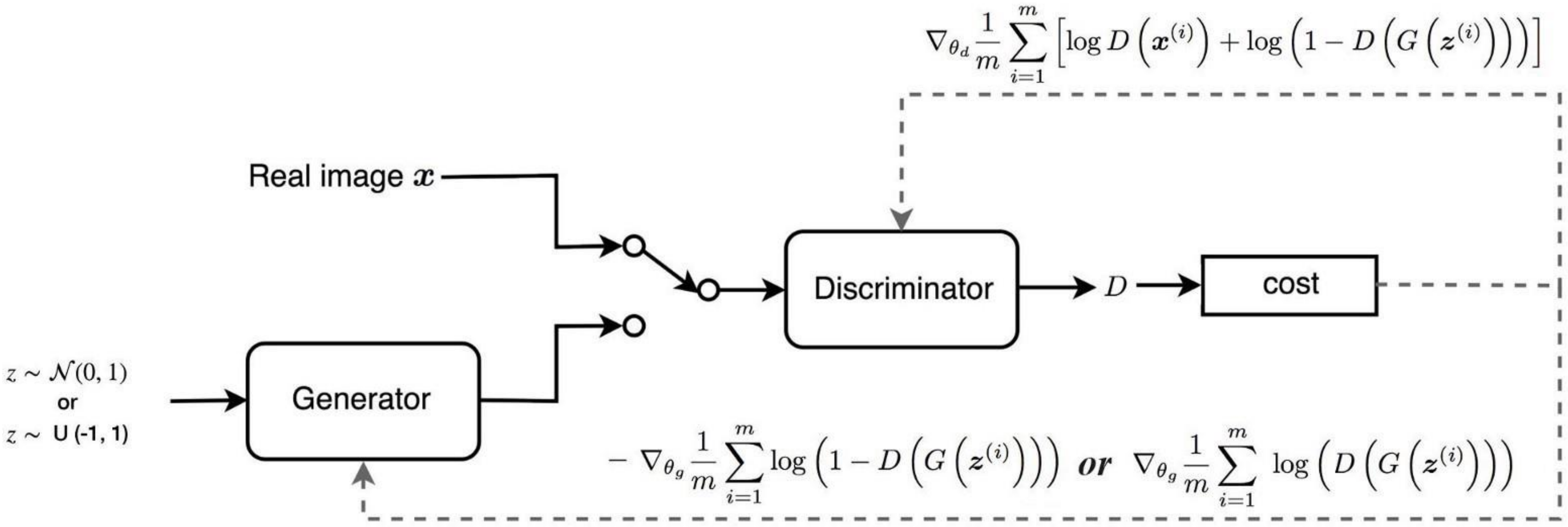
$$\min_{\theta} \mathbb{E}_{z\sim\mathcal{N}(0;1)} (1 - \log(D_\phi(G_\theta(z)))$$

We thus defined a minimax game:

$$\min_{\theta} \max_{\phi} \mathbb{E}_{x\sim p, \tilde{x}\sim q_\theta} V(f_\phi(x), f_\phi(\tilde{x})).$$

In exactly the way we wanted.

Training at a Glance



For D and G defined as neural networks, we can use backpropagation.

Optimal Solution

- For a given generator, the optimal discriminator is:

$$D^*_\phi(G) = \frac{p(x)}{p(x) + q_\theta(x)}.$$

- Incorporating that into the minimax game to yield virtual training criterion:

$$C(G) = \max_D V(G, D) =$$

$$= \mathbb{E}_{x\sim p(x)} \log(D^*_\phi(x)) + \mathbb{E}_{x\sim q_\theta(x)} \log(1 - \log(D^*_\phi(x)) =$$

$$= \mathbb{E}_{x\sim P} \log \frac{p(x)}{p(x) + q(x)} + \mathbb{E}_{x\sim q_\theta(x)} \log \frac{q(x)}{p(x) + q(x)}$$

Lower Bound Reminder

- In case of ideal discriminator:

$$C(G) = \mathbb{E}_{x\sim P} \log \frac{p(x)}{p(x)+q(x)} + \mathbb{E}_{x\sim Q} \log \frac{q(x)}{p(x)+q(x)}$$

- This can be compared to variational bound:

$$\mathrm{JS}(P||Q) \geq \mathbb{E}_{x\sim p(x)} \log \frac{2p(x)}{p(x)+q(x)} + \mathbb{E}_{x\sim q(x)} \log \frac{2q(x)}{p(x)+q(x)}$$

- Difference is only in $\log 4$

Optimal Solution

- In an optimal case $p = q_\theta$, we have $C(G) = -\log(4)$.
- We thus can write out:

$$C(G) = -\log(4) + \frac{1}{2}KL(p(x)||\frac{p(x)+q_\theta(x)}{2}) + \\ +\frac{1}{2}KL(q_\theta(x)||\frac{p(x)+q_\theta(x)}{2}).$$

- In other words, we effectively optimize Jensen-Shannon divergence:

$$C(G) = -\log(4) + JS(p(x)||q_\theta(x)).$$

- Reminder: we did it without access to $p(x)$.

GAN algorithm

1. Sample data mini-batch ($x_1, .., x_m \sim D$).
2. Sample generator mini-batch ($z_1, .., z_m \sim q_\theta$).
3. Use SGD to obtain new weights of generator:

$$\nabla_\theta V(G_\theta, D_\phi) = \frac{1}{m} \sum_{i=1}^{m} \log(1 - D_\phi(G_\theta(z_i))).$$

4. Use SGD to obtain new weights of discriminator:

$$\nabla_\phi V(G_\theta, D_\phi) = \frac{1}{m} \sum_{i=1}^{m} \left(\log D_\phi(x_i) + \log(1 - D_\phi(G_\theta(z_i)))\right).$$

5. Repeat with several epochs.

GAN results

Figure 2: Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing. a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and "deconvolutional" generator)

I. Goodfellow,et al. Generative Adversarial Networks, NIPS 2014

GAN First Paper Disclaimer

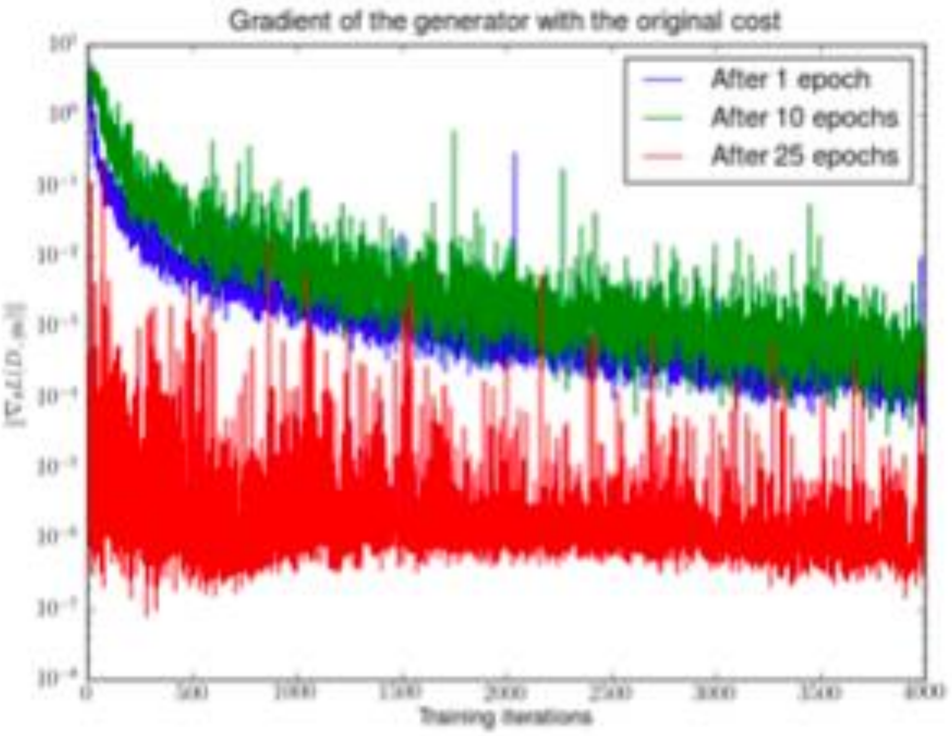
While we make no claim that these samples are better than samples generated by existing methods, we believe that these samples are at least competitive with the better generative models in the literature and highlight the potential of the adversarial framework.

I. Goodfellow, Generative Adversarial Networks, NIPS 2014

GAN problems

Game Approach Problems

- Discriminator must be optimal on every step of convergence.
- This is not true, you should not overtrain discriminator.
- Loss-function can be quite noisy.

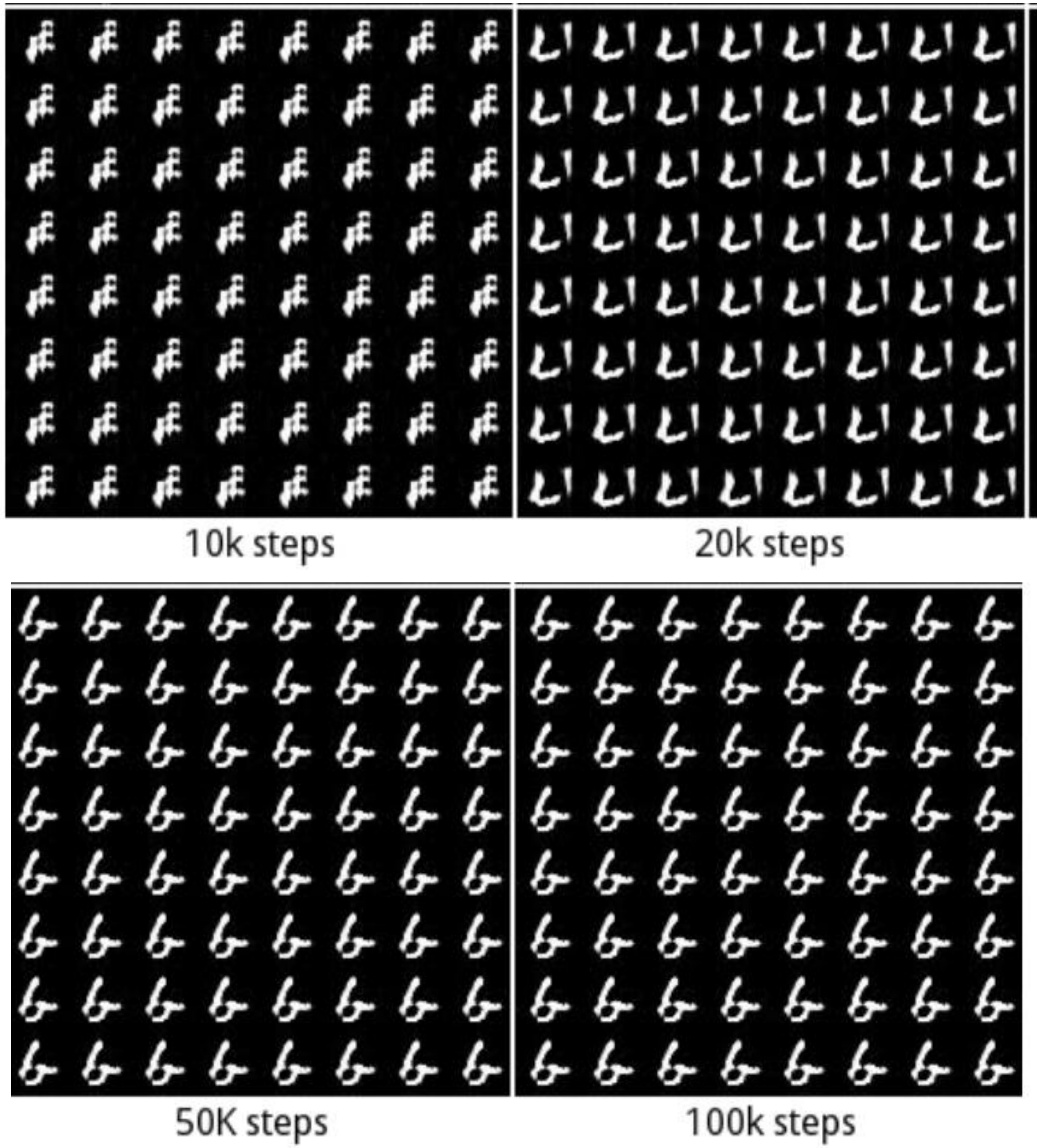


Martin Arjovsky, Towards Principled Methods for Training Generative Adversarial Networks , ICLR17

Mode Collapse

- GANs choose to generate a small number of modes due to a defect in the training procedure, rather than due to the divergence they aim to minimize.

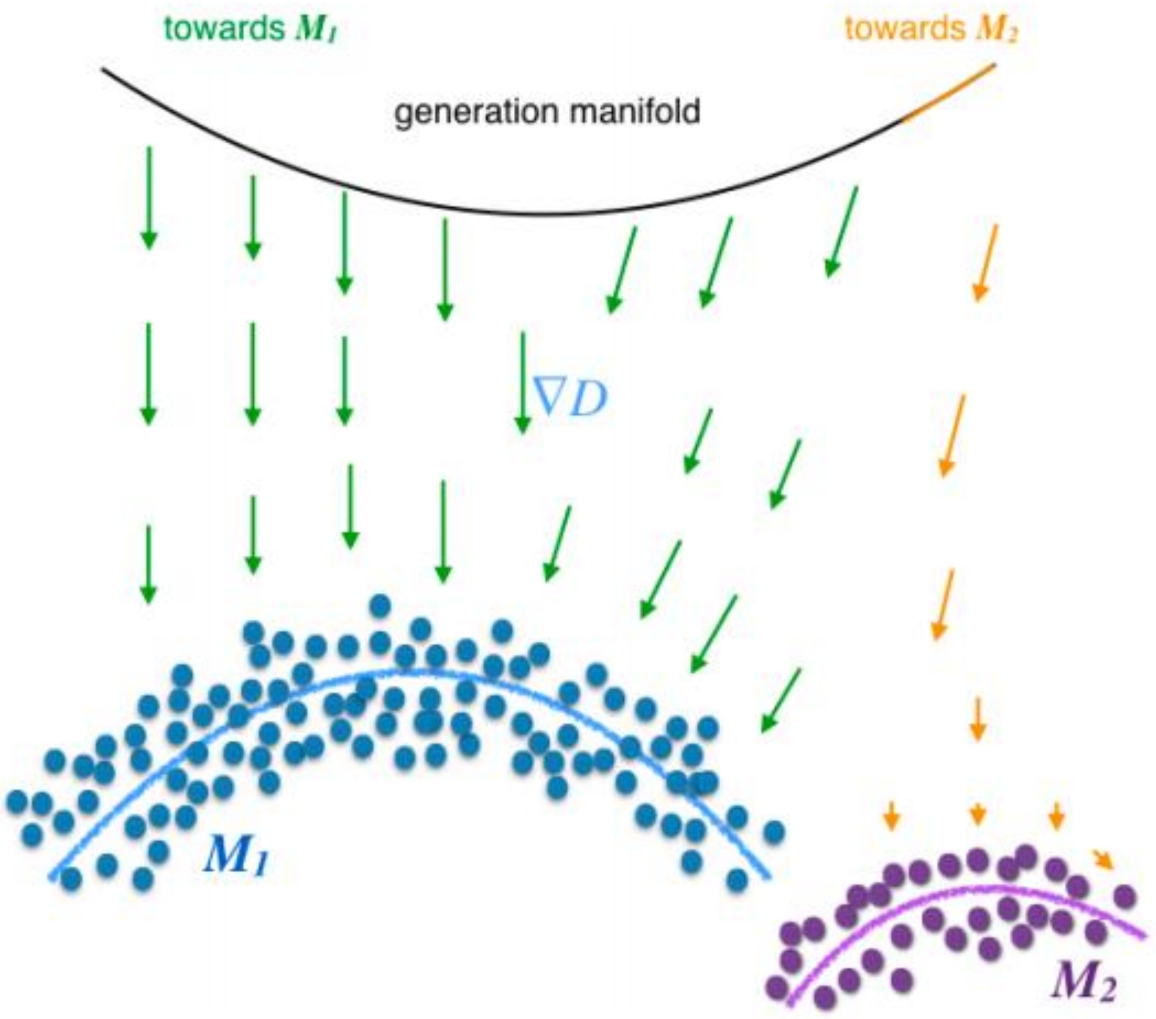
I. Goodfellow NIPS 2016 Tutorial: Generative Adversarial Network



Luke Metz et al Unrolled Generative Adversarial Networks ICLR 2017

Mode Collapse

- For fixed D:
  - G tends to converge to a point x^* that fools D the most.
  - In extreme cases, G becomes independent on z.
  - Gradient on z diminishes.



- When D restarts:
  - Easily finds this x^*.
  - Pushes G to the next point x^{**}.

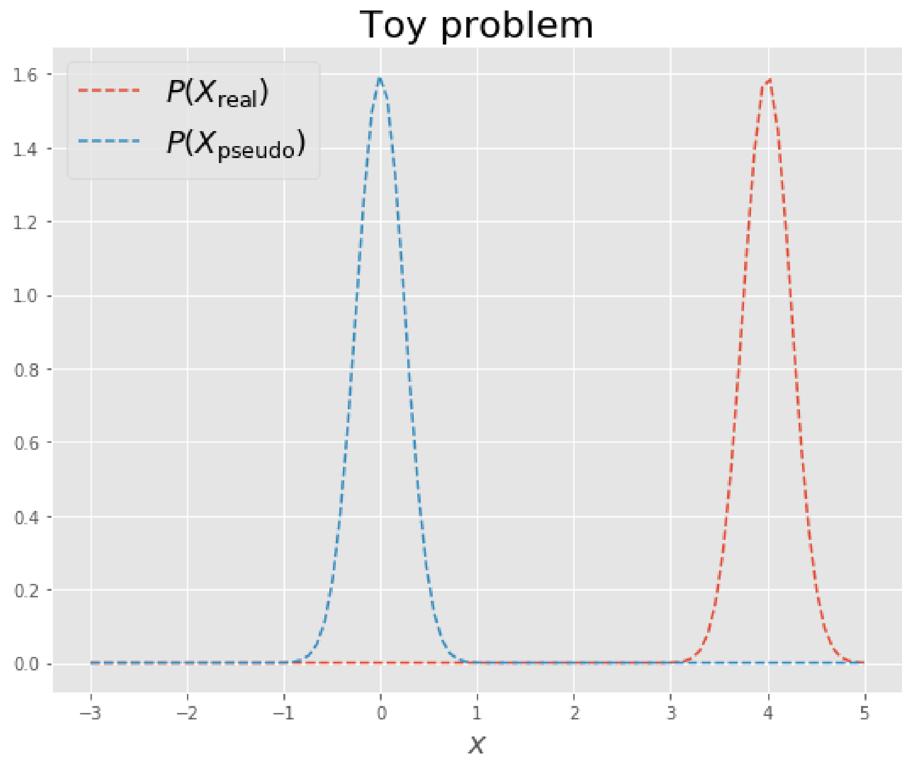
T. Che Mode Regularized Generative Adversarial Networks ICLR 2017

Vanishing Gradients

- For disjoint support of real and generated data

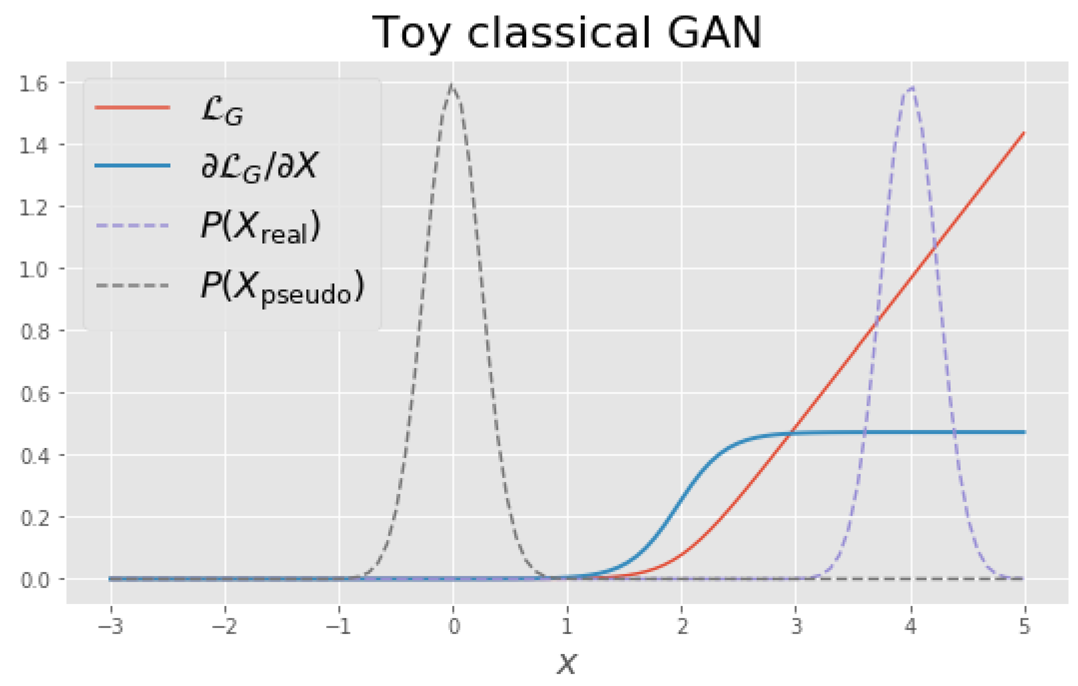
- An ideal discriminator can perfectly tell the real and generated data apart:

$$D(G(z)) \approx 0$$



Vanishing Gradients

- $L_{\mathrm{G}} = -\log D(G(z))$
- ${dD(x)}/{dx} \approx 0$ for generated x
- ${dL_{\mathrm{G}}(x)}/{dx} \approx 0$ for generated x
- Generator can't train!
- Need to start closer (how?)
- Problem is further enhanced due to noisy estimate from data.



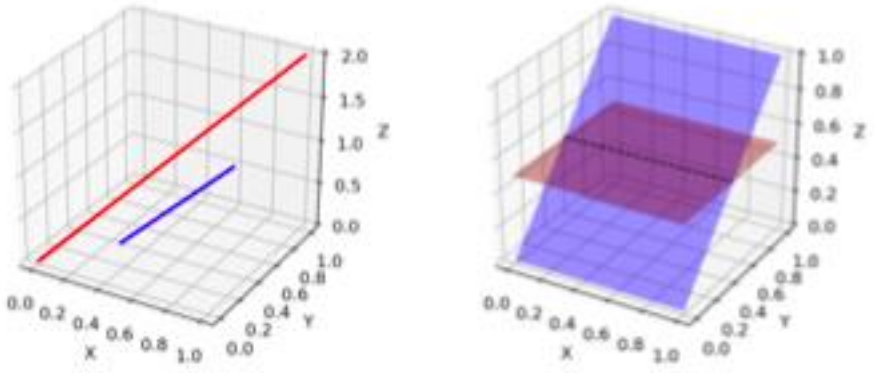
Summary so Far

- Pros:
  - Can utilize power of back-prop.
  - No explicit intractable integral.
  - No MCMC needed.
- Cons:
  - Unclear stopping criteria.
  - No explicit representation of $g_\theta(x)$.
  - Hard to train.
  - No evaluation metric so hard to compare with other models.
  - Easy to get trapped in local optima that memorize training data.
  - Hard to invert generative model to get back latent z from generated x.

GAN ways out

Diminishing Gradients

- We have seen already that signal data is located on manifold.
- GAN case is in fact more complicated, as we need a discriminator that distinguishes two supports.
- This is way too easy, if supports are disjoint.

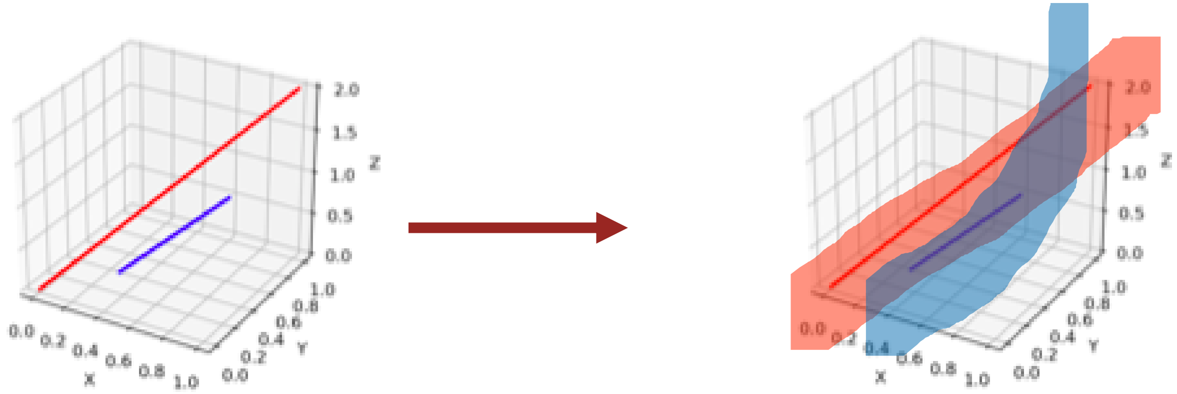


Diminishing Gradients: Noisy Supports

- Let's make the problem harder: introduce random noise $\varepsilon \sim N(0; \sigma^2 I)$:

$$\mathbb{P}_{x+\varepsilon(x)} = \mathbb{E}_{y\sim P(x)} \mathbb{P}_{\varepsilon}(x - y).$$

- This will make noisy supports, that makes it difficult for discriminator.



Martin Arjovsky, Towards Principled Methods for Training Generative Adversarial Networks , ICLR17

Feature Matching

- Change the objective of the generator:

$$||\mathbb{E}_{x\sim p(x)} f(x) - \mathbb{E}_{z\sim p_z(z)} f(G(z))||^2$$

- Here $f(x)$ can be any property we need (including the output of another network.

Danger of overtrain to match known tests!

Historical averaging

- average with previous parameter values:

$$||\theta - \frac{1}{t}\sum_{i=1}^{t}\theta[i]||^2$$

- this allows to create a fake agent that plays the game.
- and solves the problems only in low dimensions.

Salimans et al. Improved Techniques for Training GANs, NIPS16

Look into the Future: unrolled GANs

- What if we can look into the future of system?
- We could avoid local optima and optimize better.
- Algorithm:
  - At each step we train discriminator 5-10 steps ahead.
  - We DO NOT introduce it to the system.
  - We show the possible future moves to generator and update it accordingly.

Unrolled GAN: results

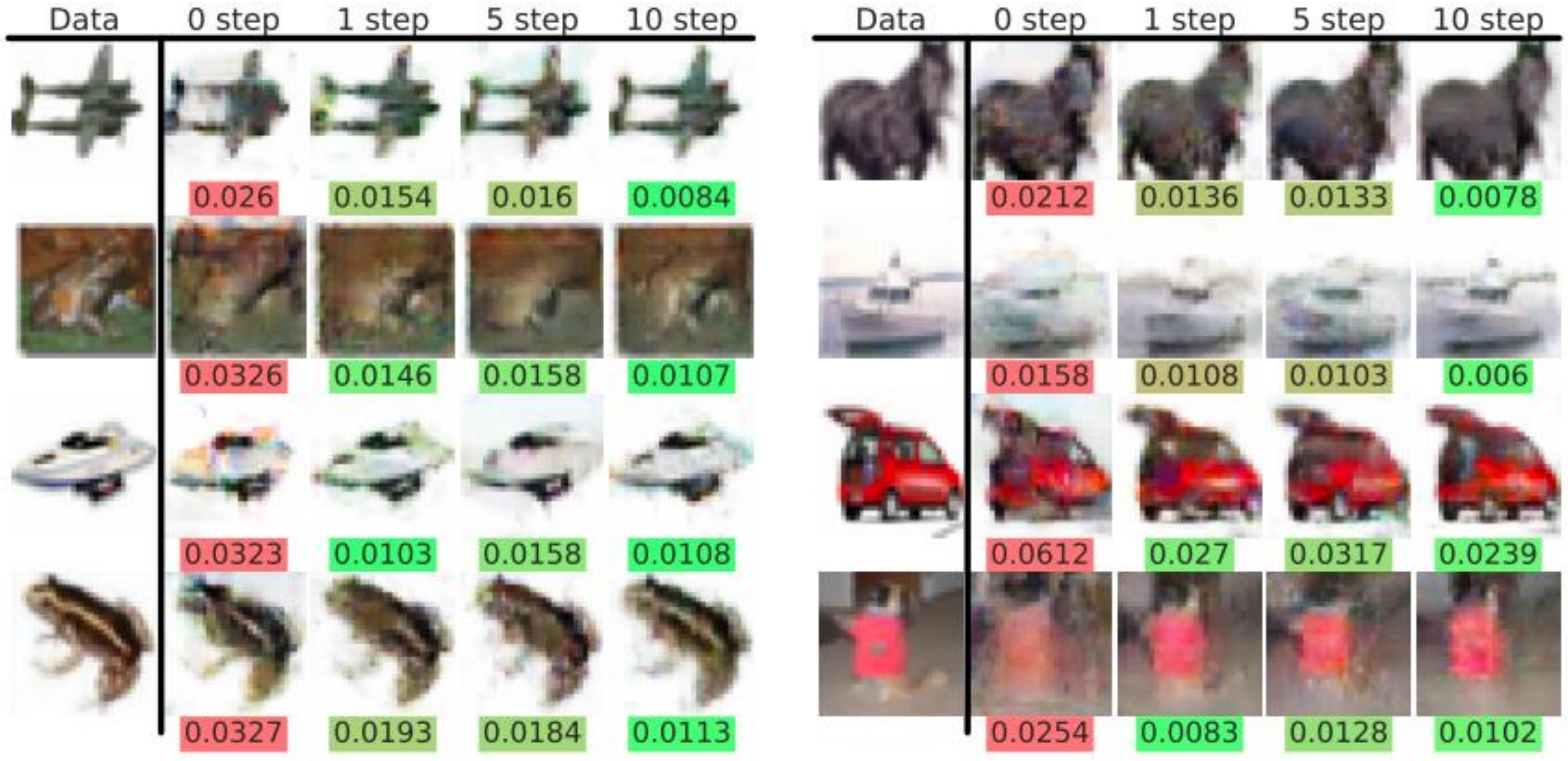


Figure 5: Training set images are more accurately reconstructed using GANs trained with unrolling than by a standard (0 step) GAN, likely due to mode dropping by the standard GAN. Raw data is on the left, and the optimized images to reach this target follow for 0, 1, 5, and 10 unrolling steps. The reconstruction MSE is listed below each sample. A random 1280 images where selected from the training set, and corresponding best reconstructions for each model were found via optimization. Shown here are the eight images with the largest absolute fractional difference between GANs trained with 0 and 10 unrolling steps.

Luke Metz et al Unrolled Generative Adversarial Networks ICLR 2017

Conclusions: GANs

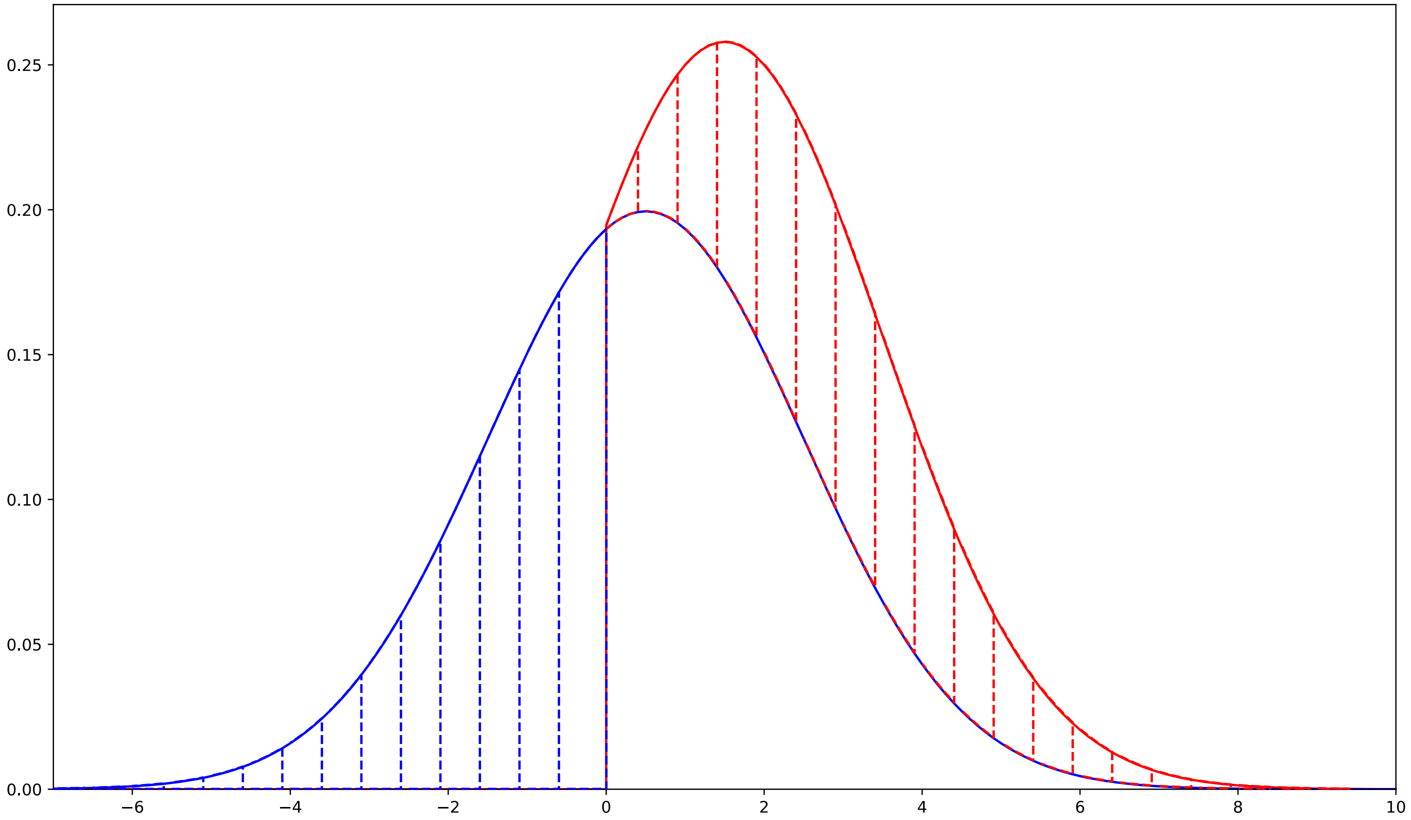
- use Generator-Discriminator game to estimate the distance from generated distribution to the true one.
- produce sharp images.
- reconstruct implicit model of target PDF.

Wasserstein Distance

Wasserstein distance

Also called "Earth mover's distance" (EMD)

- Distributions $P(x)$ and $Q(x)$ are viewed as describing the **amounts of "dirt" at point x**
- We want to convert one distribution into the other by **moving around** some amounts of dirt



- The cost of moving an amount $\boldsymbol{m}$ from $\boldsymbol{x_1}$ to $\boldsymbol{x_2}$ is $\boldsymbol{m} \times \|\boldsymbol{x_2} - \boldsymbol{x_1}\|$
- $\mathrm{EMD}(P, Q)$ = **minimum total cost** of converting P into Q

Idea of definition

- Say, we have a moving plan $\gamma(x_1, x_2) \geq 0$:

$\gamma(x_1, x_2) dx_1 dx_2 -$ how much dirt we're moving from $[x_1, x_1 + dx_1]$ to $[x_2, x_2 + dx_2]$

- Then, the cost of moving from $[x_1, x_1 + dx_1]$ to $[x_2, x_2 + dx_2]$ is:

$$\|x_2 - x_1\| \cdot \gamma(x_1, x_2) dx_1 dx_2$$

- and the total cost is:

$$C = \int_{x_1, x_2} \|x_2 - x_1\| \cdot \gamma(x_1, x_2) dx_1 dx_2 = \mathbb{E}_{x_1, x_2 \sim \gamma(x_1, x_2)} \|x_2 - x_1\|$$

- Since we want to convert P to Q, the plan has to satisfy:

$$\int_{x_1} \gamma(x_1, x_2) dx_1 = Q(x_2), \qquad \int_{x_2} \gamma(x_1, x_2) dx_2 = P(x_1)$$

Idea of Definition

- Let π be the set of all plans that convert P to Q, i.e.:

$$\pi = \left\{ \gamma: \quad \gamma \geq 0, \qquad \int_{x_1} \gamma(x_1, x_2) dx_1 = Q(x_2), \qquad \int_{x_2} \gamma(x_1, x_2) dx_2 = P(x_1) \right\}$$

- Then, the Wasserstein distance between P and Q is:

$$\mathrm{EMD}(P, Q) = \inf_{\gamma \in \pi} \mathbb{E}_{x_1, x_2 \sim \gamma} ||x_2 - x_1||$$

Optimization over all transport plans – not too friendly

Wasserstein Distance

For continuous case, there are a set of p-Wasserstein distances, with $W_p(p_x, q_y)$ defined with $x \in M$, $y \in M$ and a distance D on x, y:

$$W_p(p_x, q_y) = \inf_{\gamma \in \Pi(x,y)} \int_{M \times M} D(x, y)^p d\gamma(x, y),$$

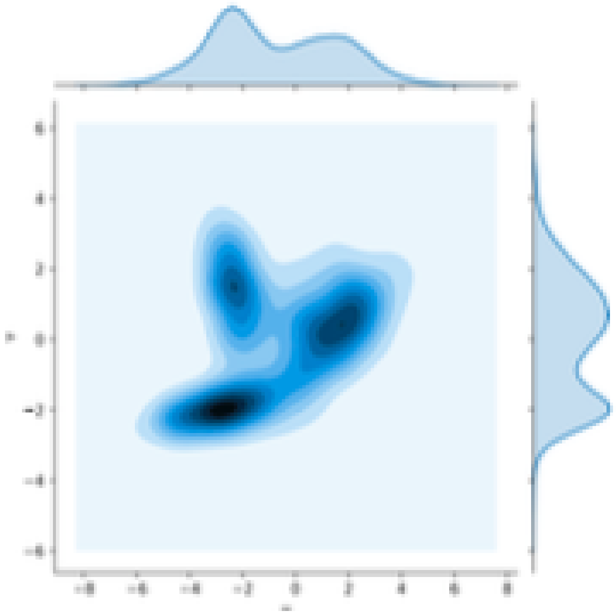
where $\Pi(x, y)$ is a set of all joint distributions having p_x, q_y as their marginals.

W_1 distance

In particular, W_1 distance with Euclidean norm is:

$$W(p_x, q_y) = \inf_{\gamma \in \Pi(x,y)} \int_{M \times M} D(x,y) d\gamma(x,y) = \inf_{\gamma \in \Pi(x,y)} \mathbb{E}(||x - y||)$$

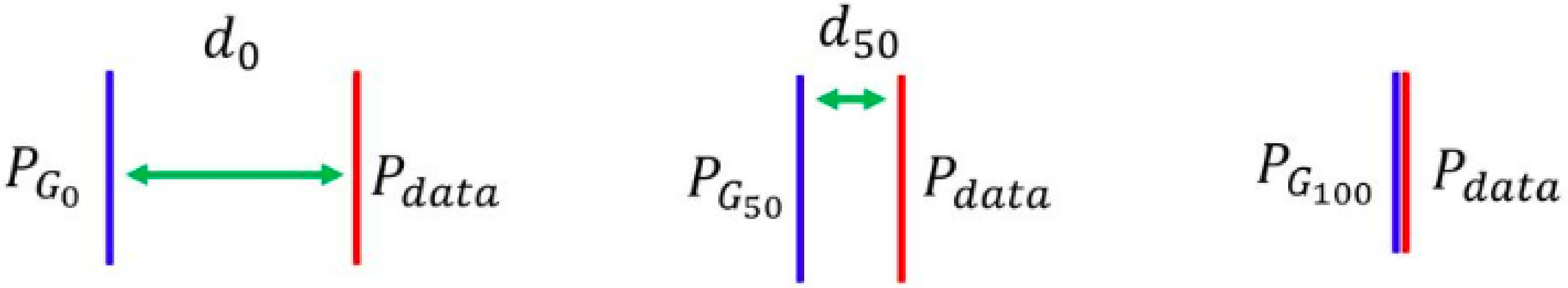
Which brings an evident connection to EMD.



Wikimedia

Two dimensional representation of the transport plan between horizontal (μ) and vertical ν pdfs. Note, that this is not unique plan. The inf must be taken over all possible plans.

Convergence Example



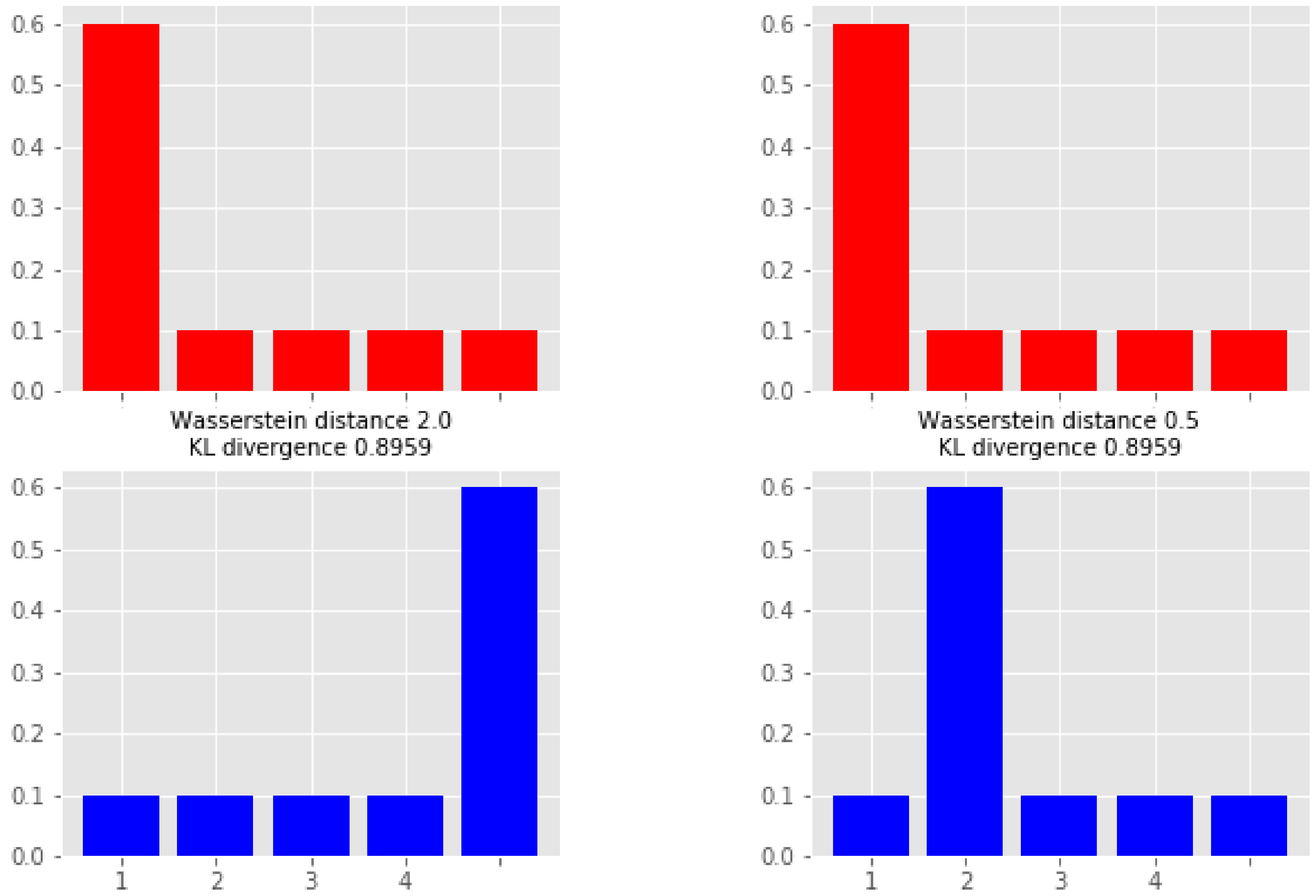
TV:	1	1	0
KL:	∞	∞	0
JS:	$\log 2$	$\log 2$	0
W:	d_0	d_{50}	0

Mass Attention

W takes into account the distance at which the differences in the distributions are located.

This is exactly what we need to take into account multiple solutions!

Wikimedia



W properties hints

P – true PDF, Q – fitted PDF.

- For a sequence of distributions Q_n:

  $KL(P||Q_n) \to 0 \rightarrow JS(P;Q_n) \to 0 \rightarrow W(P;Q_n) \to 0,\ Q_n \xrightarrow{\mathrm{D}} P$

- For $Q_\theta \sim g_\theta(z)$, $g_\theta(z)$ continuos

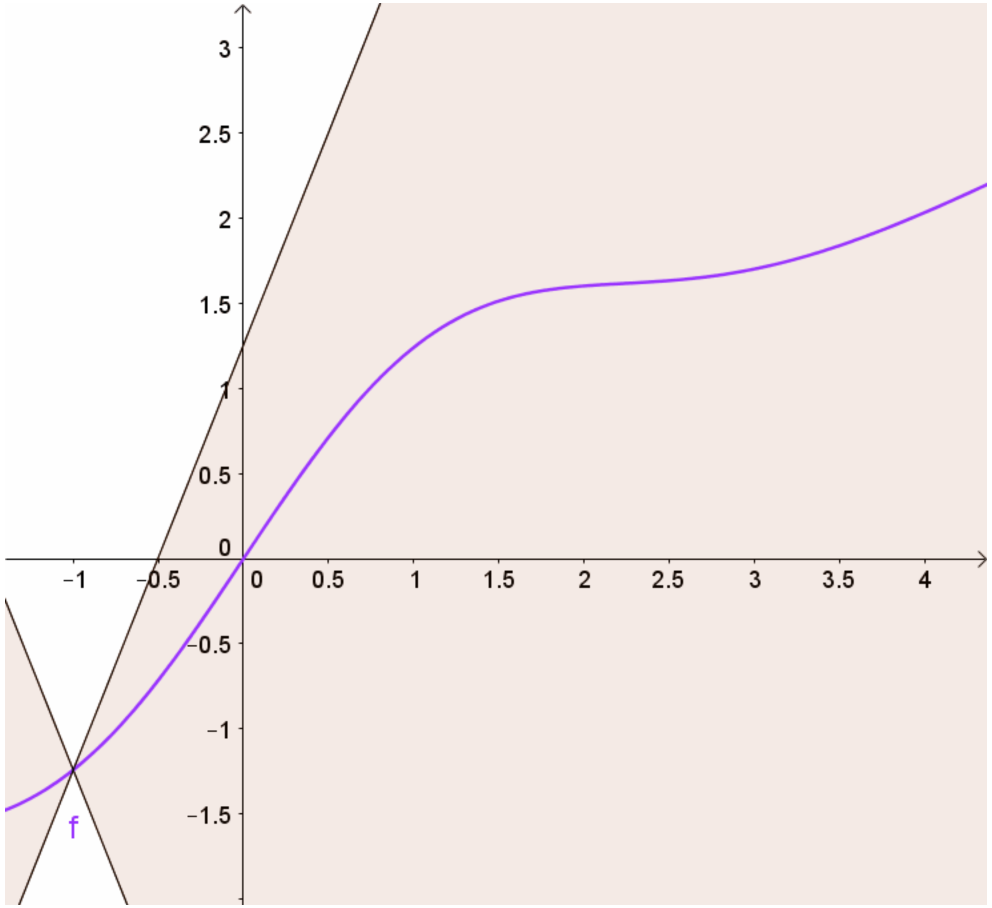
  $W(Q_\theta;Q)$ is continuous and can be restricted to differentiable almost everywhere.

Should we use directly in GAN?

Lipschitz continuity

- f is Lipschitz-k continuous if
- there exists a constant $k \geq 0$, such that for all x_1 and x_2:

$$|f(x_1) - f(x_2)| \leq \mathrm{k} \cdot \|x_1 - x_2\|$$



img from https://en.wikipedia.org/wiki/Lipschitz_continuity

Kantorovich-Rubinstein Duality

P – true PDF, Q – fitted PDF.

$$W(P;Q) = \sup_{f \in Lip_1} \left(\mathbb{E}_{x\sim P}\, f(x) - \mathbb{E}_{x\sim Q} f(x)\right),$$

where Lip_1 is 1-Lipshitz condition.

Integral Probability Metrics

$p(x), q(x)$ – PDF.

$$\gamma_{\mathcal{F}}(P, Q) = \sup\left\{\left|\int f\, dp(x) - \int f\, dq(x)\right| : f \in \mathcal{F}\right\}$$

$\mathcal{F}$ is a class of real-valued bounded measurable functions on S.

For $\mathcal{F} = \{f : ||f||_L \leq 1\}$, with 1-Lipschitz condition:

W_1 is **IPM** but **not f-divergence**

B. Sriperumbudur et al. On the empirical estimation of integral probability metrics
S. Nowozin, NIPS2016 workshop talk

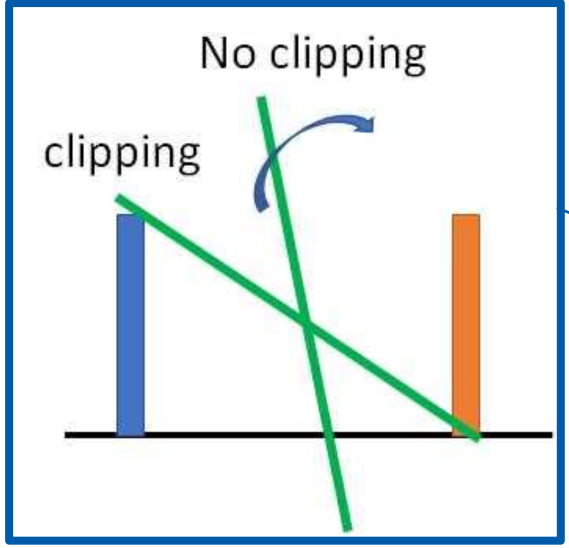
Conclusions

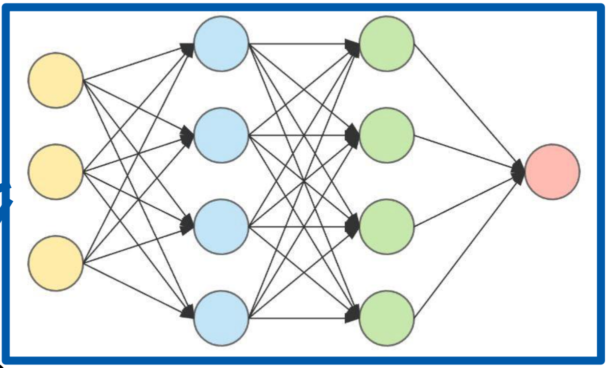
- Wasserstein-1 is a distance with desired properties.
- Kantorovich-Rubinstein duality connects Wasserstein-1 distance to IPM.
- Lipschitzeness is needed for above to work.
- Wasserstein-1 distance cannot directly be inserted into f-GAN style*.

*J. Song et al., Bridging the Gap Between f-GANs and Wasserstein GANs, ICML 2020

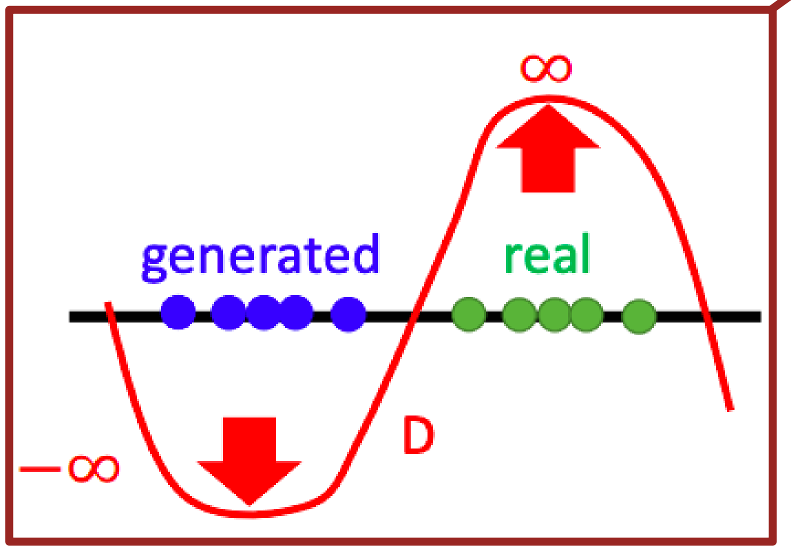
Wasserstein GAN

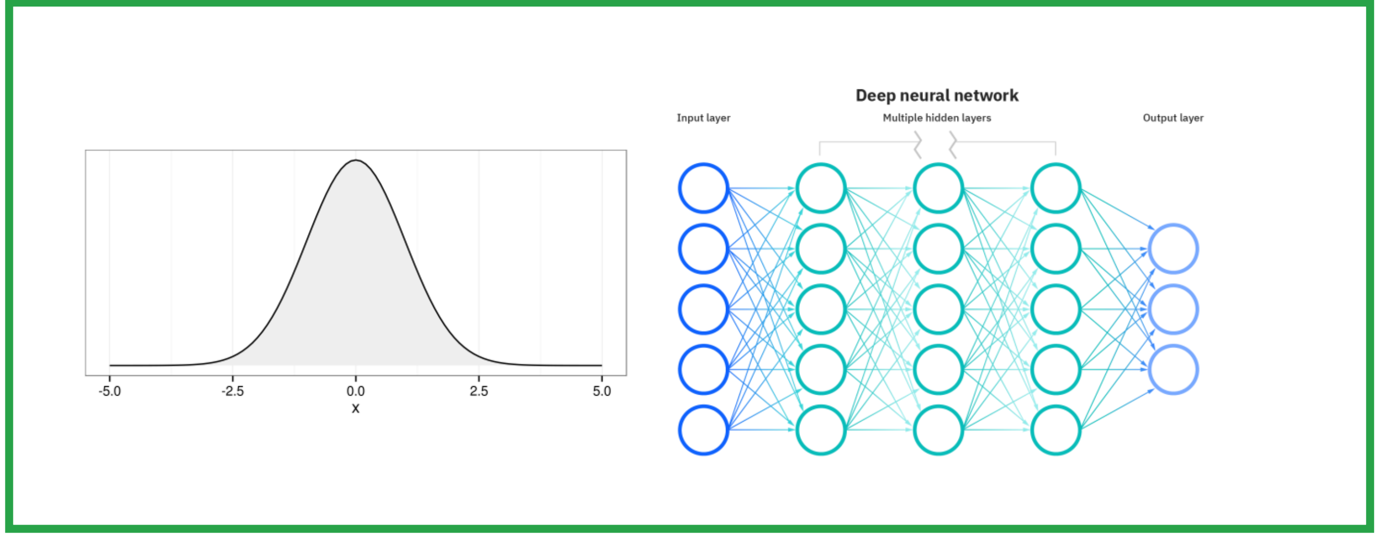
Lipschitz-1 Condition and Neural Networks





$$W(P;Q) = \sup_{f \in Lip_1} \left(\mathbb{E}_{x\sim P} f(x) - \mathbb{E}_{x\sim Q} f(x)\right),$$





Lipschitz-1 Condition and Neural Networks

$$W(P;Q) = \sup_{f \in Lip_1} \left(\mathbb{E}_{x\sim P} f(x) - \mathbb{E}_{x\sim Q} f(x) \right),$$

- f is a neural net – **discriminator** ('**critic**' in the original paper).
- The expectations is estimated from samples.
- Lipschitz-1 continuity can be replaced with Lipschitz-k continuity
  - estimate $k \times \mathrm{W}(P, Q)$
  - achieved **by clipping the weights** of the critic: $w \to \mathrm{clip}(w, -c, c)$ with some constant c.

M. Arjovsky et al. **Wasserstein GAN**

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.

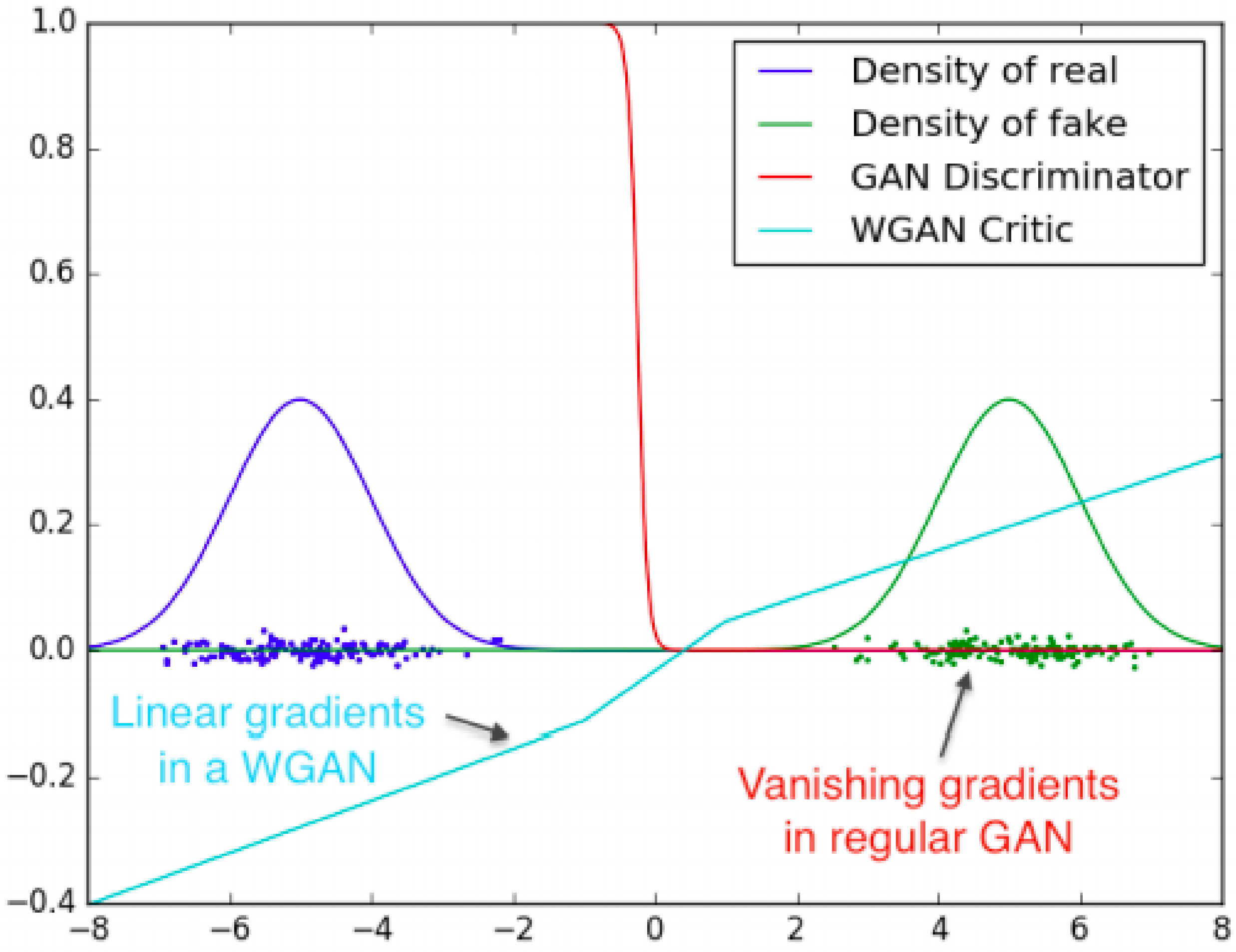
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

```
while θ has not converged do
    for t = 0, ..., n_critic do
        Sample {x^(i)}_{i=1}^m ~ P_r a batch from the real data.
        Sample {z^(i)}_{i=1}^m ~ p(z) a batch of prior samples.
        g_w ← ∇_w [ (1/m) Σ_{i=1}^m f_w(x^(i)) − (1/m) Σ_{i=1}^m f_w(g_θ(z^(i))) ]
        w ← w + α · RMSProp(w, g_w)
        w ← clip(w, −c, c)
    end for
    Sample {z^(i)}_{i=1}^m ~ p(z) a batch of prior samples.
    g_θ ← −∇_θ (1/m) Σ_{i=1}^m f_w(g_θ(z^(i)))
    θ ← θ − α · RMSProp(θ, g_θ)
end while
```

- **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: problems solved

- the vanishing gradient problem is **solved**;
- mode collapse problem is **addressed;**
- rom authors: Weight clipping is a clearly terrible way to enforce a Lipschitz constraint. :



https://arxiv.org/abs/1701.07875

WGAN: results

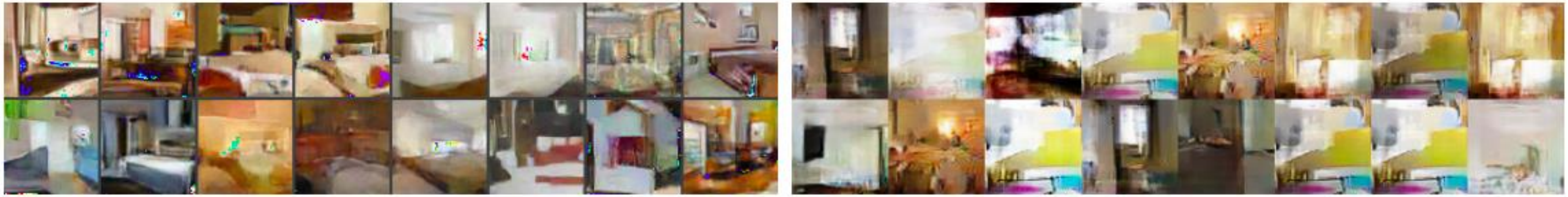


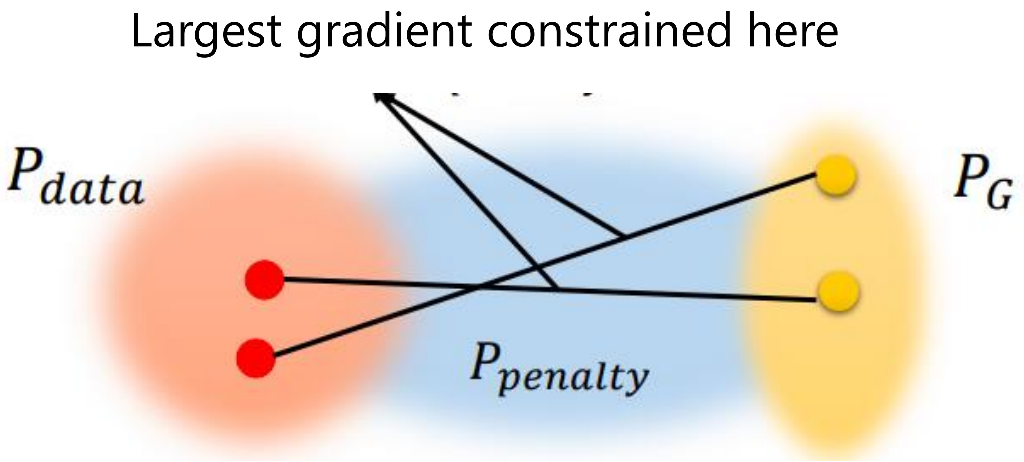
Figure 7: Algorithms trained with an MLP generator with 4 layers and 512 units with ReLU nonlinearities. The number of parameters is similar to that of a DCGAN, but it lacks a strong inductive bias for image generation. Left: WGAN algorithm. Right: standard GAN formulation. The WGAN method still was able to produce samples, lower quality than the DCGAN, and of higher quality than the MLP of the standard GAN. Note the significant degree of mode collapse in the GAN MLP.

WGAN-GP

- Weight clipping makes the critic less expressive and the training harder to converge
- Optimal f should satisfy $\|\nabla f\| = 1$ almost everywhere under P and Q
- Also: $\|f\|_L \leq 1 \iff \|\nabla f\| \leq 1$
- Can replace weight clipping with a gradient penalty term:

$$\mathrm{GP} = \lambda \int \max[(\|\nabla_{\tilde{x}} f(\tilde{x})\| - 1)^2]\, dx$$

$$\mathrm{GP} = \lambda \mathbb{E}_{\tilde{x} \sim \mathbb{P}_{\tilde{x}}}[(\|\nabla_{\tilde{x}} f(\tilde{x})\| - 1)^2]$$



$$\mathbb{P}_{\tilde{x}} : \begin{bmatrix} \tilde{x} = \alpha x_1 + (1-\alpha) x_2 \\ \alpha \sim \mathrm{Uniform}(0,1) \\ x_1 \sim P \\ x_2 \sim Q \end{bmatrix}$$

Ishaan Gulrajani **Improved Training of Wasserstein GANs**

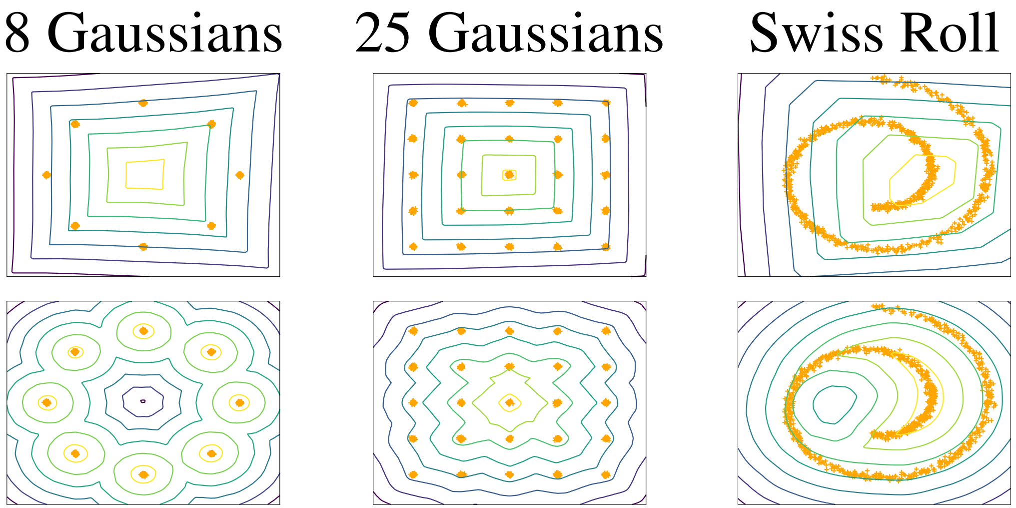
WGAN-GP

- Weight clipping makes the critic less expressive and the training harder to converge
- Optimal f should satisfy $\|\nabla f\| = 1$ almost everywhere under P and Q
- Also: $\|f\|_L \leq 1 \iff \|\nabla f\| \leq 1$
- Can replace weight clipping with a gradient penalty term:

$$\mathrm{GP} = \lambda \mathbb{E}_{\tilde{x}\sim\mathbb{P}_{\tilde{x}}}[(\|\nabla_{\tilde{x}} f(\tilde{x})\| - 1)^2]$$

or alternatively ('one-sided' penalty):

$$\mathrm{GP} = \lambda \mathbb{E}_{\tilde{x}\sim\mathbb{P}_{\tilde{x}}}[\max(0, \|\nabla_{\tilde{x}} f(\tilde{x})\| - 1)^2]$$



$$\mathbb{P}_{\tilde{x}} : \begin{bmatrix} \tilde{x} = \alpha x_1 + (1-\alpha) x_2 \\ \alpha \sim \mathrm{Uniform}(0,1) \\ x_1 \sim P \\ x_2 \sim Q \end{bmatrix}$$

Ishaan Gulrajani **Improved Training of Wasserstein GANs**

WGAN: spectral normalization

- Spectral normalisation proposes to use normalised weights:

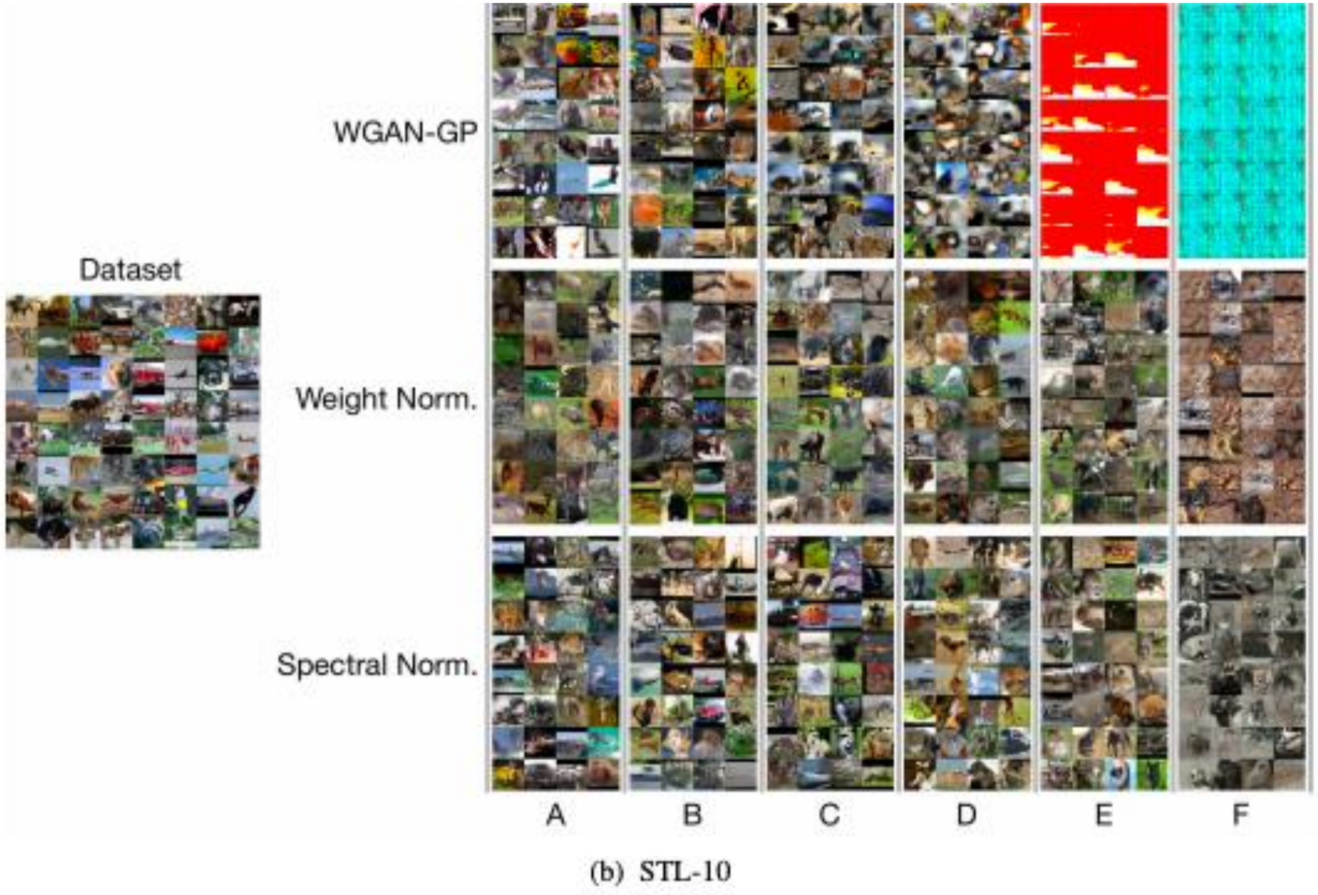
$$W_{SN} = \frac{W}{\sigma(W)}$$

where:

$$\sigma(W) = \max_{h:h\neq 0} \frac{||Wh||_2}{||h||_2}$$

- this gives constraints on gradient:

$$||f||_{Lip} \leq \prod_{i=1}^{l} \sigma(W_l).$$



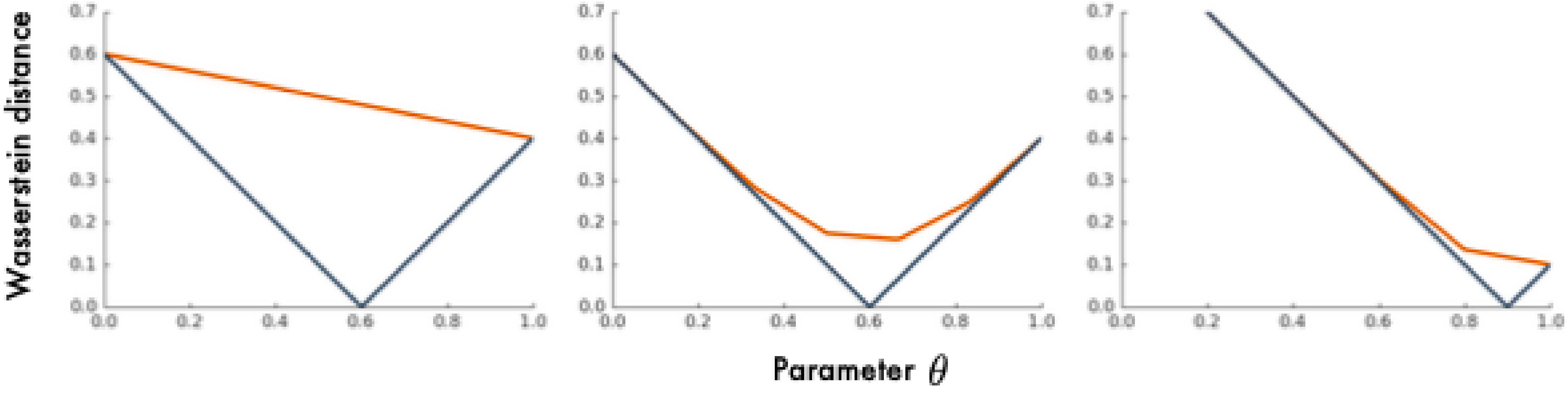
(b) STL-10

Figure 6: Generated images on different methods: WGAN-GP, weight normalization, and spectral normalization on CIFAR-10 and STL-10.

Miyato et al. Spectral Normalization for Generative Adversarial Networks, ICLR 2018

WGAN: problems

- The expected EMD gradients can differ from the true gradients.
- This leads to problems even for Bernoulli distribution.



Red for sample gradient expectation, blue is for real gradients solution.
Left to right $\theta^* = 0.6; 0.6; 0.9.$

M. Bellemare et al. The Cramer Distance as a Solution to Biased Wasserstein Gradients

Conclusions

- WGAN is a power generative model.
- Simpler training procedure but need to control Lipschitz continuity
- Several ideas how do this.
- Still problems:
  - Kantorovich-Rubinstein duality only mimicked;
  - gradient is stuck near solutions.