CS6101
Deep Unsupervised Learning
Recess Week

WGAN, WGAN-GP, Progressive GAN

Ang Yi Zhe

Course Page

Wasserstein GAN

Problems with Vanilla GANs

- Unstable training - hard to achieve Nash Equilibrium
- Low dimensional supports
- Vanishing gradient
- Mode Collapse
- Lack of a proper evaluation metric
- Not robust to architectures and hyperparameter choices

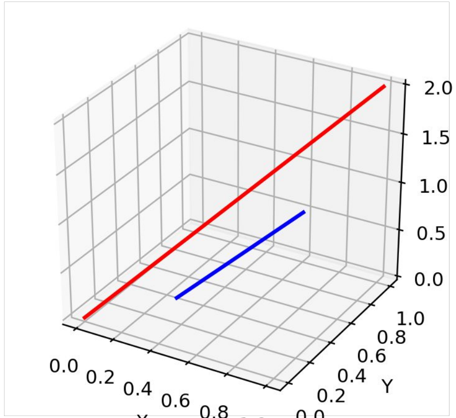
Problems with GANs - Low Dimensional Supports

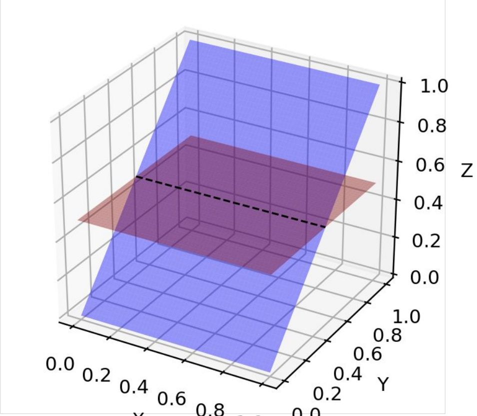
Problem arises when **supports of Pr and Pg** lie on **low dimensional manifolds**

→ Disjoint Supports

→ Easily find perfect discriminator

→ No gradient signal during training





[Towards Principled Methods for Training Generative Adversarial Networks]

Wasserstein GAN

A new GAN training algorithm

- Good **empirical** results backed up by **theory**
- Able to train the discriminator to **convergence**
    - Removing the need to balance discriminator/generator updates.
- Correlation between **discriminator loss** and **perceptual quality**
    - Easier to gauge training progress and determine stopping criteria.

Wasserstein GAN

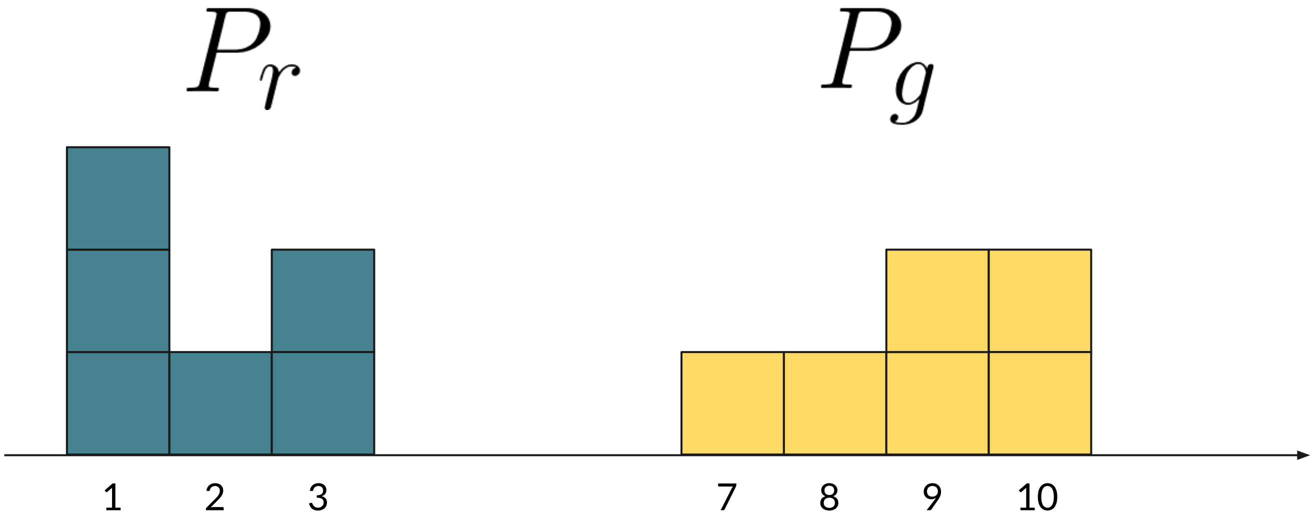
Wasserstein Distance - An Alternative Divergence Measure

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

Minimum energy cost of **moving and transforming** a pile of dirt in the shape of one probability distribution, to the shape of the other distribution, where

Energy cost = Amount of Dirt * Moving Distance

Wasserstein Distance - Explained



Transport Plan

γ

	7	8	9	10
1				
2				
3				

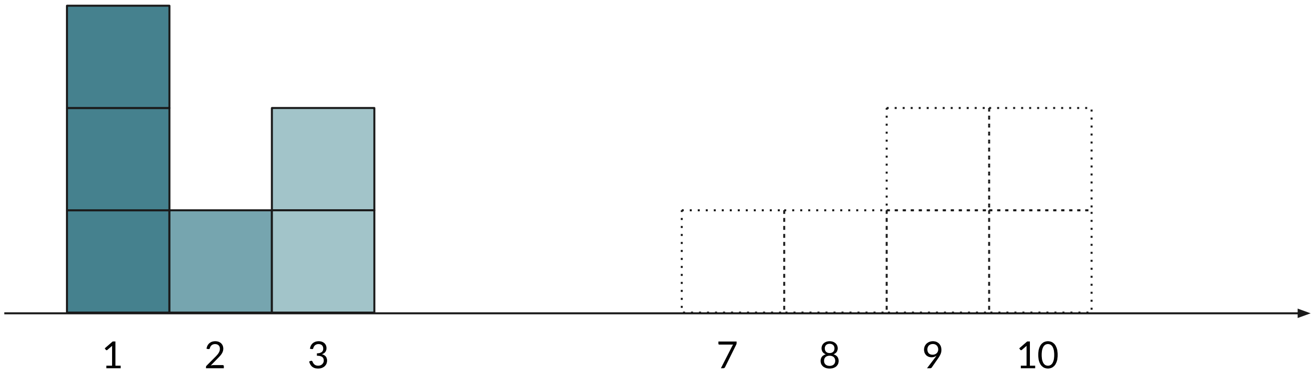
Adapted from: [Jonathan Hui, Medium]

Wasserstein Distance - Explained

Transport Plan

γ

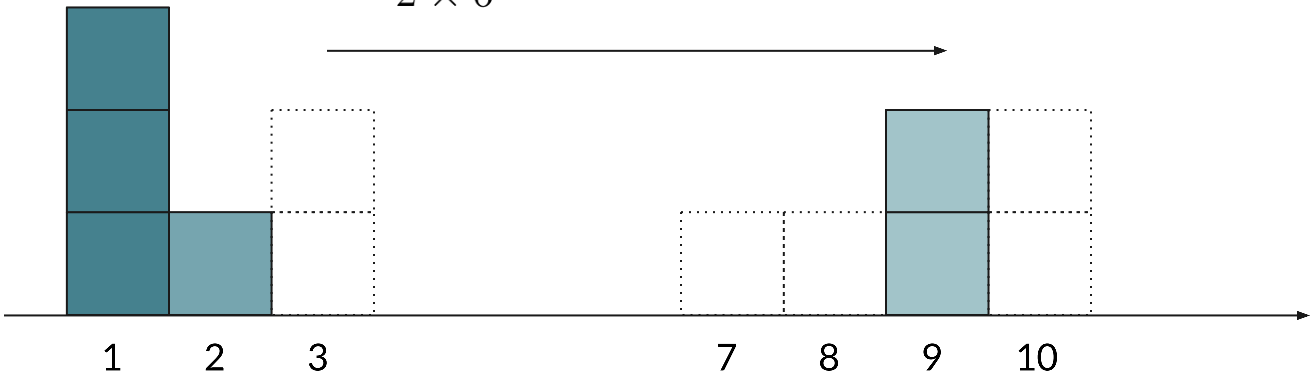
	7	8	9	10
1				
2				
3				



Adapted from: Jonathan Hui, Medium

Wasserstein Distance - Explained

$$\begin{aligned} Energy\ Cost &= Amount\ of\ \ Dirt \times Moving\ Distance \\ &= \gamma(x, y)\, \|x - y\| \\ &= 2 \times 6 \end{aligned}$$



Transport Plan

γ

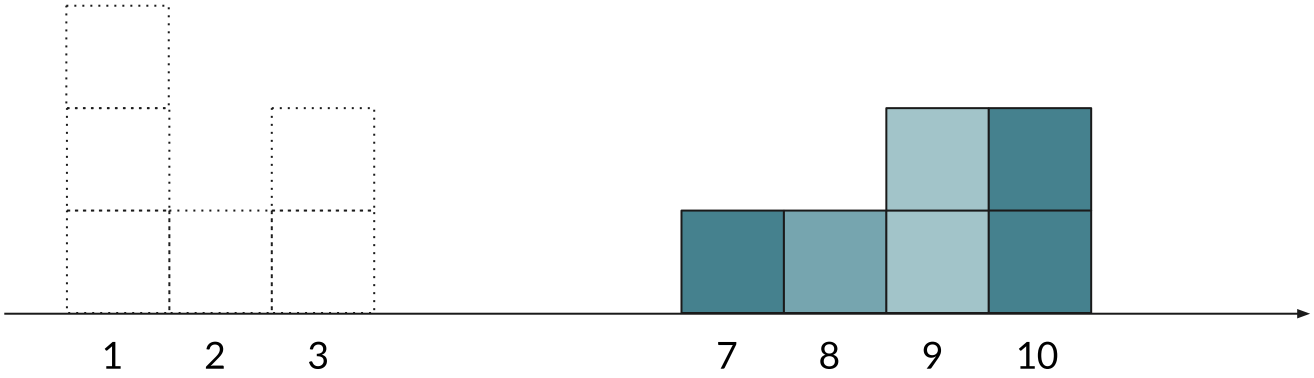
	7	8	9	10
1				
2				
3			2	

$$\gamma(3, 9) = 2$$

Adapted from: [Jonathan Hui, Medium]

Wasserstein Distance - Explained

$$Total\ Energy\ Cost = \sum_{x,y} \gamma(x,y)\,\|x-y\| = \mathbb{E}_{(x,y)\sim\gamma}\,\|x-y\|$$



γ

Transport Plan

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

Adapted from: [Jonathan Hui, Medium]

Wasserstein Distance - Explained

γ_i

P_g

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

P_r

$$\in \Pi(P_r, P_g)$$

Set of all possible Joint Probability Distributions between Pr and Pg

Wasserstein Distance - Explained

$$W(P_r, P_g) = \underbrace{\inf_{\gamma \in \Pi(P_r, P_g)}}_{\text{Find the smallest value among all valid transport plans}} \underbrace{\mathbb{E}_{(x,y)\sim\gamma}\big[\|x - y\|\big]}_{\text{Sum of distance moved, weighted by the amount of mass moved}}$$

Minimum energy cost of **moving and transforming** a pile of dirt in the shape of one probability distribution, to the shape of the other distribution, where

Energy cost = Amount of Dirt * Moving Distance

Comparison of Distance Measures

KL-Divergence

$$D_{KL}(P\|Q) = \int_x P(x) \log \frac{P(x)}{Q(x)} dx$$

JS-Divergence

$$D_{JS}(P\|Q) = \frac{1}{2} D_{KL}(P\|\frac{P+Q}{2}) + \frac{1}{2} D_{KL}(Q\|\frac{P+Q}{2})$$

Wasserstein Distance

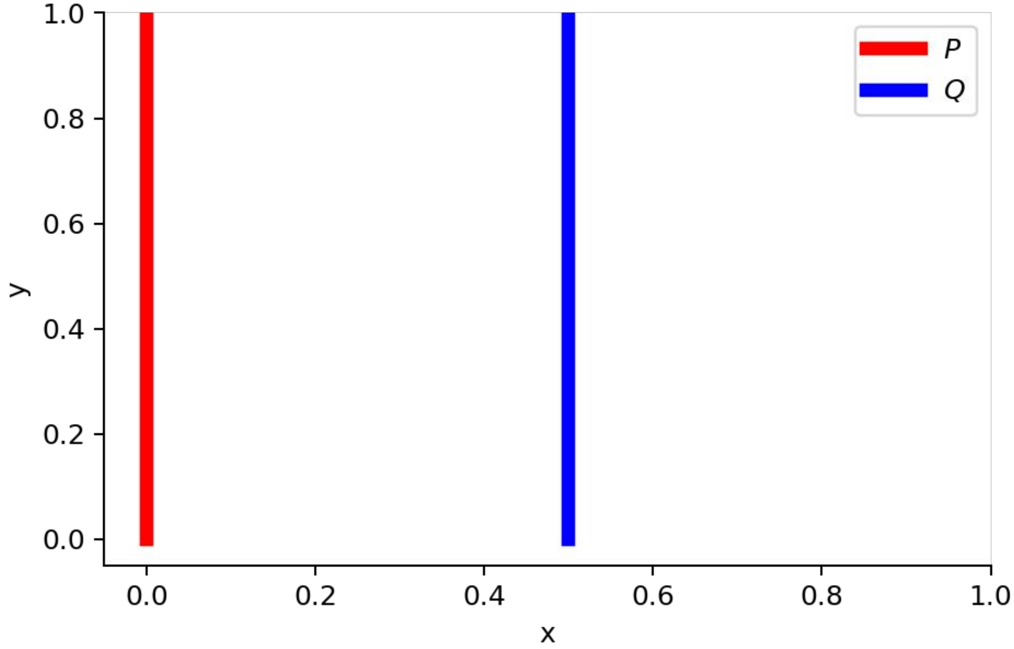
$$W(P,Q) = \inf_{\gamma \in \Pi(P,Q)} \mathbb{E}_{(x,y)\sim\gamma}[||x-y||]$$

How do these various **measures perform** when both the real and generator's data lie on **low dimensional manifolds**?

Comparison of Distance Measures

$$\forall(x, y) \in P, x = 0 \text{ and } y \sim U(0, 1)$$

$$\forall(x, y) \in Q, x = \theta, 0 \leq \theta \leq 1 \text{ and } y \sim U(0, 1)$$



What is the distance between these **two disjoint distributions**?

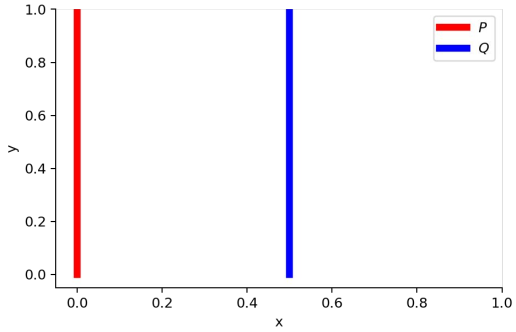
Wasserstein GAN

Comparison of Distance Measures

$$D_{KL}(P\|Q) = D_{KL}(Q\|P) = \begin{cases} +\infty & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0, \end{cases}$$

$$D_{JS}(P\|Q) = \begin{cases} \log 2 & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0, \end{cases}$$

$$W(P, Q) = |\theta|$$



There exist cases for **KL and JS** where,

- The distributions don't converge
- The gradient is always 0

→ **WD is best**; Provides a smooth measure

[Wasserstein GAN]

Kantorovich-Rubinstein Duality

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

is intractable, so the paper shows how we can compute an approximation:

Find the largest value among all K-Lipschitz continuous functions

$$\sup_{\|f\|_L \leq K} \mathbb{E}_{x\sim P_r}[f(x)] - \mathbb{E}_{x\sim P_g}[f(x)]$$

Suppose f comes from a family of K-Lipschitz continuous functions, $\{f_w\}_{w\in W}$, parameterized by w,

$$\geq \max_{w\in W} \mathbb{E}_{x\sim P_r}[f_w(x)] - \mathbb{E}_{x\sim P_g}[f_w(x))]$$

To learn **w** to find a good **f**$_{\mathbf{w}}$ to approximate the Wasserstein Distance between **P**$_{\mathbf{r}}$ and **P**$_{\mathbf{g}}$ → use a neural network!

WGAN Training

To train $P_g = g_\theta(Z)$ to match P_r using the Wasserstein Distance,

$$W(P_r, P_g) = \max_{w \in W} \mathbb{E}_{x \sim P_r}[f_w(x)] - \mathbb{E}_{z \sim Z}[f_w(g_\theta(z))]$$

01 For a fixed generator, sample from real data and generator to train $\mathbf{f}_w$ to convergence using gradient ascent, in order to approximate the Wasserstein Distance.

$$\nabla W(P_r, P_g) = -\mathbb{E}_{z \sim Z}[\nabla_\theta f_w(g_\theta(z))]$$

02 Sample from the generator, and use the approximate Wasserstein Distance to train the generator using gradient descent.

03 Repeat.

Similar to original **minimax GAN** setup!

WGAN vs GAN

Vanilla GAN

$$\min_G \max_D \mathbb{E}_{x \sim P_r}[\log D(x)] + \mathbb{E}_{z \sim Z}[\log(1 - D(G(z)))]$$

WGAN

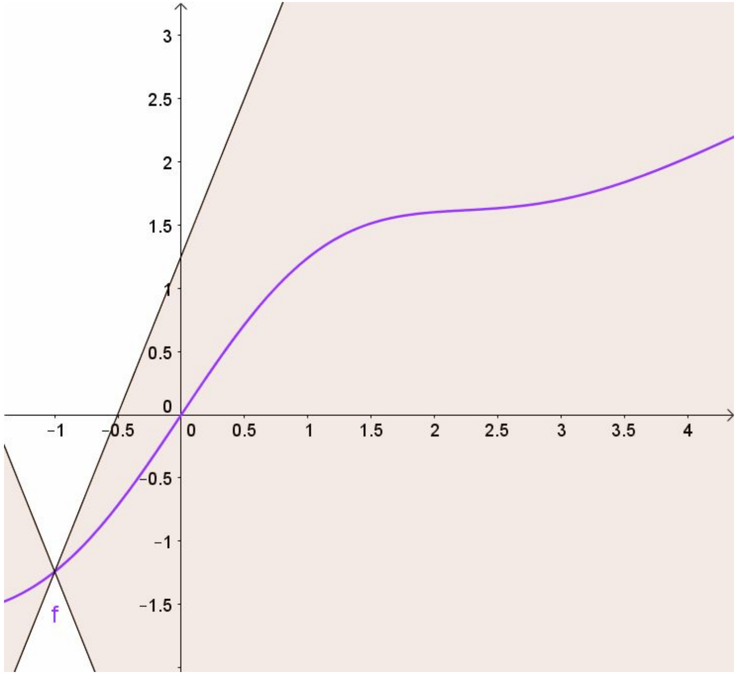
$$\min_G \underbrace{\max_D}_{\text{K-Lipschitz functions}} \mathbb{E}_{x \sim P_r}[D(x)] - \mathbb{E}_{z \sim Z}[D(G(z))]$$

- Uses Wasserstein Loss
- Predictions no longer constrained to [0, 1], but can be any real number
- Critic must be K-Lipschitz continuous (by clipping the weights)
- Train the critic multiple times for each update of generator

K-Lipschitz Continuity

There exists a real constant $K \geq 0$ s.t. for all $x_1, x_2 \in \mathbb{R}$,

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|, \qquad \left|\frac{f(x_1) - f(x_2)}{x_1 - x_2}\right| \leq K$$



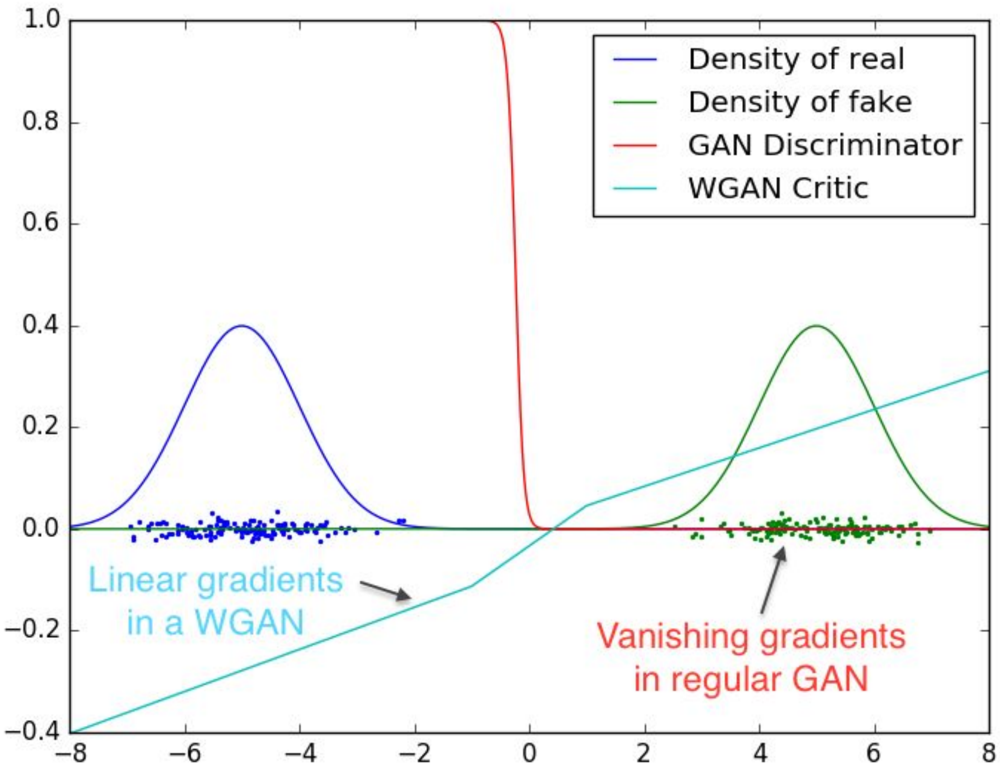
We require a **limit** on the rate at which the **predictions** can change between any **two images**

Enforce the Lipschitz constraint by **clipping the weights** of the critic to lie within a small range, after each training batch.

Taken from: [Wikipedia]

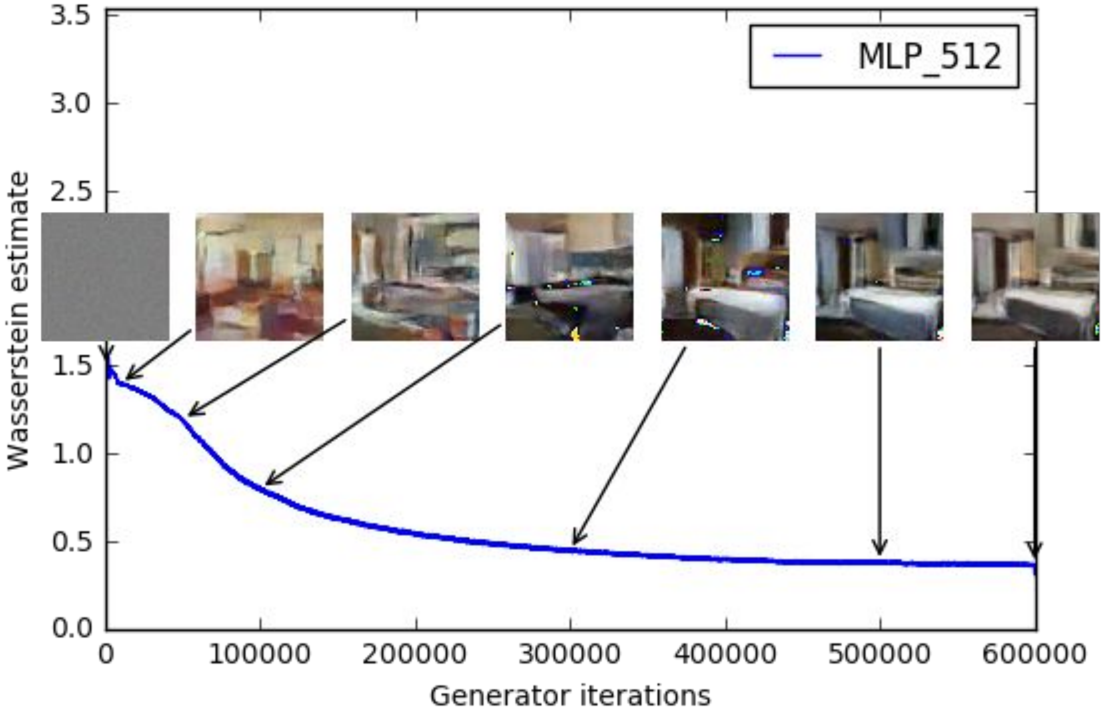
[Spectral Normalization Explained, Christian]

Results - Nice Gradients



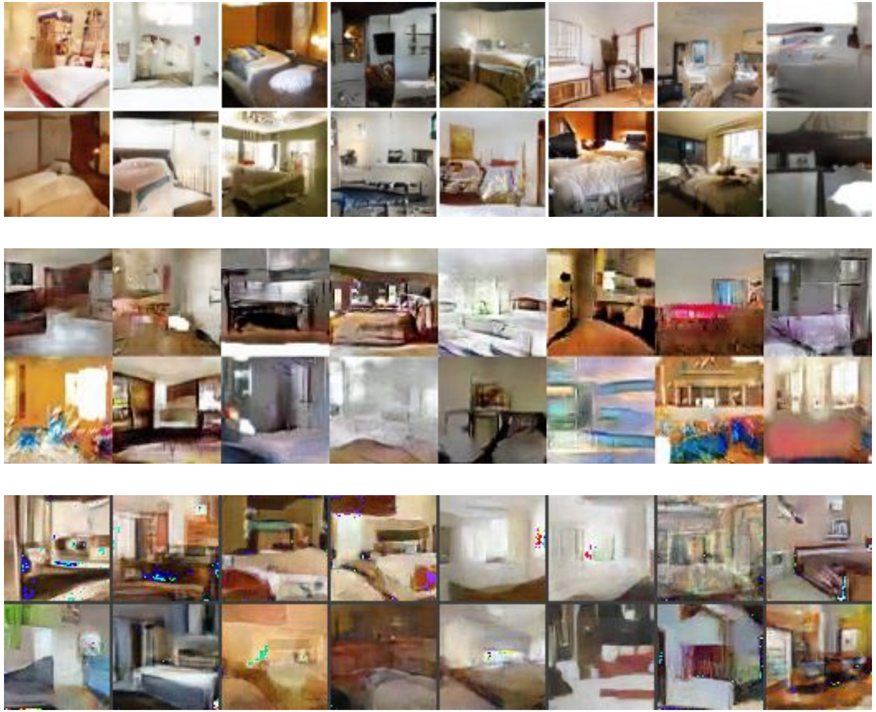
Taken from: [Wasserstein GAN]

Results - Correlates with Image Quality

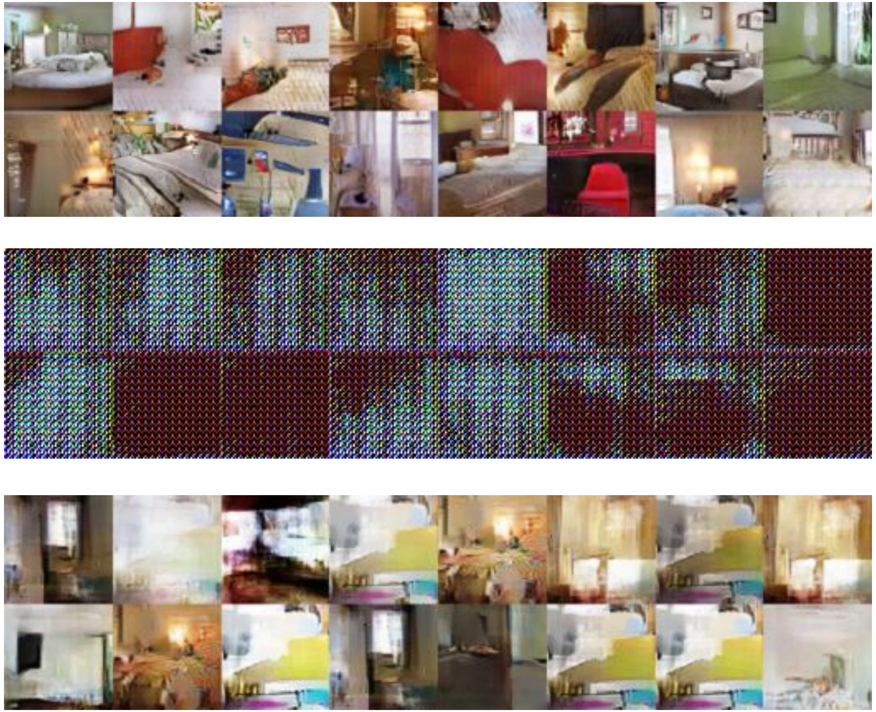


Taken from: [Wasserstein GAN]()

Results - Robust to Architectural Changes



WGAN



Vanilla GAN

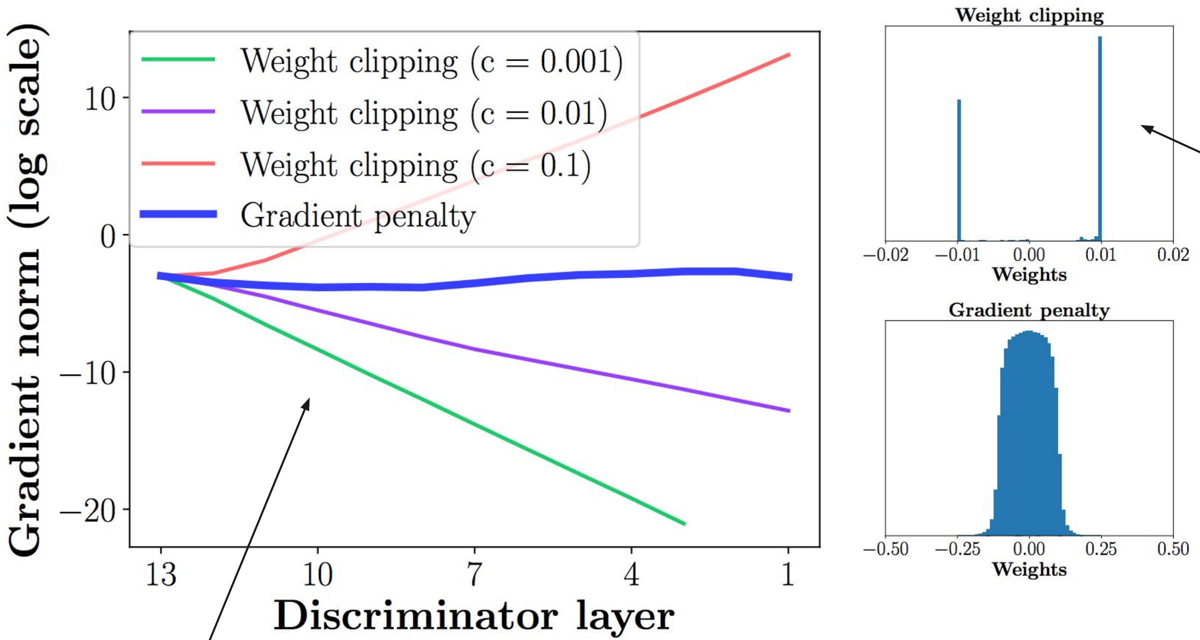
Taken from: [Wasserstein GAN]()

Wasserstein GAN - GP

Problems with WGAN

“**Weight clipping** is clearly a terrible way to enforce a Lipschitz constraint” - Authors

Weight Clipping - Instability in Training

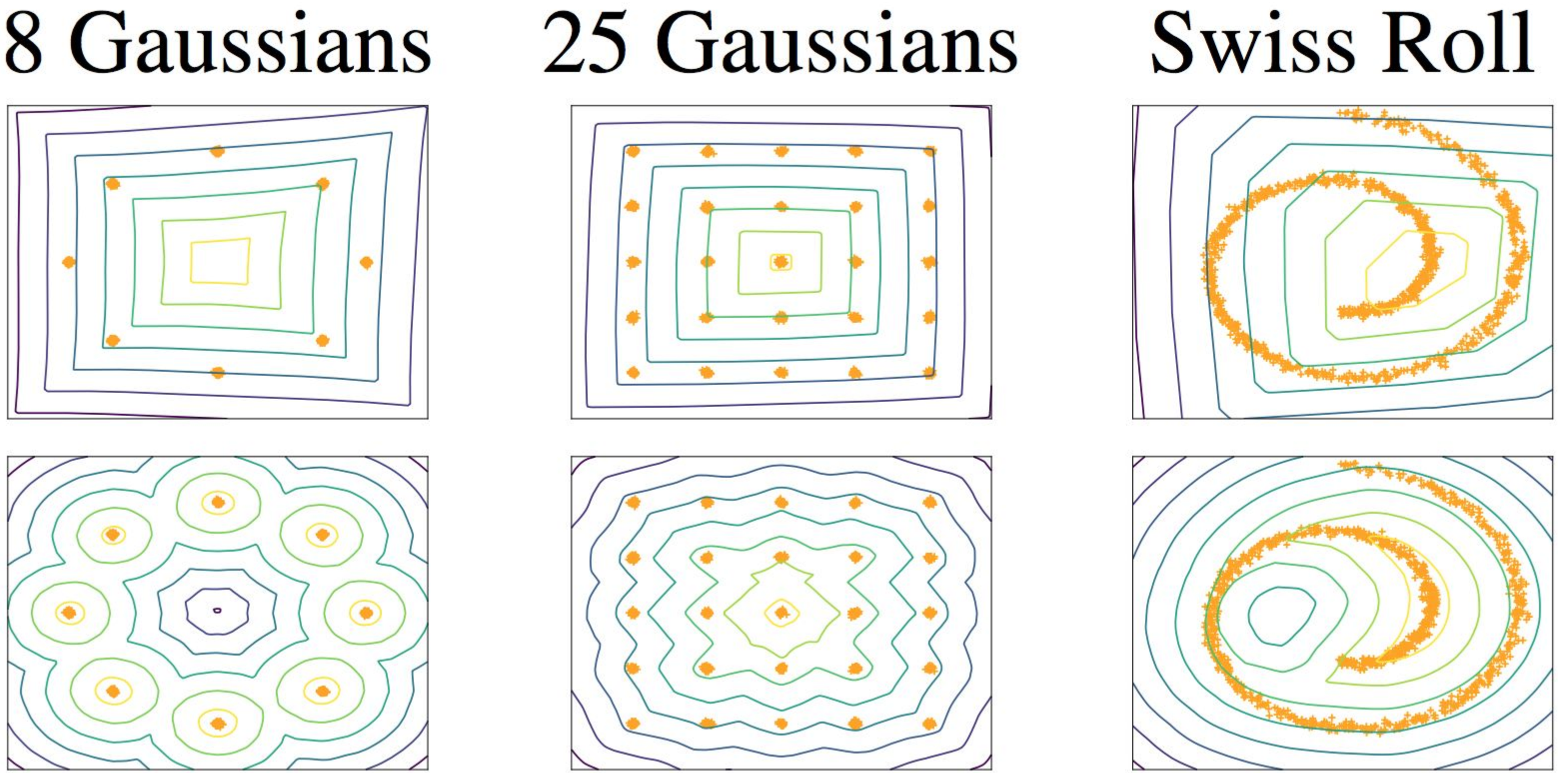


Weights concentrated at lower/upper bounds of clipping interval

Vanishing / Exploding Gradients

Taken from: [Improved Training of Wasserstein GANs]()

Weight Clipping - Reduced Capacity of Model



Taken from: [Improved Training of Wasserstein GANs]()

Gradient Penalty

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

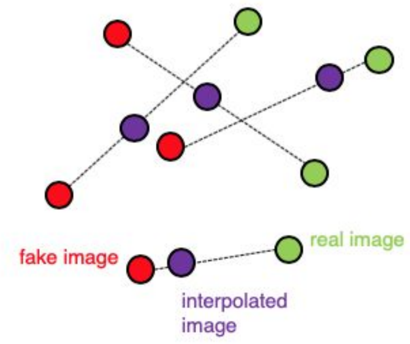
$$\max_D \mathbb{E}_{x\sim P_r}[D(x)] - \mathbb{E}_{\tilde{x}\sim P_g}[D(\tilde{x})] + \lambda \mathbb{E}_{\hat{x}\sim P_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]$$

Sample from a linear interpolation between real and fake samples

Norm of gradient of critic w.r.t. input

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

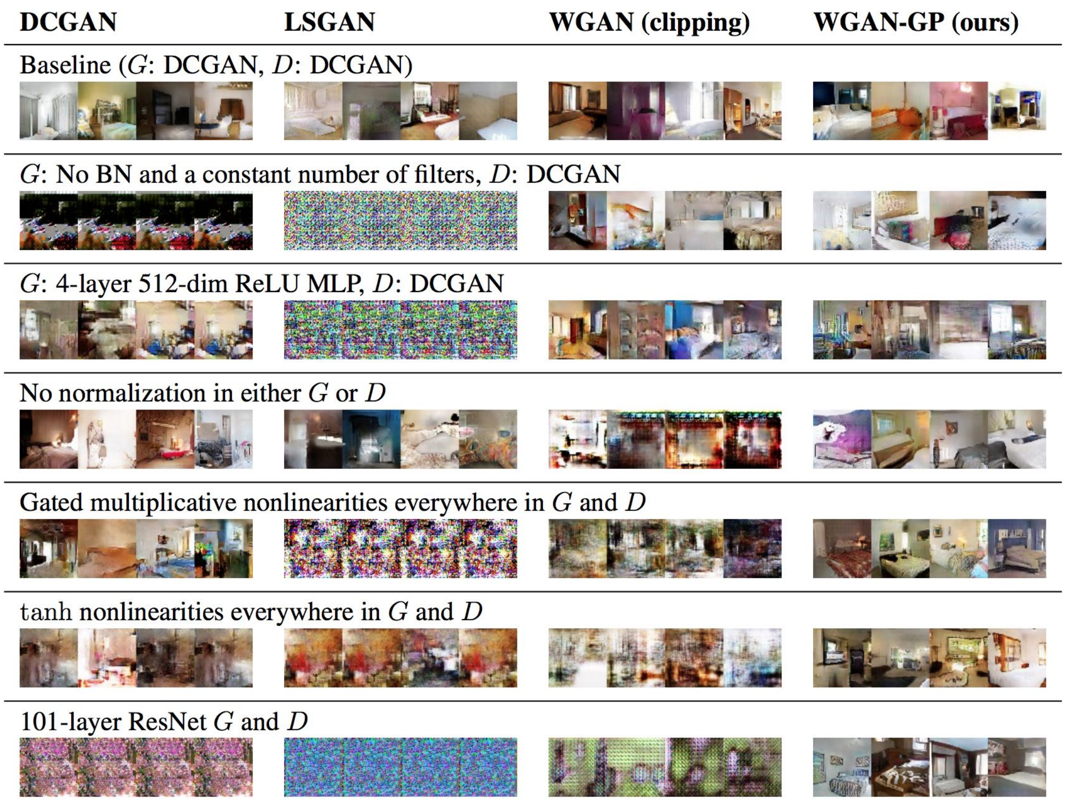
Penalise when value is away from 1



WGAN-GP vs WGAN

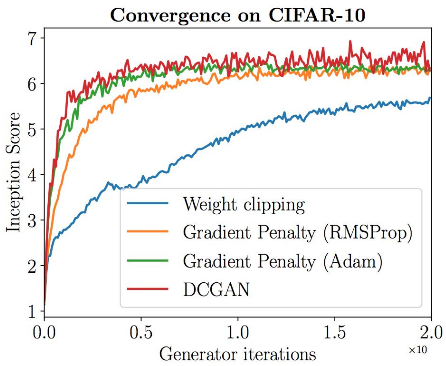
- Include a gradient penalty term in the critic loss function
- Don’t clip the weights of the critic
- Don’t use batch normalization layers in the critic
- More computationally intensive

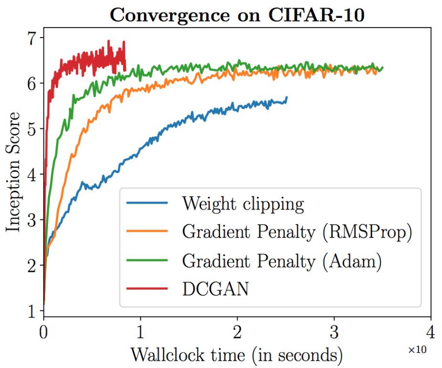
Results - Enhanced Training Stability



Taken from: [Improved Training of Wasserstein GANs]()

Results - Enhanced Training Stability





Unsupervised

Method	Score
ALI [8] (in [27])	$5.34 \pm .05$
BEGAN [4]	5.62
DCGAN [22] (in [11])	$6.16 \pm .07$
Improved GAN (-L+HA) [23]	$6.86 \pm .06$
EGAN-Ent-VI [7]	$7.07 \pm .10$
DFM [27]	$7.72 \pm .13$
WGAN-GP ResNet (ours)	$7.86 \pm .07$

Supervised

Method	Score
SteinGAN [26]	6.35
DCGAN (with labels, in [26])	6.58
Improved GAN [23]	$8.09 \pm .07$
AC-GAN [20]	$8.25 \pm .07$
SGAN-no-joint [11]	$8.37 \pm .08$
WGAN-GP ResNet (ours)	$8.42 \pm .10$
SGAN [11]	$8.59 \pm .12$

Taken from: Improved Training of Wasserstein GANs

Progressive GAN

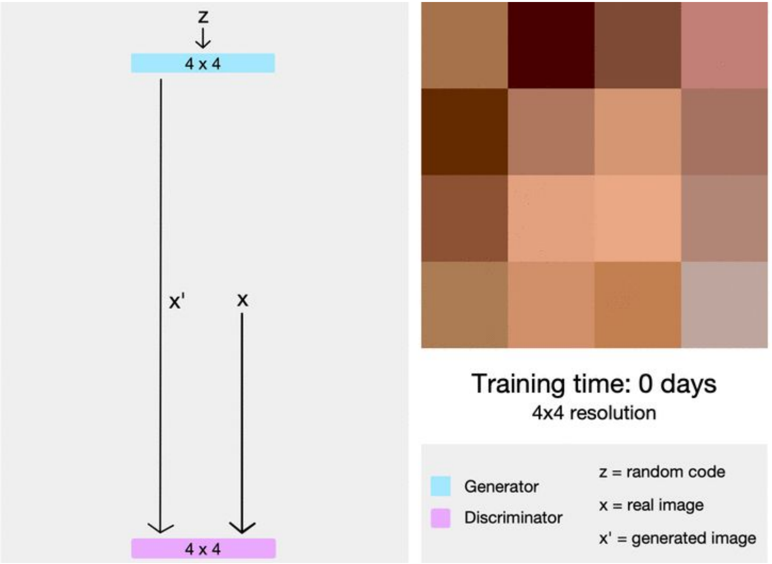
Key Innovations of ProGAN

- Progressively growing and smoothly fading in higher-resolution layers
- Mini-batch standard deviation
- Equalized learning rate
- Pixel-wise feature normalization
- **1024x1024 images (or even more)!**

Progressive Growing of GANs for Improved Quality, Stability, and Variation

Progressive Growing

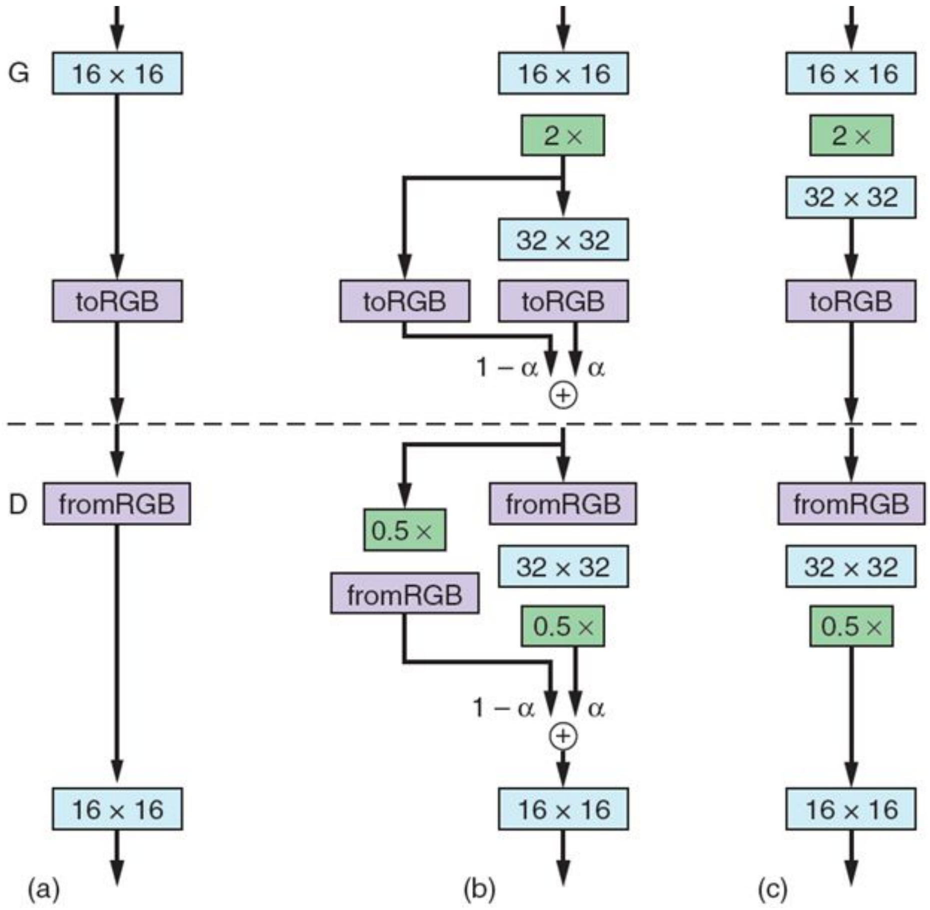
Instead of training all the layers of the generator and discriminator at once, we gradually grow the GAN, **one layer at a time**, to handle progressively higher resolution images



Start off with an easy to traverse loss landscape, and then gradually increase the complexity as we get closer to the objective

Taken from: [Sarah Wolf, Medium]

Smooth Progressive Growing



Introduce 2 new pathways:

1. Nearest Neighbour Upscaling
   a. Upscale old output
2. NNU + Convolution Layer
   a. Learned upscaling

Introduce new layer but also retain some of the previous output, **smoothly** (linearly) fading in the new layer

Mini-batch Standard Deviation

Introduce a “minibatch standard deviation” layer near the end of the discriminator - computes the standard deviations of the feature map pixels across the batch, and appends as an extra channel.

- A way for the Discriminator to tell whether the samples it is getting are varied enough
- If SD is low → fake
- Encourage Generator to increase variance of generated samples

Pixel-wise Feature Normalization

Normalize the feature vector in each pixel to have unit norm in the generator after each convolutional layer.

- Stability of training
- Less memory intensive than batch-norm
- Prevent feature map magnitudes from getting too large