GAN Crash Course

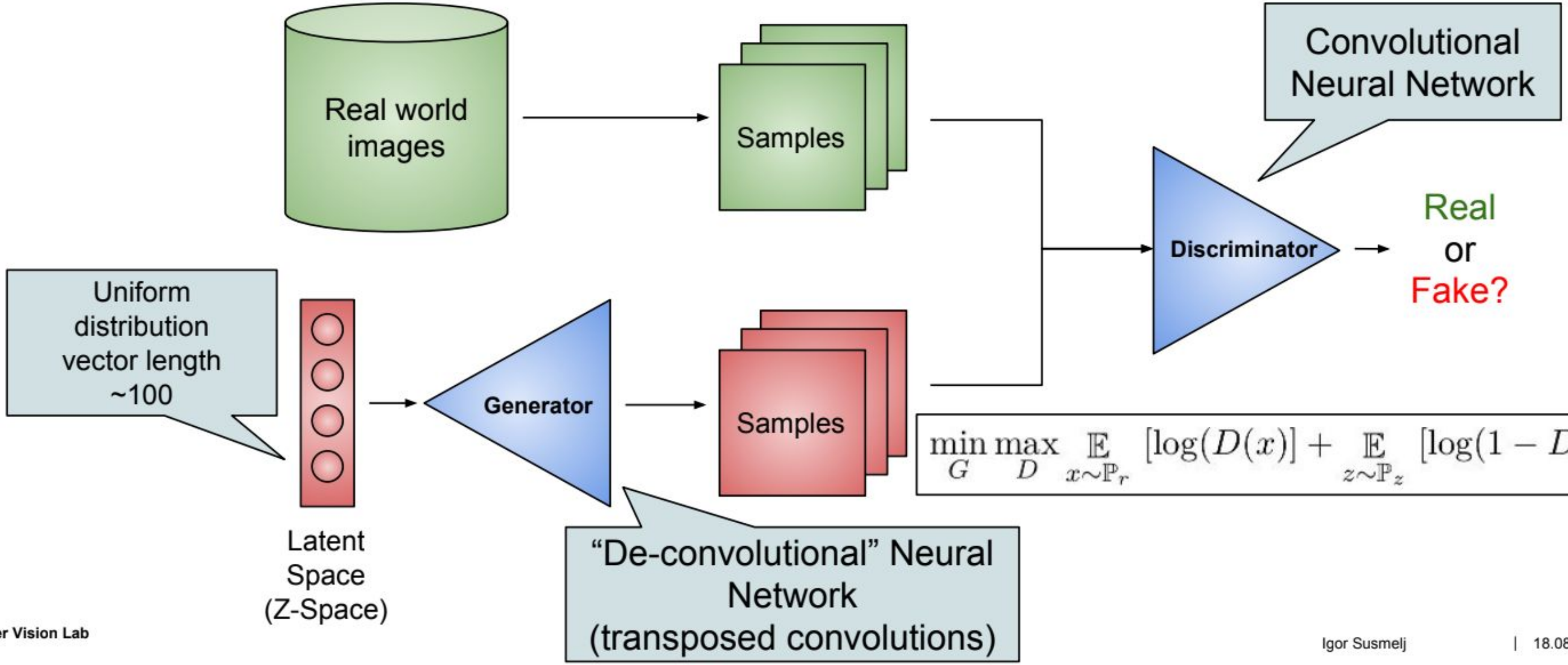
Eirikur Agustsson

Outline

- Standard GAN, theory and limitations
- Fixes and alternatives
- Wasserstein GAN
- Kantorovich duality and Lipschitz continuity
- Spectral Normalization GAN
- Improved Wasserstein GAN
- Pix2Pix & CycleGAN
- Progressive GANs

Generative Adversarial Network (GAN)

- What is a GAN and how does it work?



$$\min_G \max_D \mathop{\mathbb{E}}_{x\sim\mathbb{P}_r}[\log(D(x)] + \mathop{\mathbb{E}}_{z\sim\mathbb{P}_z}[\log(1 - D(G(z)))]$$

Theory time!

Standard GAN ideal case

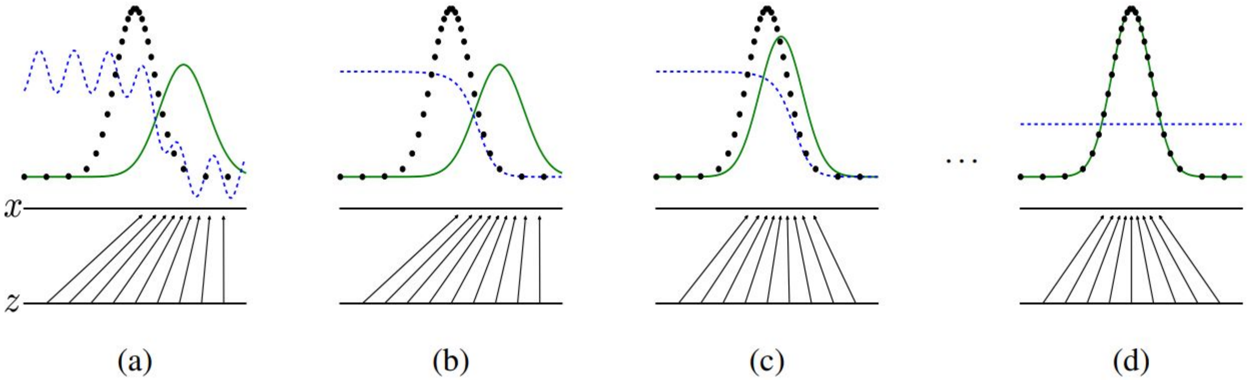
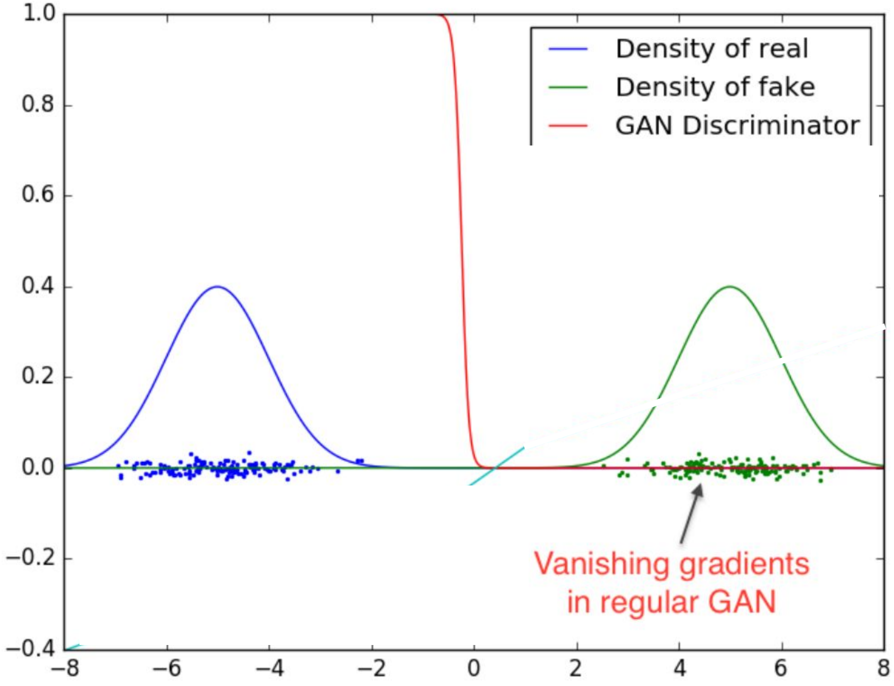


Figure 1: Generative adversarial nets are trained by simultaneously updating the **d**iscriminative distribution (D, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\boldsymbol{x}}$ from those of the **g**enerative distribution p_g (G) (green, solid line). The lower horizontal line is the domain from which $\boldsymbol{z}$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $\boldsymbol{x}$. The upward arrows show how the mapping $\boldsymbol{x} = G(\boldsymbol{z})$ imposes the non-uniform distribution p_g on transformed samples. G contracts in regions of high density and expands in regions of low density of p_g. (a) Consider an adversarial pair near convergence: p_g is similar to p_{data} and D is a partially accurate classifier. (b) In the inner loop of the algorithm D is trained to discriminate samples from data, converging to $D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x})+p_g(\boldsymbol{x})}$. (c) After an update to G, gradient of D has guided $G(\boldsymbol{z})$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if G and D have enough capacity, they will reach a point at which both cannot improve because $p_g = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(\boldsymbol{x}) = \frac{1}{2}$.

What actually happens (Vanishing gradients)

- Gradient for G zero
- The 'log trick' of original GAN paper instead gives infinite gradients for G
- Either way, you don't converge
-



Example 1 (Learning parallel lines). Let $Z \sim U[0,1]$ the uniform distribution on the unit interval. Let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$ (a 0 on the x-axis and the random variable Z on the y-axis), uniform on a straight vertical line passing through the origin. Now let $g_\theta(z) = (\theta, z)$ with θ a single real parameter. It is easy to see that in this case,

- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \, , \\ 0 & \text{if } \theta = 0 \, , \end{cases}$

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

Wasserstein - Alternative distribution distance

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

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}\big[\, \|x - y\| \,\big] \,, \tag{1}$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$. The EM distance then is the "cost" of the optimal transport plan.

Going from min-min to min-max, without coupling

On the other hand, the Kantorovich-Rubinstein duality [22] tells us that

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)] \tag{2}$$

where the supremum is over all the 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$. Note that if we replace $\|f\|_L \leq 1$ for $\|f\|_L \leq K$ (consider K-Lipschitz for some constant K), then we end up with $K \cdot W(\mathbb{P}_r, \mathbb{P}_g)$. Therefore, if we have a parameterized family of

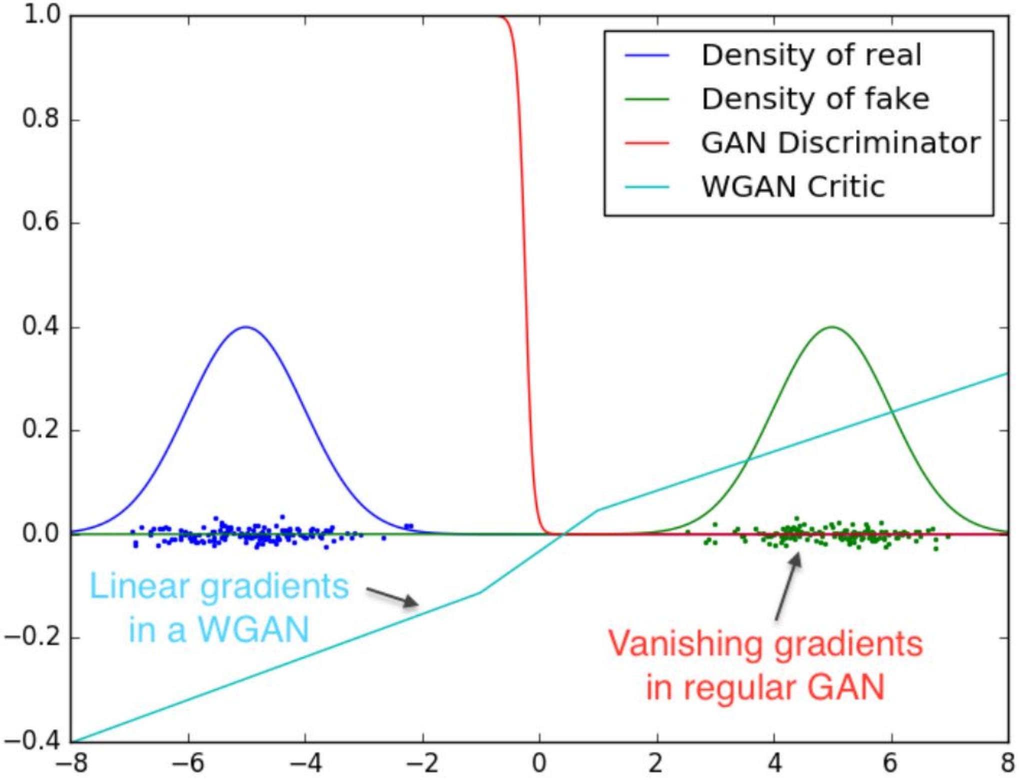
Lipschitz whaat?

Given two metric spaces (X, d_X) and (Y, d_Y), where d_X denotes the metric on the set X and d_Y is the metric on set Y, a function $f : X \to Y$ is called **Lipschitz continuous** if there exists a real constant $K \geq 0$ such that, for all x_1 and x_2 in X,

$$d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2).$$ [3]

Any such K is referred to as **a Lipschitz constant** for the function f. The smallest constant is sometimes called **the (best) Lipschitz constant**; however, in most cases, the latter notion is less relevant. If $K = 1$ the function is called a **short map**, and if $0 \leq K < 1$ the function is called a **contraction**.

Much better for stability! (No vanishing gradients)



Example 1 (Learning parallel lines). Let $Z \sim U[0,1]$ the uniform distribution on the unit interval. Let $\mathbb{P}_0$ be the distribution of $(0, Z) \in \mathbb{R}^2$ (a 0 on the x-axis and the random variable Z on the y-axis), uniform on a straight vertical line passing through the origin. Now let $g_\theta(z) = (\theta, z)$ with θ a single real parameter. It is easy to see that in this case,

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

How to enforce Lipschitz?

- If parameter space is bounded, theory tells us the lipschitz constant is bounded
  - Method of original WGAN
- If lipschitz constant is bounded (say max K), theory tells us the gradient norms are bounded. Idea of improved Wasserstein GAN:
  -

$$L = \underbrace{\underset{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}{\mathbb{E}} [D(\tilde{\boldsymbol{x}})] - \underset{\boldsymbol{x} \sim \mathbb{P}_r}{\mathbb{E}} [D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \underset{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}{\mathbb{E}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]}_{\text{Our gradient penalty}} . \tag{3}$$

Sampling distribution We implicitly define $\mathbb{P}_{\hat{\boldsymbol{x}}}$ sampling uniformly along straight lines between pairs of points sampled from the data distribution $\mathbb{P}_r$ and the generator distribution $\mathbb{P}_g$. This is motivated by the fact that the optimal critic contains straight lines with gradient norm 1 connecting coupled points from $\mathbb{P}_r$ and $\mathbb{P}_g$ (see Proposition 1). Given that enforcing the unit gradient norm constraint everywhere is intractable, enforcing it only along these straight lines seems sufficient and experimentally results in good performance.

Layer wise Lipschitz aka Spectral Normalization

To elaborate, consider two functions f_1 and f_2 which are both Lipschitz continuous, say with constants L_1 and L_2. Then we can easily prove that the composition is also Lipschitz:

$$||f_1(f_2(x)) - f_1(f_2(y))|| \leq L_1 ||f_2(x) - f_2(y)|| \leq L_1 L_2 ||x - y||$$

However, the new Lipschitz constant is the product of the two previous. This made me think what if we extend

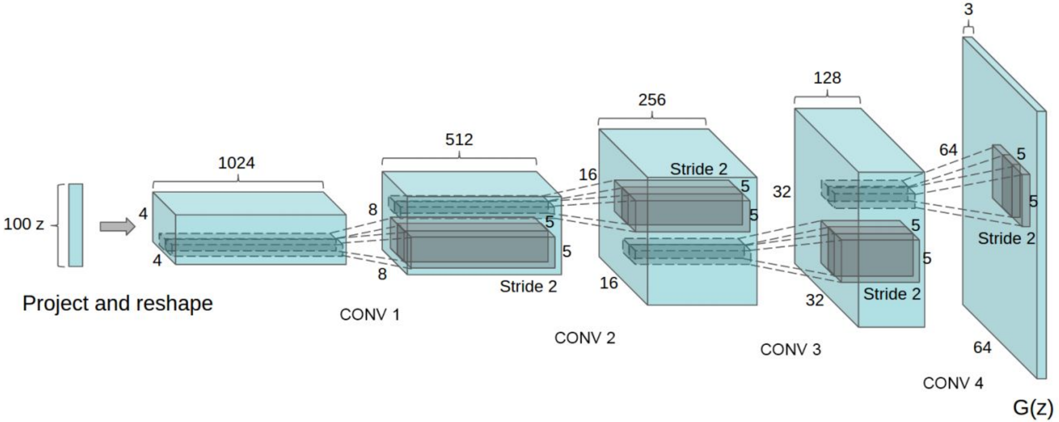
- Enough to satisfy lipschitz continuity on each layer!
- For linear layers, lipschitz constant == spectral norm == larget singular value
- Convolutions are linear
- See SPECTRAL NORMALIZATION FOR GENERATIVE ADVERSARIAL NETWORKS
- Helps for normal GANs too!

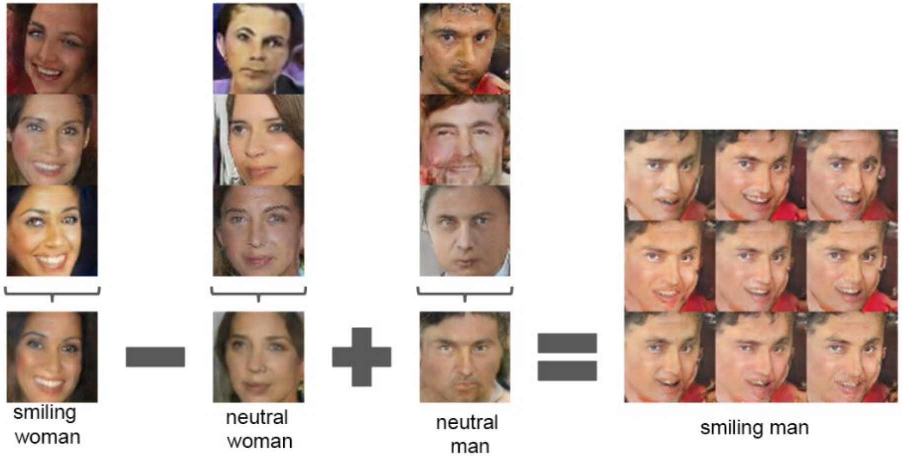
Theory summary

- Standard GANs minimize f-divergences
- Dimensional mismatch makes them unstable
- Wasserstein GAN minimize earth mover distance
    - More robust metric
    - Needs lipschitz constraint however
- Lipschitz constraints tend to also help for standard GANs
- Other regularizing approaches can also fix the standard GAN:
    - Smooth P_data and P_G so that they overlap
        - Can be implemented via a gradient based regularizer
        - (Stabilizing Training of Generative Adversarial Networks through Regularization, Roth et al, NIPS 2017)

The fun stuff!

DCGAN


100 z
Project and reshape
1024
4
4
512
8
8
5
5
Stride 2
CONV 1
256
16
16
Stride 2
5
5
CONV 2
128
32
32
5
5
Stride 2
CONV 3
3
64
64
5
5
Stride 2
CONV 4
G(z)


smiling
woman
neutral
woman
neutral
man
smiling man

Shameless Plug

Logo Synthesis and Manipulation with Clustered Generative Adversarial Networks

Alexander Sage
D-ITET, ETH Zurich
Switzerland
sagea@ee.ethz.ch

Eirikur Agustsson
D-ITET, ETH Zurich
Switzerland
aeirikur@vision.ee.ethz.ch

Radu Timofte
D-ITET, ETH Zurich
Merantix GmbH
radu.timofte@vision.ee.ethz.ch

Luc Van Gool
D-ITET, ETH Zurich
ESAT, KU Leuven
vangool@vision.ee.ethz.ch

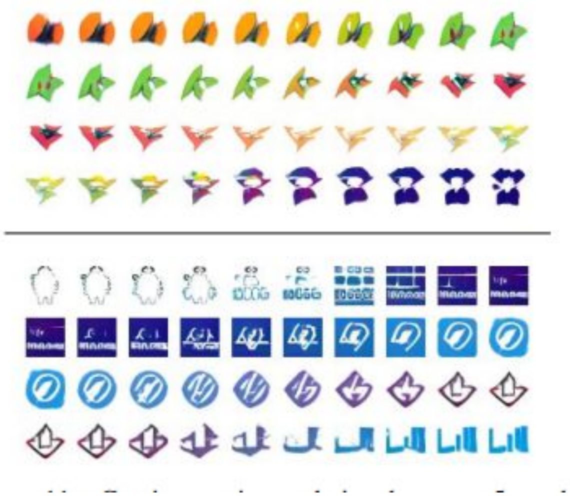
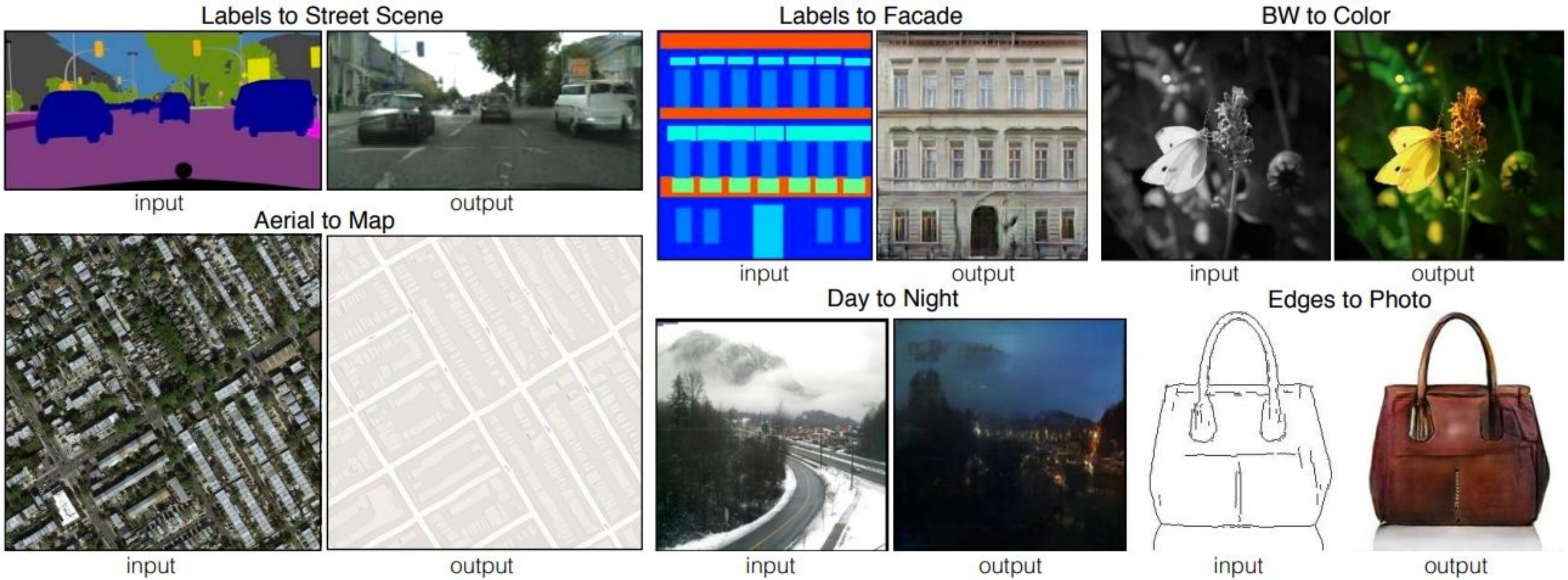


Image-to-Image Translation with Conditional Adversarial Networks

Phillip Isola Jun-Yan Zhu Tinghui Zhou Alexei A. Efros

Berkeley AI Research (BAIR) Laboratory, UC Berkeley

{isola,junyanz,tinghuiz,efros}@eecs.berkeley.edu



Pix2Pix: Main differences

- Conditional on some given data x (semantic map, low res image, etc)
- Uses content losses:
  - Pix2Pix: L1 loss between G(x,z) and y
    - Requires pairs!
  - Pix2Pix HD
    - VGG + GAN Features
    - Multiscale D
    - Different G architecture

The objective of a conditional GAN can be expressed as

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))], \quad (1)$$

where G tries to minimize this objective against an adversarial D that tries to maximize it, i.e. $G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D)$.

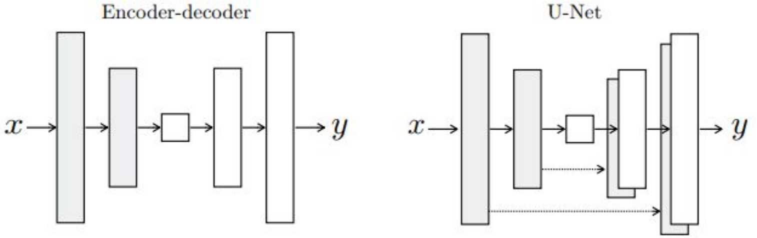


Figure 3: Two choices for the architecture of the generator. The "U-Net" [49] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Jun-Yan Zhu* Taesung Park* Phillip Isola Alexei A. Efros
Berkeley AI Research (BAIR) laboratory, UC Berkeley

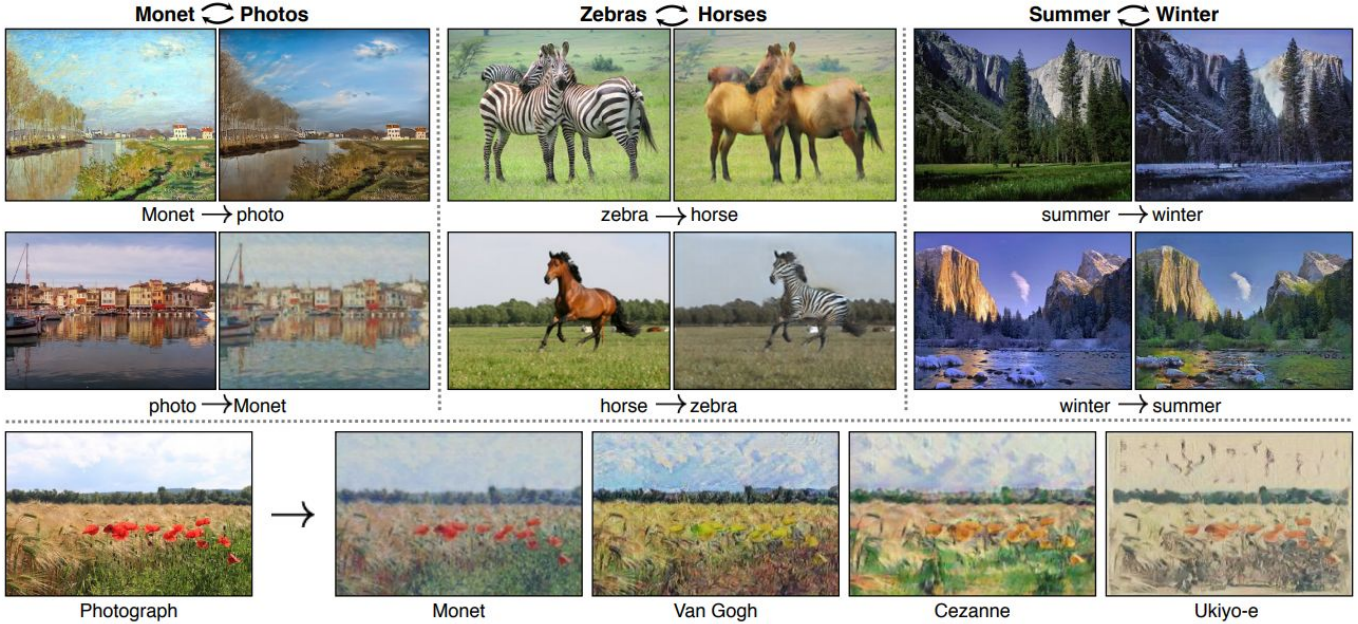


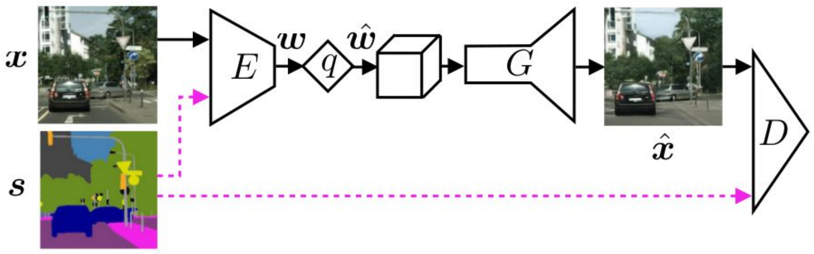
Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite photos from Flickr. Example application *(bottom)*: using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

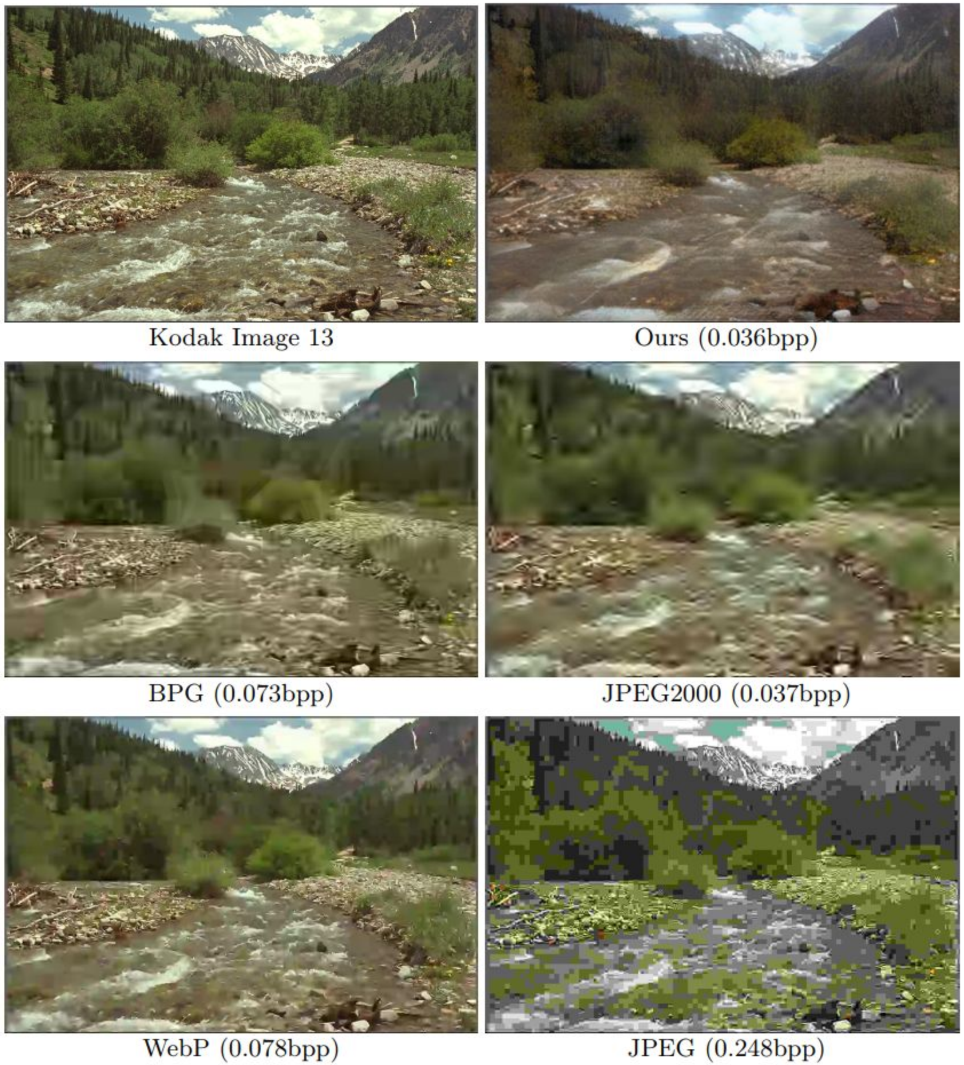
Shameless Plug

Generative Adversarial Networks for Extreme Learned Image Compression

Anonymous ECCV submission

Paper ID 2186





CycleGAN - Pix2Pix without pairs!

- Pix2Pix only trains a G+D from domain X -> Y
- CycleGAN trains G_XY, D_Y and G_YX, D_Y
- Can't do content loss directly, so instead do it in a cycle!
- LS-GAN instead of Vanilla GAN

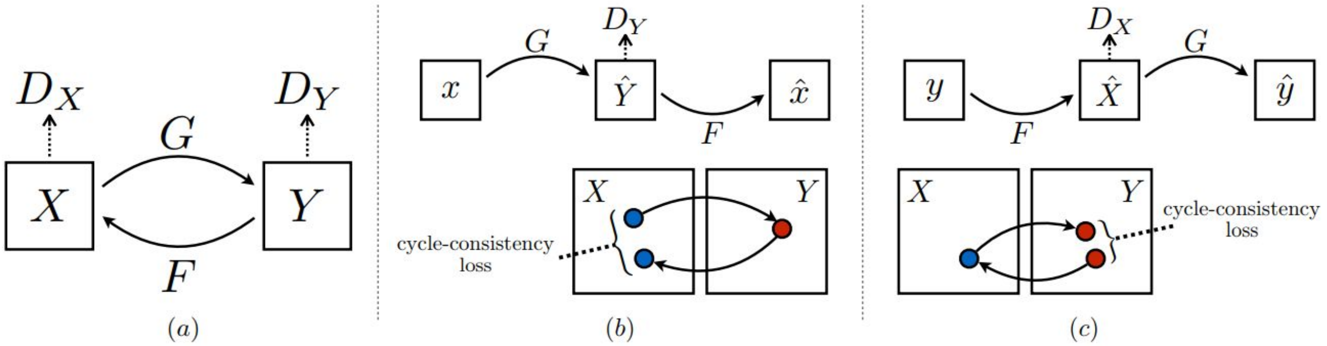


Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X and F. To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

CycleGAN, full objective

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]. \quad (2)$$

Our full objective is:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F), \quad (3)$$

where λ controls the relative importance of the two objectives. We aim to solve:

$$G^*, F^* = \arg \min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y). \quad (4)$$

results. In particular, for a GAN loss $\mathcal{L}_{\text{GAN}}(G, D, X, Y)$, we train the G to minimize $\mathbb{E}_{x \sim p_{\text{data}}(x)}[(D(G(x)) - 1)^2]$ and train the D to minimize $\mathbb{E}_{y \sim p_{\text{data}}(y)}[(D(y) - 1)^2] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[D(G(x))^2]$.

Shameless Plug: ComboGAN

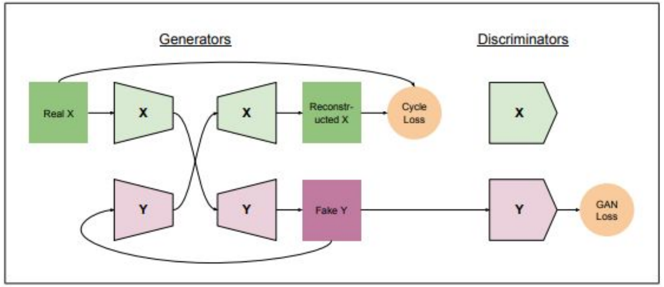


Figure 2. Generator training pass for direction $X \rightarrow Y$, where $X, Y \in \{1, .., n\} : X \neq Y$ are randomly chosen from our n domains at the start of every iteration. This pass is always repeated symmetrically for direction $Y \rightarrow X$ as well.

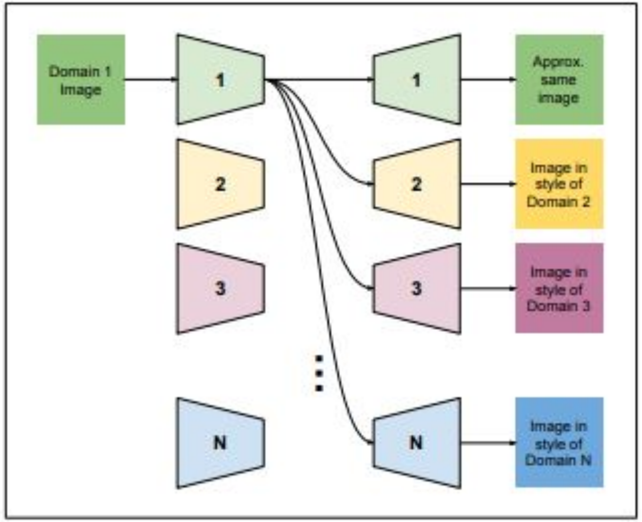
ComboGAN:
Unrestrained Scalability for Image Domain Translation

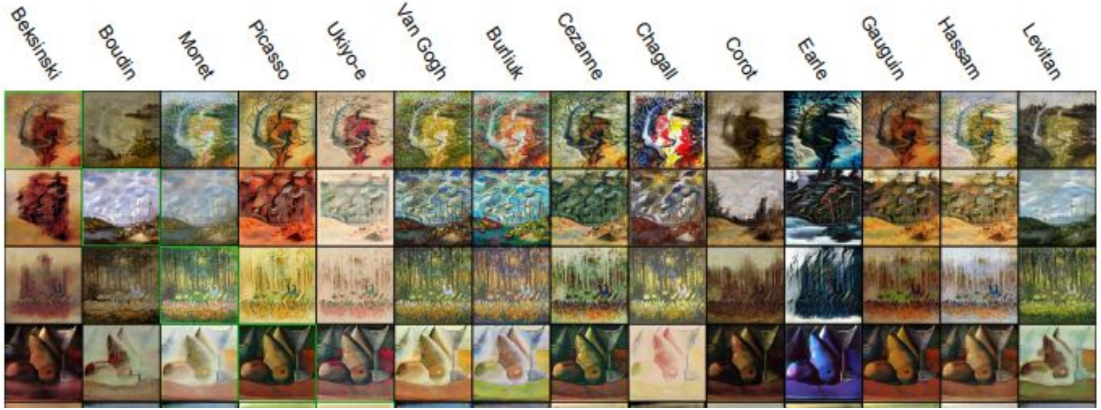
Asha Anoosheh
Computer Vision Lab
ETH Zürich
ashaa@ethz.ch

Eirikur Agustsson
Computer Vision Lab
ETH Zürich
aeirikur@ethz.ch

Radu Timofte
ETH Zürich
Merantix GmbH
timofter@ethz.ch

Luc Van Gool
ETH Zürich
KU Leuven
vangool@ethz.ch





Progressive growing of GANs

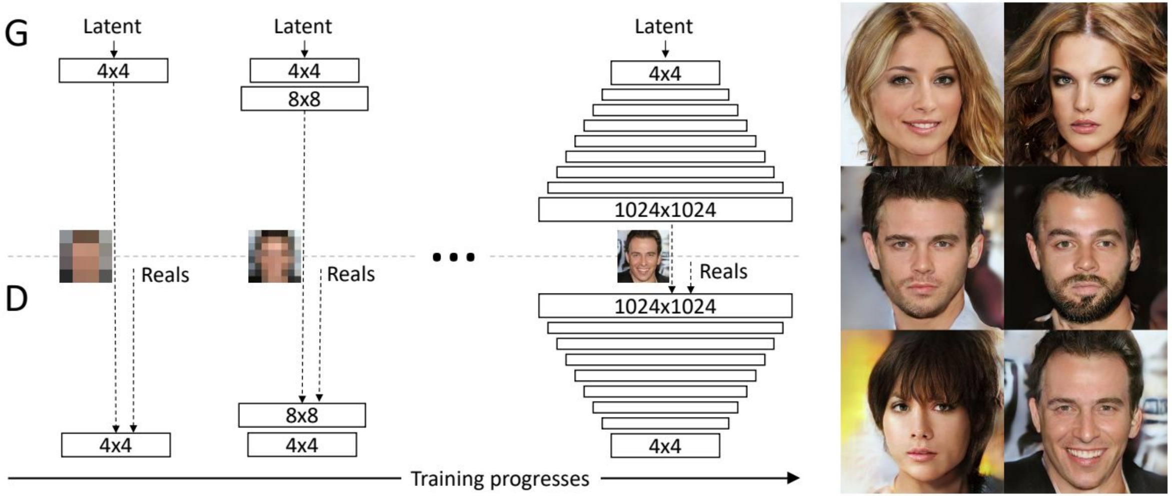


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $\boxed{N \times N}$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably. One the right we show six example images generated using progressive growing at 1024×1024.

Progressive growing of GANs

- Need to smoothly transition
- toRGB/fromRGB avoid going down to 3 channels when building up
-

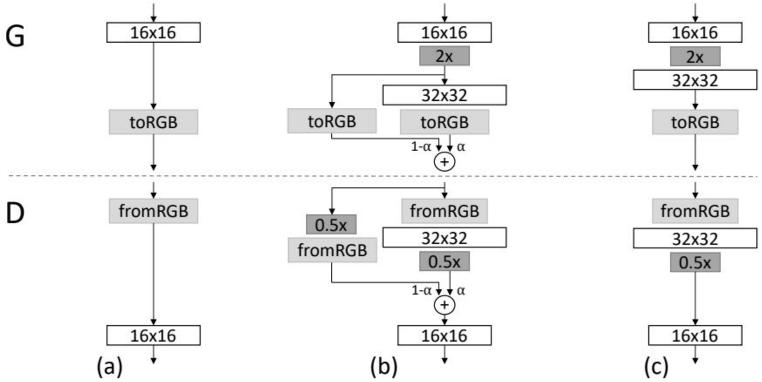


Figure 2: When doubling the resolution of the generator (G) and discriminator (D) we fade in the new layers smoothly. This example illustrates the transition from 16×16 images (a) to 32×32 images (c). During the transition (b) we treat the layers that operate on the higher resolution like a residual block, whose weight α increases linearly from 0 to 1. Here $\boxed{2\times}$ and $\boxed{0.5\times}$ refer to doubling and halving the image resolution using nearest neighbor filtering and average pooling, respectively. The $\boxed{\text{toRGB}}$ represents a layer that projects feature vectors to RGB colors and $\boxed{\text{fromRGB}}$ does the reverse; both use 1×1 convolutions. When training the discriminator, we feed in real images that are downscaled to match the current resolution of the network. During a resolution transition, we interpolate between two resolutions of the real images, similarly to how the generator output combines two resolutions.

Tricks and details

- Use improved WGAN loss
  - say LS-GAN with tricks is also OK
- Need tricks to make small minibatch work

Training configuration	CelebA				
	Sliced Wasserstein distance $\times 10^3$				
	128	64	32	16	Avg
(a) Gulrajani et al. (2017)	12.99	7.79	7.62	8.73	9.28
(b) + Progressive growing	4.62	**2.64**	3.78	6.06	4.28
(c) + Small minibatch	75.42	41.33	41.62	26.57	46.23
(d) + Revised training parameters	9.20	6.53	4.71	11.84	8.07
(e*) + Minibatch discrimination	10.76	6.28	6.04	16.29	9.84
(e) Minibatch stddev	13.94	5.67	2.82	5.71	7.04
(f) + Equalized learning rate	4.42	3.28	2.32	7.52	4.39
(g) + Pixelwise normalization	**4.06**	3.04	**2.02**	**5.13**	**3.56**
(h) Converged	2.42	2.17	2.24	4.99	2.96