Generative Adversarial Networks

AI602: Recent Advances in Deep Learning

Lecture 8

Slide made by

Jongjin Park

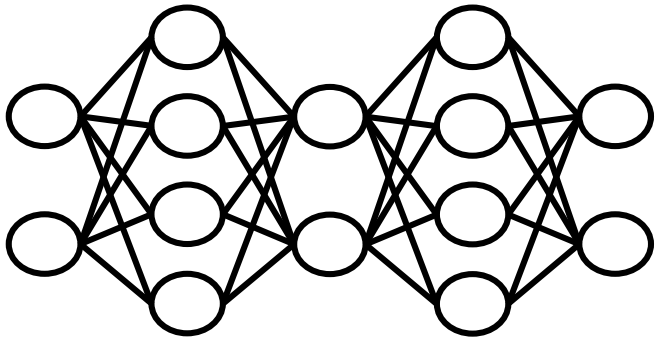
KAIST Graduate School of AI

Table of Contents

Table of Contents

Generative Model and Discriminative Model

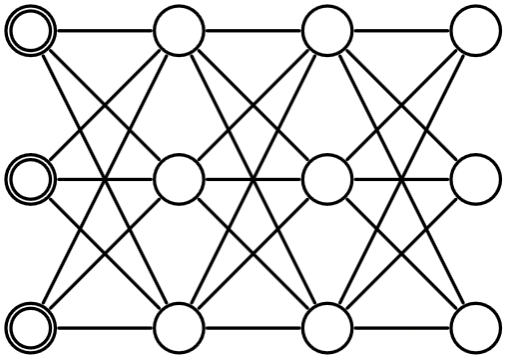
- Given an observed variable $\mathbf{x}$ and a target variable $\mathbf{y}$
- **Discriminative model** is a model of a conditional distribution $P(\mathbf{y}|\mathbf{x})$
  - e.g., neural network classifiers



$$P\left(\mathbf{y} = \text{cat} \middle| \right)$$

- **Generative model** is a model of a joint distribution $P(\mathbf{x}, \mathbf{y})$ (or $P(\mathbf{x})$)
  - e.g., Boltzmann machines, sum-product networks

$\mathbf{y} = \text{cat}$

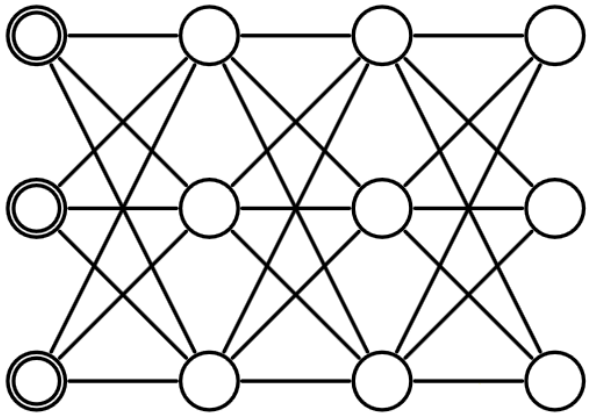


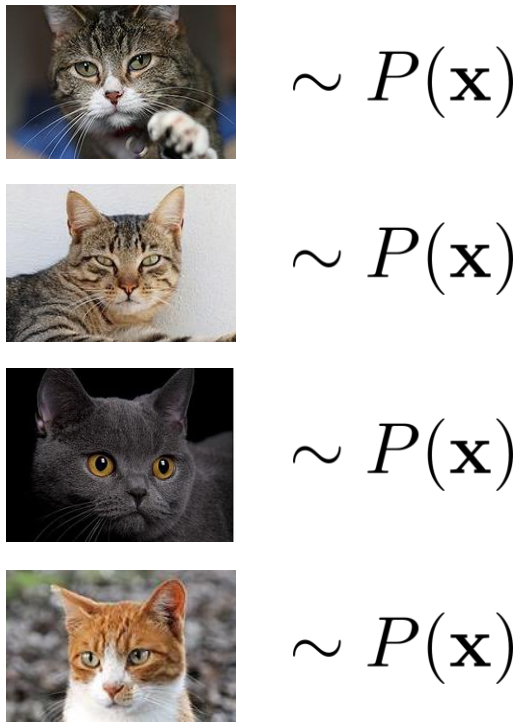
$$P\left(\mathbf{y} = \text{cat},\right)$$

* source : https://en.wikipedia.org/wiki/Cat#/media/File:Kittyply_edit1.jpg

Why Generative Model?

- Generative models model a full probability distribution of given data
- $P(\mathbf{x})$ enables us to generate new data similar to existing (training) data
  - This is impossible under discriminative models
- **Sampling methods** are required for generation



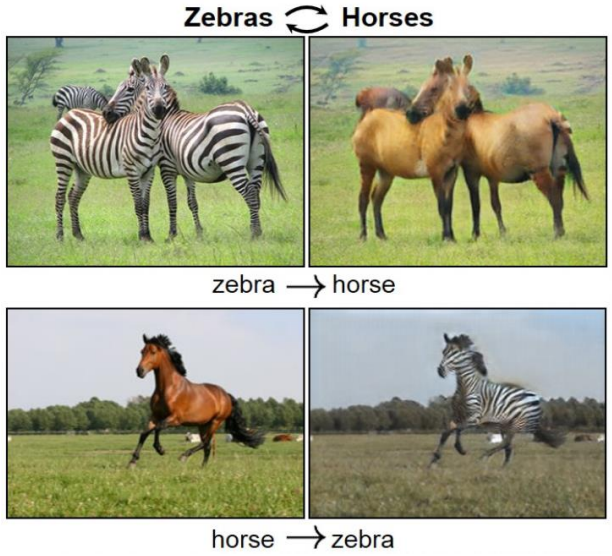


* source : https://en.wikipedia.org/wiki/File:Cat_poster_1.jpg

Why Generative Model?

- Generate new samples from the same distribution with training data
- Many real-world applications are related with generating data
- Common applications
  - Vision: super-resolution, style transfer, and image inpainting, etc.
  - Audio: synthesizing audio, speech generation, voice conversion, etc.
  - And many more..

Super-resolution [Ledig, et. al., 2017]



Style transfer [Zhu, et. al., 2017]

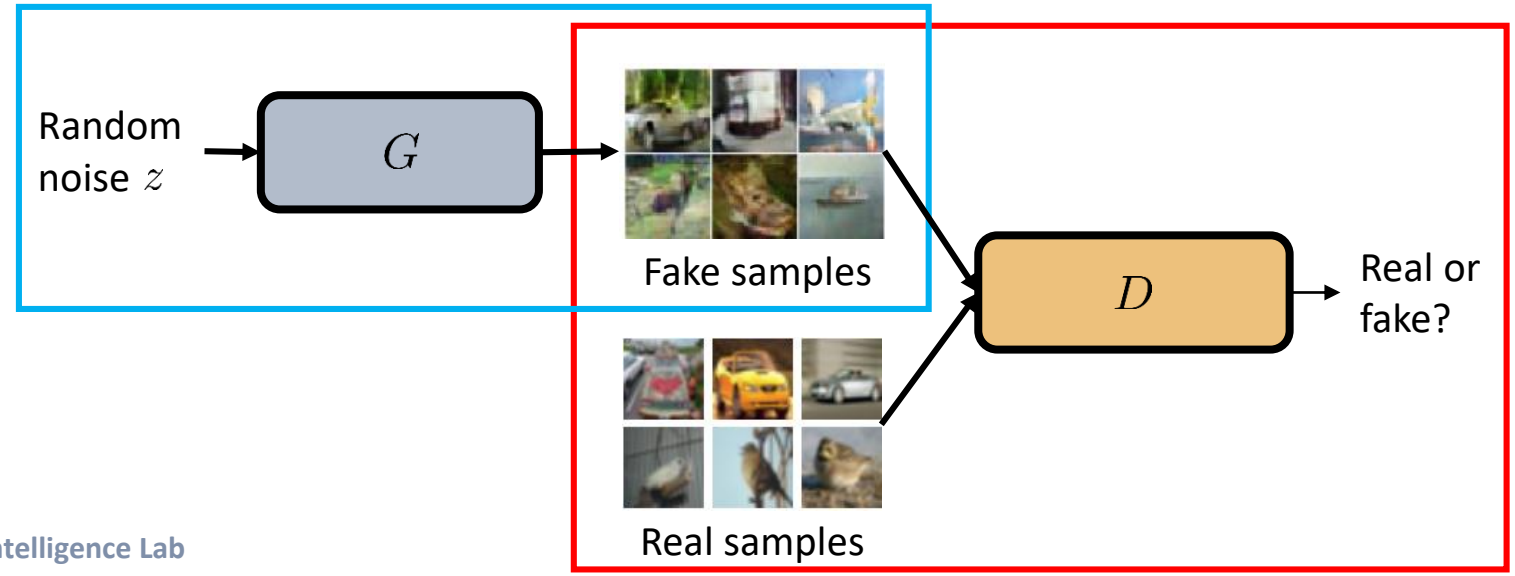
High-res image generation [Karras, et. al., 2018]

Examples of Generative Models

- Modeling a joint distribution of $\mathbf{x}$ with an explicit probability density function (which we will study in the next lecture)
  - Multivariate Gaussian distributions
    - $P(\mathbf{x}) \propto \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)\Sigma^{-1}(\mathbf{x}-\mu)\right)$
    - Tractable inference, low expressive power
  - Graphical models (e.g., RBM, DBM, etc.)
    - $P(\mathbf{x}) \propto \exp\left(\sum_i b_i x_i + \sum_{i,j} w_{ij} x_i x_j\right)$
    - Intractable inference, high expressive power with compact representations
- Modeling a joint distribution of $\mathbf{x}$ with an implicit density function
  - **Generative adversarial networks (GAN)**
    - Use function approximation capacity of neural networks
    - Modeling the data distribution with implicit density function using neural networks
    - Sampling: simple forward propagation of a generator neural network

Table of Contents

- Many previous approaches (explicit generative models) have difficulties in
  - Sampling from **high-dimensional** and **complex** distributions
  - And make it **realistic**
- Basic idea of GAN [Goodfellow, et. al., 2014]
  - Do not use any explicit density function $p_{\text{model}}(\mathbf{x})$
  - **Two player game** between discriminator network D and generator network G
  - D **tries to distinguish** real data and samples generated by G (fake samples)
  - G **tries to fool** the D by generating real-looking images
  - Utilizes **large capacity of neural nets** to model the sampling function



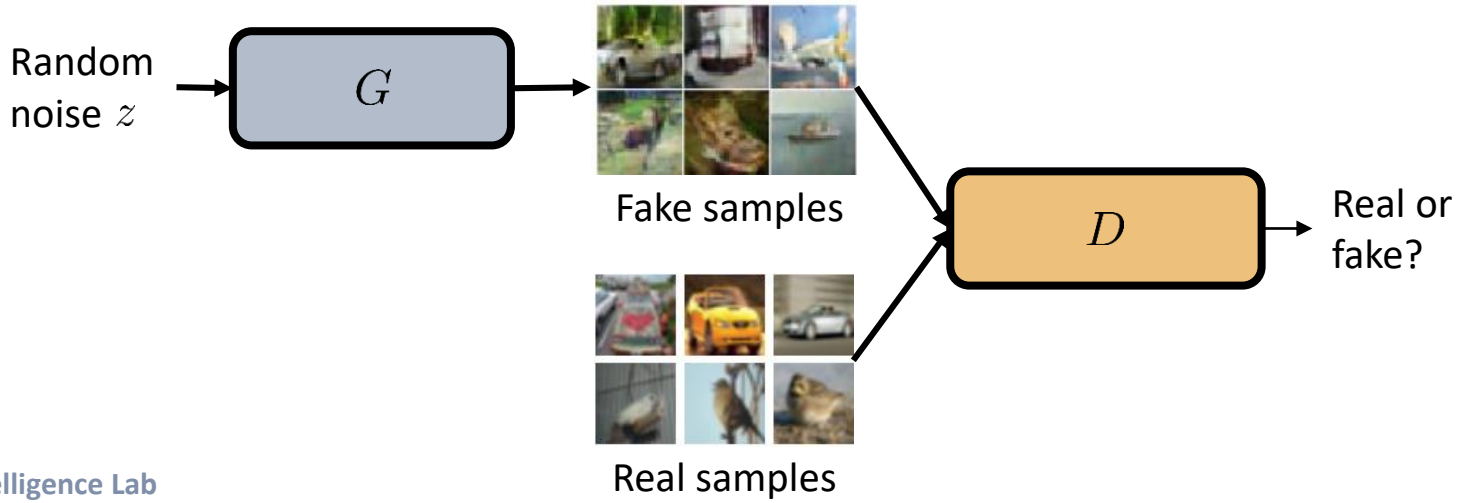
- D **tries to distinguish** real data and samples generated by G (fake samples)
- G **tries to fool** the D by generating real-looking images
- Objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Discriminator output for real data

Discriminator output for generated fake data

  - For D, maximize objective by making $D(x)$ is close to 1 and $D(G(z))$ is close to 0
  - For G, minimize objective by making $D(G(z))$ is close to 1



Training GAN

- Objective function [Goodfellow, et. al., 2014]:

$$\min_{\theta_g} \max_{\theta_d} V(\theta_d, \theta_g) = \left[\mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

- Alternative training between D and G
  - For D

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

  - For G

$$\min_{\theta_g} \mathbb{E}_{z \sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

  - In practice, optimizing generator objective does not work well (details in later slides)

What is Optimized in GAN Objective?

- Discriminator
  - For fixed G, the D optimizes:

$$
\begin{aligned}
V(\theta_d, \theta_g) &= \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \\
&= \int_x p_{\text{data}}(x) \log(D_{\theta_d}(x)) dx + \int_z p_z(z) \log(1 - D_{\theta_d}(G_{\theta_g}(z)) dz \\
&= \int_x p_{\text{data}}(x) \log(D_{\theta_d}(x)) + p_g(x) \log(1 - D_{\theta_d}(x)) dx
\end{aligned}
$$

  - Optimal discriminator is

$$
D_{\theta_d^*}(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})}
$$

  - If $p_{\text{data}} = p_g$, optimal discriminator $D_{\theta_d^*}(\mathbf{x}) = \frac{1}{2}$

What is Optimized in GAN Objective?

- Generator
  - For fixed $D_{\theta_d^*}$, the G optimizes:

$$
\begin{aligned}
V(\theta_d^*, \theta_g) &= \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d^*}(x) + \mathbb{E}_{z \sim p_z} \log(1 - D_{\theta_d^*}(G(z))) \\
&= \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d^*}(x) + \mathbb{E}_{x \sim p_g} \log(1 - D_{\theta_d^*}(x)) \\
&= \mathbb{E}_{x \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(\mathbf{x})} \right] + \mathbb{E}_{x \sim p_g} \left[\log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)} \right] \\
&= -\log 4 + KL\left(p_{\text{data}} \,\middle\|\, \frac{p_{\text{data}} + p_g}{2} \right) + KL\left(p_g \,\middle\|\, \frac{p_{\text{data}} + p_g}{2} \right) \\
&= -\log 4 + 2 \cdot \boxed{JS(p_{\text{data}} \,\|\, p_g)}
\end{aligned}
$$

- When discriminator is optimal
  - **Generator objective** becomes **minimizing the Jensen-Shannon (JS) divergence**
  - Many previous generative models use KL divergence (maximum likelihood)
  - Unlike KL divergence, JS divergence helps to
    - Generate sharp, clear images but causes a missing mode problem

GAN Training Algorithm: in Practice

- Alternative training of discriminator and generator
  - Recall: G optimizes JS divergence when D is optimal
  - But D is not optimal generally
  - By updating discriminator k-steps per each iteration of generator, this problem could be reduced

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

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

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

* source : Goodfellow, et. al., Generative adversarial nets, NIPS 2014

GAN Training Algorithm: in Practice

- Alternative training between D and G
  - For D

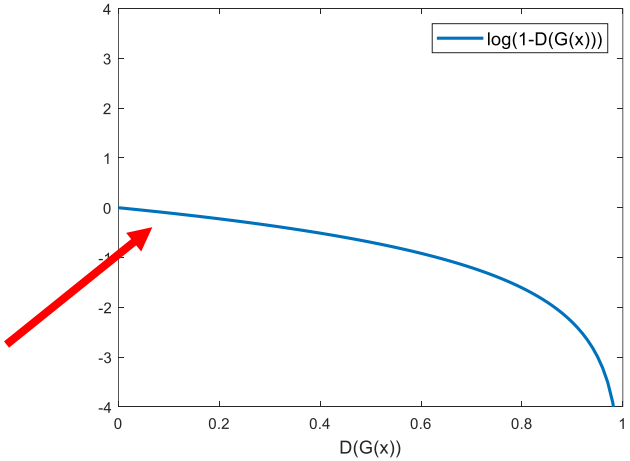
$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

  - For G

$$\min_{\theta_g} \mathbb{E}_{z \sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

  - In practice, optimizing generator objective does not work well
  - When generated sample looks **bad** (at beginning of training) **gradient is relatively flat**
    - Learning by back-prop becomes difficult

Flat gradients when a sample is really bad



* source : http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture12.pdf

GAN Training Algorithm: in Practice

- Alternative training between D and G
  - For D

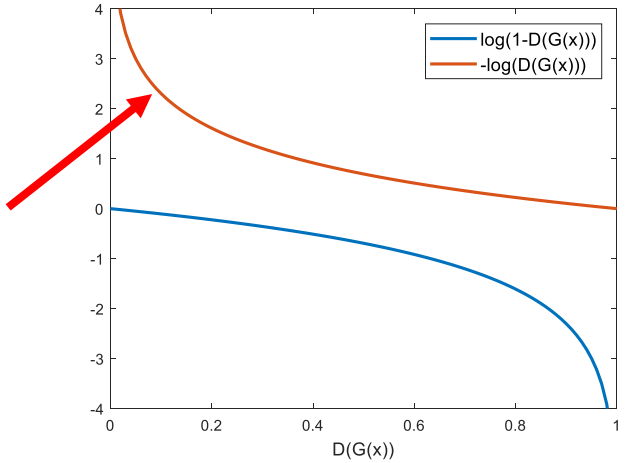
$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

  - In practice, G is optimized by

$$\min_{\theta_g} \mathbb{E}_{z \sim p_z} - \log(D_{\theta_d}(G_{\theta_g}(z)))$$

  - $-\log(D_{\theta_d}(G_{\theta_g}(z)))$ gives **stronger gradients** early in learning

Stronger gradients when a sample is really bad

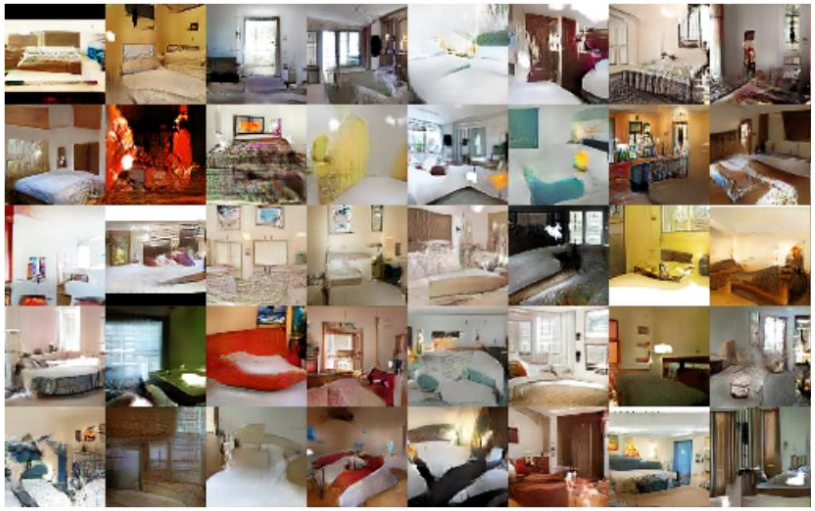


* source : http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture12.pdf

Table of Contents

Generated Samples with GAN

- GAN generates sharp, clear images compared to previous generative models
  - Most previous works are suffered by blurred unrealistic generated samples



Bedroom images

Faces images

ImageNet

- Then, **what makes GAN be able to generate realistic samples**?
  - GAN utilizes the function approximation power of neural networks
    - But it is also the cases for other models (e.g., Variational auto encoder; VAE)
  - What else?

* source : Radford, et. al., Unsupervised representation learning with deep convolutional generative adversarial networks.

Difference with Previous Generative Models

- Maximum likelihood methods (= KL divergence minimization)

$$KL(p_{\text{data}} \parallel p_g) = \int_x p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_g(x)} dx$$

  - $p_{\text{data}}(x) > p_g(x)$
    - When $p_{\text{data}}(x) > 0, p_g(x) \to 0$, the integrand grows quickly to infinity
    - **High penalty** when generator's distribution **does not cover parts of the train data**
  - $p_{\text{data}}(x) < p_g(x)$
    - When $p_{\text{data}}(x) \to 0, p_g(x) > 0$, the integrand goes to 0
    - **Low penalty** for generating **fake looking samples**
  - KL divergence solution tends to **cover all the modes**

- Inverse KL divergence $KL(p_g \parallel p_{\text{data}})$ tends to **fit single mode**

Difference with Previous Generative Models

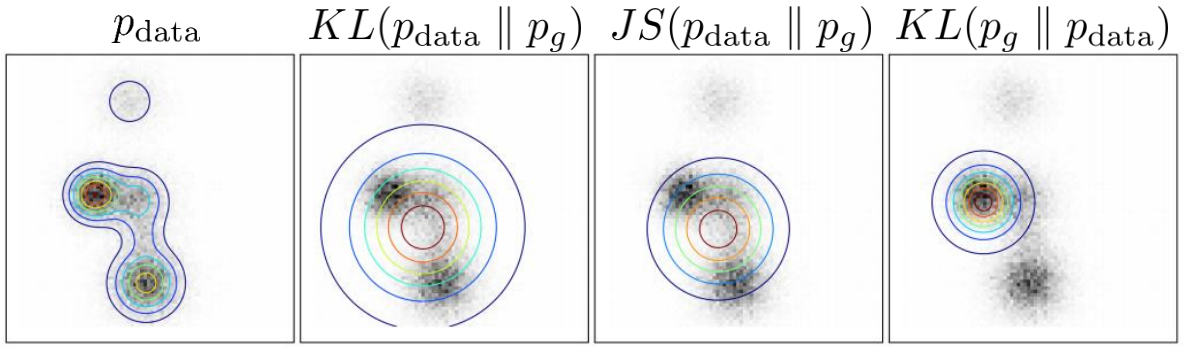
- Maximum likelihood methods (= KL divergence minimization)

$$KL(p_{\text{data}} \parallel p_g) = \int_x p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_g(x)} dx$$

  - KL divergence solution tends to **cover all the modes**
- Inverse KL divergence $KL(p_g \parallel p_{\text{data}})$ tends to **fit single mode**
- Jensen-Shannon divergence

$$JS(p_{\text{data}} \parallel p_g) = KL\left(p_{\text{data}} \,\middle\|\, \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \,\middle\|\, \frac{p_{\text{data}} + p_g}{2}\right)$$

  - (A bit like a) combination of the two divergences
  - Using **JS divergence instead of KL divergence** helps to generate realistic images [Huszar 2015]



* source : https://www.inference.vc/how-to-train-your-generative-models-why-generative-adversarial-networks-work-so-well-2/

Issues of GAN: Intractable Nash and Mode Collapse

- Hard to achieve Nash equilibrium to a two-player non-cooperative game [Salimans, et. al., 2016]
  - Each model updates its own objective function
  - Modification of θ_d that reduces D's objective can increase G's, and vice versa

- Mode collapse
  - Generator collapse to parameters that produces the same outputs
  - Generator can fool if it is really good at making only **a** good looking sample
  - JS divergence does not penalize missing mode as hard as KL divergence

Examples of mode collapse in GAN.

* source : Arjovsky, et. al., Wasserstein GAN, ICML 2017

Table of Contents

Conditional GAN

- In the standard (unconditional) GAN, there is no control on modes of the data being generated.
- Conditional GANs aim for modeling the distribution better by incorporating its attribute or class y, therefore they have advantages of to its class-wise controllability and improved quality for complex generation tasks.

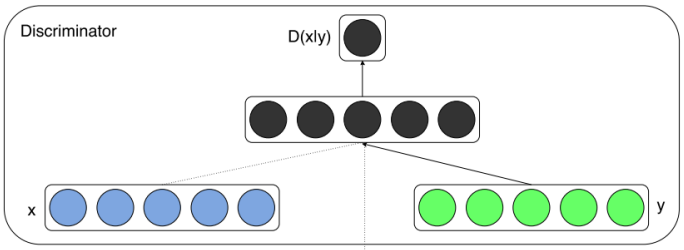
- Recall the objective function of unconditional GAN [Goodfellow et al., 2014]:

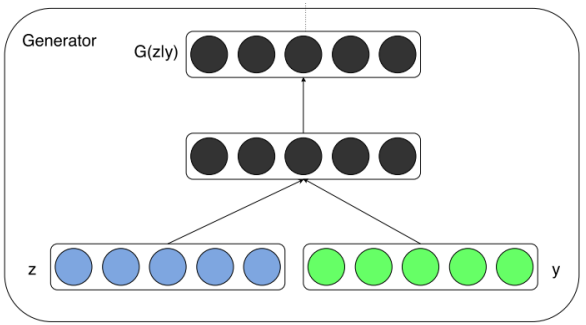
$$\min_G \max_D \left[\mathbb{E}_{x\sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

- The objective function of conditional GAN is as follows [Mirza et al., 2014]:

$$\min_G \max_D \left[\mathbb{E}_{x\sim p_{\text{data}}} \log D_{\theta_d}(x|y) + \mathbb{E}_{z\sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z|y)))\right]$$

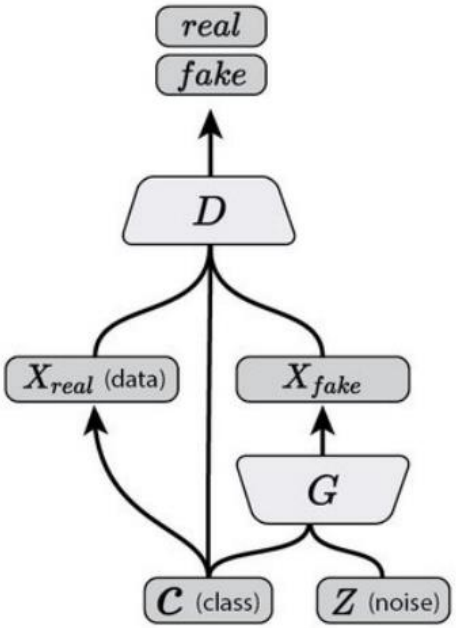
There are many works to concatenate (embedded) y to the input or to the middle layers [Reed et al. ICML 2016].



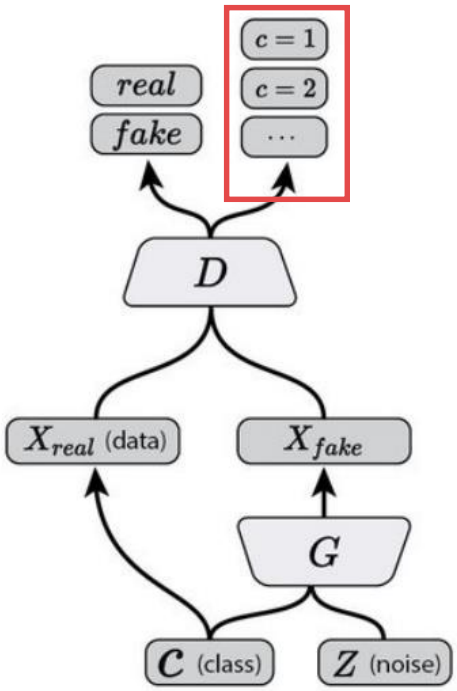


Conditioning of GANs: Auxiliary Classifier GAN (ACGAN)

- For better results and to stabilize training, several methods to feed conditional information to GANs are studied.
- Instead of feeding class information to the discriminator, [Odena et al., ICML 2017] tasks the discriminator with reconstructing class information.
- They modified the discriminator to contain an auxiliary classifier which classifies the classes of both real and fake inputs.



Conditional GAN
(Mirza et al., 2014)



ACGAN
(Odena et al., 2017)

* source : Basart , Analysis of Generative Adversarial Models, 2017

- The training objective function consists of two parts:
  - The log-likelihood of the correct source, L_S (i.e. the original GAN loss).

$$L_S = \mathbb{E}_{x\sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$
$$= \mathbb{E}_{x\sim p_{\text{data}}} \log P(S = \text{real}|x) + \mathbb{E}_{z\sim p_z} \log P(S = \text{fake}|G_{\theta_g}(z))$$

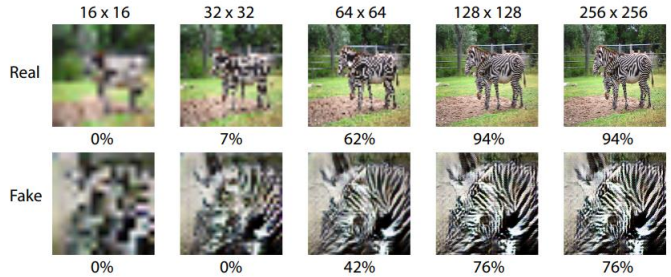
  - The log-likelihood of the correct class, L_C.

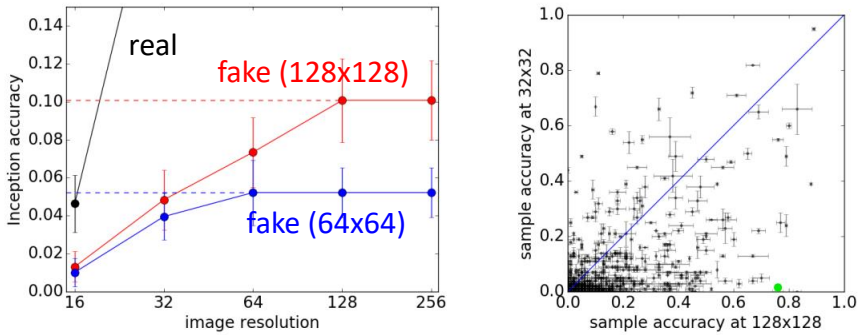
$$L_C = \mathbb{E}_{x\sim p_{\text{data}}} \log P(C = c|x) + \mathbb{E}_{z\sim p_z, c\sim p_c} \log P(C = c|G_{\theta_g}(z, c)))$$

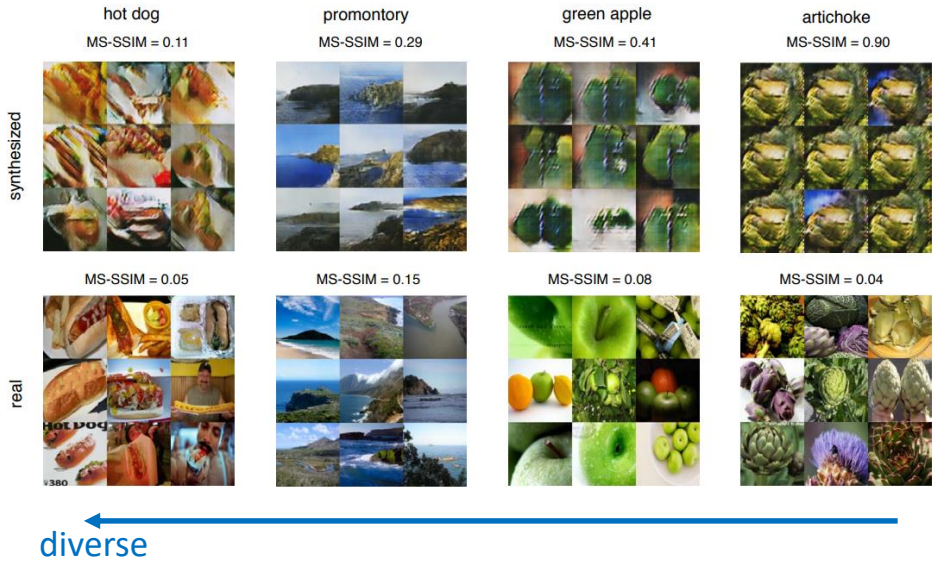
- The discriminator trained to maximize $L_S + L_C$, and the generator trained to maximize $-L_S + L_C$.
  - Note that the classification loss L_C is used not only for the discriminator, but also for the generator.
  - The balancing weight of both losses can be tuned for better training. (i.e. The discriminator and generator maximize $L_S + \lambda_1 L_C$ and $-L_S + \lambda_2 L_C$, respectively.)

Conditioning of GANs: Auxiliary Classifier GAN (ACGAN)

- ACGAN can generate diverse, higher resolution images than concatenation of class labels in the discriminator.
  - The authors show that synthesizing higher resolution images leads to increased discriminability by feeding images to a pre-trained Inception network (left).
  - To evaluate the diversity of generated images quantitatively, they also measure the multiscale structural similarity (MS-SSIM) between randomly chosen pairs of images within a given class (right).







Inception accuracy: higher is better

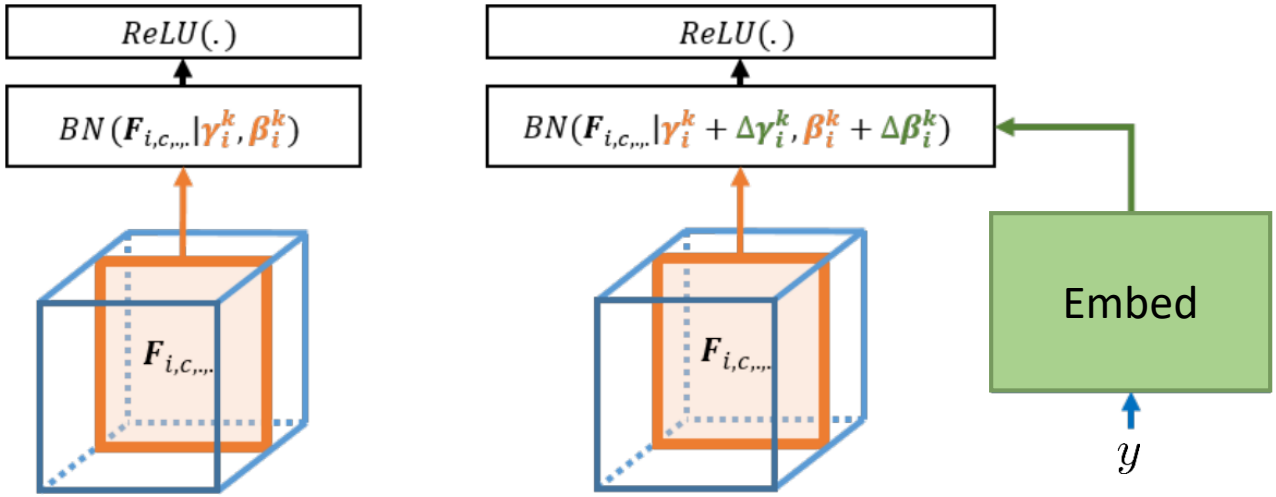
MS-SSIM: lower is better

* source : Odena et al., Conditional Image Synthesis with Auxiliary Classifier GANs, ICML 2017

Conditioning of GANs : Conditional Batch Normalization (CBN)

- Instead of concatenating one-hot vectors to the generator, [Dumoulin et al., ICLR 2017, de Vries et al., NIPS 2017] made the conditions modulate Batch Normalization (BN) layers.
- The key idea is to predict the affine scaling parameters, γ and β of the batch normalization from an embedding of conditional information, y.

$$z = \gamma(y) \cdot \left(\frac{x-\mu(x)}{\sigma(x)}\right) + \beta(y)$$



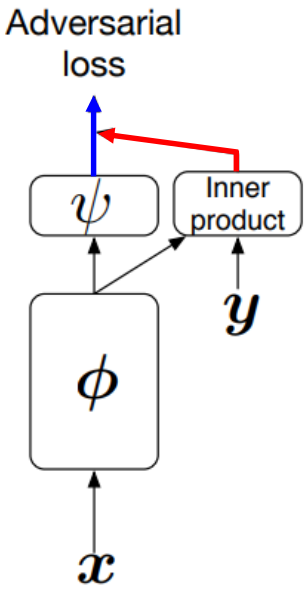
Batch Normalization

Conditional Batch Normalization

* source : de Vries et. al., Modulating early visual processing by language, NIPS 2017

Conditioning of GANs: Projection Discriminator

- [Miyato et al., ICLR 2018] suggested another approach to feed conditional information to the discriminator.
- They show a *projection* of one-hot vector is much better than concatenation.
- The architecture of the proposed projection discriminator is as follows:



$$D(x, y; \theta) := \mathcal{A}(f(x, y; \theta))$$

where f is a function of x and y, and $\mathcal{A}$ is an activation function of the users' choice (e.g. sigmoid for vanilla GAN).

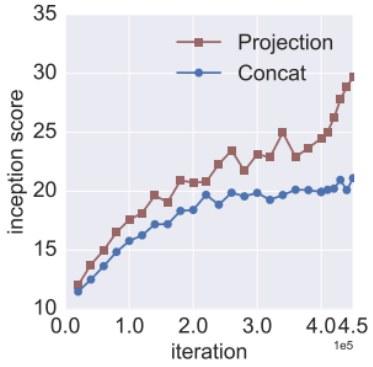
$$f(x, y) := y^{\mathrm{T}} V \phi(x) + \psi(\phi(x))$$

The discriminator is modeled by a inner-product (projection) of the class embedded vector y.

* source : Miyato et. al., cGANS with Projection Discriminator, ICLR 2018

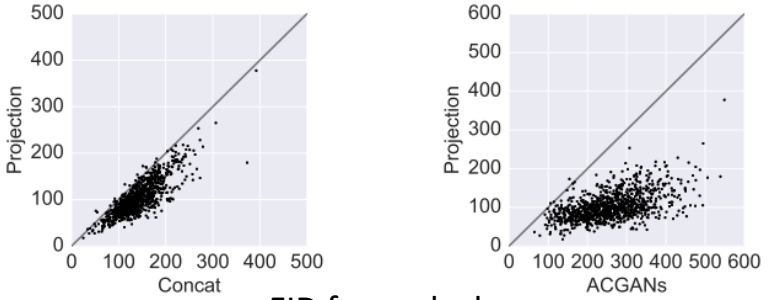
Conditioning of GANs: Projection Discriminator

- Projection discriminator significantly outperforms concatenation of one-hot vector and ACGAN.

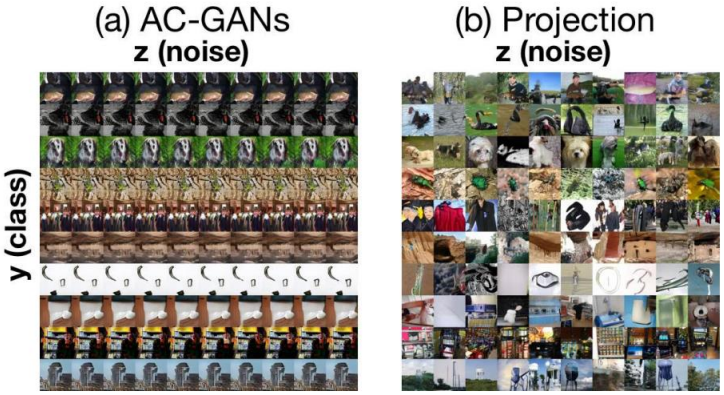


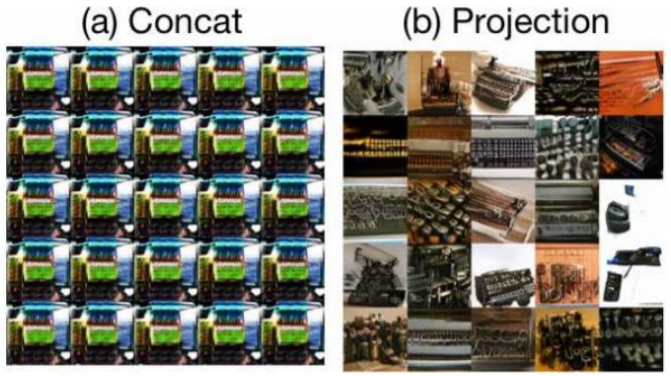
Method	Inception Score	Intra FID
AC-GANs	28.5±.20	260.0
concat	21.1±.35	141.2
projection	**29.7**±.61	**103.1**
*projection (850K iteration)	**36.8**±.44	**92.4**

Inception score: higher is better
Intra (class-wise) FID: lower is better



FID for each class





The projection discriminator is also more robust for mode-collapse than prior methods.

Table of Contents

Wasserstein Distance

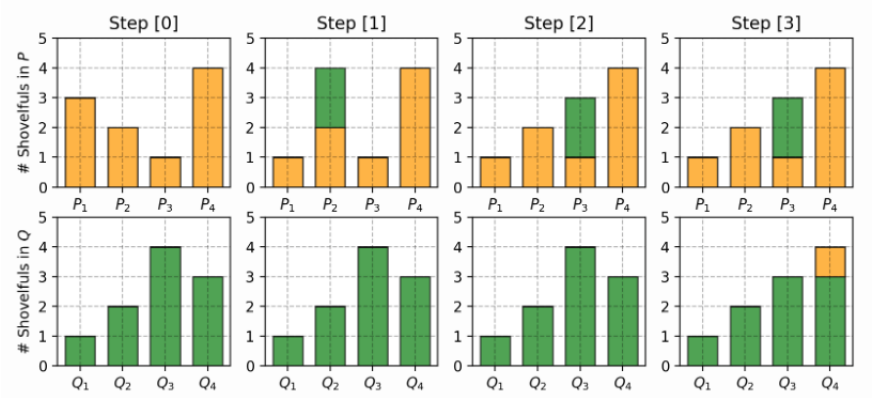
- Some heuristics can alleviate the issue for training GAN
  - But, they are not fundamental solutions and are not clear to work in general

- Wasserstein distance: **measure of the distance between two probability distributions** (also called Earth Mover's distance)

$$W(p_{\text{data}}, p_g) = \inf_{\gamma \in \Pi(p_{\text{data}}, p_g)} \mathbb{E}_{(x,y)\sim\gamma} \|x - y\|$$

  - Intuitively, minimal total amount of work to transform one *heap of dirt* into the other
  - Work is defined as the amount of *dirt* in a chunk times the distance it was moved
  - Example
    - $W(P, Q)$: the minimum amount of work from distribution P to Q

$$P_1 = 3, P_2 = 2, P_3 = 1, P_4 = 4$$
$$Q_1 = 1, Q_2 = 2, Q_3 = 4, Q_4 = 3$$
$$W(P, Q) = 5$$



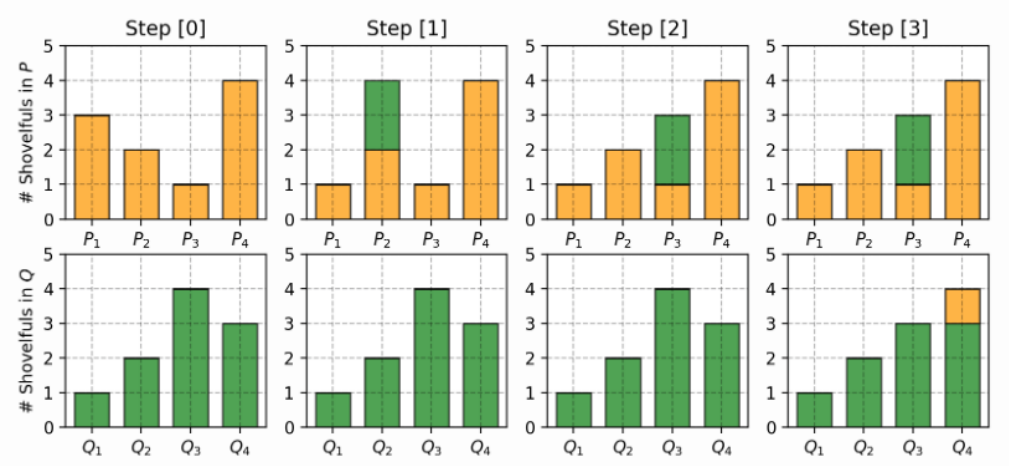
* source : https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html#wasserstein-gan-wgan

Wasserstein Distance

- Wasserstein distance: **measure of the distance between two probability distributions** (also called Earth Mover's distance)

$$W(p_{\text{data}}, p_g) = \inf_{\gamma \in \Pi(p_{\text{data}}, p_g)} \mathbb{E}_{(x,y)\sim\gamma} \|x - y\|$$

  - Intuitively, minimal total amount of work to transform one *heap of dirt* into the other
  - Work is defined as the amount of *dirt* in a chunk times the distance it was moved
  - Example
    - $\Pi(p_{\text{data}}, p_g)$ is the set of all possible joint probability distributions between p_{data} and p_g
    - Infimum over joint distribution γ (each γ corresponds to one dirt transport plan like in example in a slide before)



* source : https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html#wasserstein-gan-wgan

Comparison between Wasserstein Distance and Other Distance Metrics

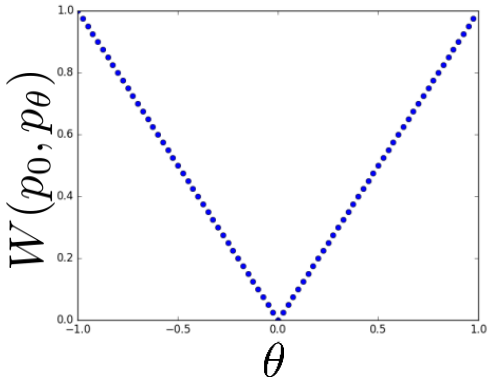
- When two distributions are located without overlaps
  - Still provides meaningful and smooth representation of the distance (and gradients)

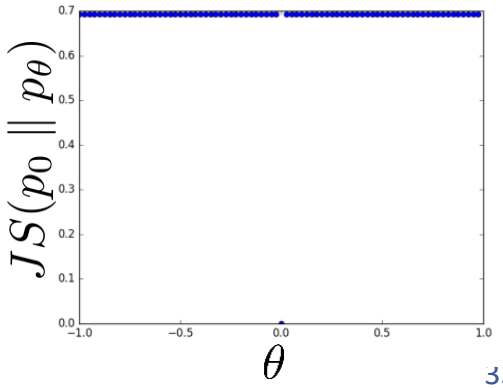
- Example [Arjovsky, et. al., 2017]
  - Let $Z \sim U[0,1]$, p_0 be the distribution of $(0, Z) \in \mathbb{R}^2$
  - $g_\theta(Z) = (\theta, Z)$ with θ, a single real parameter, and p_θ is the distribution of $g_\theta(Z)$
  - Distance between two distributions are:

$$W(p_0, p_\theta) = |\theta|$$

$$JS(p_0 \parallel p_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0 \end{cases}$$

$$KL(p_0 \parallel p_\theta) = KL(p_\theta \parallel p_0) = \begin{cases} \infty & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0 \end{cases}$$



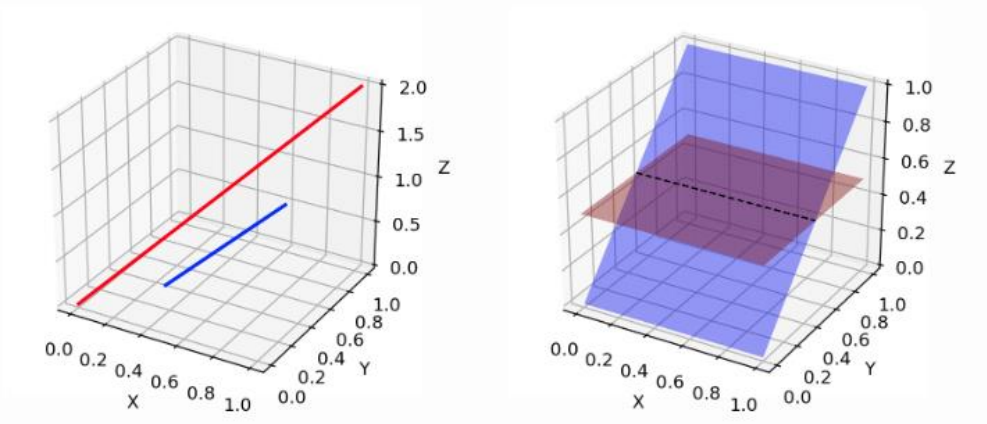


  - Parameter θ can be learned on the Wasserstein distance
  - Parameter θ cannot be learned on JS or KL divergence

Comparison between Wasserstein Distance and Other Distance Metrics

- This example shows that there exist distributions that
  - **Don't converge under the JS, KL, or inverse KL**
    - For the **JS, KL, and inverse KL, there are cases where the gradient is always 0**
    - This is especially not good from an optimization perspective
  - **Do converge under the Wasserstein distance**

- Easy to get similar results, if p_{data} and p_g are on low-dimensional manifolds in high dimensional space



Low dimensional manifolds in high dimension space can hardly have overlaps.
(Left) two lines in a 3-d space. (Right) two surfaces in 3-d space

* source: https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html#wasserstein-gan-wgan

Wasserstein Distance in GAN Objective

- Infimum over joint distribution $\gamma \in \Pi(p_{\text{data}}, p_g)$ is computationally **intractable**

- Using Kantorovich-Rubinstein duality [Villani, 2009], Wasserstein distance becomes:

$$W(p_{\text{data}}, p_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p_{\text{data}}}[f(x)] - \mathbb{E}_{x \sim p_g}[f(x)]$$

  - The Supremum is over all the 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$
  - Let f is parameterized by w, then one could consider solving the problem

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim p_{\text{data}}}[f_w(x)] - \mathbb{E}_{z \sim p_z}\left[f_w(g_{\theta_g}(z))\right]$$

  - To enforce the Lipschitz constraint, **clamp the weights** to a fixed box (e.g., $\mathcal{W} = [-0.01, 0.01]^\ell$, where ℓ is dimension of parameter $w \in \mathcal{W}$)

* source: WGAN and Kantorovich-Rubinstein duality https://vincentherrmann.github.io/blog/wasserstein/

- Comparison of GAN and WGAN
  - Discriminator (outputs probability of real or fake) becomes a continuous function to help compute Wasserstein distance (with weight clamping)

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

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

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

end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

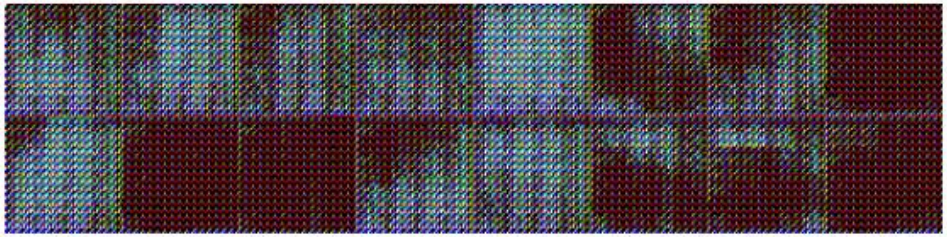
$$g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$$
$$\theta \leftarrow \theta - \alpha \cdot \mathrm{RMSProp}(\theta, g_\theta)$$

end for

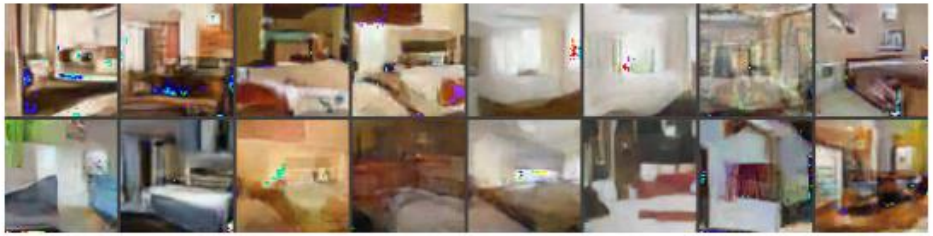
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

WGAN vs GAN


(Left) WGAN vs. (Right) GAN with DCGAN architecture . Both produce high quality samples

(Left) WGAN vs. (Right) GAN with less parameter models and without batch normalization




(Left) WGAN vs. (Right) GAN with MLP generator.
Vanilla GAN does mode collapse, while WGAN still produces good samples

* source : Arjovsky, et. al., Wasserstein GAN, ICML 2017

Enforcing the Lipschitz Constraint of Discriminator

- To maintain Lipschitz constraint WGAN uses weight clamping
  - But it is naïve and no guaranteed method
  - Weight clamping leads to optimization difficulties sometimes

- **Recent works try to improve the method for maintaining Lipschitz constraint**
  - Improved training of Wasserstein GANs (**WGAN-GP**) [Gulrajani, et. al., 2017]
    - Use **gradient penalty** to maintain Lipschitz constraint

$$\mathbb{E}_{\hat{x}\sim p_{\hat{x}}}\left[\left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1\right)^2\right]$$

where $\hat{x} = \varepsilon x + (1-\varepsilon)G(z)$

  - **Spectral normalization** for generative adversarial networks [Miyato, et. al., 2018]
    - Control the Lipschitz constant of D by **constraining the spectral norm of each layer**

$$\bar{W}_{SN}(W) = W/\sigma(W)$$

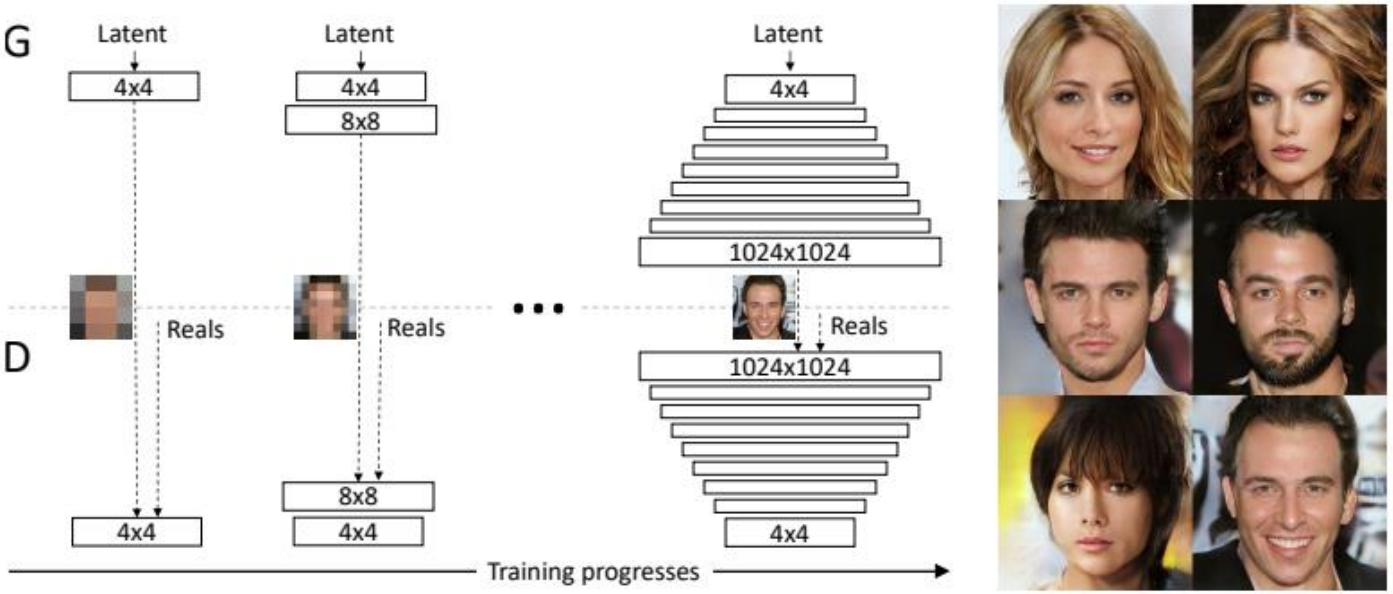
where $\sigma(W)$ is the spectral norm of W

- Nevertheless, stabilizing training GAN is still an on-going research topic!

Table of Contents

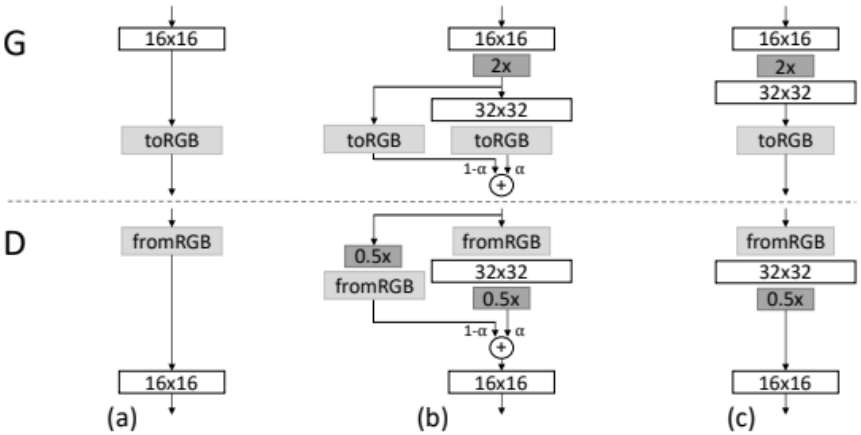
Progressive GAN: High-Resolution Image Generation

- GANs produce sharp images
  - But only in fairly small resolutions and with somewhat limited variation
- Training continues to be unstable despite recent progress
- Generating **high resolution image is difficult**
  - It is **easier to tell the generated images** from training images in high-res images [Karras, et. al., 2018]
  - Grow both generator and discriminator **progressively**
  - **Start learning from easier** low-resolution images
  - Add new layers that introduce higher-resolution details as the training progress



* source : Karras, et. al., Progressive growing of GANs for improved quality, stability, and variation, ICLR 2018

- Fade in the new layers smoothly
  - Prevent sudden *shocks* to the already well-trained, smaller-resolution layers



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

- **Simplified** minibatch discrimination [Salimans, et. al., 2016]
  - **Compute standard deviation** for each feature in each spatial location and average it
  - Use it as an additional feature map for the input of the next layer

* source : Karras, et. al., Progressive growing of GANs for improved quality, stability, and variation, ICLR 2018

Progressive GAN: Results

1024x1024 images generated using the CELEBA-HQ dataset

https://www.youtube.com/watch?v=G06dEcZ-QTg&feature=youtu.be

Mao et al. (2016b) (128 × 128) Gulrajani et al. (2017) (128 × 128) Our (256 × 256)

Visual quality comparison: LSUN bedroom

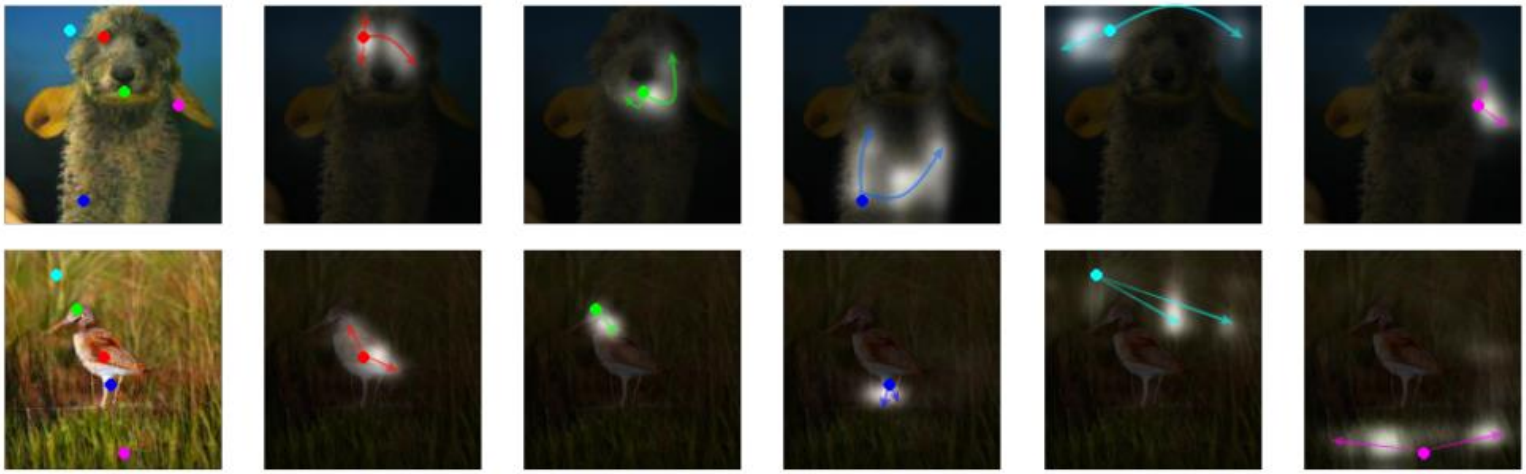
LSUN other categories generated image (256x256)

* source : Karras, et. al., Progressive growing of GANs for improved quality, stability, and variation, ICLR 2018

Table of Contents

Self-Attention GAN: Attention-Driven Image Generation Tasks

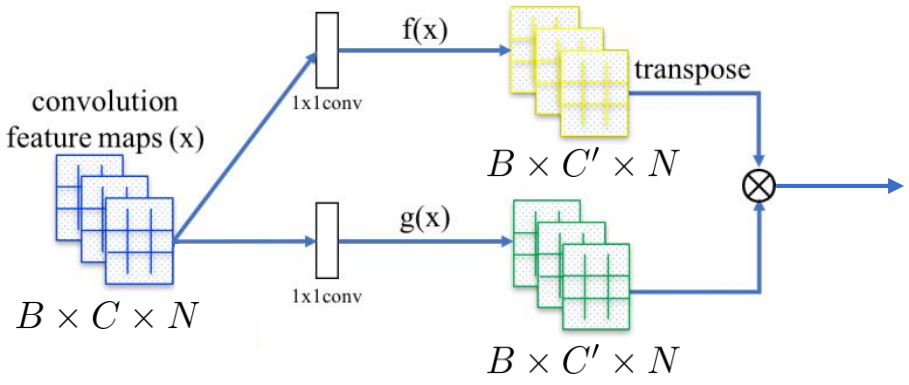
- Previous GANs fail to capture geometric or structural patterns that occur consistently in some classes.
- Convolution process is local, thus using conv layers alone is computationally inefficient for modeling long-range dependencies in images.
- They adapt the non-local model (i.e. self-attention module) of [Wang et al., CVPR 2018] for both the generator and the discriminator to efficiently model relationships between spatial regions.

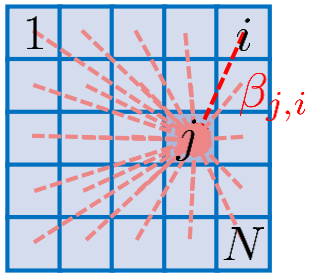


→ : attention between regions

* source : Zhang, et. al., Self-Attention Generative Adversarial Networks, ICML 2019

- The self-attention module for the SAGAN





$\beta_{j,i}$ for $i = 1, \cdots, N$
$j = 1, \cdots, N$

  - The Image features are first transformed into two feature spaces.

$$f(x) = W_f x,\ g(x) = W_g x$$

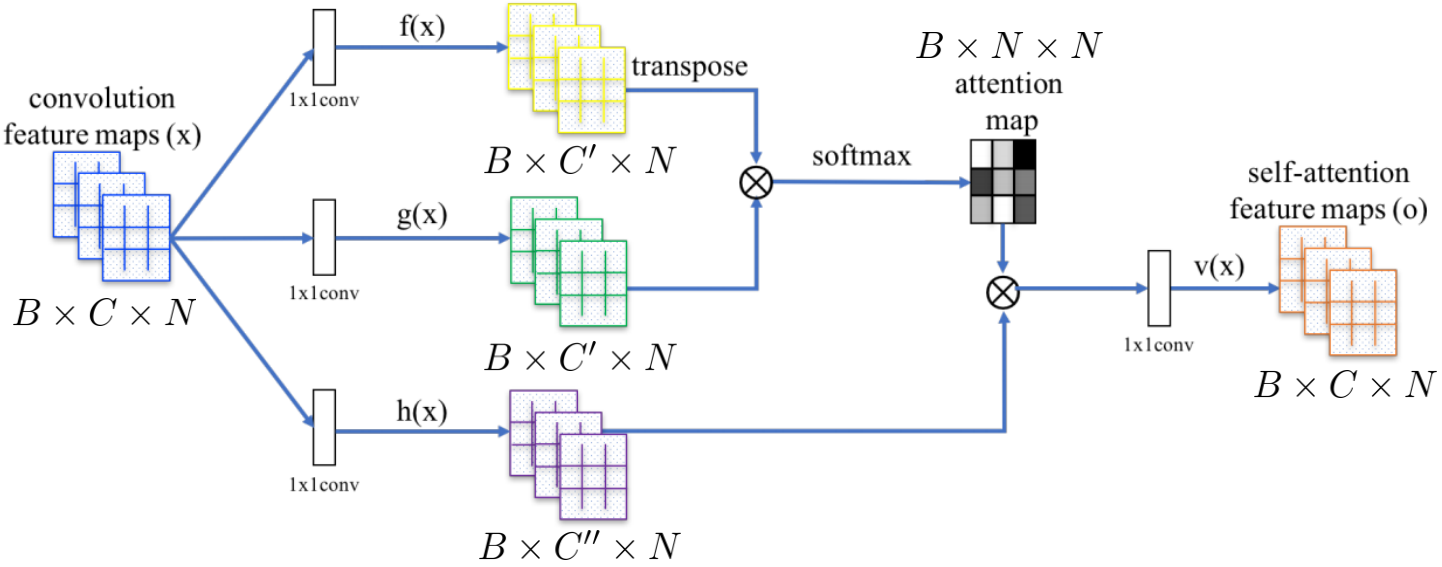
  - Then calculate the attention.

$$\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^{N} \exp(s_{ij})},\ \text{where } s_{ij} = f(x_i)^T g(x_j)$$

  - $\beta_{j,i}$ indicates the extent to which the model attends to the i th location when synthesizing the j th region.

* source : Zhang, et. al., Self-Attention Generative Adversarial Networks, ICML 2019

- The self-attention module for the SAGAN



- Here the output of the attention layer is:

$$o_j = v\left(\sum_{i=1}^{N} \beta_{j,i} h(x_i)\right), \; h(x_i) = W_h x_i, \; v(x_i) = W_v x_i.$$

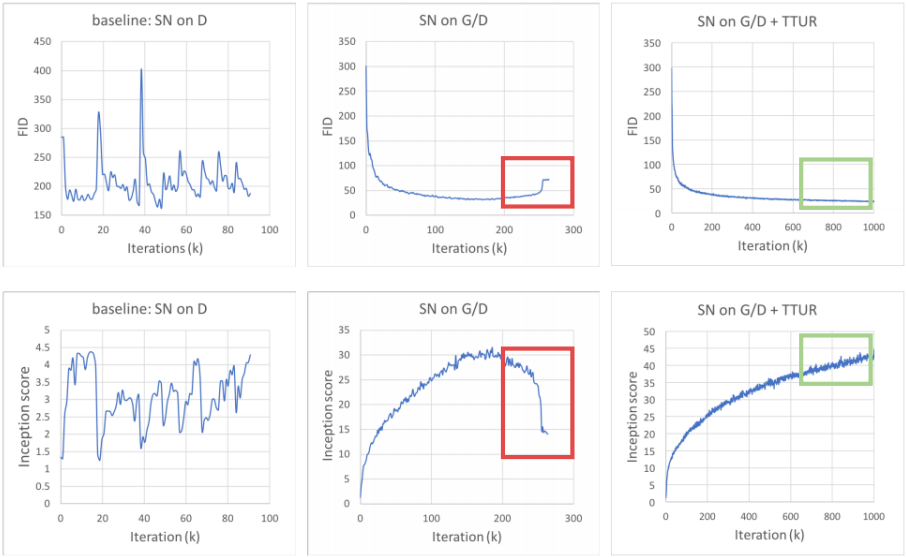
- In addition, multiply the output of the attention layer by a scale parameter and add back the input feature map (as similar as Residual block).

$$y_i = \gamma o_i + x_i$$

* source : Zhang, et. al., Self-Attention Generative Adversarial Networks, ICML 2019

Self-Attention GAN: Attention-Driven Image Generation Tasks

- They additionally proposed some stabilization techniques, such as Spectral Normalization on both G/D and two-timescale learning rates (TTUR), which is using separate learning rates for the G/D.



Training of G/D are well-balanced

- Comparison of Self-Attention and Residual block on GANs. These blocks are added into different features of the network.

Model	no attention	SAGAN				Residual			
		$feat_8$	$feat_{16}$	$feat_{32}$	$feat_{64}$	$feat_8$	$feat_{16}$	$feat_{32}$	$feat_{64}$
FID	22.96	22.98	22.14	**18.28**	18.65	42.13	22.40	27.33	28.82
IS	42.87	43.15	45.94	51.43	**52.52**	23.17	44.49	38.50	38.96

The improvements depend not only on residual connections, but also on attentions.

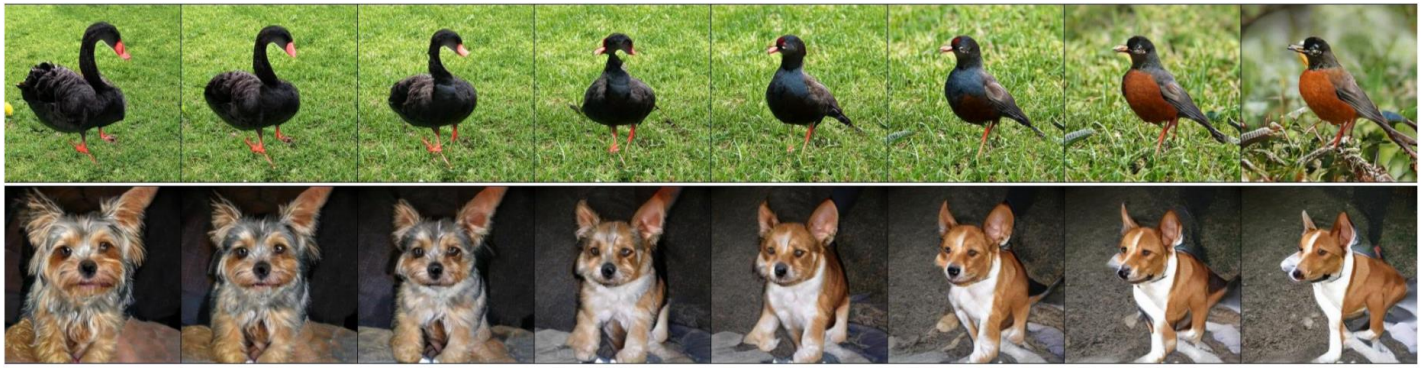
* source : Zhang, et. al., Self-Attention Generative Adversarial Networks, ICML 2019

Table of Contents

BigGAN: High-resolution, Diverse Image Generation

- BigGAN is a massive aggregation of recent techniques for training GANs.
- The model is a kind of conditional GANs, which can generate high-resolution, diverse samples from complex datasets such as ImageNet successfully.

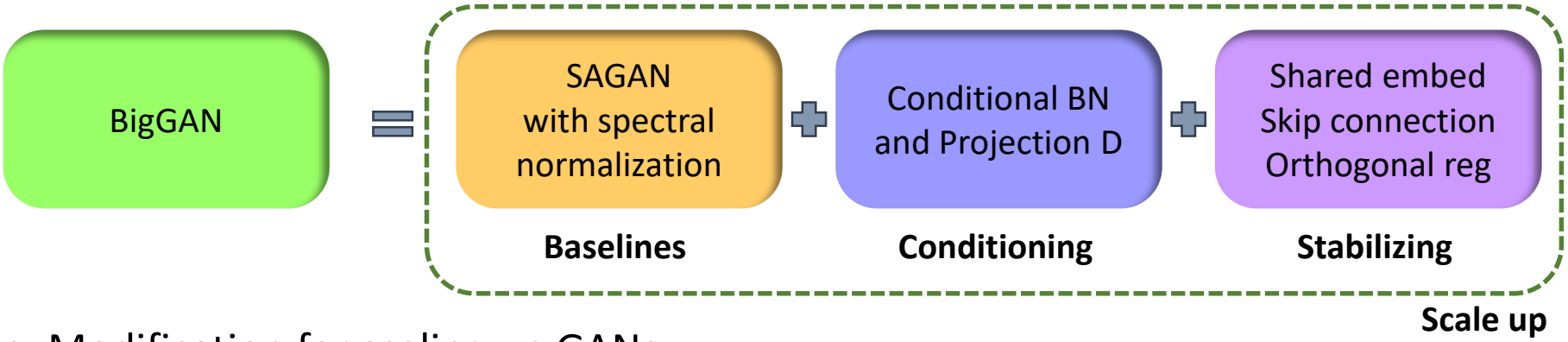
- When trained on ImageNet at 128×128 resolution, BigGAN improves the state-of-the-art Inception Score (IS) and Fréchet Inception Distance (FID).



* source : Brock, et. al., Large Scale GAN Training for High Fidelity Natural Image Synthesis, ICLR 2019

BigGAN: High-resolution, Diverse Image Generation

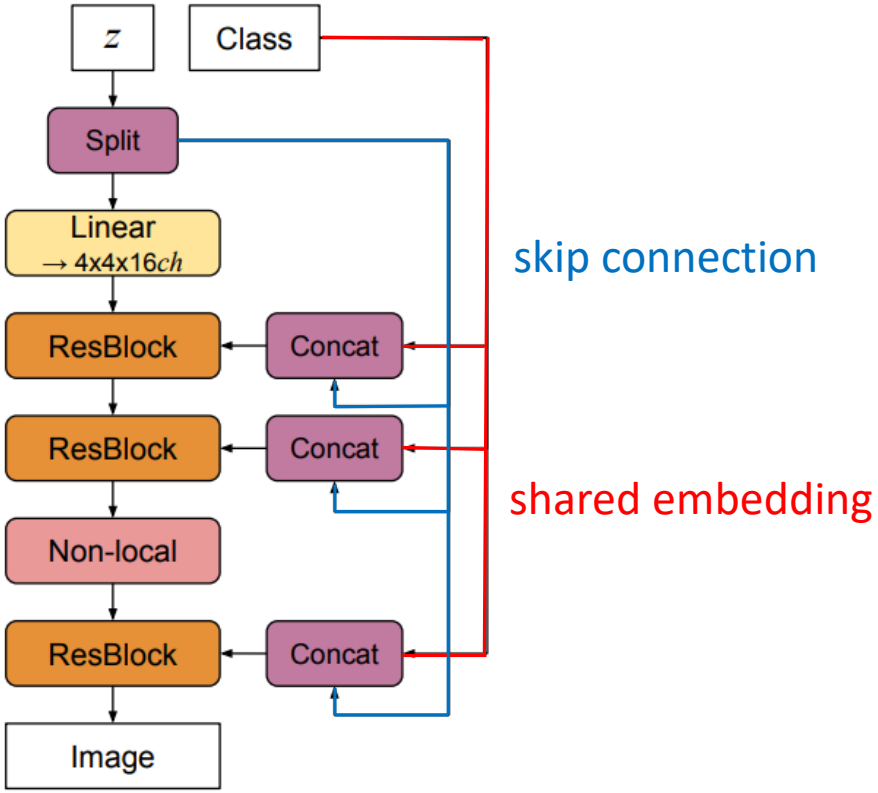
- Aggregation of previous techniques
  - The base model is Self-Attention GAN [Zhang et al., ICML 2019] using spectral normalization [Miyato et al., ICLR 2018].
  - They provide class information to the generator with class-conditional BatchNorm [Dumoulin et al., ICLR 2017; de Vries et al., NIPS 2017] and to the discriminator with projection [Miyato et al., ICLR 2018].
  - Progressive growing [Karras et al., ICLR 2018] is not used.



- Modification for scaling up GANs
  - Increasing the size of models and batches leads to a IS and FID improvement (This is why we call the model *big*).
  - Some techniques such as shared embedding, skip connection, and orthogonal regularization are used for stabilized training.

* source : Brock, et. al., Large Scale GAN Training for High Fidelity Natural Image Synthesis, ICLR 2019

- Instead of having a separate layer for each embedding [Miyato et al., ICLR 2018], they use a shared embedding, which is linearly projected to each layer's gains and biases [Perez et al., 2018].
- Skip connections (skip-z) from the noise vector z to multiple layers of G rather than just the initial layer. This design allows G to use the latent space to directly influence features at different resolutions and levels of hierarchy.



The architecture of the BigGAN

BigGAN: High-resolution, Diverse Image Generation

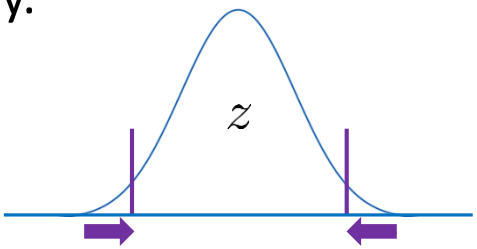
Batch	Ch.	Param (M)	Shared	Skip-z	Ortho.	Itr $\times 10^3$	FID	IS
256	64	81.5	SA-GAN Baseline			1000	18.65	52.52
512	64	81.5	✗	✗	✗	1000	15.30	58.77(±1.18)
1024	64	81.5	✗	✗	✗	1000	14.88	63.03(±1.42)
2048	64	81.5	✗	✗	✗	732	12.39	76.85(±3.83)
2048	96	173.5	✗	✗	✗	295(±18)	9.54(±0.62)	92.98(±4.27)
2048	96	160.6	✓	✗	✗	185(±11)	9.18(±0.13)	94.94(±1.32)
2048	96	158.3	✓	✓	✗	152(±7)	8.73(±0.45)	98.76(±2.84)
2048	96	158.3	✓	✓	✓	165(±13)	8.51(±0.32)	99.31(±2.10)
2048	64	71.3	✓	✓	✓	371(±7)	10.48(±0.10)	86.90(±0.61)

Scale up

Stabilize

- Simply increasing the batch size by a factor of 8 improves the state-of-the-art IS by 46%. Increasing the width (number of channels) in each layer by 50% leads to a further IS improvement of 21%.

- In the test time, "*truncation trick*", which reduces the variance of the generator's input, allows fine control over the trade-off between sample fidelity and variety.

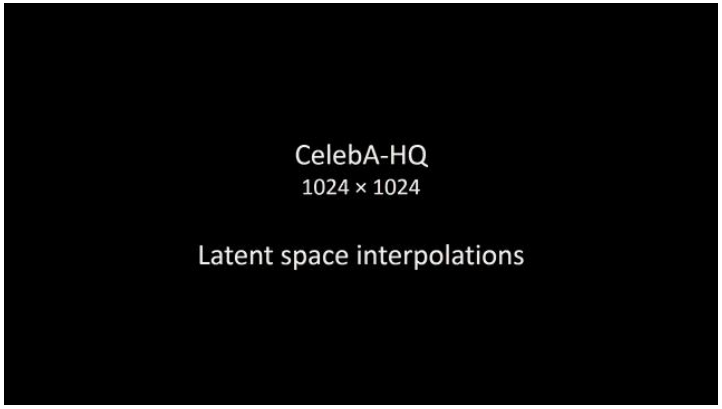


variety fidelity

* source : Brock, et. al., Large Scale GAN Training for High Fidelity Natural Image Synthesis, ICLR 2019

Table of Contents

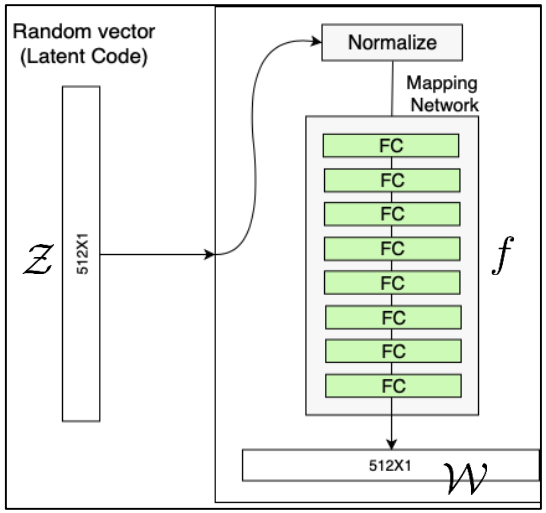
- Interpolation of latent-space vectors yields non-linear changes in the image. For example, features that are absent in either endpoint may appear in the middle of a linear interpolation path as belows:



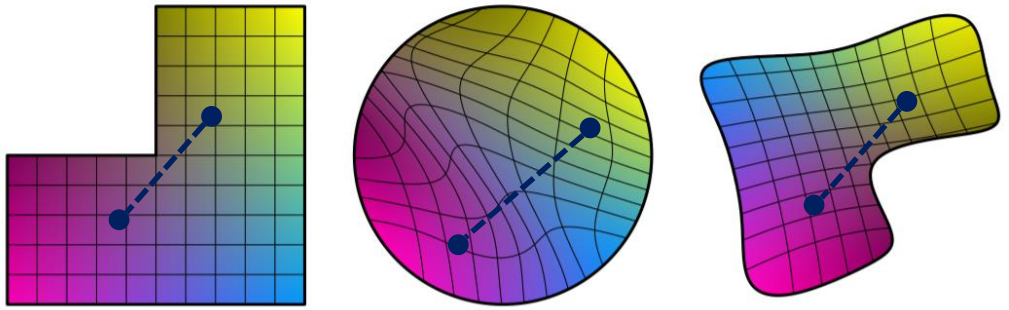
Latent space interpolations with Progressive GAN

- The input latent space must follow the probability density of the training data, and this leads to some degree of **unavoidable entanglement**.
- [Karras et al., 2019] proposed an intermediate latent space which is free from that restriction and is therefore **allowed to be disentangled**.

- Instead of using the latent variable as an input directly, authors embed it into an *intermediate latent space* using a non-linear mapping network $f : \mathcal{Z} \rightarrow \mathcal{W}$.
- The mapping network f is implemented using an 8-layer MLP.



The mapping network.



(a) Distribution of features in training set

(b) Mapping from $\mathcal{Z}$ to features

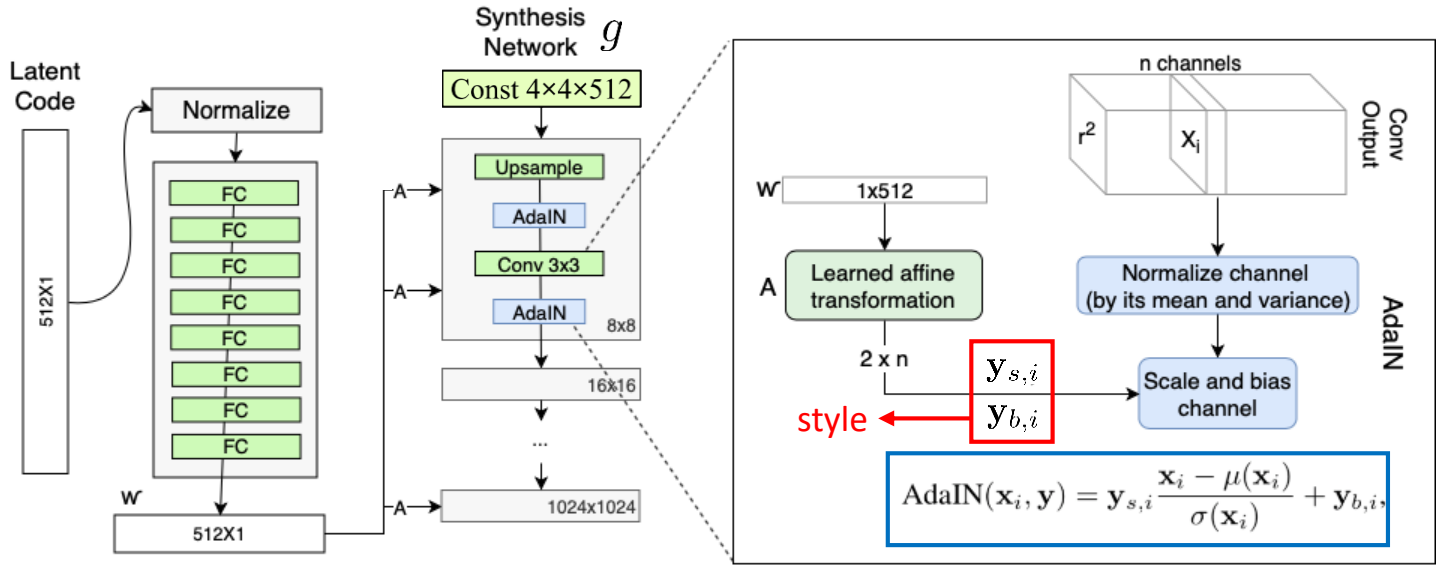
(c) Mapping from $\mathcal{W}$ to features

Illustration of disentanglement.

- As illustrated above, complex data distribution forces the mapping from $\mathcal{Z}$ to features to become curved (b), while non-linear mapping $\mathcal{W}$ is well adapted (c).
- A curved latent space is highly entangled.

StyleGAN: An Alternative Style-Based Generator Architecture

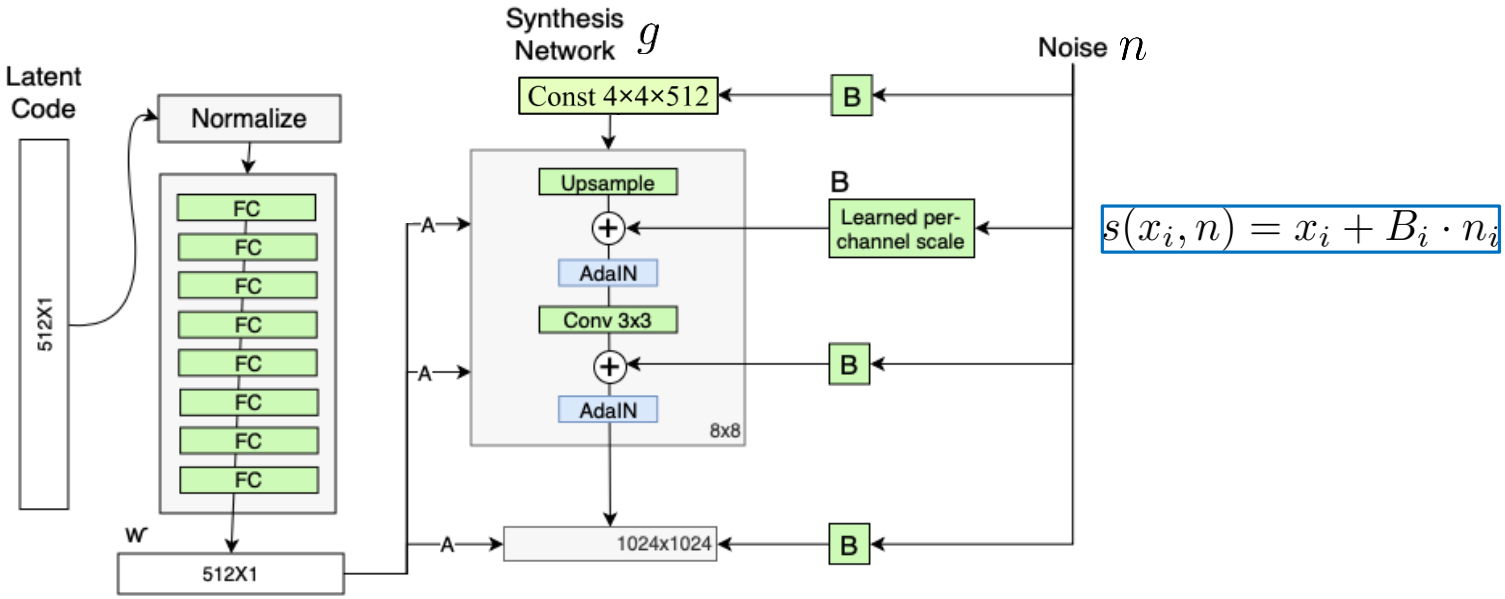
- Motivated by style transfer literature [Huang et al., 2017], the generator starts from a learned constant input and adjusts the "style" which is a learned affine-transformation of an intermediate latent code.
- Styles $\mathbf{y}$ control adaptive instance normalization (AdaIN) [Huang et al., 2017] operations after each convolution layer of the synthesis network g.



- The styles control high-level attributes (e.g., pose, identity of face images).

* source : Rani Horev's blog

- To generate stochastic detail, the authors introduce explicit noise inputs. These are single-channel images consisting of uncorrelated Gaussian noise, and they are fed to each layer of the synthesis network.
- The noise image is broadcasted to all feature maps using learned per feature scaling factors which are denoted as B, and then added to the output of the corresponding convolution, as illustrated as below.

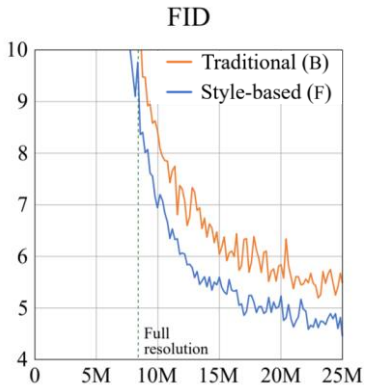


- The noise controls stochastic variation (e.g., freckles, hair of face images).

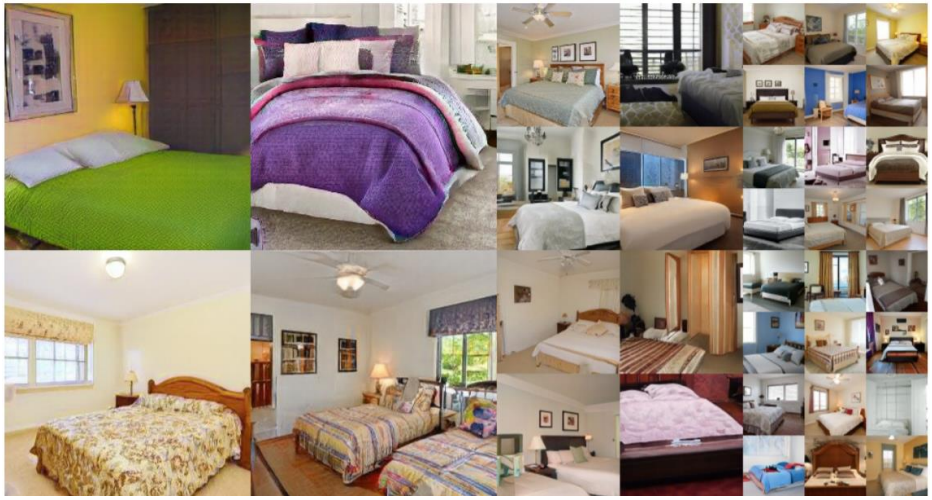
* source : Rani Horev's blog

- The generator improves the state-of-the-art in terms of traditional distribution quality metrics such as FID on face image dataset.

Method	CelebA-HQ	FFHQ
A Baseline Progressive GAN [30]	7.79	8.04
B + Tuning (incl. bilinear up/down)	6.11	5.25
C + Add mapping and styles	5.34	4.85
D + Remove traditional input	5.07	4.88
E + Add noise inputs	**5.06**	4.42
F + Mixing regularization	5.17	**4.40**



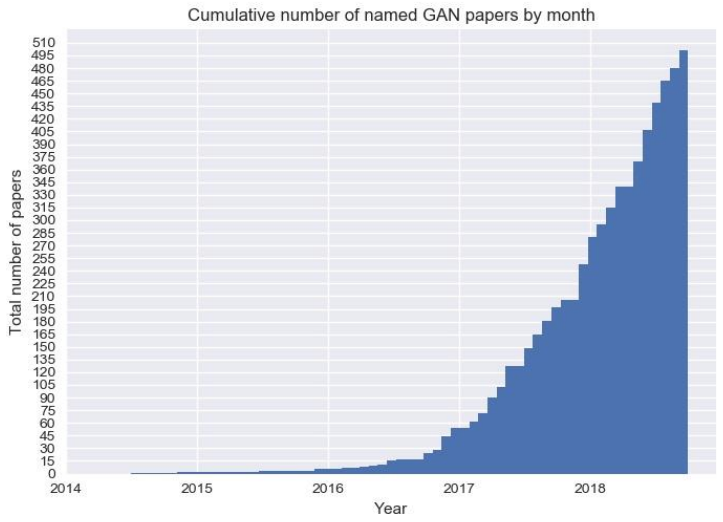
- It also leads to demonstrably better interpolation properties, and also better disentangles the latent factors of variation.



The GAN-Zoo

- Lots of *GAN papers* are published since 2014
- Hundreds of papers about theories and applications
  - About better training and various applications to many types of dataset/tasks
  - If you are interested for more, see the-gan-zoo (https://github.com/hindupuravinash/the-gan-zoo)

- 3D-ED-GAN - Shape Inpainting using 3D Generative Adversarial Network and Recurrent Convolutional Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling (github)
- 3D-IWGAN - Improved Adversarial Systems for 3D Object Generation and Reconstruction (github)
- 3D-PhysNet - 3D-PhysNet: Learning the Intuitive Physics of Non-Rigid Object Deformations
- 3D-RecGAN - 3D Object Reconstruction from a Single Depth View with Adversarial Learning (github)
- ABC-GAN - ABC-GAN: Adaptive Blur and Control for improved training stability of Generative Adversarial Networks (github)
- ABC-GAN - GANs for LIFE: Generative Adversarial Networks for Likelihood Free Inference
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- ACGAN - Coverless Information Hiding Based on Generative adversarial networks
- acGAN - On-line Adaptative Curriculum Learning for GANs
- ACtuAL - ACtuAL: Actor-Critic Under Adversarial Learning
- AdaGAN - AdaGAN: Boosting Generative Models
- Adaptive GAN - Customizing an Adversarial Example Generator with Class-Conditional GANs
- AdvEntuRe - AdvEntuRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples
- AdvGAN - Generating adversarial examples with adversarial networks
- AE-GAN - AE-GAN: adversarial eliminating with GAN
- AE-OT - Latent Space Optimal Transport for Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AF-DCGAN - AF-DCGAN: Amplitude Feature Deep Convolutional GAN for Fingerprint Construction in Indoor Localization System
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AIM - Generating Informative and Diverse Conversational Responses via Adversarial Information Maximization



References

[Goodfellow, et. al., 2014] Generative adversarial nets, NIPS 2014
link: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[Theis, et. al., 2016] A note on the evaluation of generative models, ICLR 2016
link: http://bethgelab.org/media/publications/1511.01844v1.pdf

[Radford, et. al., 2015] Unsupervised representation learning with deep convolutional generative adversarial networks.
link: https://arxiv.org/pdf/1511.06434.pdf

[Ledig, et. al., 2017] Photo-realistic single image super-resolution using a generative adversarial networks, CVPR 2017
link: http://openaccess.thecvf.com/content_cvpr_2017/papers/Ledig_Photo-Realistic_Single_Image_CVPR_2017_paper.pdf

[Zhu, et. al., 2017] Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017
link: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8237506

[Karras, et. al., 2018] Progressive growing of GANs for improved quality, stability, and variation, ICLR 2018
link: https://arxiv.org/abs/1710.10196

[Salimans, et. al., 2016] Improved techniques for training GANS, NIPS 2016
link: https://arxiv.org/abs/1606.03498

[Huszar 2015] How (not) to train your generative model: scheduled sampling, likelihood, adversary?
link: https://arxiv.org/pdf/1511.05101.pdf

[Mirza et al., 2014] Conditional Generative Adversarial Nets
link: https://arxiv.org/pdf/1411.1784.pdf

References

[Odena, et. al., 2017] Conditional Image Synthesis with Auxiliary Classifier GANs, ICML 2017
link: https://arxiv.org/pdf/1610.09585.pdf

[Basart et. al., 2017] Analysis of Generative Adversarial Models
link: https://newtraell.cs.uchicago.edu/files/ms_paper/xksteven.pdf

[Dumoulin, et. al., 2017] A Learned Representation for Artistic Style, ICLR 2017
link: https://arxiv.org/pdf/1610.07629.pdf

[de Vries, et. al., 2017] Modulating early visual processing by language, NIPS 2017
link: https://arxiv.org/pdf/1707.00683.pdf

[Miyato, et. al., 2018] cGANs with Projection Discriminator, ICLR 2018
link: https://arxiv.org/pdf/1802.05637.pdf

[Arjovsky, et. al., 2017] Wasserstein GAN, ICML 2017
link: https://arxiv.org/pdf/1701.07875.pdf

[Arjovsdky and Bottou, 2017] Towards principled methods for training generative adversarial networks, ICLR 2017
link: https://arxiv.org/pdf/1701.04862.pdf

[Villani, 2009] Optimal transport: old and new, Grundlehren der mathematischen wissenschaften 2009
link: http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf

[Reed, et. al., 2016] Generative adversarial text to image synthesis, ICML 2016
link: https://arxiv.org/pdf/1605.05396.pdf

[Wang, et. al., 2004] Image quality assessment: from error visibility to structural similarity, IEEE transactions on image processing 2004
link: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1284395

References

[Radford, et. al., 2015] Unsupervised representation learning with deep convolutional generative adversarial networks, 2015
link: https://arxiv.org/pdf/1511.06434.pdf

[Gulrajani, et. al., 2017] Improved training of Wasserstein GANs, NIPS 2017
link: https://arxiv.org/pdf/1704.00028.pdf

[Miyato, et. al., 2018] Spectral normalization for generative adversarial networks, ICLR 2018

link: https://arxiv.org/pdf/1802.05957.pdf

[Zhang, et. al., 2019] Self-Attention Generative Adversarial Networks, ICML 2019
link: https://arxiv.org/pdf/1805.08318.pdf

[Wang, et. al., 2018] Non-local neural networks, CVPR 2018

link: https://arxiv.org/pdf/1711.07971.pdf

[Brock, et. al., 2019] Large Scale GAN Training for High Fidelity Natural Image Synthesis, ICLR 2019

link: https://arxiv.org/pdf/1809.11096.pdf

[Perez, et. al., 2019] FiLM: Visual Reasoning with a General Conditioning Layer, AAAI 2018

link: https://arxiv.org/pdf/1709.07871.pdf

[Karras, et. al., 2019] Large Scale GAN Training for High Fidelity Natural Image Synthesis, CVPR 2019

link: https://arxiv.org/pdf/1809.11096.pdf

[Huang, et. al., 2017] Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization, ICCV 2017

link: https://arxiv.org/pdf/1703.06868.pdf