

Advanced Deep Generative Models I: Implicit Density Models

AI602: Recent Advances in Deep Learning
Lecture 5

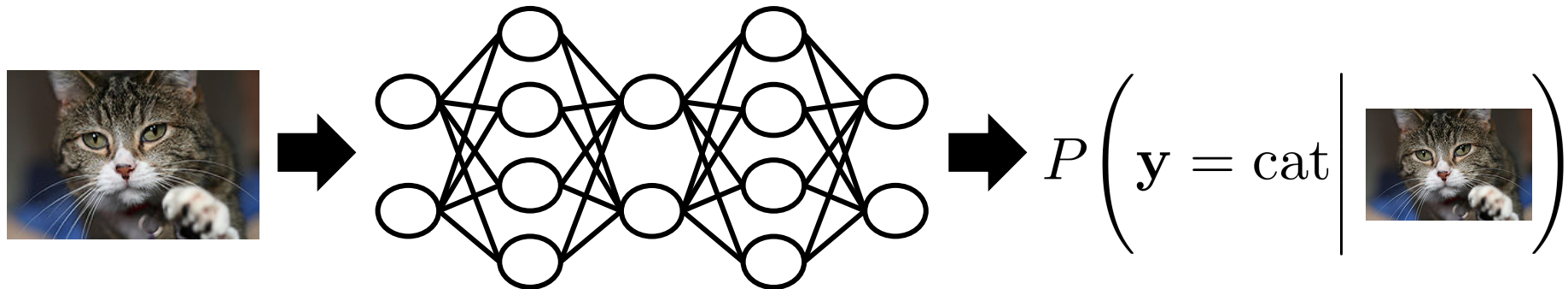
Slide made by

Jongheon Jeong, Jongjin Park, Hankook Lee, Sihyun Yu

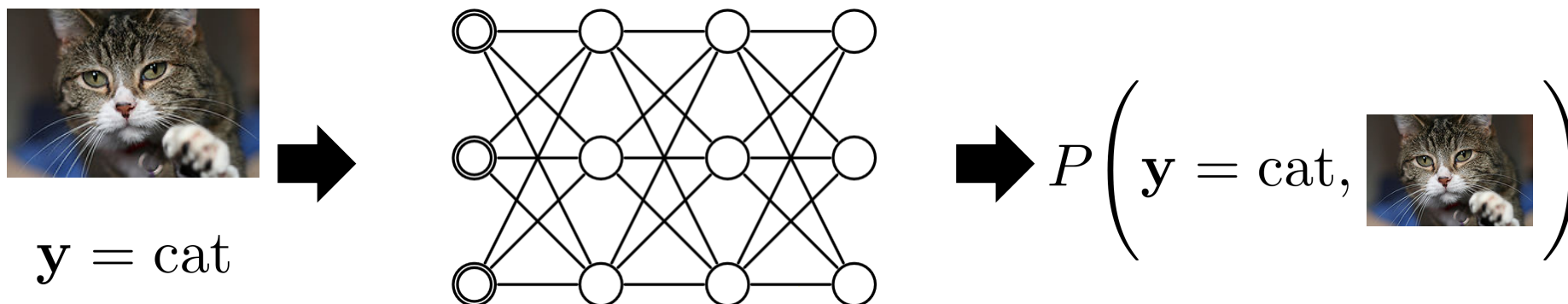
KAIST AI & EE

Generative Model and Discriminative Model

- Given an **observed variable** \mathbf{x} and a **target variable** \mathbf{y}
- Discriminative modeling** estimates the **conditional distribution** $P(\mathbf{y}|\mathbf{x})$
 - Example:** ImageNet classifiers

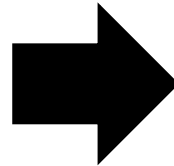
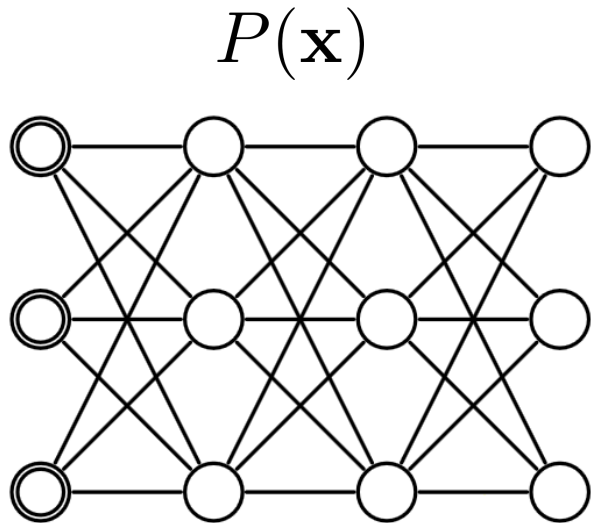


- Generative modeling** estimates the **joint distribution** $P(\mathbf{x}, \mathbf{y})$
 - Example:** Boltzmann machines, sum-product networks



Why Generative Model?

- Without assuming y , generative models learn $P(\mathbf{x})$ from given data
- $P(\mathbf{x})$ enables us to **generate new data** similar to the training dataset
- We can use various **sampling methods** for generation based on $P(\mathbf{x})$
 - Is it possible to do the same thing with discriminative models?



$\sim P(\mathbf{x})$



$\sim P(\mathbf{x})$



$\sim P(\mathbf{x})$



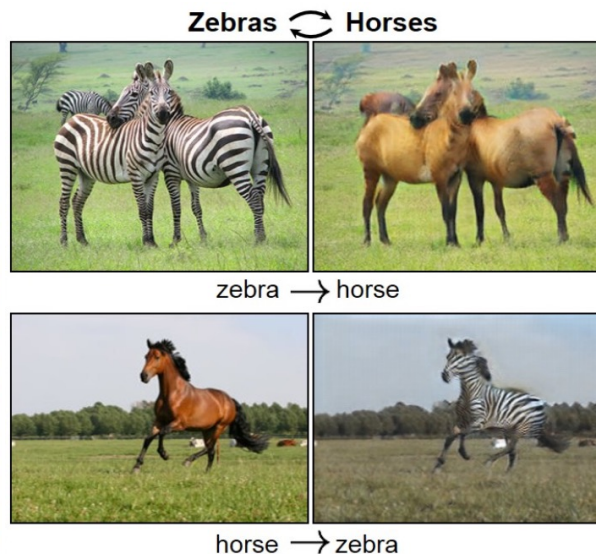
$\sim P(\mathbf{x})$

Why Generative Model?

- $P(\mathbf{x})$ enables us to **generate new data** similar to the training dataset
- Many real-world problems can be formulated assuming generative models
- Common applications?
 - Vision: super-resolution, style transfer, image inpainting, ...
 - Audio: audio synthesis, speech generation, voice conversion, ...
 - And many more..



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



Style transfer [Zhu et. al., 2017]



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

Two Types of Generative Models

- **Explicit models** directly estimate the (usually “unnormalized”) data distribution
 - **Example 1:** 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
 - **Example 2:** Graphical models (RBM, DBM, ...)
 - $P(\mathbf{x}) \propto \exp\left(\sum_i b_i x_i + \sum_{i,j} w_{ij} x_i x_j\right)$
 - Intractable inference, high expressive power with compact representations
 - **Many more examples:** Variational Auto-encoder, Flow-based models, ...
 - ... which we will be discussed more in the next lecture
- **Generative adversarial network (GAN)** is an instance of **implicit models**
 - One does not have to access $P(\mathbf{x})$ for sampling \rightarrow More efficient in some cases
 - $P(\mathbf{x})$ is rather **implicitly defined** by its model
 - GANs assume that $P(x) \sim G(z)$
 - z = “random noise”, and G = “a neural network”
 - **Sampling?** A simple forward computation of $G(z)$

Two Types of Generative Models

- **Explicit models** directly estimate the (usually “unnormalized”) data distribution
 - **Example 1:** 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
 - **Example 2:** Graphical models (RBM, DBM, ...)
 - $P(\mathbf{x}) \propto \exp\left(\sum_i b_i x_i + \sum_{i,j} w_{ij} x_i x_j\right)$
 - Intractable inference, high expressive power with compact representations
 - **Many more examples:** Variational Auto-encoder, Flow-based models, ...
 - ... which we will be discussed more in the next lecture
- **Generative adversarial network (GAN)** is an instance of **implicit models**
 - One does not have to access $P(\mathbf{x})$ for sampling \rightarrow More efficient in some cases
 - $P(\mathbf{x})$ is rather **implicitly defined** by its model
 - GANs assume that $P(x) \sim G(z)$
 - z = “random noise”, and G = “a neural network”
 - **Sampling?** A simple forward computation of $G(z)$

1. Basics of Generative Adversarial Networks (GAN)

- Advantages and disadvantages of GAN
- Conditional GANs

2. Advanced Topics of GANs

- Loss, regularization and normalization
- Data augmentations for GANs
- GAN architectures
- GAN training with pretrained networks

3. Beyond Image Generation: More Applications

- GANs for image manipulation
- GANs for video generation
- GANs for medical diagnosis

1. Basics of Generative Adversarial Networks (GAN)

- Advantages and disadvantages of GAN
- Conditional GANs

2. Advanced Topics of GANs

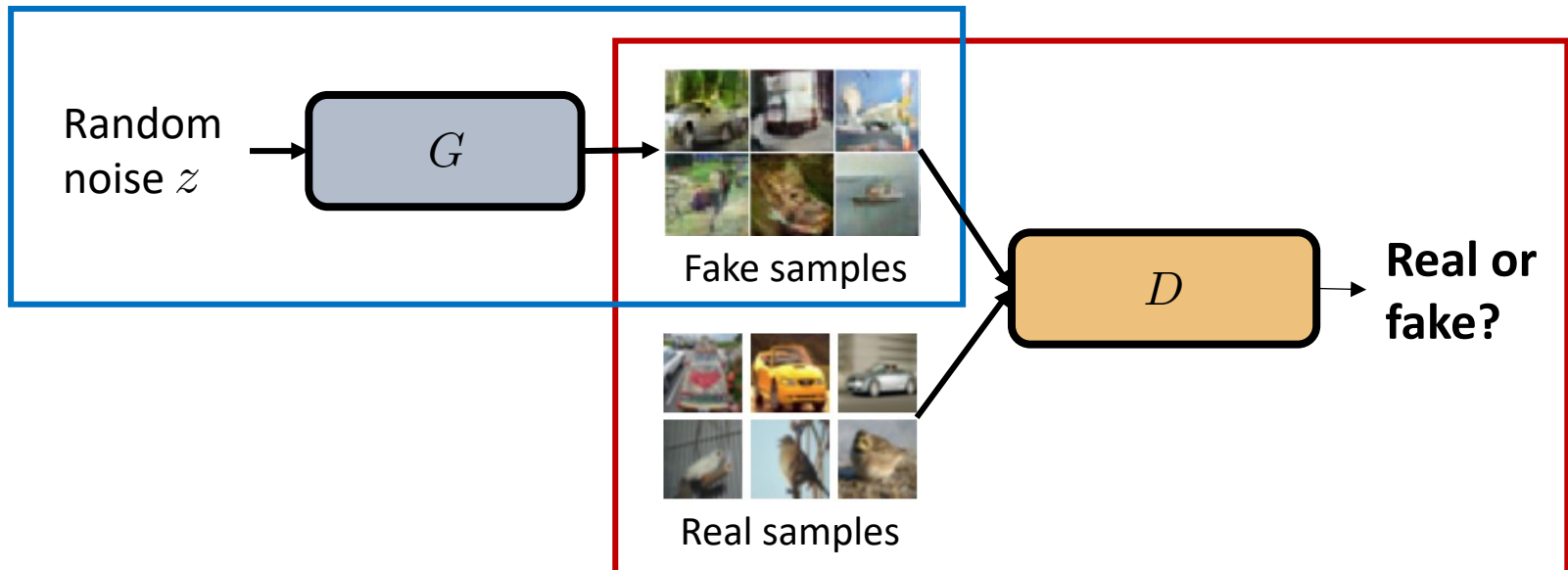
- Loss, regularization and normalization
- Data augmentations for GANs
- GAN architectures
- GAN training with pretrained networks

3. Beyond Image Generation: More Applications

- GANs for image manipulation
- GANs for video generation
- GANs for medical diagnosis

Generative Adversarial Networks (GAN)

- Classical (usually explicit) generative methods struggle on complex data
 - Sampling from **high-dimensional, complex** distributions can be intractable
- GANs [Goodfellow, et. al., 2014] do not explicitly model $p_{\text{model}}(\mathbf{x})$
 - **Two player game** between **discriminator network** D and **generator network** G
 - D **tries to discriminate** real data and samples generated by G (“fake” samples)
 - G **tries to fool** D by generating more “realistic” images
 - GAN utilizes **neural networks** to model the sampling function itself



Generative Adversarial Networks (GAN)

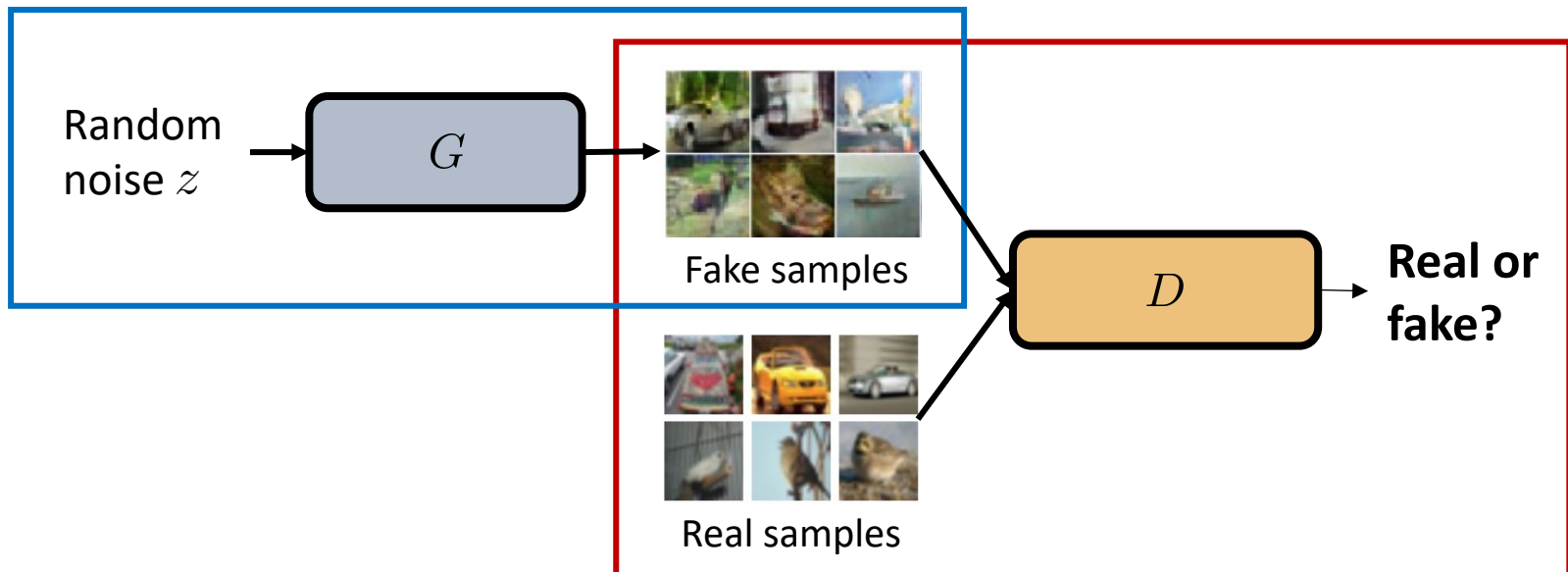
- **Two player game** between **discriminator network** D and **generator network** G
- **Training objective:**

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

Discriminator output
for real data

Discriminator output
for generated fake data

- D **maximizes** the objective: $D(x) \rightarrow 1$ and $D(G(z)) \rightarrow 0$
- G **minimizes** the objective: $D(G(z)) \rightarrow 1$



- **Training objective** [Goodfellow, et. al., 2014]:

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

- Alternative training between D and G

- **Objective for D :**

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

- **Objective for G :**

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

- In practice, directly optimizing the **G-objective** can be problematic
 - (cont'd) ... will be discussed in the later slides

- **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 Happens in the GAN Objective?

- **Generator**

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

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

- Provided that the discriminator (D) is **optimal**
 - G-objective = minimizing the **Jensen-Shannon (JS) divergence**
 - Many previous generative models used the KL divergence (a.k.a. Max. likelihood)
 - **KL divergence vs JS divergence?**
 - JS helps to capture sharper and clearer modes in the distribution
 - But JS can cause a missing mode problem (“**mode collapse**”)

- Training GANs via alternating updates between D and G
- **Recall:** G optimizes JS divergence when D is optimal
 - **Q:** But how can we ensure that D is indeed “optimal”?
 - Simplest practice: Just update D more (e.g., for **k -steps**) per each G update

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

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

end for

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

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

end for

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

- Alternative training between D and G

- **Objective for D :**

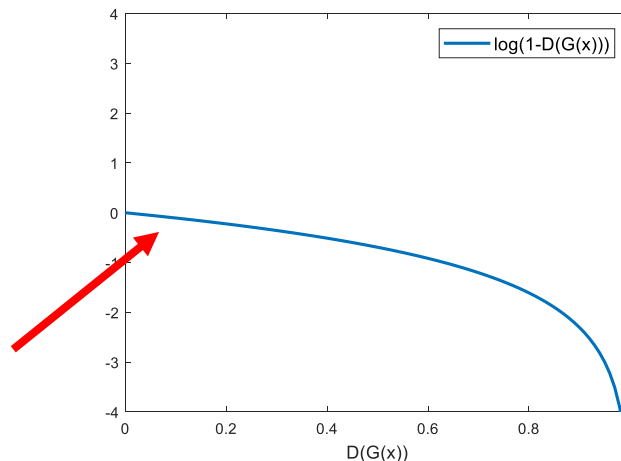
$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

- **Objective for G :**

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

- In practice, directly optimizing the **G-objective** can be problematic
 - **Gradient vanishing:** Especially when $G(z)$ looks “bad” to D (e.g., early of training)
 - Learning via back-propagation becomes significantly difficult

Gradients $\rightarrow 0$ when
 G is too worse than D



- Alternative training between D and G

- **Objective for D :**

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

- **Objective for G :**

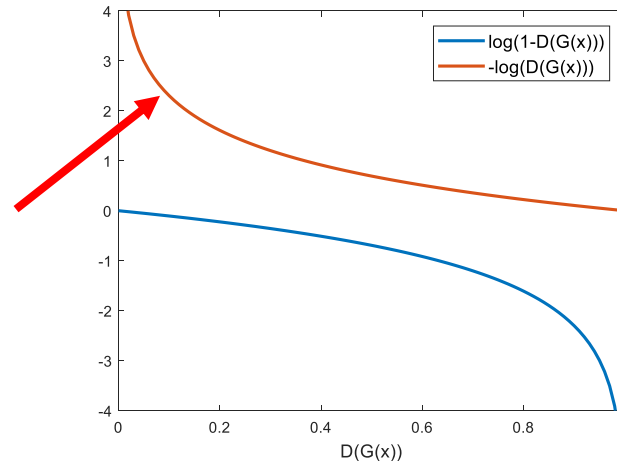
$$\min_{\theta_g} \mathbb{E}_{z \sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

- **Non-saturating loss** is practically more favorable in this respect

$$-\log(D_{\theta_d}(G_{\theta_g}(z)))$$

- This G-objective gives much stronger gradients in this scenario

Stronger gradients when
G is too worse than D



1. Basics of Generative Adversarial Networks (GAN)

- Advantages and disadvantages of GAN
- Conditional GANs

2. Advanced Topics of GANs

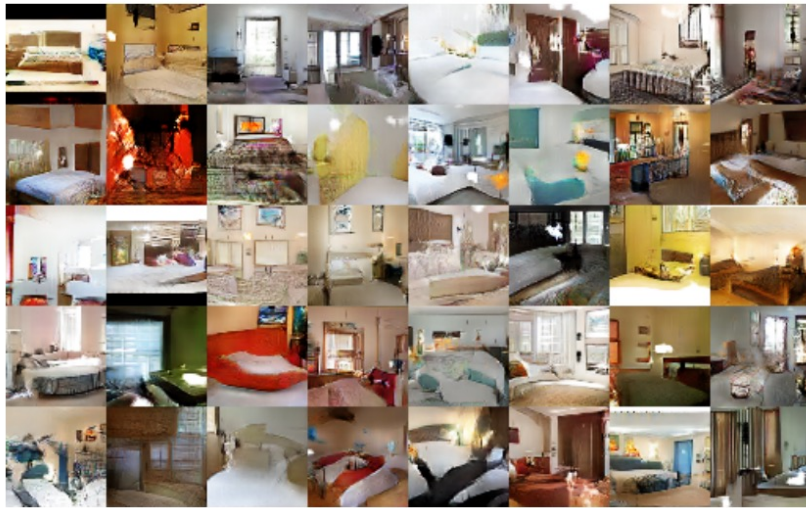
- Loss, regularization and normalization
- Data augmentations for GANs
- GAN architectures
- GAN training with pretrained networks

3. Beyond Image Generation: More Applications

- GANs for image manipulation
- GANs for video generation
- GANs for medical diagnosis

Generated Samples with GAN

- GAN could generate “sharper” and “clearer” images than previous approaches
 - Most of the previous works suffered from “blurred”, unrealistic generations



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 can be a possible explanation?

- 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 $p_{\text{data}}(x) > 0, p_g(x) \rightarrow 0$, 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 $p_{\text{data}}(x) \rightarrow 0, p_g(x) > 0$, 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

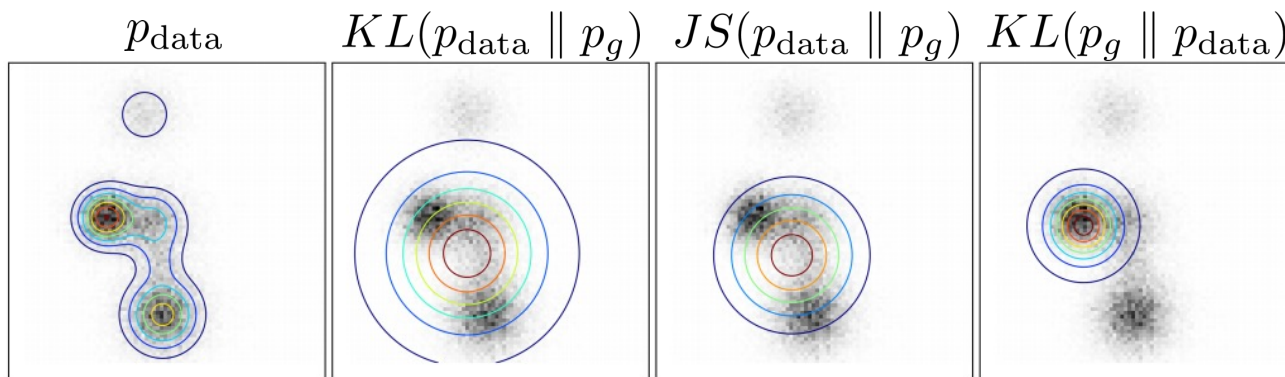
- Maximum likelihood methods (= KL divergence minimization)

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

- 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

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

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



1. Training instability

- GANs are notoriously **unstable to train**, with much sensitivity to hyperparameters
- GAN as a two-player non-cooperative game [Salimans, et. al., 2016]
 - The **Nash equilibrium** of such games can be extremely hard to achieve
 - Reducing the D-objective can significantly increase the G's, and vice versa

2. Mode collapse problem

- G can “collapse” to **produce the same outputs** to beat D
- G may easily fool D if it is good at making a **single perfect image**
- JS itself does not explicitly penalize such cases as much as KL



Examples of the mode collapse problem in GAN

1. Basics of Generative Adversarial Networks (GAN)

- Advantages and disadvantages of GAN
- Conditional GANs

2. Advanced Topics of GANs

- Loss, regularization and normalization
- Data augmentations for GANs
- GAN architectures
- GAN training with pretrained networks

3. Beyond Image Generation: More Applications

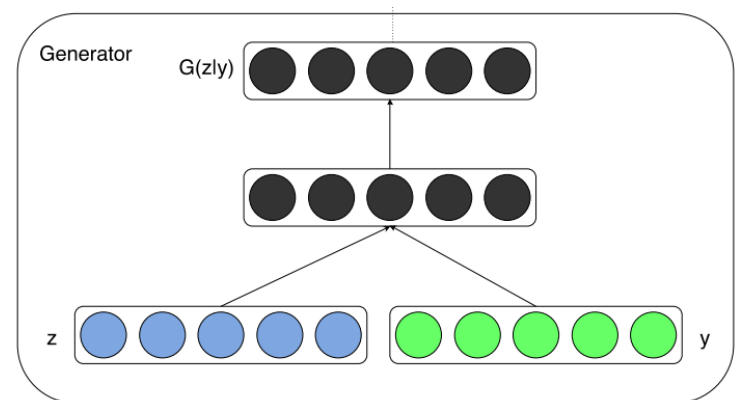
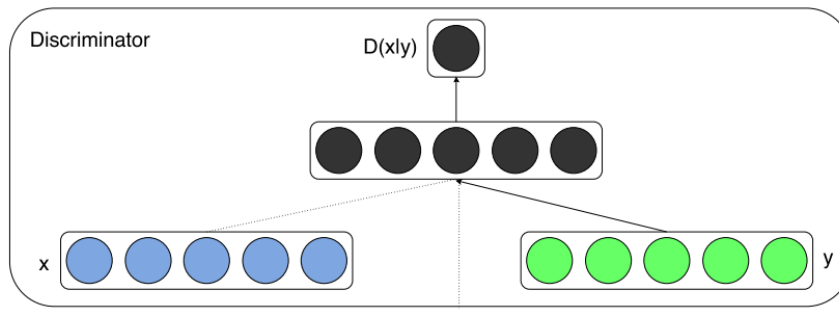
- GANs for image manipulation
- GANs for video generation
- GANs for medical diagnosis

- By default, the standard GANs are **unconditional**
 - One cannot control **the mode of the distribution** to be generated
- **Conditional GANs** (cGANs) aim to incorporate **an additional attribute** y
 - (+) Controllable generation (e.g., class-wise generation)
 - (+) Improved quality for complex generation tasks
- **Recall:** Training objective for unconditional GAN [Goodfellow et al., 2014]:

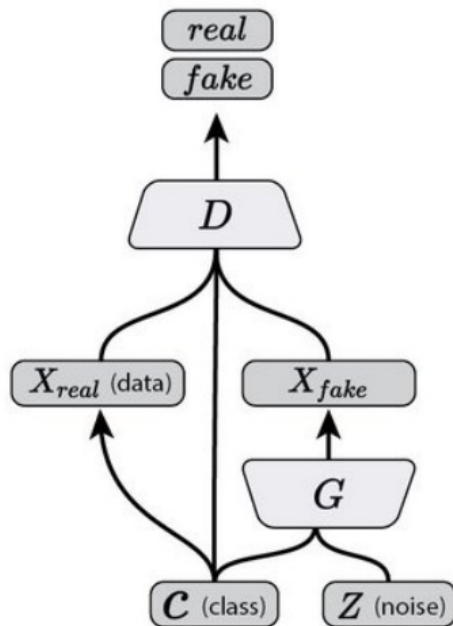
$$\min_G \max_D [\mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$$

- Mirza et al. (2014) formulated a cGAN objective by:

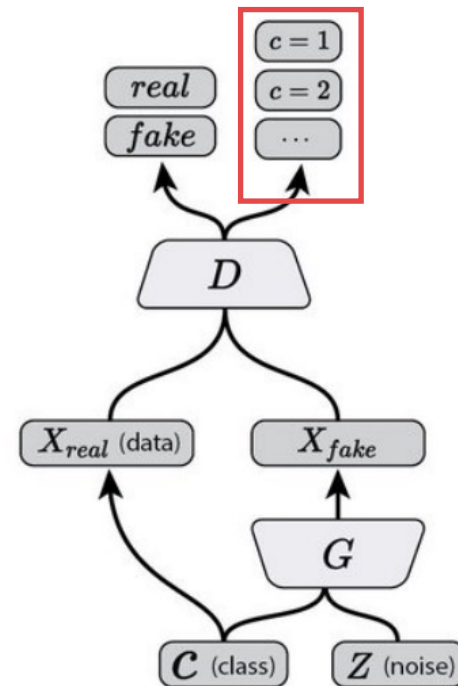
$$\min_G \max_D [\mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x|y) + \mathbb{E}_{z \sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z|y)))]$$



- Many works have been proposed then to better encode y
 - e.g., Reed et al. (2016): Concatenate y to inputs for hidden features of D
- Odena et al. (2017): **Auxiliary Classifier GAN (ACGAN)**
 - Modified D to have an **auxiliary classifier** for the class of both real and fake inputs
 - D should preserve the information to **reconstruct the class** as well as “real vs. fake”



Conditional GAN
[Mirza et al., 2014]



ACGAN
[Odena et al., 2017]

- The training objective function consists of two parts:

- **GAN loss:** the **log-likelihood of the correct source, L_S**

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

- **Classification loss:** the **log-likelihood of the correct class, L_C**

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

- D maximizes $L_S + L_C$, while G maximizes $-L_S + L_C$

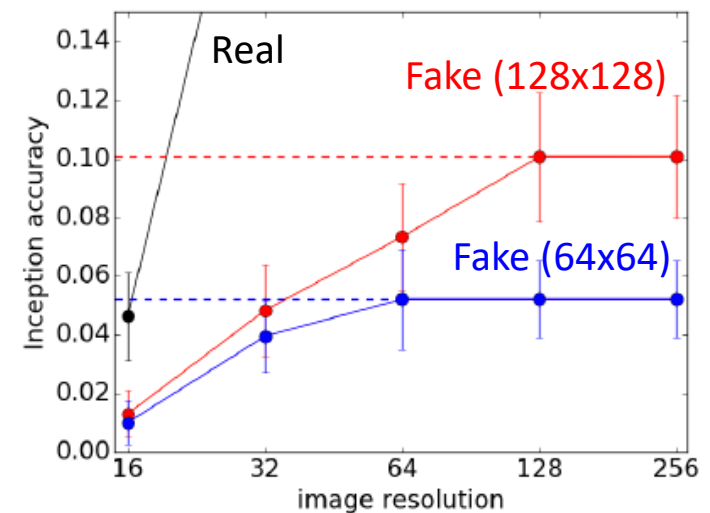
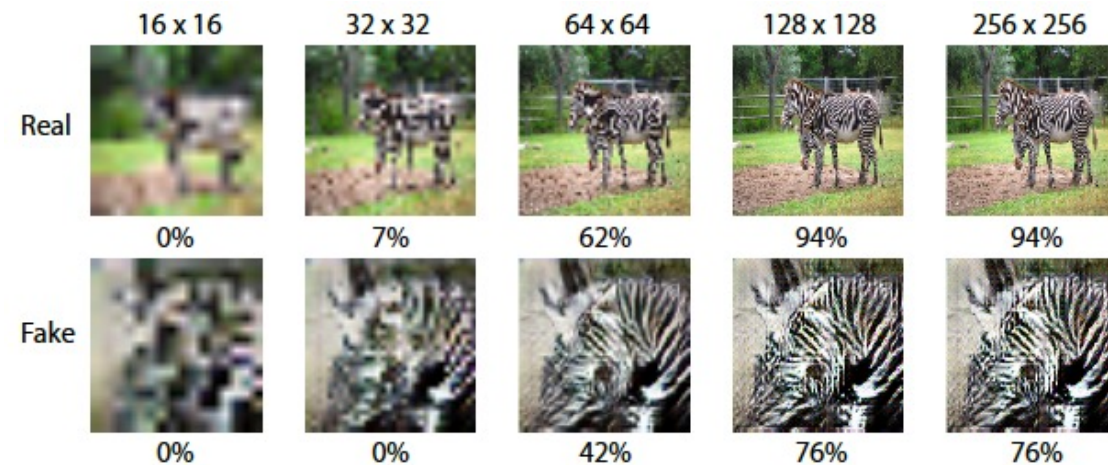
- **Remark:** L_C is used not only for D, but also G
- **Remark:** There can be some balancing weight for both losses for better training
 - i.e., D and G maximize $L_S + \lambda_1 L_C$ and $-L_S + \lambda_2 L_C$, respectively

Conditional GANs: Auxiliary Classifier GAN (ACGAN)

- ACGAN could allow **diverse & higher resolution images** than previous cGANs
 - The first cGAN approach that could scale up to the **ImageNet dataset**

1. Evaluation of cGAN conditioning via Inception accuracy

- Increased discriminability on Inception → Better conditioning
- Higher-res conditional generations via ACGAN improves Inception accuracy

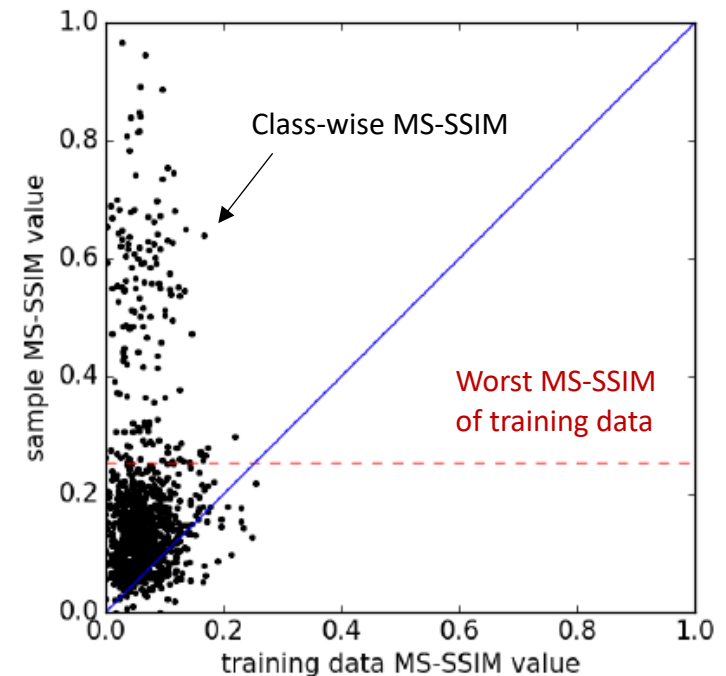
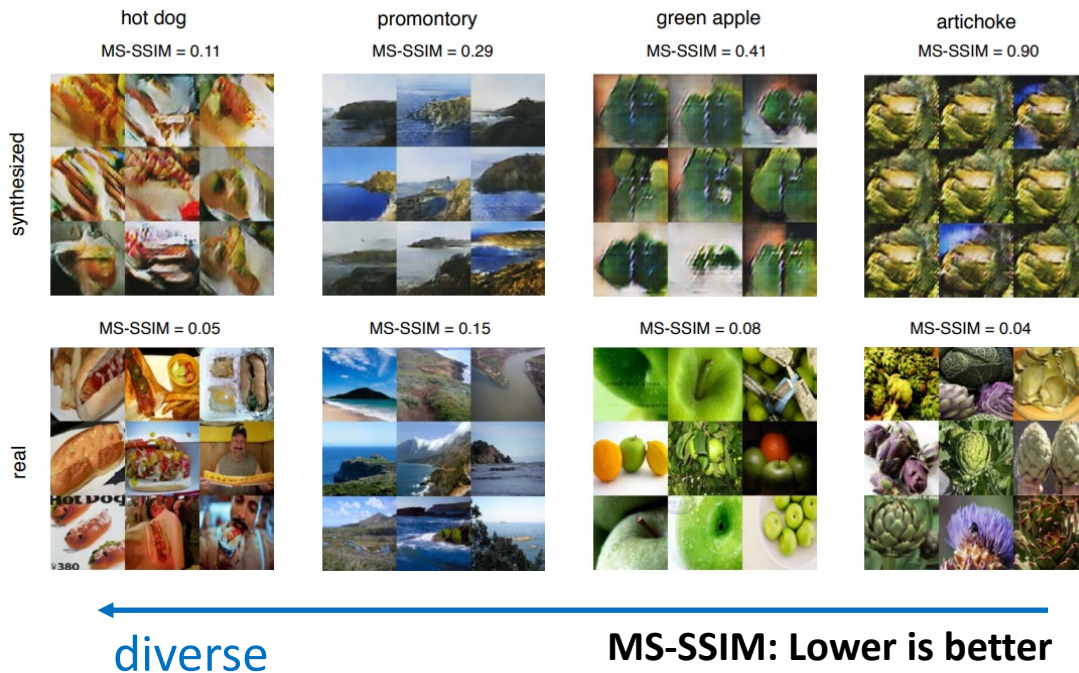


Conditional GANs: Auxiliary Classifier GAN (ACGAN)

- ACGAN could allow **diverse & higher resolution images** than previous cGANs
 - The first cGAN approach that could scale up to the **ImageNet dataset**

2. Comparison of multiscale structural similarity (MS-SSIM)

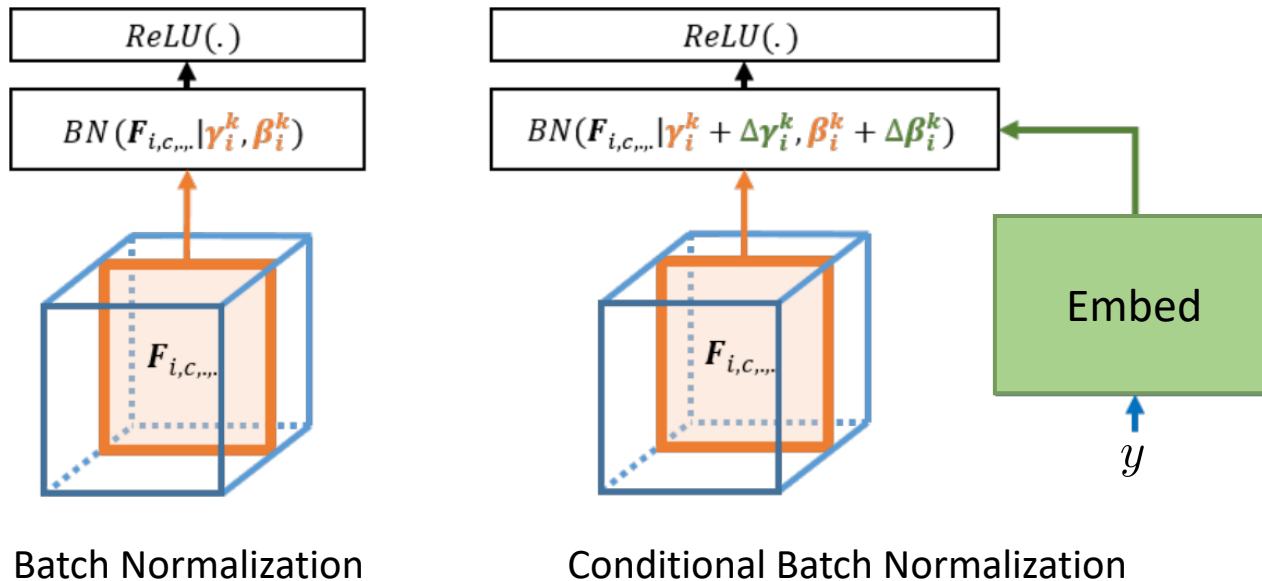
- MS-SSIM ranges 0.0 ~ 1.0; Higher MS-SSIM → perceptually more similar
- ACGAN achieves similar MS-SSIM to the training set on many of ImageNet classes



Conditional GANs: Conditional Batch Normalization (CBN)

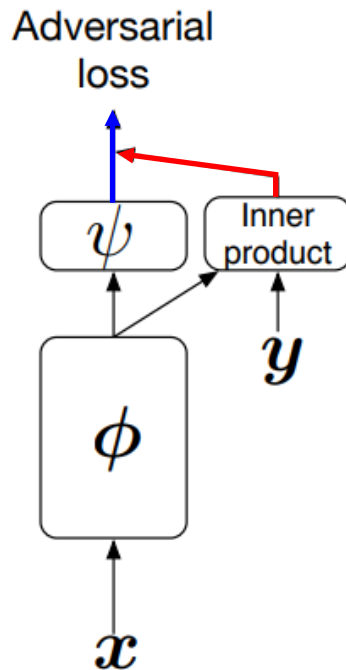
- **Conditional BN** [Dumoulin et al., 2017, DeVries et al., 2017]
 - Recent practice of designing **cGAN generator** instead of concatenating y
 - Modulate **Batch Normalization (BN) layers** depending on the condition
- **Idea:** Predict the affine scaling parameters, γ and β in BN, from y

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



Conditional GANs: Projection Discriminator

- **Projection discriminator** [Miyato et al., 2018]
 - Recent practice of designing **cGAN discriminator** instead of feeding y
- Miyato et al. (2018): **projecting** y into D-representation is very effective



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

\mathcal{A} : an activation function of design choice

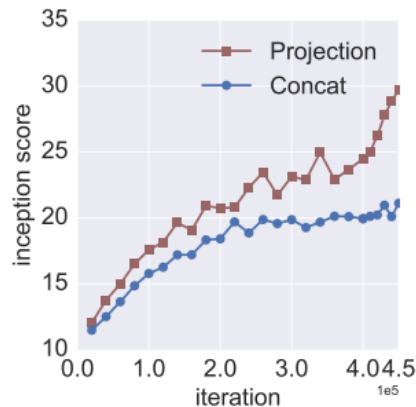
- e.g., sigmoid for vanilla GAN

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

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

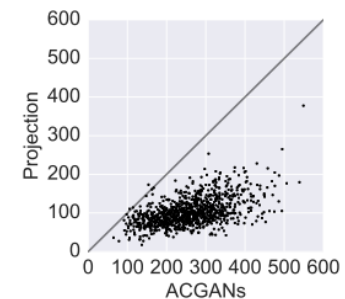
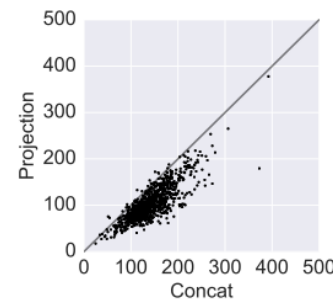
Conditional GANs: Projection Discriminator

- Projection discriminator significantly outperforms “concat” and “ACGAN”



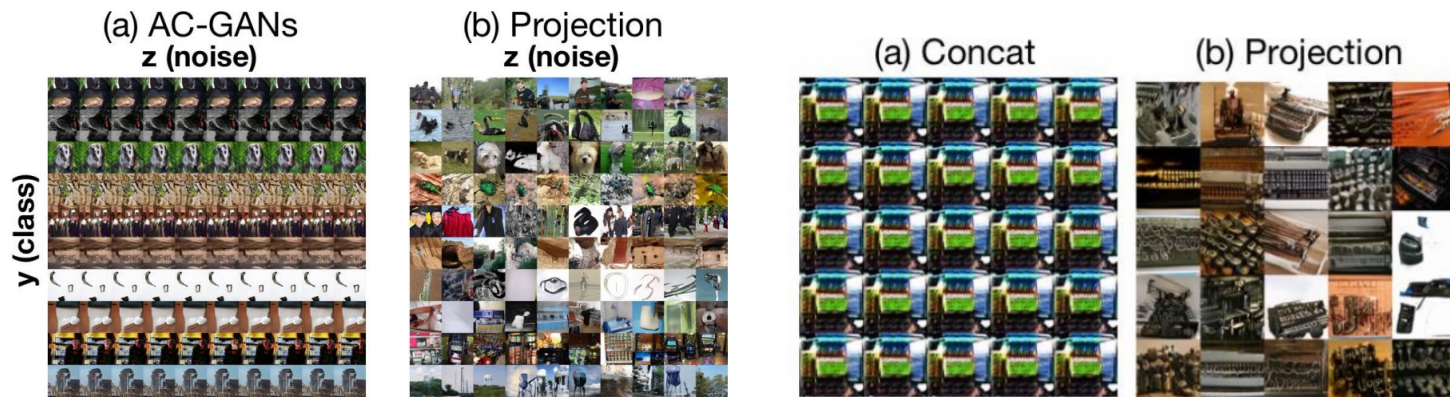
Method	Inception Score	Intra FID
AC-GANs	$28.5 \pm .20$	260.0
concat	$21.1 \pm .35$	141.2
projection	$29.7 \pm .61$	103.1
*projection (850K iteration)	$36.8 \pm .44$	92.4

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



FID for each class

- “Projection” is also more robust against mode-collapse than others



1. Basics of Generative Adversarial Networks (GAN)

- Advantages and disadvantages of GAN
- Conditional GANs

2. Advanced Topics of GANs

- Loss, regularization and normalization
- Data augmentations for GANs
- GAN architectures
- GAN training with pretrained networks

3. Beyond Image Generation: More Applications

- GANs for image manipulation
- GANs for video generation
- GANs for medical diagnosis

1. Basics of Generative Adversarial Networks (GAN)

- Advantages and disadvantages of GAN
- Conditional GANs

2. Advanced Topics of GANs

- Loss, regularization and normalization
- Data augmentations for GANs
- GAN architectures
- GAN training with pretrained networks

3. Beyond Image Generation: More Applications

- GANs for image manipulation
- GANs for video generation
- GANs for medical diagnosis

Wasserstein Distance

- Many heuristics have been proposed to alleviate training issues in GANs
 - However, it was hard to explain why they actually work in general

- 1-Wasserstein distance** (a.k.a. Earth Mover's distance):

- A distance measure between two probability distributions**

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

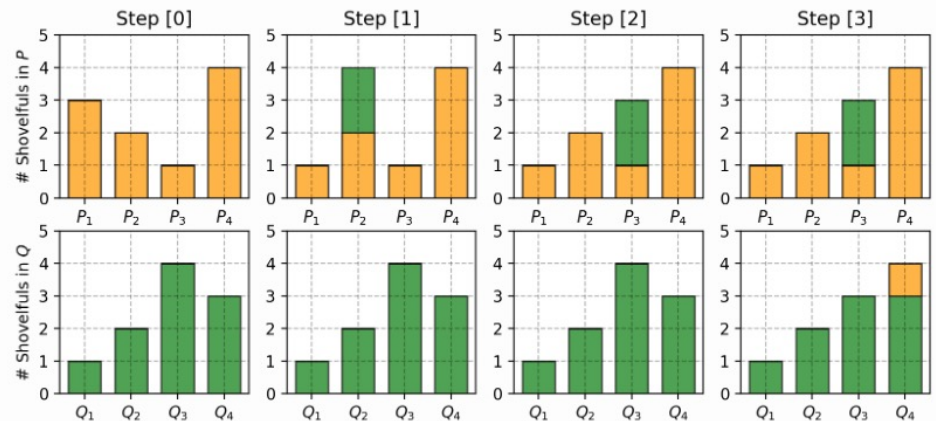
- Minimal amount of “work” to transform a distribution P to Q
- Work?** the amount of *dirt* in a chunk times the distance it was moved

- Example:**

$$P_1 = 3, P_2 = 2, P_3 = 1, P_4 = 4$$

$$Q_1 = 1, Q_2 = 2, Q_3 = 4, Q_4 = 3$$

$$W(P, Q) = 5$$



Comparison between Wasserstein Distance and Other Distance Metrics

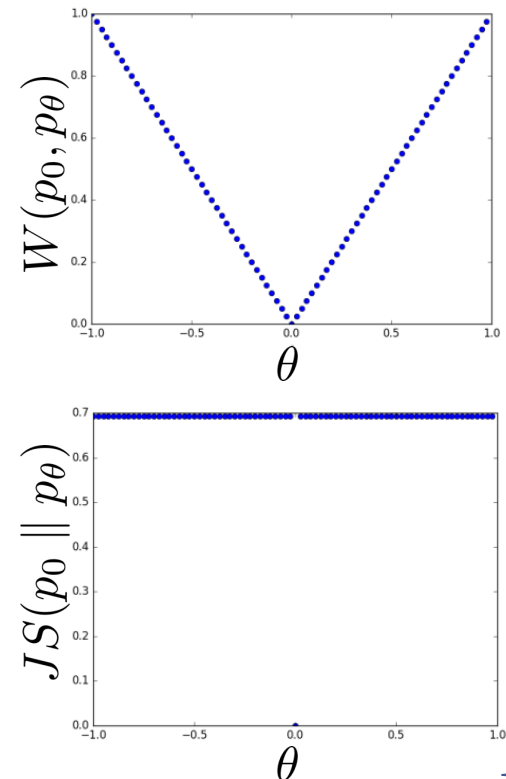
- **Why Wasserstein?** - When two distributions have no overlap
 - It still gives non-zero and smooth notion of the distance (and gradients)
- **Example** [Arjovsky, et. al., 2017]: **Wasserstein vs JS (or KL)**
 - Let $Z \sim U[0, 1]$, p_0 be the distribution of $(0, Z) \in \mathbb{R}^2$
 - $g_\theta(Z) = (\theta, Z)$ with θ , a single real parameter, and p_θ is the distribution of $g_\theta(Z)$
 - Distance between two distributions are:

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

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

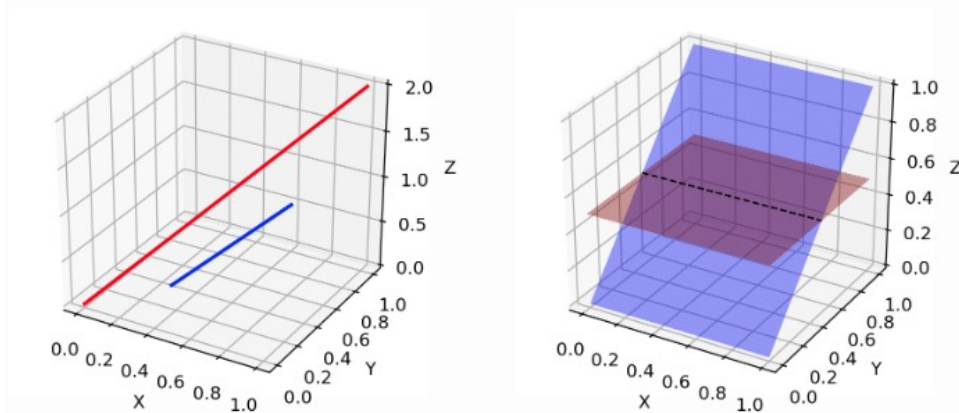
$$KL(p_0 \parallel p_\theta) = KL(p_\theta \parallel p_0) = \begin{cases} \infty & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0 \end{cases}$$

- 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

- Infimum over joint distribution $\gamma \in \Pi(p_{\text{data}}, p_g)$ is computationally **intractable**
- Using Kantorovich-Rubinstein duality [Villani, 2009]:

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

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

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

- 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 $w \in \mathcal{W}$)

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

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

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

end for

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

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

$$\begin{aligned} g_\theta &\leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)})) \\ \theta &\leftarrow \theta - \alpha \cdot \text{RMSPProp}(\theta, g_\theta) \end{aligned}$$

end for

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

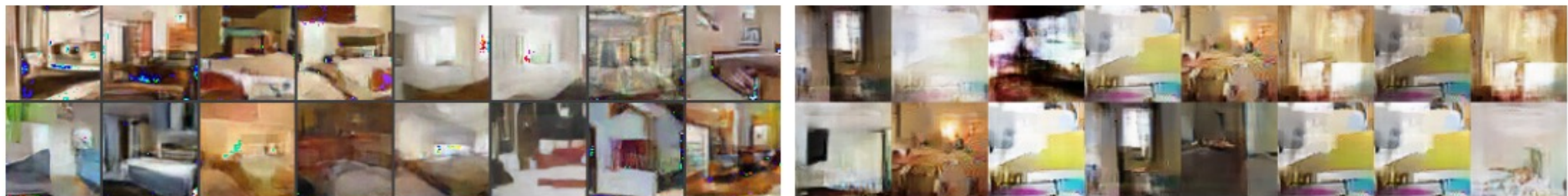
WGAN vs GAN



(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

- WGAN uses the **weight clamping** to maintain Lipschitz constraint
 - (-) Still naïve, ad-hoc and heuristic
 - (-) It often leads to significant optimization difficulties
- **Two representative methods for the direct Lipschitz constraint on D**
 1. **Gradient penalty** on Wasserstein GANs (WGAN-GP) [Gulrajani, et. al., 2017]
 - Use **gradient penalty** to maintain Lipschitz constraint

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

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

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

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

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

- Stabilizing GAN dynamics is still an open research topic

- Significant research efforts has been made to stabilize GANs
 - Different GAN losses [Arjovsky et al., 2017; Mao et al., 2017; Berthelot et al., 2017]
 - Regularization on D [Gulrajani et al., 2017; Roth et al., 2017; Kodali et al., 2017]
 - Normalization [Miyato et al., 2018]

GAN	DISCRIMINATOR LOSS	GENERATOR LOSS
MM GAN	$\mathcal{L}_D^{\text{GAN}} = -\mathbb{E}_{x \sim p_d} [\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$	$\mathcal{L}_G^{\text{GAN}} = \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$
NS GAN	$\mathcal{L}_D^{\text{NSGAN}} = -\mathbb{E}_{x \sim p_d} [\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$	$\mathcal{L}_G^{\text{NSGAN}} = -\mathbb{E}_{\hat{x} \sim p_g} [\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_D^{\text{WGAN}} = -\mathbb{E}_{x \sim p_d} [D(x)] + \mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})]$	$\mathcal{L}_G^{\text{WGAN}} = -\mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})]$
WGAN GP	$\mathcal{L}_D^{\text{WGANGP}} = \mathcal{L}_D^{\text{WGAN}} + \lambda \mathbb{E}_{\hat{x} \sim p_g} [(\nabla D(\alpha x + (1 - \alpha)\hat{x}) _2 - 1)^2]$	$\mathcal{L}_G^{\text{WGANGP}} = -\mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})]$
LS GAN	$\mathcal{L}_D^{\text{LSGAN}} = -\mathbb{E}_{x \sim p_d} [(D(x) - 1)^2] + \mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})^2]$	$\mathcal{L}_G^{\text{LSGAN}} = -\mathbb{E}_{\hat{x} \sim p_g} [(D(\hat{x}) - 1)^2]$
DRAGAN	$\mathcal{L}_D^{\text{DRAGAN}} = \mathcal{L}_D^{\text{GAN}} + \lambda \mathbb{E}_{\hat{x} \sim p_d + \mathcal{N}(0, c)} [(\nabla D(\hat{x}) _2 - 1)^2]$	$\mathcal{L}_G^{\text{DRAGAN}} = \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$
BEGAN	$\mathcal{L}_D^{\text{BEGAN}} = \mathbb{E}_{x \sim p_d} [x - \text{AE}(x) _1] - k_t \mathbb{E}_{\hat{x} \sim p_g} [\hat{x} - \text{AE}(\hat{x}) _1]$	$\mathcal{L}_G^{\text{BEGAN}} = \mathbb{E}_{\hat{x} \sim p_g} [\hat{x} - \text{AE}(\hat{x}) _1]$

- Then, which method should we actually use to train our GANs?**
 - How to choose a proper combination of hyperparameters?
 - Should one use a completely different method for different datasets?

- Significant research efforts has been made to stabilize GANs
- **Then, which method should we actually use to train our GANs?**
 - How to choose a proper combination of hyperparameters?
 - Should one use a completely different method for different datasets?
- Two large-scale studies empirically evaluate various existing GAN techniques
 - [Lucic et al., 2018] “Are GANs Created Equal? A Large-Scale Study”
 - [Kurach et al., 2019] “A Large-Scale Study on Regularization and Normalization in GANs”
- **TL;DR:** No evidence that “non-saturating loss” < most of existing methods

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

1. ... Given that there could be sufficient hyperparameter search
2. “Spectral normalization (SN)” is the only that showed consistent gain

- Kurach et al. (2019): An extensive comparison over various GAN practices
 - **Regularization/normalization**
 - Gradient penalty [Gulrajani et al., 2017] (**GP**)
 - DRAGAN [Kodali et al., 2017] (**DR**)
 - Spectral normalization [Miyato et al., 2018] (**SN**)
 - LayerNorm [Ba et al., 2016] (**LN**)
 - BatchNorm [Ioffe & Szegedy, 2015] (**BN**)
 - L2 regularization (**L2**)
 - **Loss functions**
 - Non-saturating loss [Goodfellow et al., 2014] (**NS**)
 - Least-squares loss [Mao et al., 2017] (**LS**)
 - Wasserstein loss [Arjovsky et al., 2017] (**WGAN**)
 - **Hyperparameter choices: (a) Fixed or (b) Bayesian optimization**

PARAMETER	DISCRETE VALUE
Learning rate α	{0.0002, 0.0001, 0.001}
Reg. strength λ	{1, 10}
$(\beta_1, \beta_2, n_{dis})$	{(0.5, 0.900, 5), (0.5, 0.999, 1), (0.5, 0.999, 5), (0.9, 0.999, 5)}

Table 1. Hyperparameter ranges used in this study. The Cartesian product of the fixed values suffices to uncover most of the recent results from the literature.

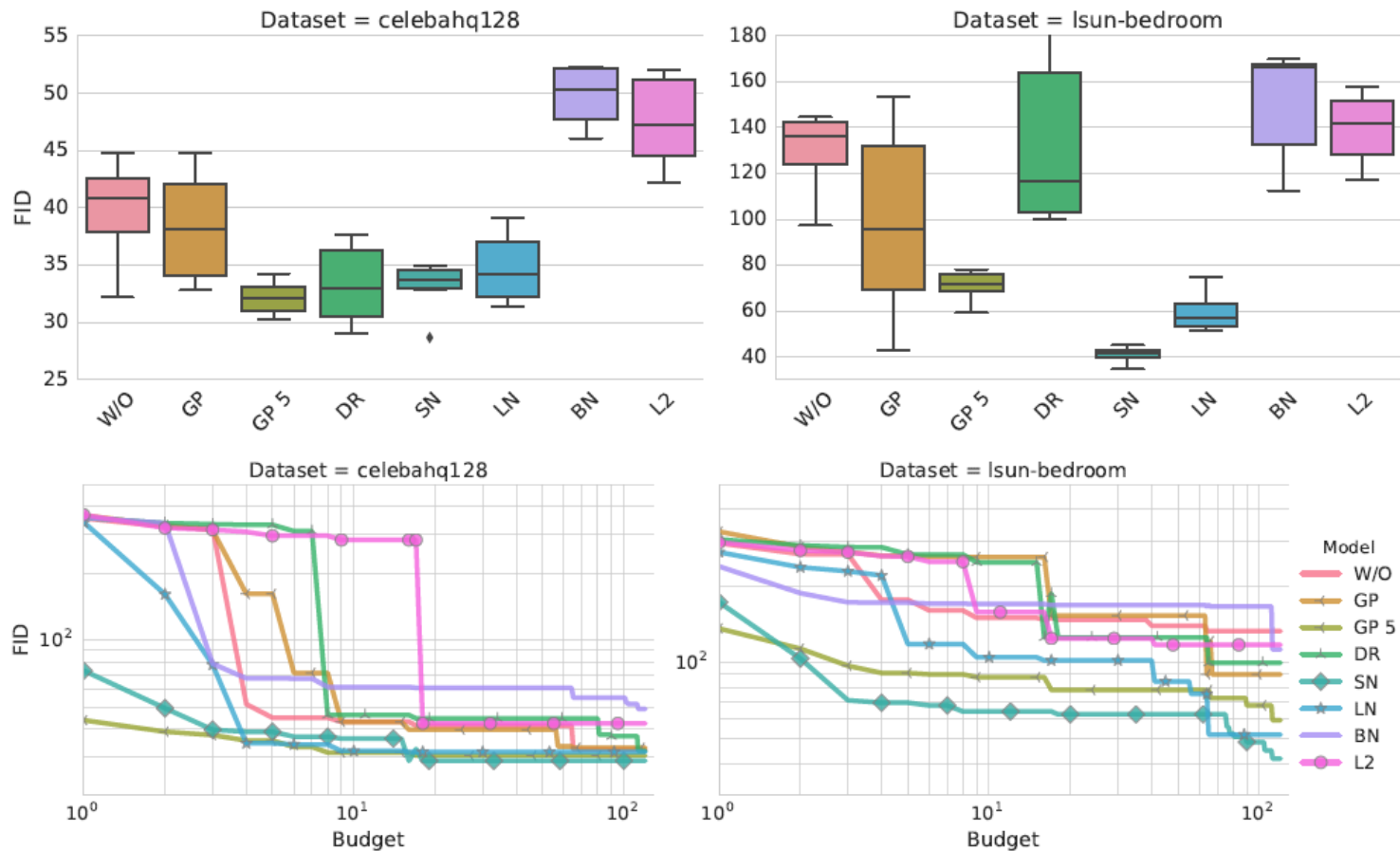
PARAMETER	RANGE	LOG
Learning rate α	$[10^{-5}, 10^{-2}]$	Yes
λ for L_2	$[10^{-4}, 10^1]$	Yes
λ for non- L_2	$[10^{-1}, 10^2]$	Yes
$\beta_1 \times \beta_2$	$[0, 1] \times [0, 1]$	No

Table 2. We use sequential Bayesian optimization (Srinivas et al., 2010) to explore the hyperparameter settings from the specified ranges. We explore 120 hyperparameter settings in 12 rounds of optimization.

- Kurach et al. (2019): An extensive comparison over various GAN practices

1. Effect of different regularization and normalization

- All the models are trained with non-saturating loss (NS)
- Compared the FID distribution for top 5% models over HPs (lower is better)

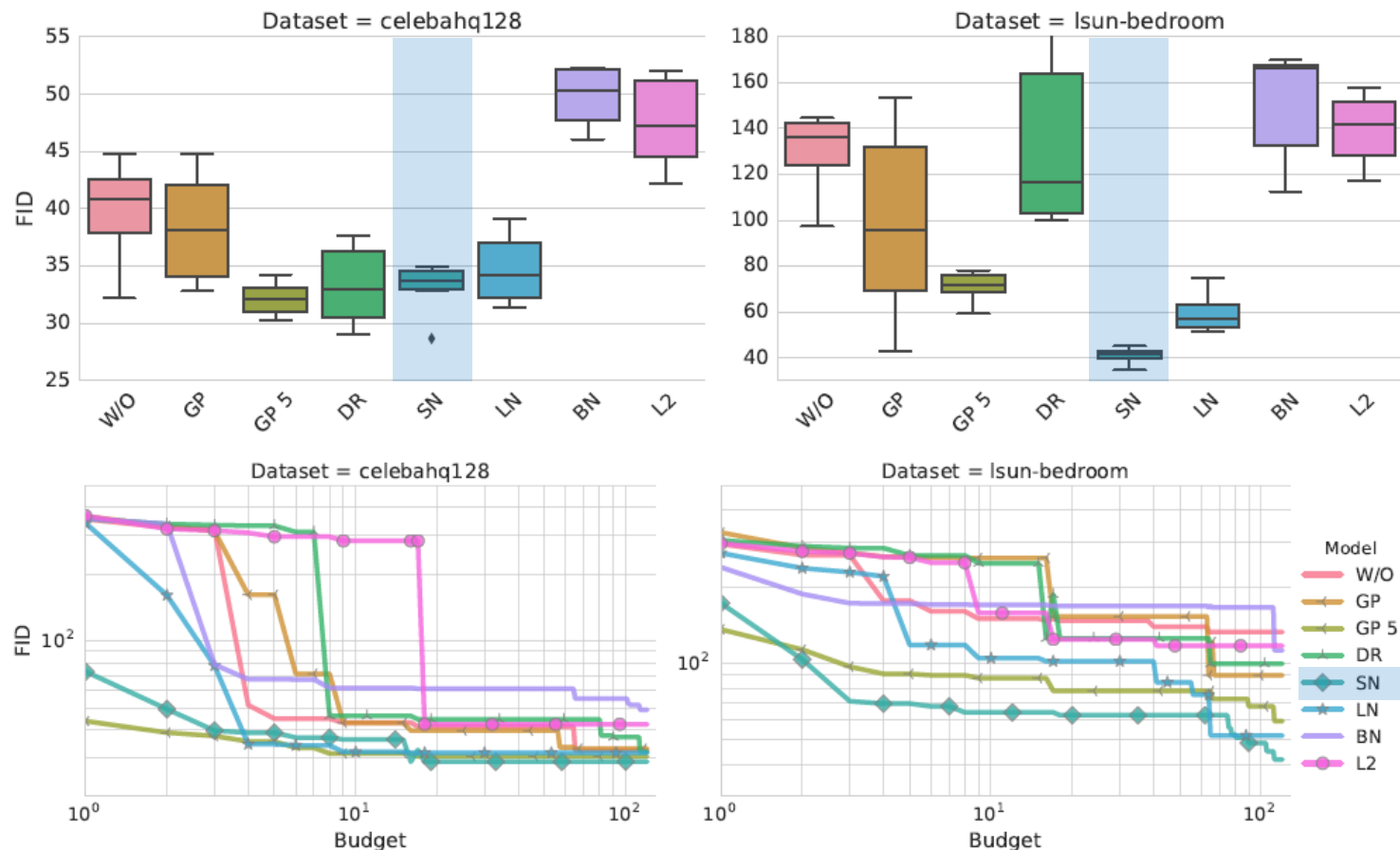


- Kurach et al. (2019): An extensive comparison over various GAN practices

1. Effect of different regularization and normalization

Remark 1. **None of them** fully address the stability issues

Remark 2. **Spectral normalization (SN)** is generally a better practical choice

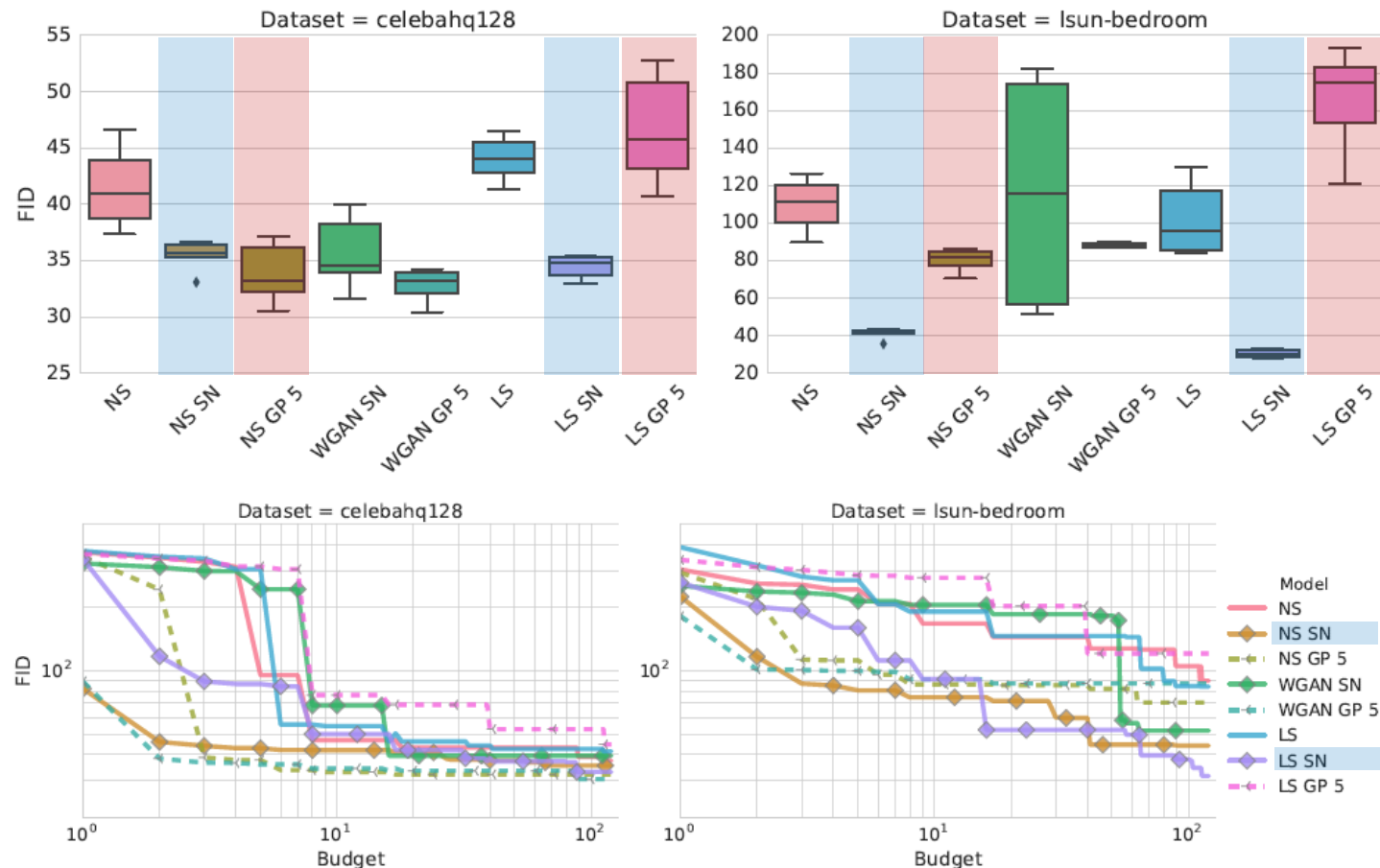


- Kurach et al. (2019): An extensive comparison over various GAN practices

2. Effect of different training loss

Remark 1. Non-saturating loss (NS) was enough to achieve good FIDs

Remark 2. SN still consistently improves FID, while GP makes some mixed conclusion



1. Basics of Generative Adversarial Networks (GAN)

- Advantages and disadvantages of GAN
- Conditional GANs

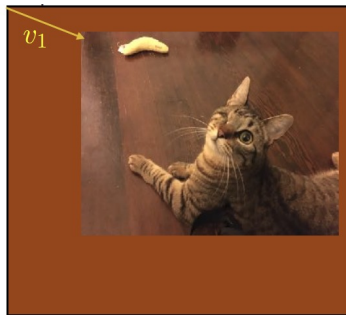
2. Advanced Topics of GANs

- Loss, regularization and normalization
- Data augmentations for GANs
- GAN architectures
- GAN training with pretrained networks

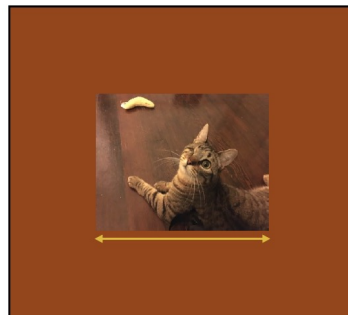
3. Beyond Image Generation: More Applications

- GANs for image manipulation
- GANs for video generation
- GANs for medical diagnosis

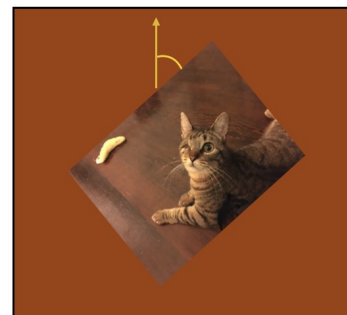
- **Collecting more data** is perhaps the best way to generalize better
- **Data augmentation (DA)** makes **artificial data** instead of collecting more
 - Requires some knowledge on **making “good” artificial data**
- Have been especially effective for **discriminative modeling**
- **Example:** Rigid transformation symmetries
 - Translation, dilation, rotation, mirror symmetry, ...
 - Forms an affine group on pixels:
$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \mapsto \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$



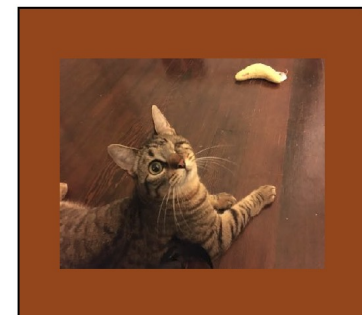
Translation



Dilation



Rotation



Mirror symmetry

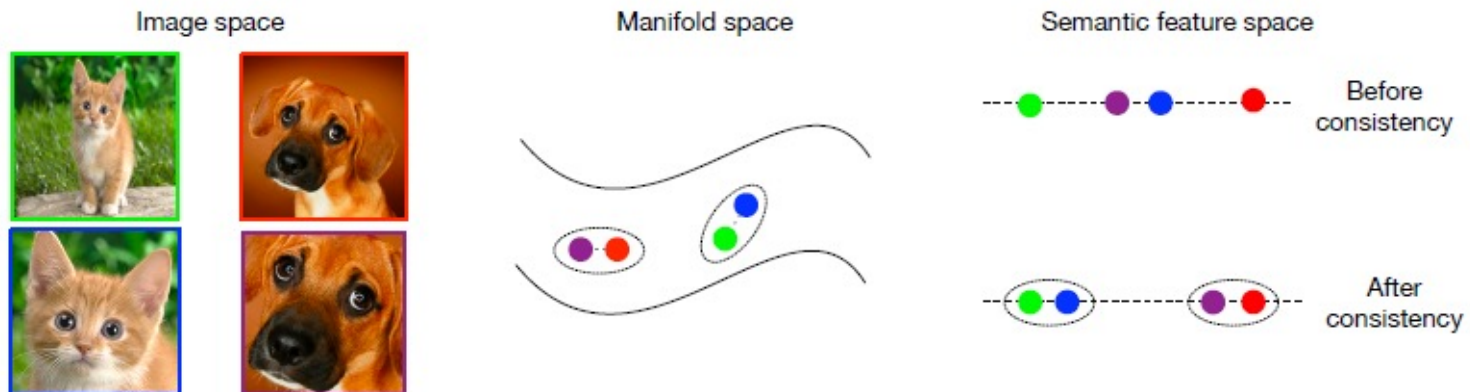
- **Collecting more data** is perhaps the best way to generalize better
- **Data augmentation (DA)** makes **artificial data** instead of collecting more
 - Requires some knowledge on **making “good” artificial data**
- Have been especially effective for **discriminative modeling**
- **DA for GANs?** (or for generative modeling in general?)
 - Not much explored until very recently [Zhang et al., 2019]
 - **Why?** Current DA practices for discriminative modeling might be **too strong**
 - How can we incorporate the distribution shifts $P(x) \rightarrow P(T(x))$?



- How can we incorporate the distribution shifts $P(x) \rightarrow P(T(x))$?
 - Naïve augmentation of real images would shift the data distribution



- Zhang et al. (2019): [Consistency regularization](#) for GANs (CRGAN)
 - Enforcing only “consistency”** can effectively incorporate $T(x)$



- **Enforcing consistency** can effectively incorporate $P(x) \rightarrow P(T(x))$
 - Training data is not directly augmented by T , but **only consider** $D(x) \approx D(T(x))$
 - D should learn representation that is **invariant** to T

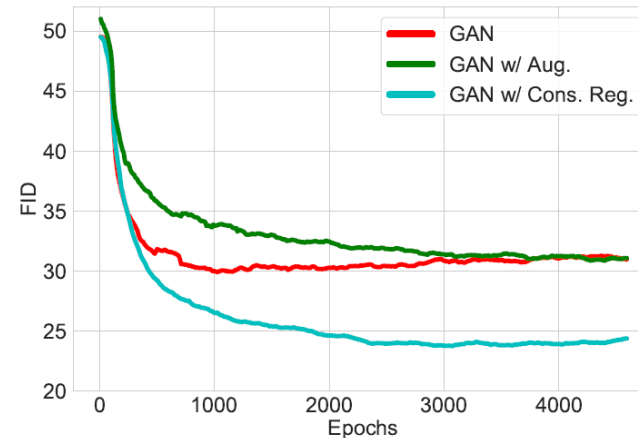
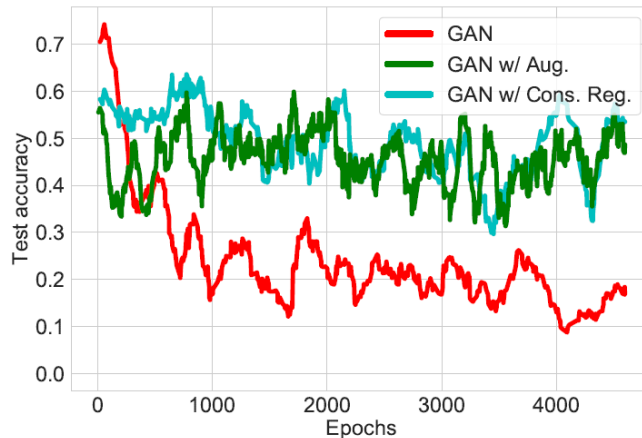
$$L_{cr} = \|D(x) - D(T(x))\|^2,$$
$$L_D^{cr} = L_D + \lambda L_{cr}, \quad L_G^{cr} = L_G.$$

Algorithm 1 Consistency Regularized GAN (CR-GAN). We use $\lambda = 10$ by default.

Input: generator and discriminator parameters θ_G, θ_D , consistency regularization coefficient λ , Adam hyperparameters α, β_1, β_2 , batch size M , number of discriminator iterations per generator iteration N_D

```
1: for number of training iterations do
2:   for  $t = 1, \dots, N_D$  do
3:     for  $i = 1, \dots, M$  do
4:       Sample  $z \sim p(z), x \sim p_{\text{data}}(x)$ 
5:       Augment  $x$  to get  $T(x)$ 
6:        $L_{cr}^{(i)} \leftarrow \|D(x) - D(T(x))\|^2$  ← Only real images are augmented
7:        $L_D^{(i)} \leftarrow D(G(z)) - D(x)$ 
8:     end for
9:      $\theta_D \leftarrow \text{Adam}(\frac{1}{M} \sum_{i=1}^M (L_D^{(i)} + \lambda L_{cr}^{(i)}), \alpha, \beta_1, \beta_2)$ 
10:  end for
11:  Sample a batch of latent variables  $\{z^{(i)}\}_{i=1}^M \sim p(z)$ 
12:   $\theta_G \leftarrow \text{Adam}(\frac{1}{M} \sum_{i=1}^M (-D(G(z^{(i)}))), \alpha, \beta_1, \beta_2)$  ← G is trained in the standard way
13: end for
```

- Does CR really learn differently than simple augmentation?

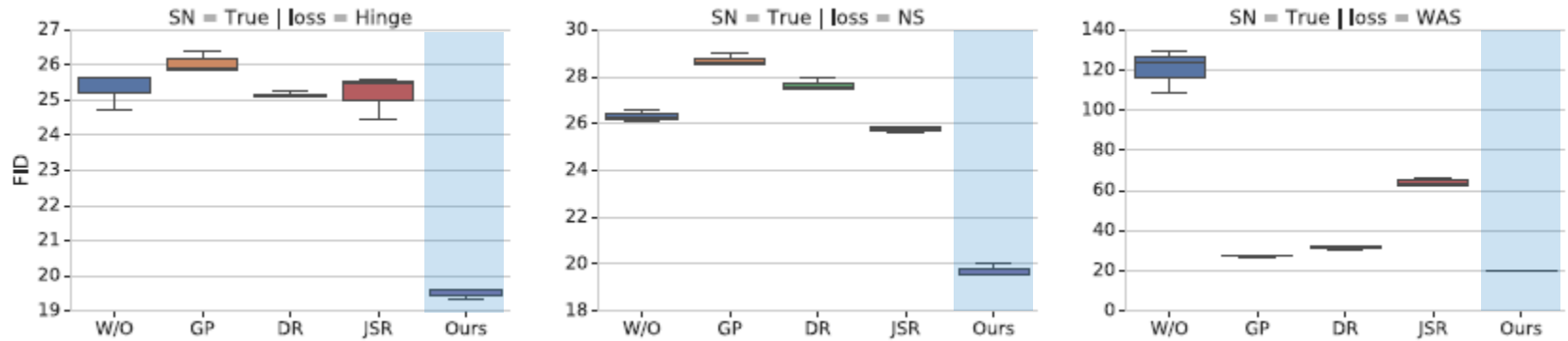


- Both CR and Aug. prevent overfitting of the discriminator
 - However, CR is the one that could only meaningfully improve FIDs
- Which augmentations should we use?**
 - The choice of augmentation does matter in GAN training
 - For CR, a simple choice of “Random shift & flip” worked best

Metric	Gaussian Noise	Random shift & flip	Cutout	Cutout w/ random shift & flip
FID	21.91±0.32	16.04±0.17	17.10±0.29	19.46±0.26

Table 3: FID scores on CIFAR-10 for different types of image augmentation. Gaussian noise is the worst, and random shift and flip is the best, consistent with general consensus on the best way to perform image optimization on CIFAR-10 (Zagoruyko & Komodakis, 2016).

- CR surprisingly stabilizes GAN training on various existing practices



- CR further improves state-of-the-art BigGAN training

Dataset	SNGAN	SAGAN	BigGAN	BigGAN*	CR-BigGAN*
CIFAR-10	17.5	/	14.73	20.42	11.48
ImageNet	27.62	18.65	8.73	7.75	6.66

Comparison of FIDs (lower is better)

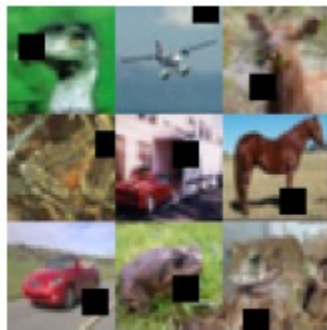


BigGAN



CR-BigGAN

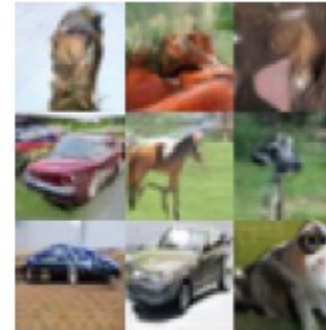
- **Recall:** How can we incorporate the distribution shifts $P(x) \rightarrow P(T(x))$?
- **Then would it be just enough with CR for GANs?**
 - Still, CR does not perfectly prevent the shifting issue in GAN
 - For certain augmentations, e.g., CutOut, CR often make “leakages”



(a) 8×8 cutout.



(b) CR samples.



(c) bCR samples.

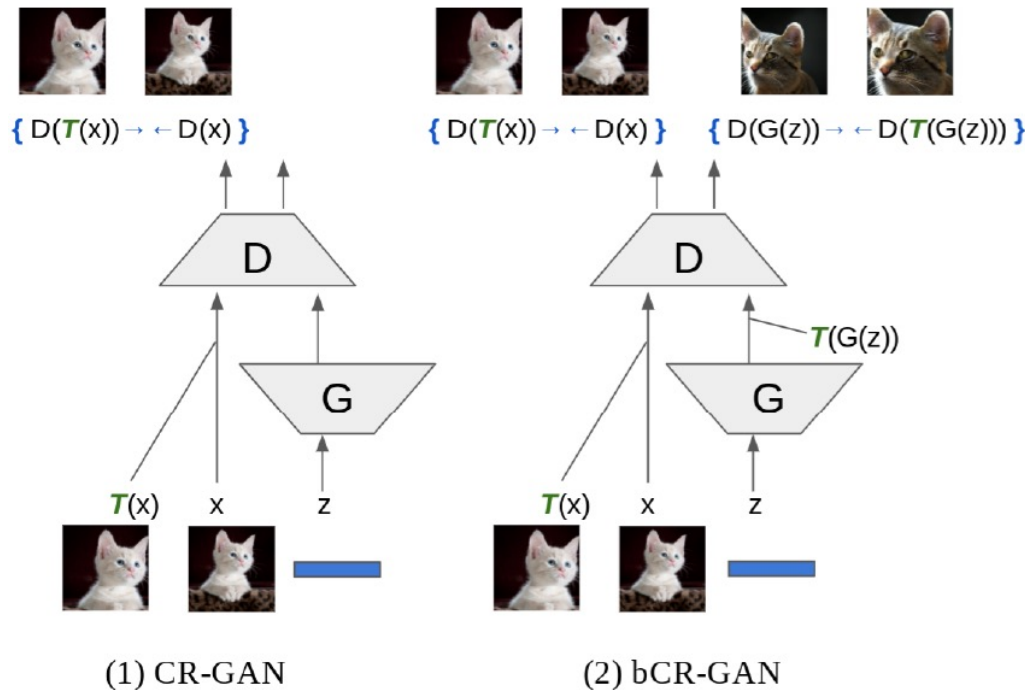
- Zhao et al. (2020): **Balanced** Consistency regularization (bCR)
 - bCR alleviates such leakages by **also giving consistency to “fake” images**

$$L_{\text{real}} \leftarrow \|D(x) - D(T(x))\|^2$$
$$L_{\text{fake}} \leftarrow \|D(G(z)) - D(T(G(z)))\|^2$$

- Zhao et al. (2020): **Balanced** Consistency regularization (bCR)
 - bCR alleviates such leakages by **also giving consistency to “fake” images**

$$L_{\text{real}} \leftarrow \|D(x) - D(T(x))\|^2$$

$$L_{\text{fake}} \leftarrow \|D(G(z)) - D(T(G(z)))\|^2$$



Algorithm 1 Balanced Consistency Regularization (bCR)

Input: parameters of generator θ_G and discriminator θ_D , consistency regularization coefficient for real images λ_{real} and fake images λ_{fake} , number of discriminator iterations per generator iteration N_D , augmentation transform T (for images, e.g. shift, flip, cutout, etc).

for number of training iterations **do**

for $t = 1$ **to** N_D **do**

 Sample batch $z \sim p(z)$, $x \sim p_{\text{real}}(x)$

 Augment both real $T(x)$ and fake $T(G(z))$ images

$L_D \leftarrow D(G(z)) - D(x)$

$L_{\text{real}} \leftarrow \|D(x) - D(T(x))\|^2$

$L_{\text{fake}} \leftarrow \|D(G(z)) - D(T(G(z)))\|^2$

$\theta_D \leftarrow \text{AdamOptimizer}(L_D + \lambda_{\text{real}}L_{\text{real}} + \lambda_{\text{fake}}L_{\text{fake}})$

end for

 Sample batch $z \sim p(z)$

$L_G \leftarrow -D(G(z))$

$\theta_G \leftarrow \text{AdamOptimizer}(L_G)$

end for

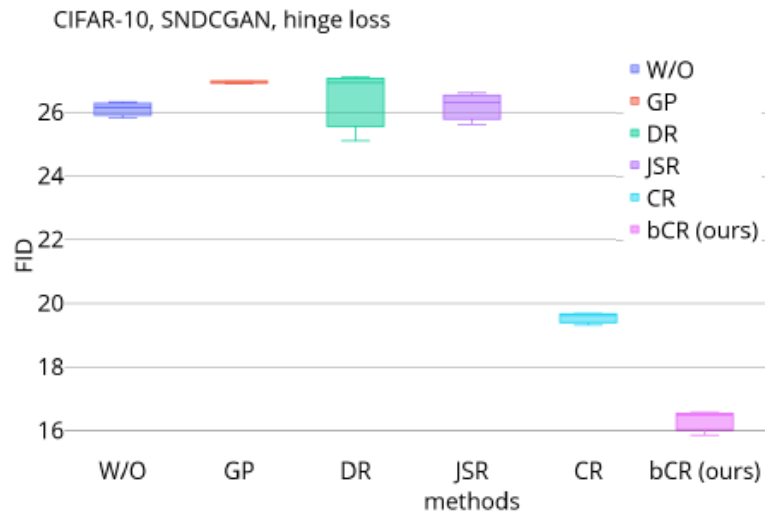
G is still trained in the standard way

“Improved” Consistency Regularization for GANs [Zhao et al., 2020a]

- Zhao et al. (2020): **Balanced** Consistency regularization (bCR)
- Despite its simplicity, bCR could achieve state-of-the-art BigGAN training

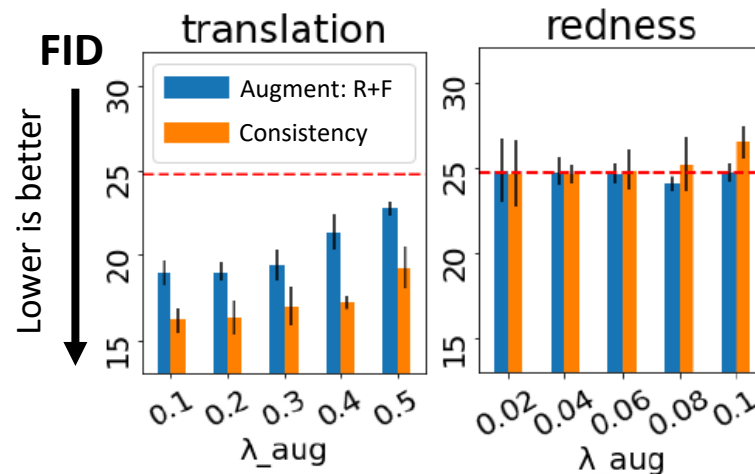
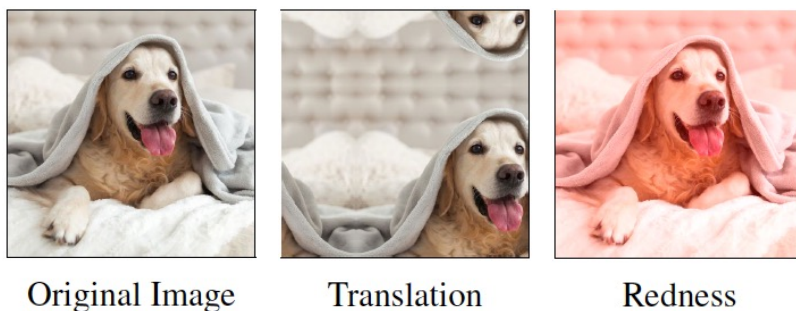
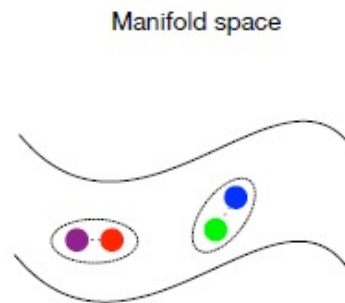


(a) Monarch Butterfly (our ICR vs baseline CR)

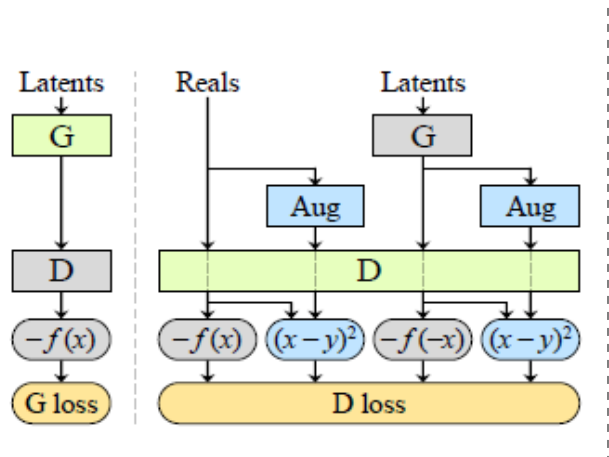


Models	CIFAR-10	ImageNet
SNGAN	17.50	27.62
BigGAN	14.73	8.73
CR-BigGAN	11.48	6.66
ICR-BigGAN (ours)	9.21	5.38

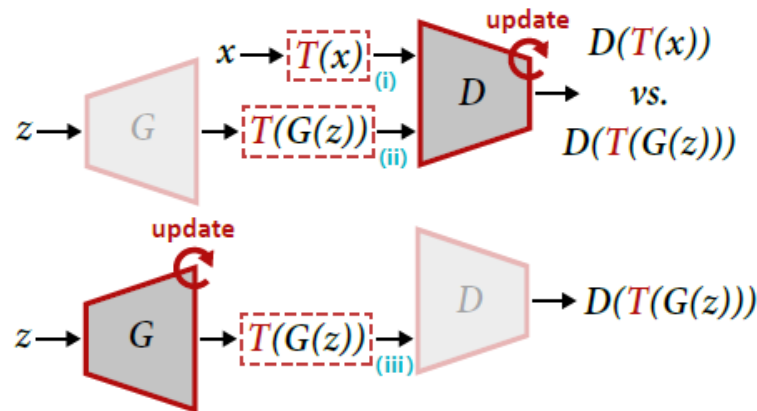
- Is CR really necessary for GANs to incorporate data augmentations?
- **Limitation of CR:** Fundamentally hard to incorporate **stronger augmentations**
- **Example:** Color jittering
 - The “redness” is not helpful to improve FID with CR
 - Forcing CR for such a stronger augmentation might be **too restrictive for D representation**
- How can we incorporate stronger augmentations for GANs?



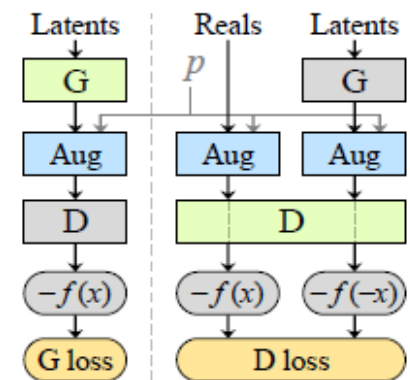
- Is CR really necessary for GANs to incorporate data augmentations?
- **How can we incorporate stronger augmentations for GANs?**
- Two concurrent works propose a “even simpler” scheme for GANs
 - [Zhao et al., 2020b] “Differentiable Augmentation for Data-Efficient GAN Training”
 - [Karras et al., 2020b] “Training Generative Adversarial Networks with Limited Data”



Balanced CR (bCR)

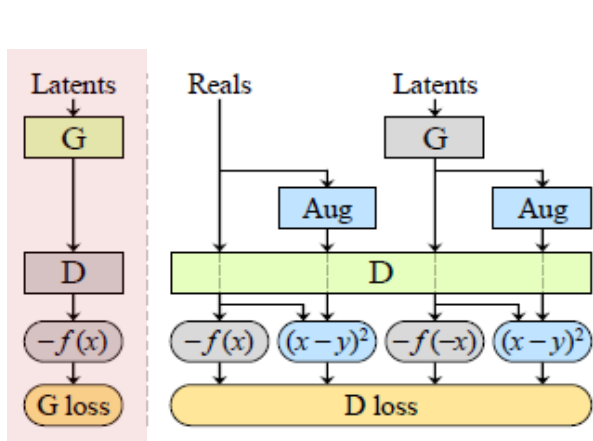


DiffAug

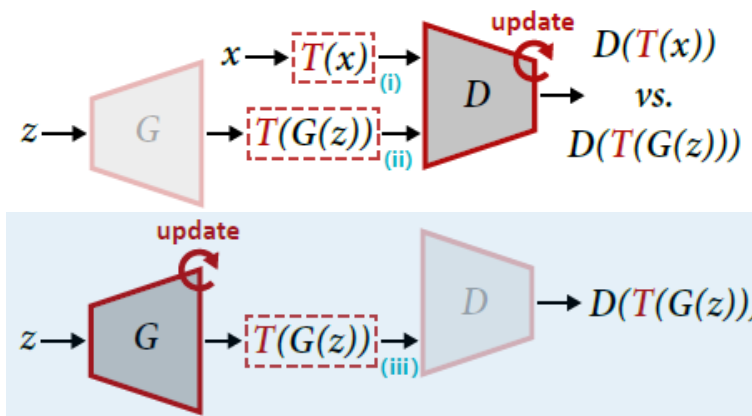


ADA

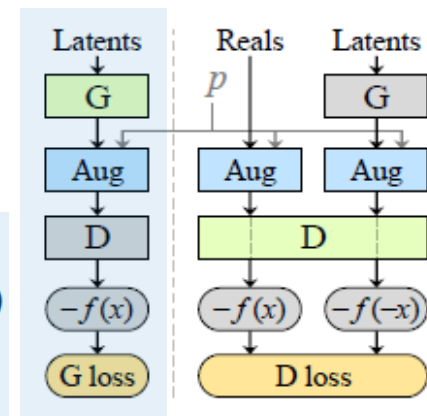
- Is CR really necessary for GANs to incorporate data augmentations?
- **How can we incorporate stronger augmentations for GANs?**
- Two concurrent works propose a “even simpler” scheme for GANs
 - [Zhao et al., 2020b] “Differentiable Augmentation for Data-Efficient GAN Training”
 - [Karras et al., 2020b] “Training Generative Adversarial Networks with Limited Data”
- **Idea:** Simply augment every input before D, **even when G is trained**
 - No CR needed anymore, and accept stronger augmentations without leakages



Balanced CR (bCR)

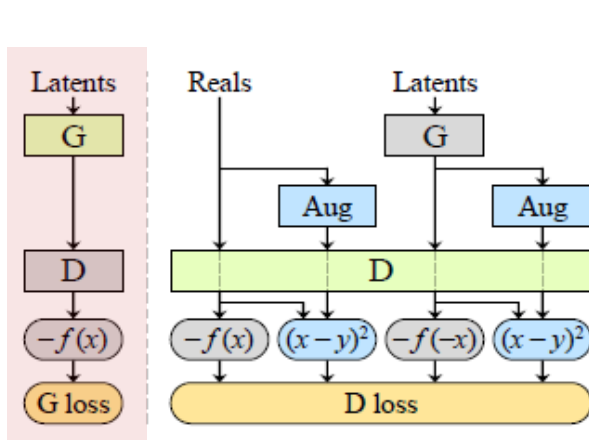


DiffAug

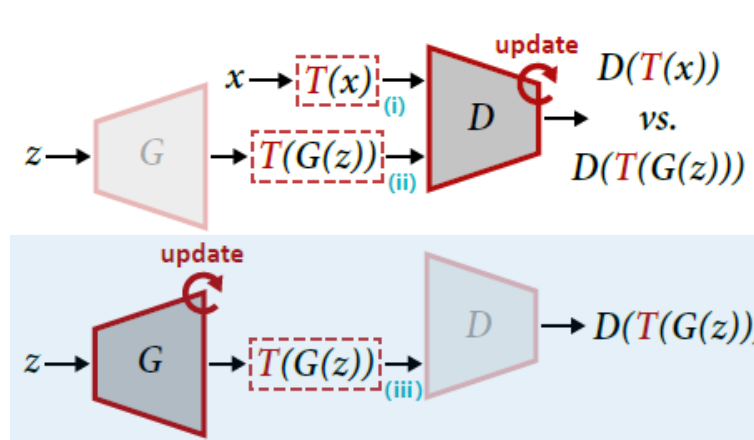


ADA

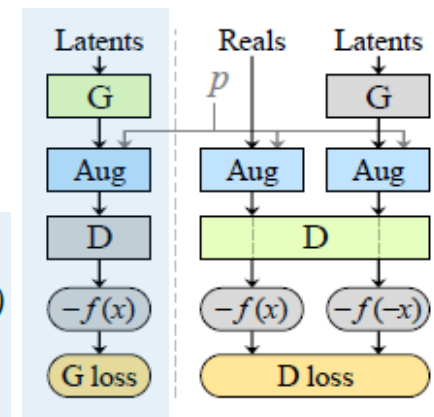
- Two concurrent works propose a “even simpler” scheme for GANs
- Idea:** Simply augment every input before D, **even when G is trained**
- Then, how could this approach have not been explored so far?
 - This requires a **differentiable implementation** of $T(\cdot)$ for training G
 - Example:** Non-saturating loss should minimize $\mathbb{E}_z[-\log(D(T(G(z))))]$
 - Nevertheless, most of the previous implementations of T were **non-differentiable**
 - ... as they were rather considered as pre-processing steps
 - In this respect, the “**differentiability**” of T is becoming increasingly important



Balanced CR (bCR)

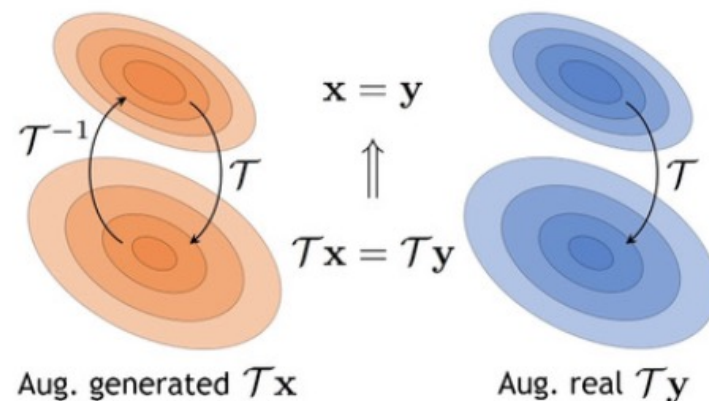


DiffAug

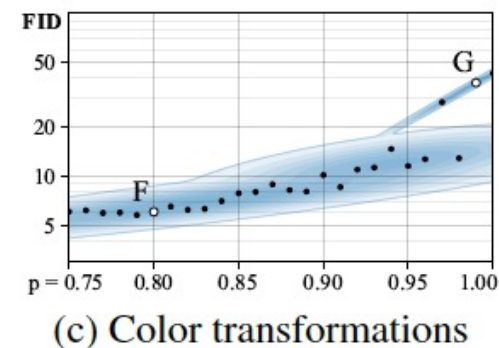
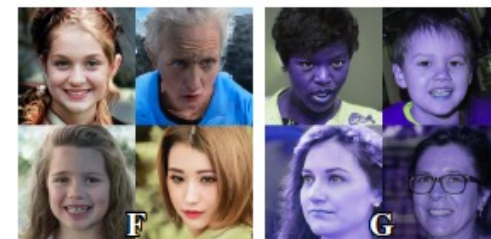
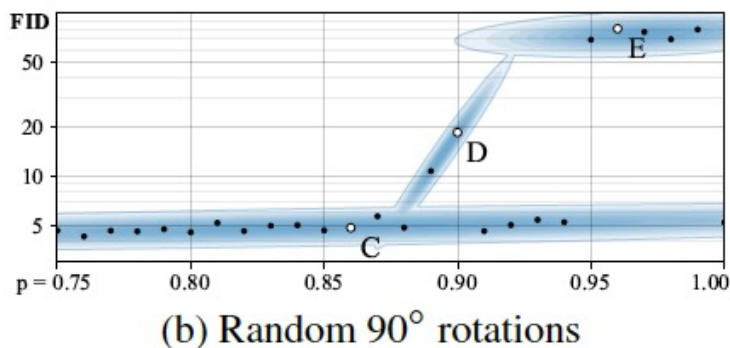
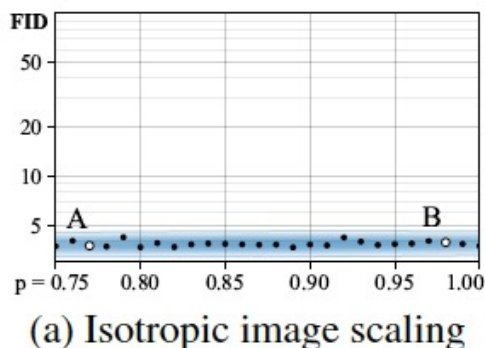
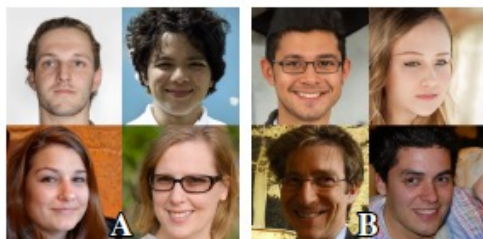


ADA

- **Adaptive Discriminator Augmentation (ADA)** [Karras et al., 2020b]
- Which augmentation should we use?
 - **Key point:** There should be **no leakage** of augmentations
- **Example:** Random 90° rotations as \mathcal{T}
 - Assume \mathbf{x} : **generated** distribution and \mathbf{y} : **target** distribution
 - **Q:** ADA matches $\mathcal{T}\mathbf{x} = \mathcal{T}\mathbf{y}$: then, does it always imply $\mathbf{x} = \mathbf{y}$?
 - **A:** No, imagine when \mathbf{x} goes like “E” below → **augmentation leakage**



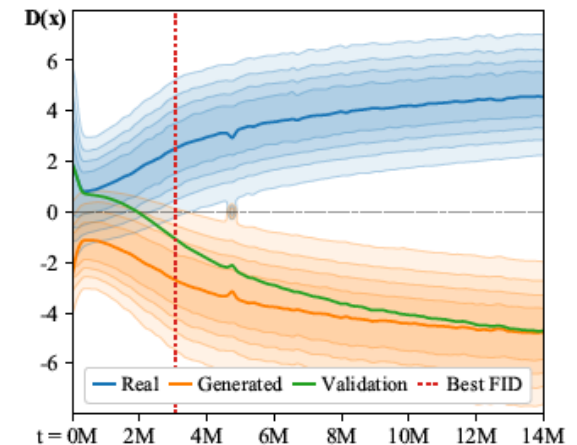
- **Adaptive Discriminator Augmentation (ADA)** [Karras et al., 2020b]
- Which augmentation should we use?
 - **Key point:** There should be **no leakage** of augmentations
- **Example:** Random 90° rotations as \mathcal{T}
- **Idea:** The leakage of any \mathcal{T} can be controlled by setting $p \in [0, 1]$ The prob. of executing \mathcal{T}



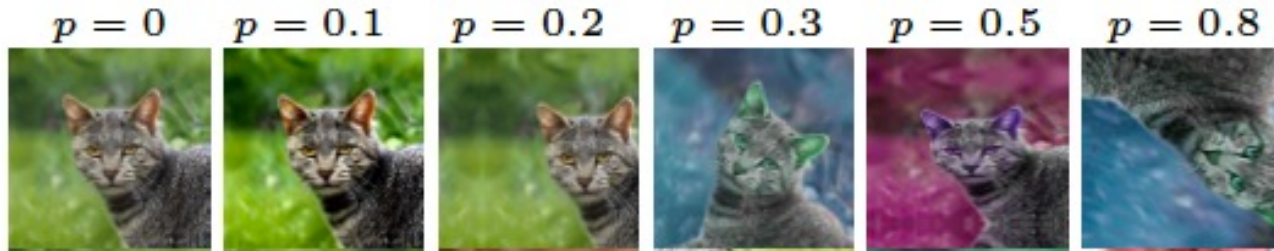
- **Adaptive Discriminator Augmentation (ADA)** [Karras et al., 2020b]
- Which augmentation should we use?
 - **Key point:** There should be **no leakage** of augmentations
- **Idea:** The leakage of any \mathcal{T} can be controlled by setting $p \in [0, 1]$ The prob. of executing \mathcal{T}
- ADA also proposes a heuristic to **adaptively set p** in training by observing r_v

$$r_v = \frac{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{validation}}]}{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{generated}}]}$$

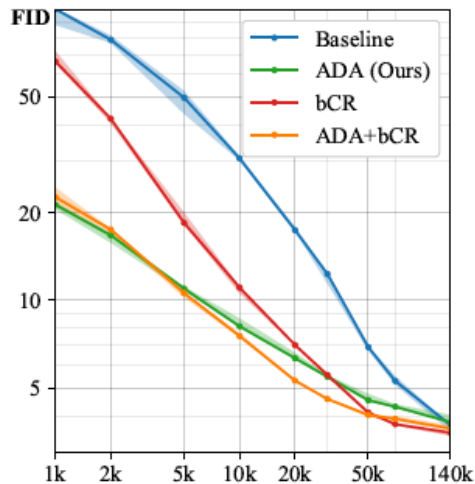
- $r_v = 0$: No overfitting / $r_v = 1$: Complete overfitting
- p of the augmentation is initially set to 0
- Increase/decrease p when r_v is low/high, resp.



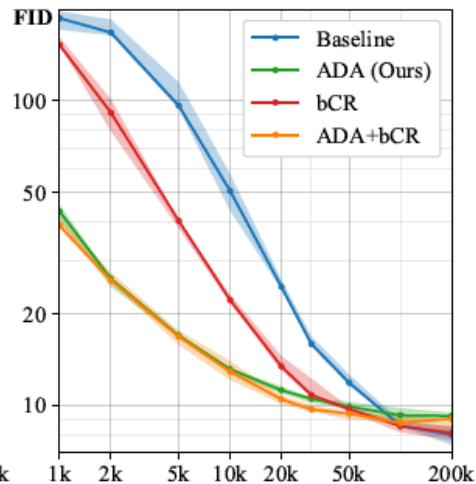
- **Adaptive Discriminator Augmentation (ADA)** [Karras et al., 2020b]
- ADA successfully incorporate wider augmentations than bCR



- ADA works significantly better than bCR when # sample is small



(a) FFHQ (256×256)



(b) LSUN CAT (256×256)

Dataset		Baseline	ADA	+ bCR
FFHQ	1k	100.16	21.29	22.61
	5k	49.68	10.96	10.58
	10k	30.74	8.13	7.53
	30k	12.31	5.46	4.57
	70k	5.28	4.30	3.91
	140k	3.71	3.81	3.62
LSUN CAT	1k	186.91	43.25	38.82
	5k	96.44	16.95	16.80
	10k	50.66	13.13	12.90
	30k	15.90	10.50	9.68
	100k	8.56	9.26	8.73
	200k	7.98	9.22	9.03

(c) Median FID

- **Adaptive Discriminator Augmentation (ADA)** [Karras et al., 2020b]
- ADA significantly improves GAN training especially on limited-sized datasets

Dataset	Method	Scratch		Transfer KID $\times 10^3$	+ Freeze-D KID $\times 10^3$
		FID	KID $\times 10^3$		
METFACES	Baseline	57.26	35.66	3.16	2.05
	ADA	18.22	2.41	0.81	1.33
BRECAHAD	Baseline	97.72	89.76	18.07	6.94
	ADA	15.71	2.88	3.36	1.91
AFHQ CAT	Baseline	5.13	1.54	1.09	1.00
	ADA	3.55	0.66	0.44	0.35
AFHQ DOG	Baseline	19.37	9.62	4.63	2.80
	ADA	7.40	1.16	1.40	1.12
AFHQ WILD	Baseline	3.48	0.77	0.31	0.12
	ADA	3.05	0.45	0.15	0.14

(a) Small datasets

Method	Unconditional		Conditional	
	FID ↓	IS ↑	FID ↓	IS ↑
ProGAN [19]	15.52	8.56±0.06	—	—
AutoGAN [13]	12.42	8.55±0.10	—	—
BigGAN [5]	—	—	14.73	9.22
+ Tuning [22]	—	—	8.47	9.07±0.13
MultiHinge [22]	—	—	6.40	9.58±0.09
FQ-GAN [52]	—	—	5.59±0.12	8.48
Baseline	8.32±0.09	9.21±0.09	6.96±0.41	9.53±0.06
+ ADA (Ours)	5.33±0.35	10.02±0.07	3.49±0.17	10.24±0.07
+ Tuning (Ours)	2.92±0.05	9.83±0.04	2.42±0.04	10.14±0.09

(b) CIFAR-10



1. Basics of Generative Adversarial Networks (GAN)

- Advantages and disadvantages of GAN
- Conditional GANs

2. Advanced Topics of GANs

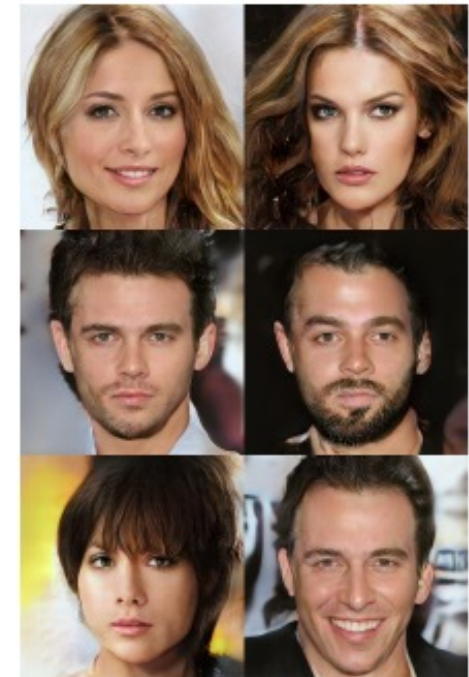
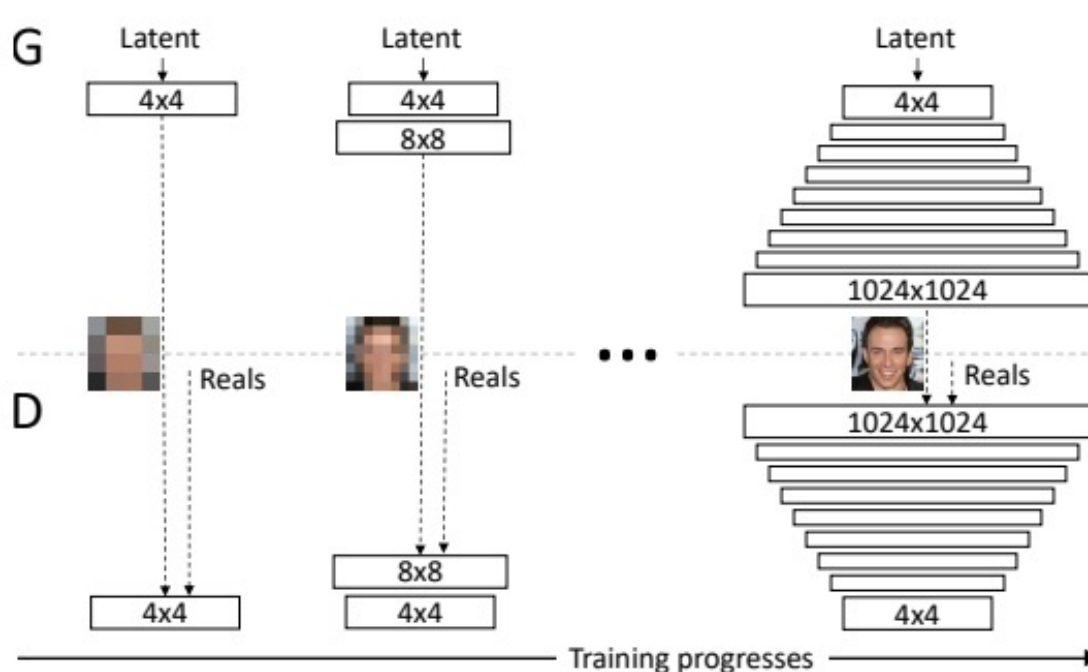
- Loss, regularization and normalization
- Data augmentations for GANs
- GAN architectures
- GAN training with pretrained networks

3. Beyond Image Generation: More Applications

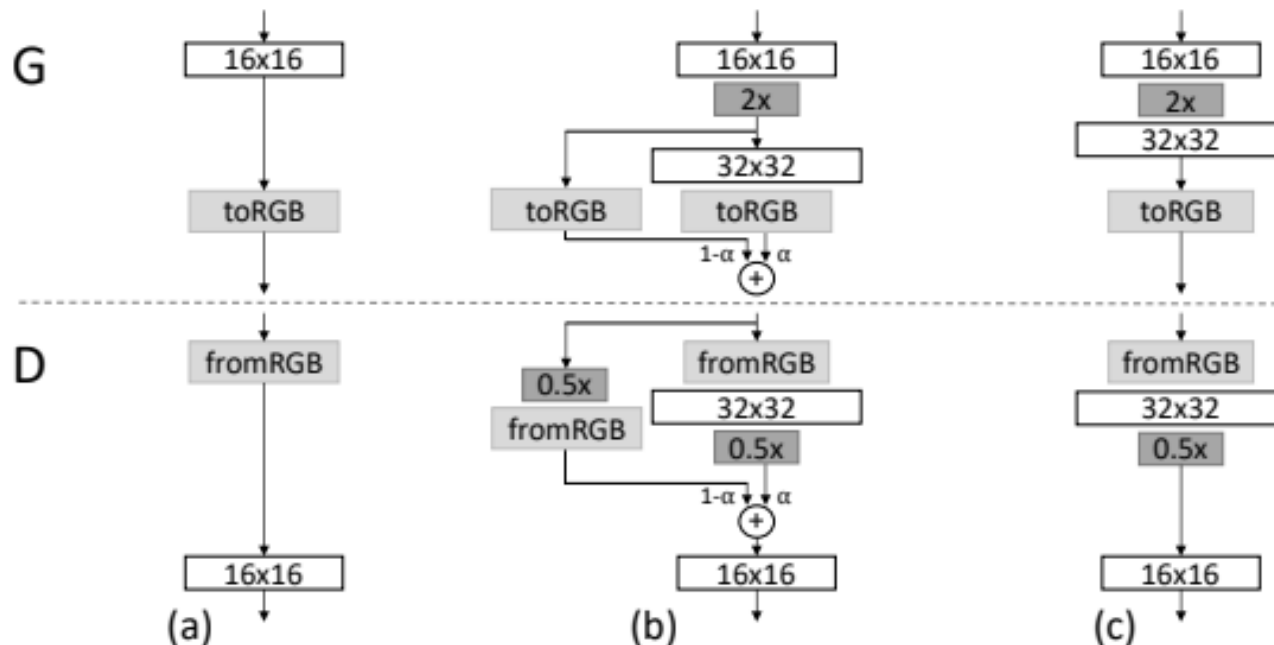
- GANs for image manipulation
- GANs for video generation
- GANs for medical diagnosis

Progressive GAN: High-Resolution Image Generation [Karras et al., 2018]

- Previous GANs could produce sharp images, but only at **small resolutions**
 - It was still unstable on higher-resolution training despite some progress
- Karras et al. (2018): **Progressive growing of G and D** (Progressive-GAN)
 - Training GANs to directly generate high-res image might be too difficult!
 - Progressive-GAN starts from **learning low-resolution images**
 - It **adds new layers to G and D** during training for up-scaling into higher-resolution



- **Smooth fade-in** to the new layers during up-scale training
 - To prevent “**sudden shocks**” to the pre-trained smaller-resolution layers
- **Example:** Upscaling transition **(b)** from 16×16 to 32×32 **((a) \rightarrow (c))**



- Simply treat the higher resolution like a residual block
- The fade-in weight α increases linearly from 0 to 1 during training

Progressive GAN: High-Resolution Image Generation [Karras et al., 2018]



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)

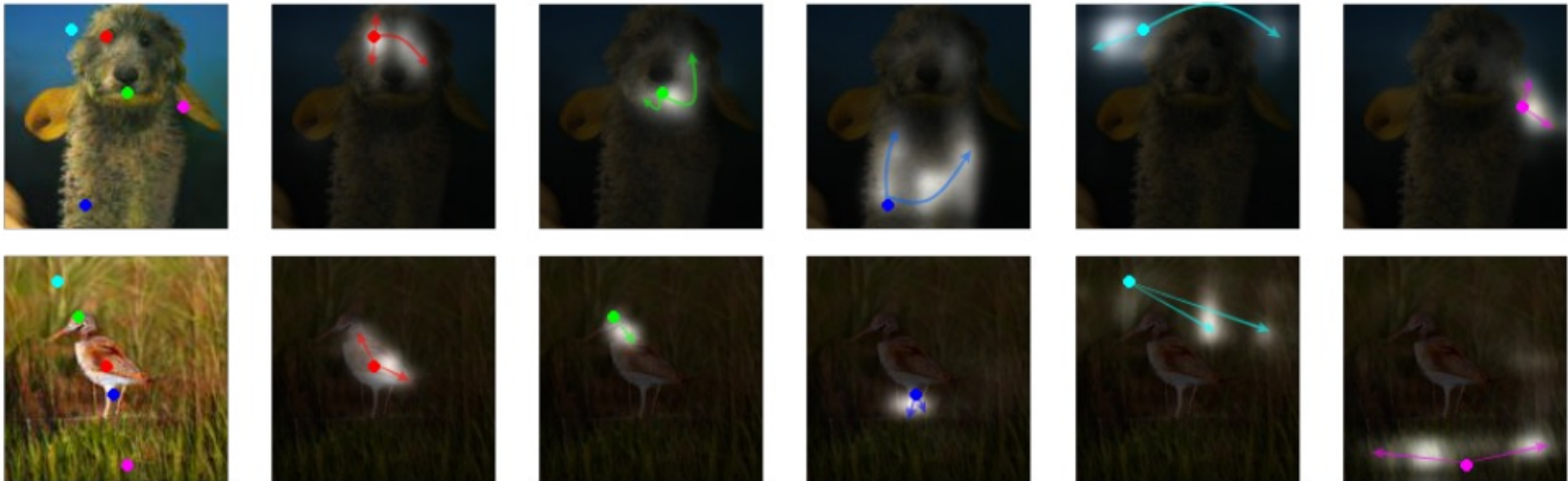
Visual quality comparison: LSUN bedroom



LSUN other categories generated image (256x256)

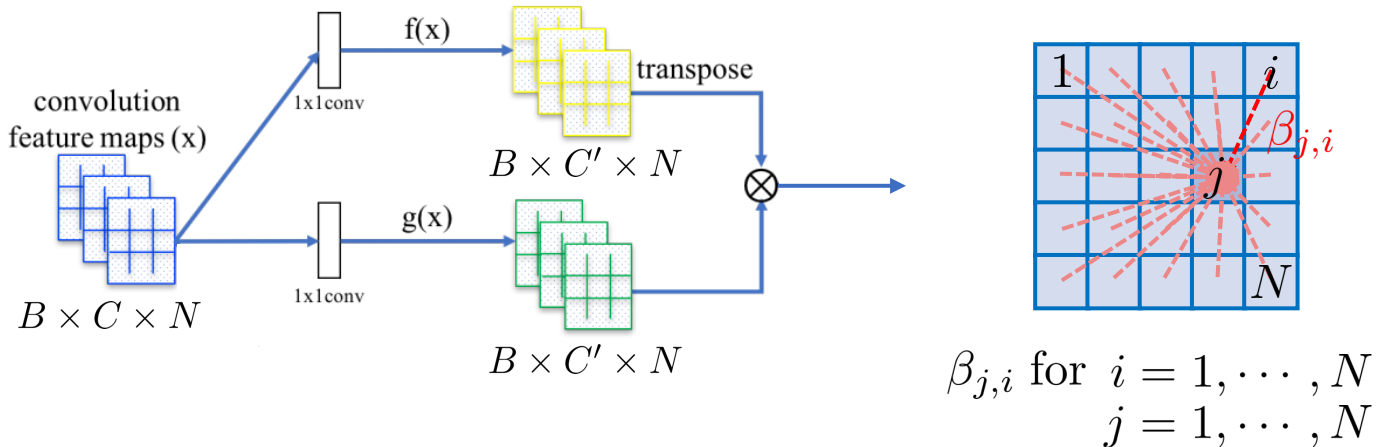
* source : Karras, et. al., Progressive growing of GANs for improved quality, stability, and variation, ICLR 2018 68

- Previous GANs often failed to capture geometric or structural patterns
- Using only convolutional layers may be computationally inefficient
 - Especially for modeling **long-range dependencies** in images
- **Self-Attention GAN (SAGAN)** [Wang et al., 2018]
 - The **non-local model (i.e. self-attention module)** of for both G and D
 - To efficiently model the relationships between spatial regions



→ : attention between regions

- The self-attention module of SAGAN



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

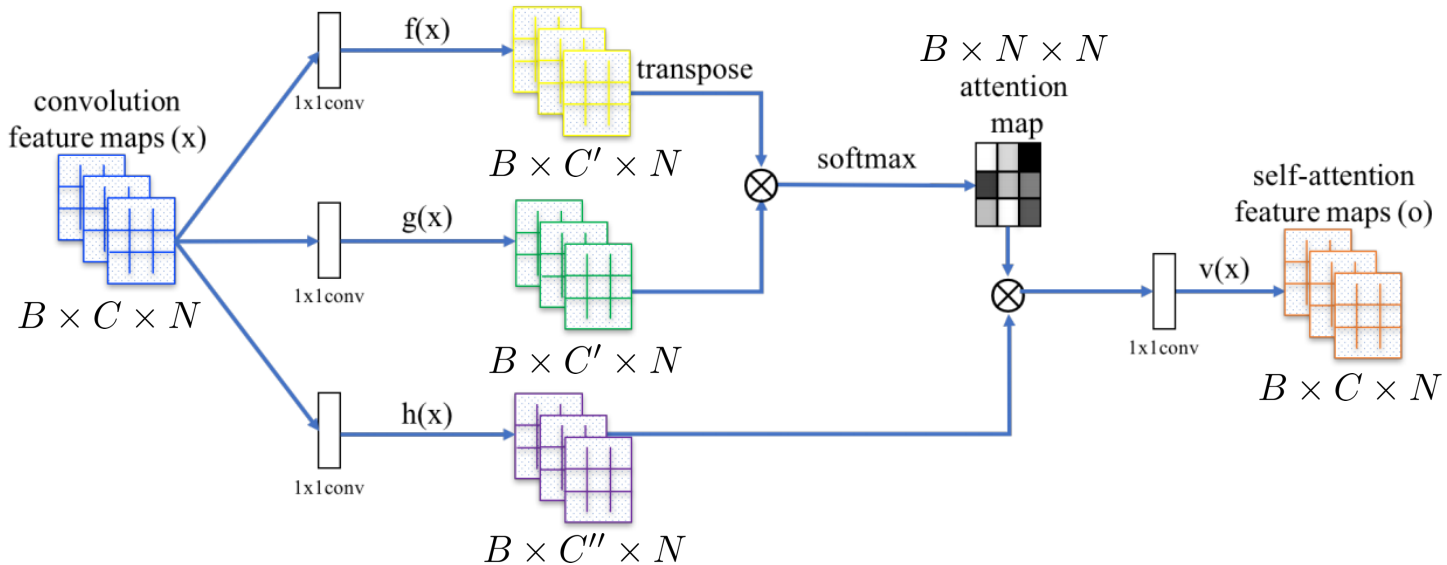
$$f(x) = W_f x, \quad g(x) = W_g x$$

- Then calculate the attention.

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

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

• The self-attention module of SAGAN



- Here the output of the attention layer is:

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

- 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).

$$y_i = \gamma o_i + x_i$$

- SAGAN improves upon state-of-the-art class-conditional ImageNet generation

Model	Inception Score	Intra FID	FID
AC-GAN (Odena et al., 2017)	28.5	260.0	/
SNGAN-projection (Miyato & Koyama, 2018)	36.8	92.4	27.62*
SAGAN	52.52	83.7	18.65

Table 2. Comparison of the proposed SAGAN with state-of-the-art GAN models (Odena et al., 2017; Miyato & Koyama, 2018) for class conditional image generation on ImageNet. FID of SNGAN-projection is calculated from officially released weights.



Visualization of generated samples & their attention maps

- Comparison between Self-Attention and Residual block in GANs
 - Ablation on the features index where the blocks added

Model	no attention	SAGAN				Residual			
		<i>feat₈</i>	<i>feat₁₆</i>	<i>feat₃₂</i>	<i>feat₆₄</i>	<i>feat₈</i>	<i>feat₁₆</i>	<i>feat₃₂</i>	<i>feat₆₄</i>
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

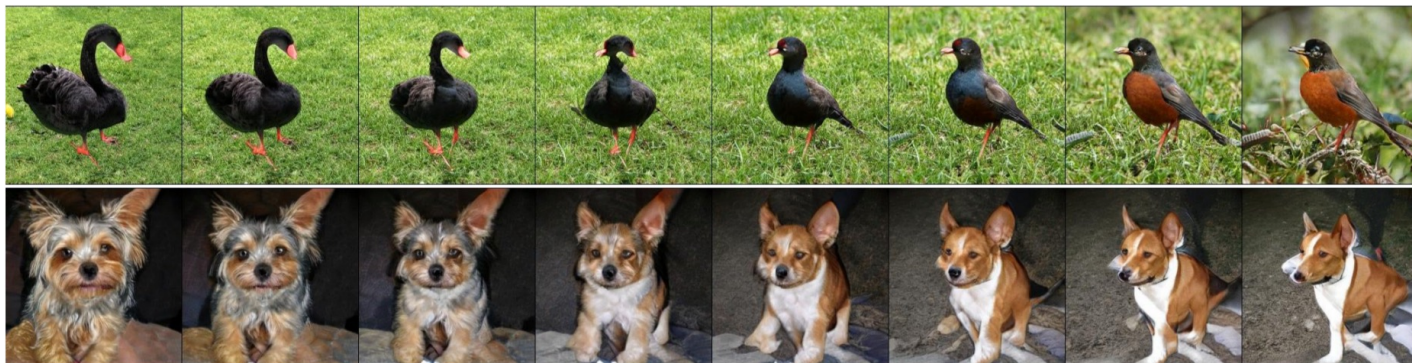
The improvements depend not only on residual connections, but also on attentions

BigGAN: High-resolution, Diverse Image Generation [Brock et al., 2019]

- BigGAN is a holistic approach of recent techniques for training GANs
- Current cGAN techniques can be successfully scaled up to generate **high-resolution, diverse samples** from complex datasets such as ImageNet



- Achieved state-of-the-art results in terms of IS and FID on 128×128 ImageNet



- **A holistic approach of previous GAN techniques**

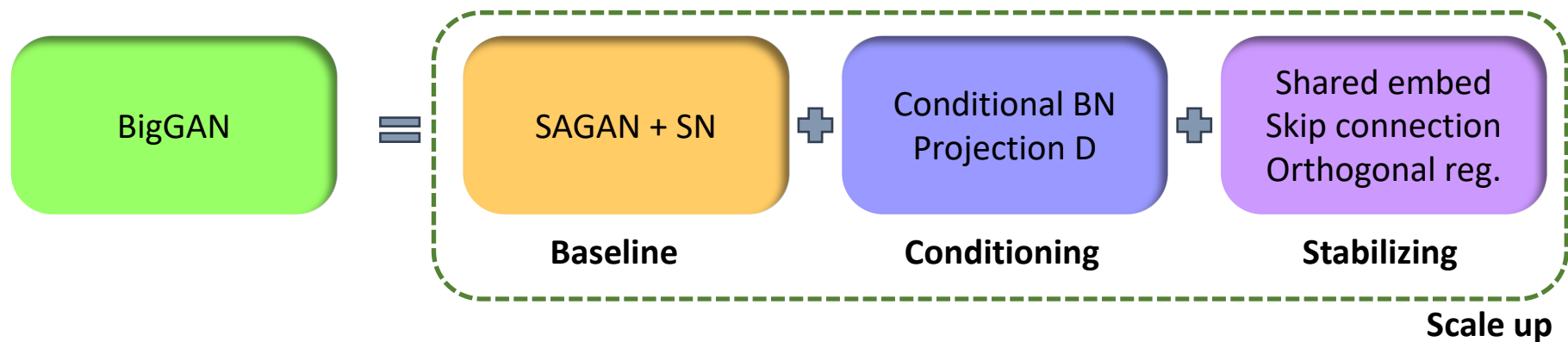
1. Based on SAGAN [Zhang et al., 2019] + Spectral normalization [Miyato et al., 2018]
2. Class-conditional modeling
 - G: **Class-conditional BatchNorm** [Dumoulin et al., 2017]
 - D: **Projection discriminator** [Miyato et al., 2018]

- **Several further techniques needed to stabilize the large-scale training**

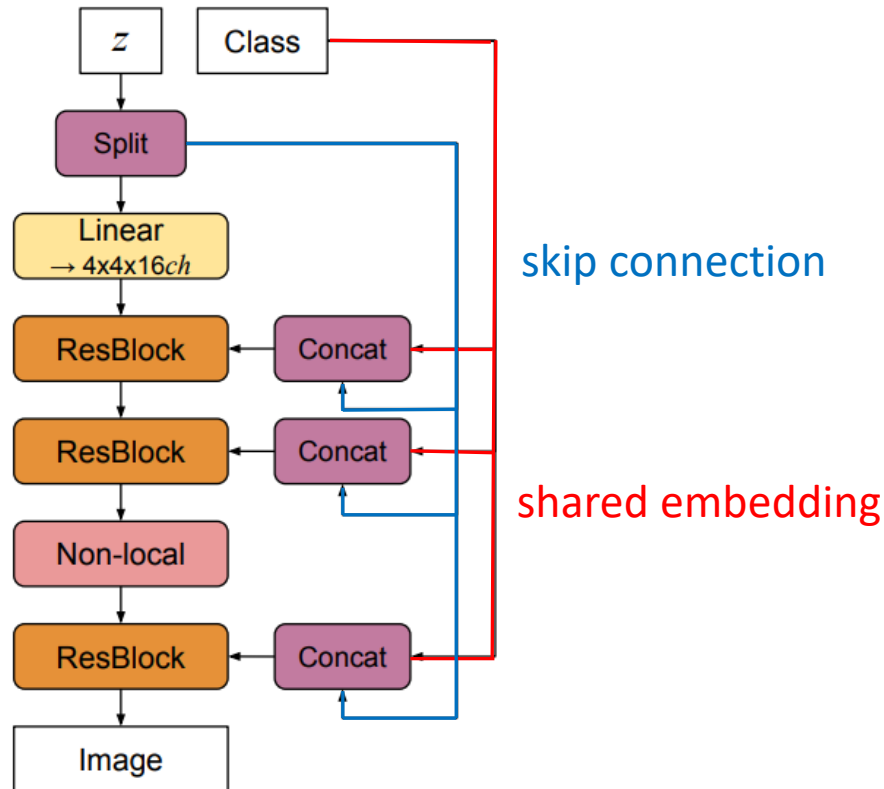
1. Shared embedding of y across multiple layers
2. Skip connection (residual) of the latent variable
3. Orthogonal regularization

$$R_{\beta}(W) = \beta \|W^{\top} W \odot (\mathbf{1} - I)\|_F^2$$

weight matrix



- Shared embedding of class information
 - Instead of having a separate layer at the end for embedding [Miyato et al., 2018]
 - Linearly projected to each layer's gains and biases [Perez et al., 2018]
- Skip connections (skip- z) from z across multiple layers of G
 - Allows z to **directly influence** the features at different resolutions



The BigGAN architecture

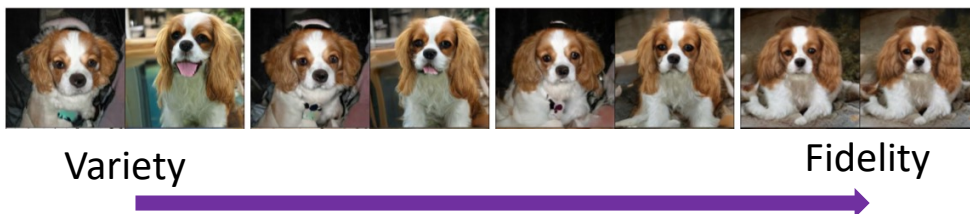
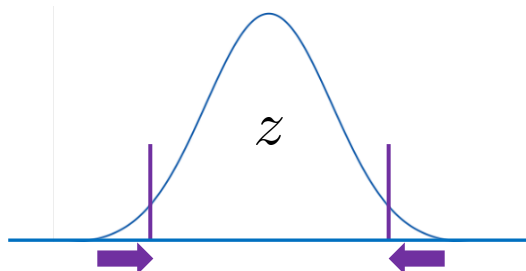
BigGAN: High-resolution, Diverse Image Generation [Brock et al., 2019]

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)

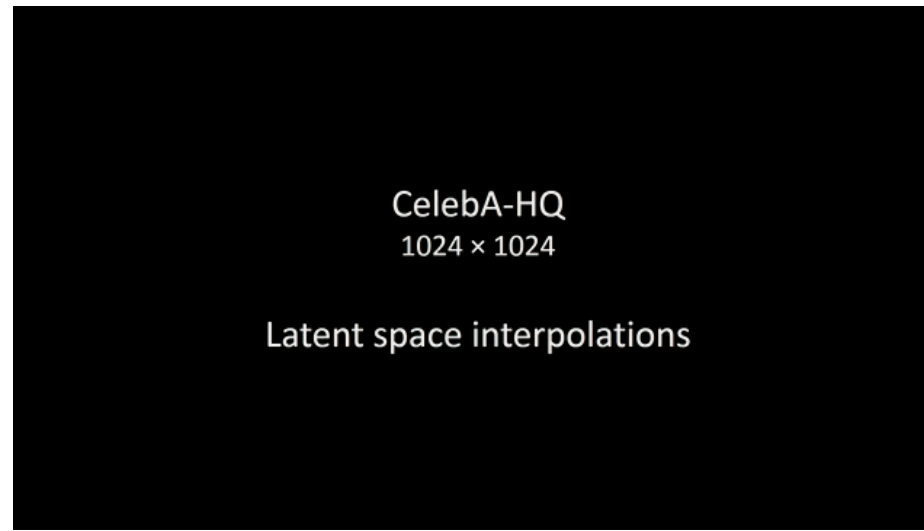
Scale up (blue arrow pointing up from the 2048x96x158.3 configuration to the 1024x64x81.5 configuration)

Stabilize (red arrow pointing down from the 2048x96x173.5 configuration to the 2048x96x158.3 configuration)

- Increasing the **batch size by 8x** improves the state-of-the-art IS by 46%
- Increasing the **width (# channels) by 1.5x** leads to a further improvement
- Truncation trick** could further fine-control FID
 - Trade-off between variety vs. fidelity
 - Simply truncate the variance of the latent variable



- Interpolation on the **latent space of GAN** yields smooth, but non-linear changes
 - Features not in both end-points appear along the interpolation path



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): **Intermediate latent space** representing a “style”
 - Significantly relaxes the restriction, and allowed to be **disentangled**

- StyleGAN proposes to use a **non-linear mapping network** $f : \mathcal{Z} \rightarrow \mathcal{W}$
 - Implemented using an 8-layer fully-connected neural network

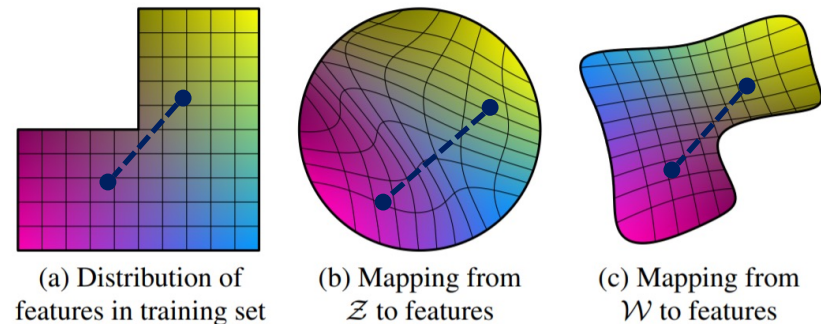
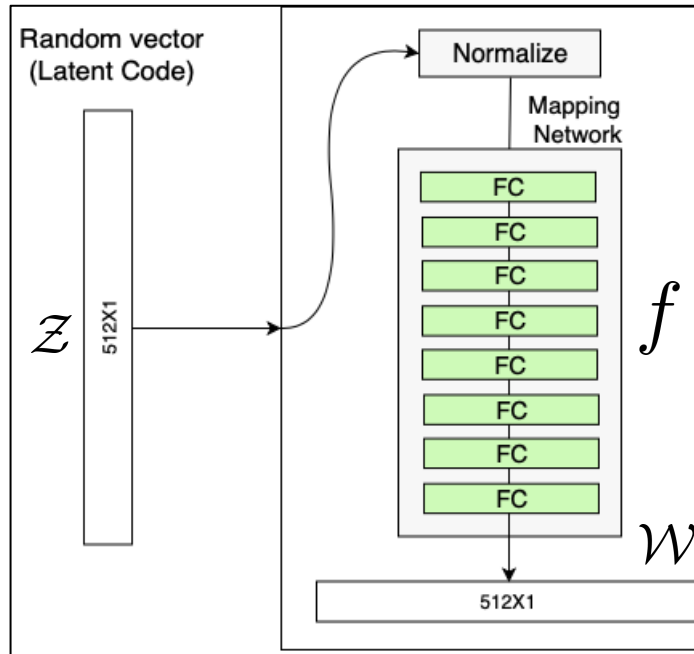


Illustration of disentanglement

- Direct mapping from \mathcal{Z} to meaningful features might be too complex
- Mapping from \mathcal{W} to the features, on the other hand, can be more simpler

- **Adaptive instance normalization (AdaIN)**

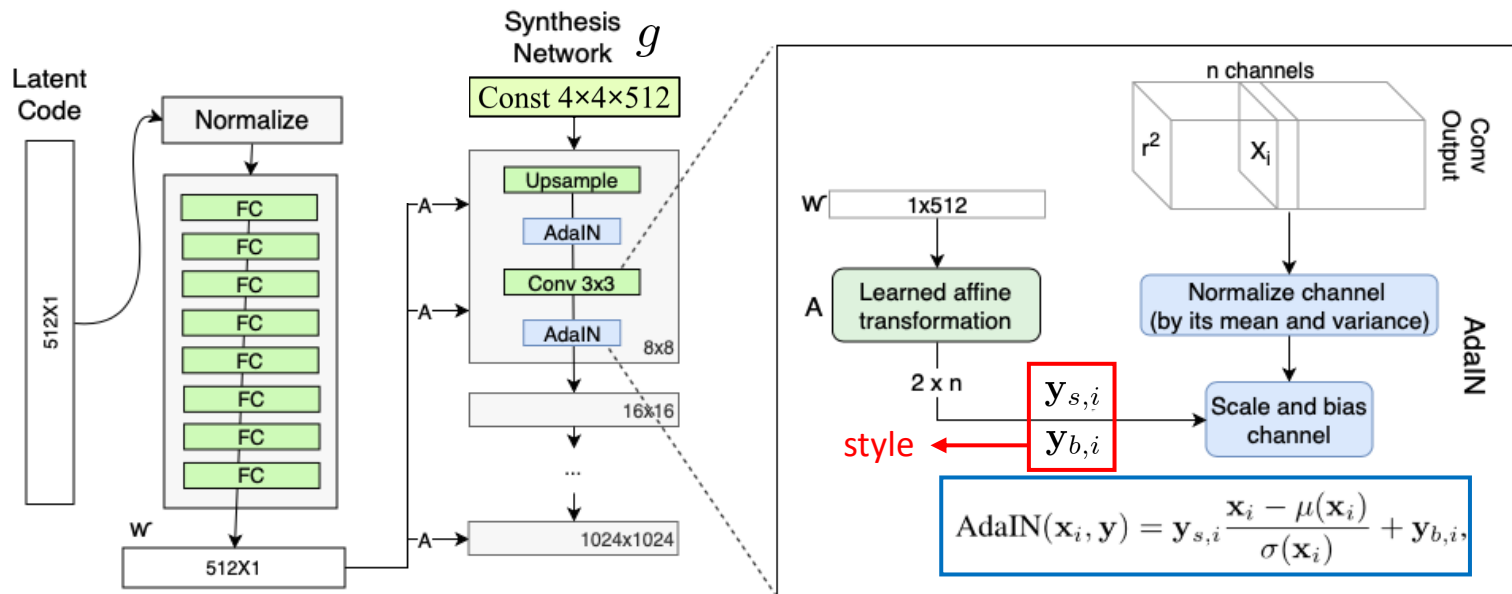
- Motivated by the instance normalization [Huang et al., 2017]

$$\text{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i},$$

- $\mathbf{y} = (\mathbf{y}_s, \mathbf{y}_b)$ is called by a “style”

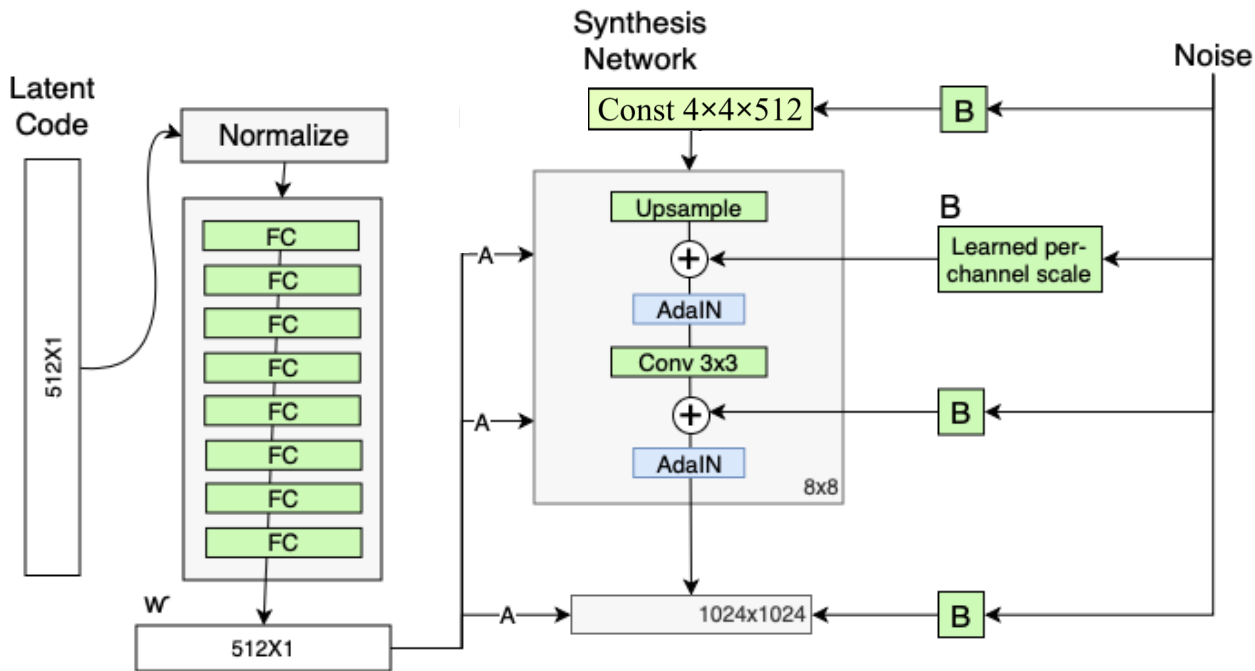
- A learned affine-transformation of $w \in \mathcal{W}$
- Controls high-level attributes (e.g., pose, identity of face images)

- Applied after all the convolutional layer in the **synthesis network** g



- **Explicit noise inputs for stochastic variation**
 - Single-channel images of **Gaussian noise**
 - Aims to control the stochastic details, e.g., freckles, hair of face images
- A noise channel n is fed to every layer of the synthesis network g
 - Broadcasted across features with **learned per-feature scaling factors** B

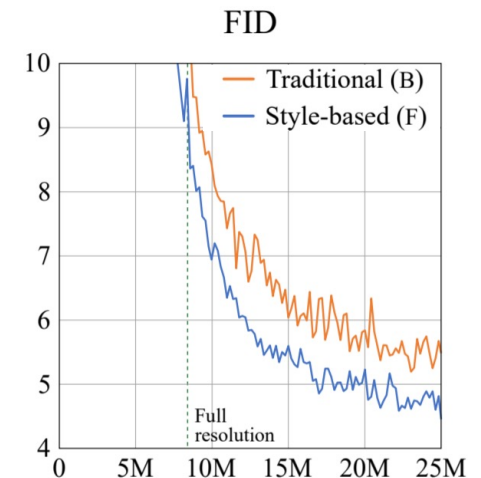
$$s(x_i, n) = x_i + B_i \cdot n$$



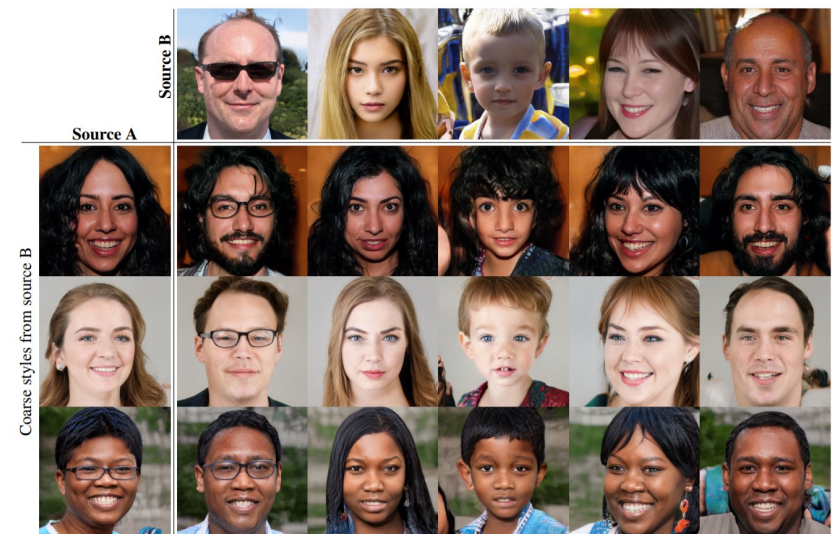
StyleGAN: A Style-Based Generator Architecture [Karras et al., 2019]

- StyleGAN improves state-of-the-art in terms of FID

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



- Better interpolation properties, and disentangles the latent factors of variation



- Karras et al. (2020a): Some **buggy-artifacts** in StyleGAN samples
 - **Blob-shaped artifacts** found in most of StyleGAN images (and hidden features)



Figure 1. Instance normalization causes water droplet -like artifacts in StyleGAN images. These are not always obvious in the generated images, but if we look at the activations inside the generator network, the problem is always there, in all feature maps starting from the 64x64 resolution. It is a systemic problem that plagues all StyleGAN images.

- **StyleGAN2** includes several design modifications to address this issue

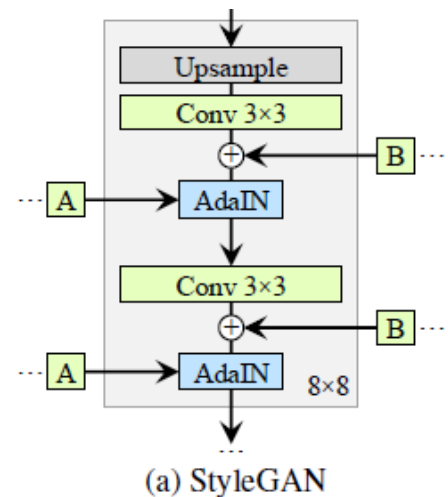
Configuration	FFHQ, 1024×1024				LSUN Car, 512×384			
	FID ↓	Path length ↓	Precision ↑	Recall ↑	FID ↓	Path length ↓	Precision ↑	Recall ↑
A Baseline StyleGAN [24]	4.40	212.1	0.721	0.399	3.27	1484.5	0.701	0.435
B + Weight demodulation	4.39	175.4	0.702	0.425	3.04	862.4	0.685	0.488
C + Lazy regularization	4.38	158.0	0.719	0.427	2.83	981.6	0.688	0.493
D + Path length regularization	4.34	122.5	0.715	0.418	3.43	651.2	0.697	0.452
E + No growing, new G & D arch.	3.31	124.5	0.705	0.449	3.19	471.2	0.690	0.454
F + Large networks (StyleGAN2)	2.84	145.0	0.689	0.492	2.32	415.5	0.678	0.514
Config A with large networks	3.98	199.2	0.716	0.422	—	—	—	—

- **Blob-shaped artifacts** found in most of StyleGAN images (and hidden features)
 1. The anomaly starts to appear around 64×64 resolution
 2. It becomes progressively stronger at higher resolutions



Figure 1. Instance normalization causes water droplet -like artifacts in StyleGAN images. These are not always obvious in the generated images, but if we look at the activations inside the generator network, the problem is always there, in all feature maps starting from the 64×64 resolution. It is a systemic problem that plagues all StyleGAN images.

- If so, **why the discriminator could not detect** those artifacts?
- Karras et al. (2020a): **AdaIN operation** can be problematic
 - AdaIN normalizes each feature map **separately**
 - This can destroy any magnitude information in the features relative to each other
- **Hypothesis:** they “sneak” some information past AdaIN



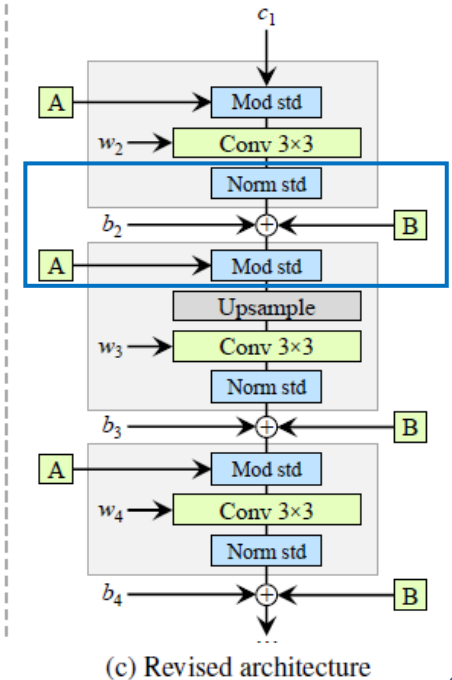
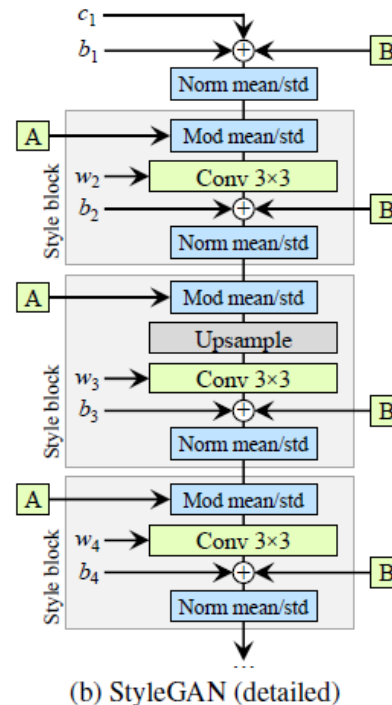
- Karras et al. (2020a): **AdaIN operation** can be problematic
 - AdaIN normalizes each feature map **separately**
 - This can destroy any magnitude information in the features relative to each other
- **Hypothesis:** they “sneak” some information past AdaIN
 - **Observation:** the artifacts disappear when the normalization step is removed
- **Generator architecture revisited \Rightarrow No artifacts anymore!**

1. Bias outside the style block

- StyleGAN applies bias & noise “within” the style block
 - Inversely proportional impact to **the current magnitude**
- This design is more **predictable**

2. No norm/mod for means

- It was possible after (1) is made
- Much simplifies the design



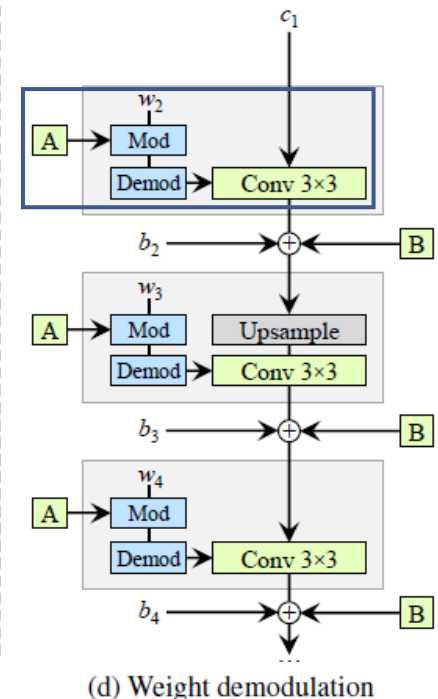
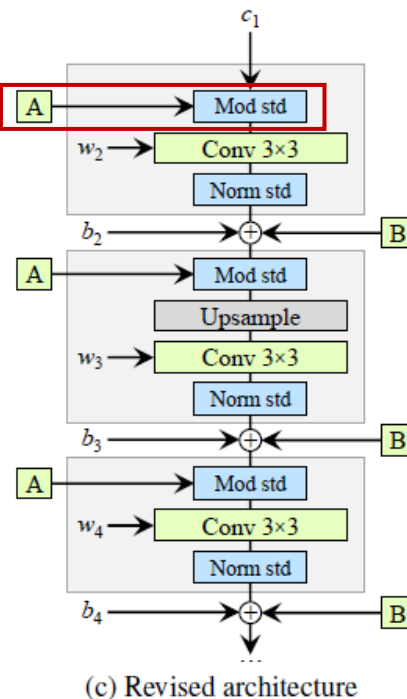
- Karras et al. (2020a): **AdaIN operation** can be problematic
 - AdaIN normalizes each feature map **separately**
 - This can destroy any magnitude information in the features relative to each other
- **Hypothesis:** they “sneak” some information past AdaIN
 - **Observation:** the artifacts disappear when the normalization step is removed
- **Generator architecture revisited \Rightarrow No artifacts anymore!**

3. Weight de-modulation

- A “weaker notion” of AdaIN
- AdaIN is originally for removing the effect of **input modulation**
- StyleGAN2 instead implement these “**Mod + AdaIN**” by **weight re-scaling**

$$w'_{ijk} = s_i \cdot w_{ijk},$$

$$w''_{ijk} = w'_{ijk} / \sqrt{\sum_{i,k} w'_{ijk}{}^2 + \epsilon},$$



• Path length regularization

- Recall the mapping network $f : \mathcal{Z} \rightarrow \mathcal{W}$
- Prior:** a fixed step in \mathcal{W} results in a **fixed-sized change** in $g(w)$

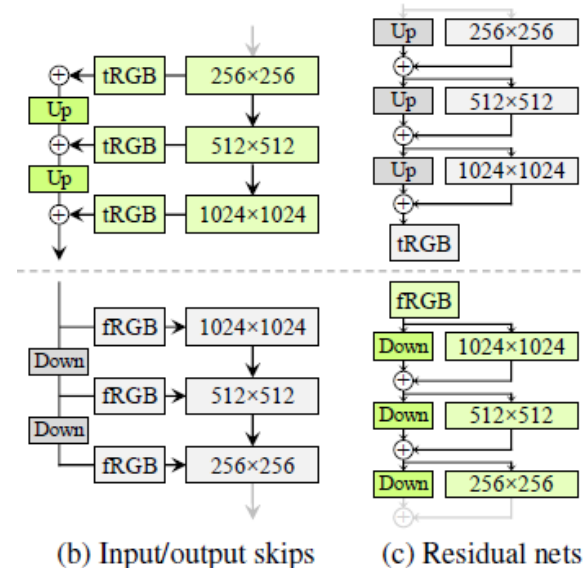
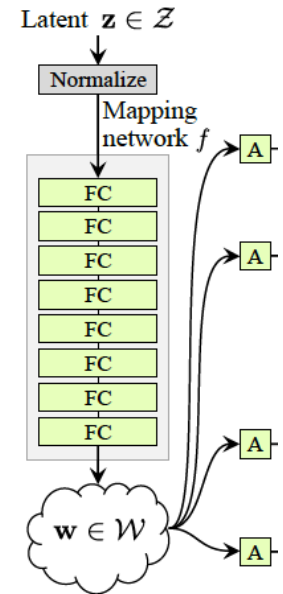
$$\mathbb{E}_{\mathbf{w}, \mathbf{y} \sim \mathcal{N}(0, \mathbf{I})} \left(\left\| \mathbf{J}_{\mathbf{w}}^T \mathbf{y} \right\|_2 - a \right)^2$$

- $\mathbf{w} \sim f(\mathbf{z})$; \mathbf{y} : random image
- $\mathbf{J}_{\mathbf{w}} = \partial g(\mathbf{w}) / \partial \mathbf{w}$: The Jacobian matrix

• Improved architectural design

- StyleGAN follows simple feedforward designs
- StyleGAN2 considers better architectural choices
 - Skip connections for G**
 - Residual network design for D**

FFHQ	D original		D input skips		D residual	
	FID	PPL	FID	PPL	FID	PPL
G original	4.32	265	4.18	235	3.58	269
G output skips	4.33	169	3.77	127	3.31	125
G residual	4.35	203	3.96	229	3.79	243



(b) Input/output skips

(c) Residual nets

StyleGAN2: Analyzing and Improving the Image Quality of StyleGAN [Karras et al., 2020a]

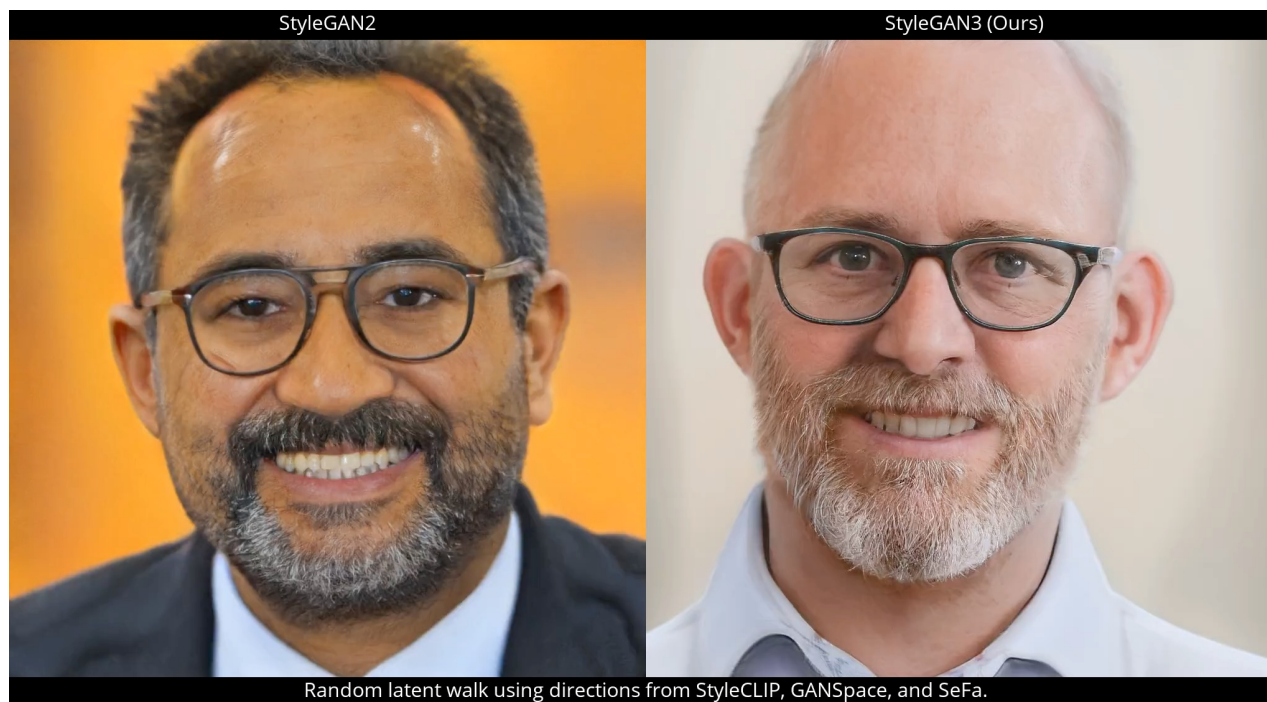
- StyleGAN2 successfully removes the **buggy-artifacts** of StyleGAN
 - **Weight de-modulation** significantly improves the **recall** of generations
 - Simply using **larger StyleGAN** could not be comparable with StyleGAN2

Configuration	FFHQ, 1024×1024				LSUN Car, 512×384			
	FID ↓	Path length ↓	Precision ↑	Recall ↑	FID ↓	Path length ↓	Precision ↑	Recall ↑
A Baseline StyleGAN [24]	4.40	212.1	0.721	0.399	3.27	1484.5	0.701	0.435
B + Weight demodulation	4.39	175.4	0.702	0.425	3.04	862.4	0.685	0.488
C + Lazy regularization	4.38	158.0	0.719	0.427	2.83	981.6	0.688	0.493
D + Path length regularization	4.34	122.5	0.715	0.418	3.43	651.2	0.697	0.452
E + No growing, new G & D arch.	3.31	124.5	0.705	0.449	3.19	471.2	0.690	0.454
F + Large networks (StyleGAN2)	2.84	145.0	0.689	0.492	2.32	415.5	0.678	0.514
Config A with large networks	3.98	199.2	0.716	0.422	—	—	—	—



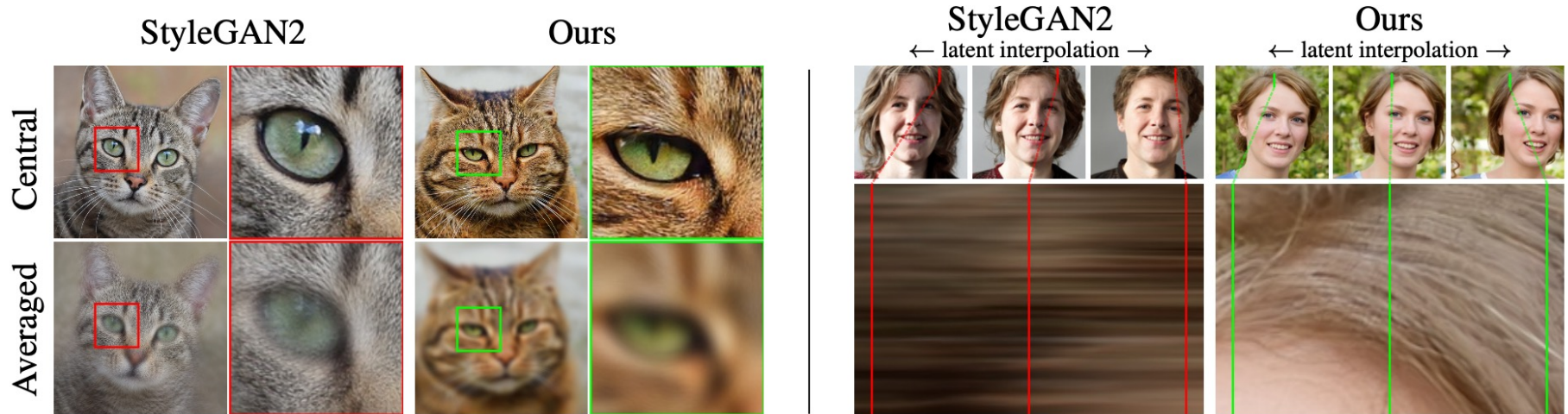
StyleGAN3: Alias-Free Generative Adversarial Networks [Karras et al., 2021]

- Karras et al. (2021): Still, some **buggy-artifacts** in StyleGAN2 latent space
 - **Texture striking** problem in most StyleGAN2 images
 - Textures (or details) appear to be **fixed** in pixel coordinates



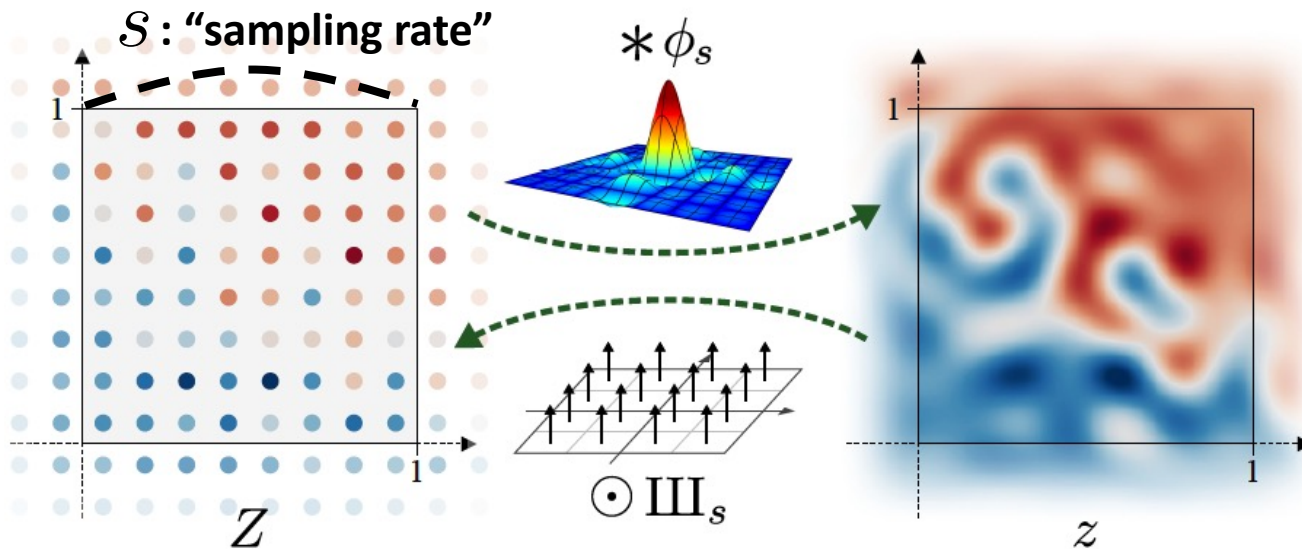
- **StyleGAN3** includes several design modifications to address this issue

- Current GANs do not synthesize images in a natural hierarchical manner
 1. Previous coarse features **control the “presence” of finer features**
 2. Yet, such details to be fixed in pixel coordinates



- **Why?** Unintentional positional references drawn on the intermediate layers
 - Faint after-images of the pixel grid from non-ideal up-sampling (e.g., nearest)
 - Pointwise application of non-linearities (e.g., ReLU)
- **Goal:** *Continuous equivariance* to sub-pixel translation (& rotation) in all layers

- Nyquist-Shannon sampling theorem [Shannon, 1949]
 - Regular, discrete signal can represent any continuous signals of frequencies 0- $s/2$
- Whittaker-Shannon interpolation formula [Shannon, 1949]



$$z(\mathbf{x}) = (\phi_s * Z)(\mathbf{x}), \text{ and } Z(\mathbf{x}) = \mathbb{I}_s \odot z(\mathbf{x})$$

$$\phi_s(\mathbf{x}) := \text{sinc}(sx_0) \cdot \text{sinc}(sx_1), \text{ where } \text{sinc}(x) := \frac{\sin(\pi x)}{\pi x}$$

$$\mathbb{I}_s(\mathbf{x}) := \sum_{X \in \mathbb{Z}^2} \delta(\mathbf{x} - (X + \frac{1}{2})/s)$$

- Continuous representation of network layers

- A neural network operation \mathbf{F} works on a **discrete** feature map: $Z' = \mathbf{F}(Z)$
- Consider its **continuous counterpart**, $z' = \mathbf{f}(z)$, from the correspondence of $z \leftrightarrow Z$

$$\mathbf{f}(z) = \phi_{s'} * \mathbf{F}(\mathbb{I}\mathbb{I}_s \odot z) \iff \mathbf{F}(Z) = \mathbb{I}\mathbb{I}_{s'} \odot \mathbf{f}(\phi_s * Z)$$

(a) (b)

- ... as long as **both (a) and (b)** are **band-limited** (to $s/2$ and $s'/2$, resp.)

“Equivariant” network layer (w.r.t. a spatial transformation \mathbf{t})?

- $\mathbf{f} \circ \mathbf{t} = \mathbf{t} \circ \mathbf{f}$ (commute) in the continuous domain
- \mathbf{f} (and \mathbf{t}) must not generate frequency above the output bandlimit of $s'/2$

Four operations in ConvNets: Convolution, Up/down-sampling, and Nonlinearity

- Here, we primarily discuss on the case of **translation** equivariance
- See the full paper for additional tricks to handle the **rotation** equivariance

Case 1: Convolution – $\mathbf{F}_{\text{conv}}(Z) = K * Z$

- Consider a discrete kernel K with sampling rate s

$$\mathbf{f}_{\text{conv}}(z) = \underbrace{\phi_s * (K * (\mathbb{I}\mathbb{I}\mathbb{I}_s \odot z))}_{\text{Commutative}} = K * \underbrace{(\phi_s * (\mathbb{I}\mathbb{I}\mathbb{I}_s \odot z))}_{\mathcal{Z}} = K * z$$

- Introduces **no new frequencies** – the bandlimit requirement fulfilled
- Convolution **commutes with translation** in the continuous domain

Case 2: Up-sampling ($s' > s$) – $\mathbf{f}_{\text{up}}(z) = z$

- Translation (and rotation) equivariance follow from being an identity in the continuous domain

$$\mathbf{F}_{\text{up}}(Z) = \mathbb{I}\mathbb{I}\mathbb{I}_{s'} \odot (\phi_s * Z)$$

- The operation can be simpler to implement when $s' = ns$ (n : an integer)
- First interleave Z with zeros, and then convolve it with $\mathbb{I}\mathbb{I}\mathbb{I}_{s'} \odot \phi_s$

Case 3: Down-sampling ($s' < s$) – $\mathbf{f}_{\text{down}}(z) = \psi_{s'} * z$, where $\psi_s := s^2 \cdot \phi_s$

- Low-pass filter to z to remove frequencies above the output bandlimit

$$\begin{aligned}\mathbf{F}_{\text{down}}(Z) &= \text{III}_{s'} \odot (\psi_{s'} * (\phi_s * Z)) = 1/s^2 \cdot \text{III}_{s'} \odot (\psi_{s'} * \psi_s * Z) \\ &= (s'/s)^2 \cdot \text{III}_{s'} \odot (\phi_{s'} * Z)\end{aligned}$$

- In case of $s = ns'$: a discrete convolution by dropping sample points
- Translation equivariance follows from the commutativity of \mathbf{f}_{down}

Case 4: Non-linearities (e.g., ReLU)

- Pointwise nonlinearity σ commutes with translation (and rotation) in the **continuous** domain
- However, The **bandlimit constraint** is problematic
 - ReLU in the continuous domain may introduce arbitrarily high frequencies
 - **Solution:** **Low-pass filtering** of $\sigma(z)$ via ψ_s

$$\mathbf{f}_\sigma(z) = \psi_s * \sigma(z) = s^2 \cdot \phi_s * \sigma(z)$$

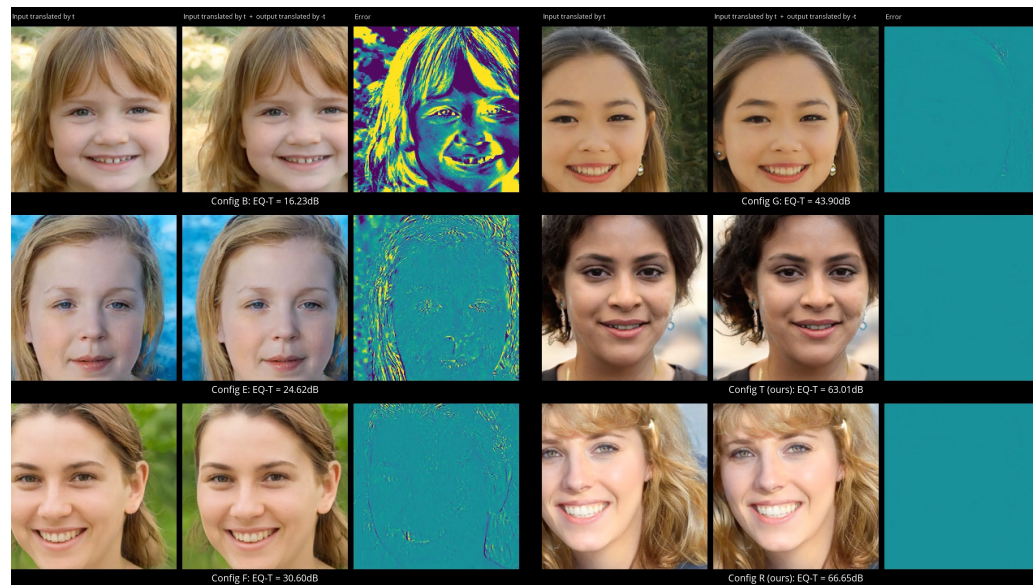
$$\mathbf{F}_\sigma(Z) = s^2 \cdot \mathbb{I}\mathbb{I}\mathbb{I}_s \odot (\phi_s * \sigma(\phi_s * Z))$$

- In practice, $\mathbf{F}_\sigma(Z)$ can't be realized without temporarily entering the continuous z
 - **Solution:** **Upsampling – ReLU – Downsampling**
 - Only a $2\times$ temporary upsampling was sufficient for high-quality equivariance

StyleGAN3: Alias-Free Generative Adversarial Networks [Karras et al., 2021]

- StyleGAN3 successfully achieves equivariance and avoid the texture-striking
 - Several techniques (e.g., Fourier features) to compensate worse FID

Configuration	FID↓	EQ-T↑	EQ-R↑
A StyleGAN2	5.14	—	—
B + Fourier features	4.79	16.23	10.81
C + No noise inputs	4.54	15.81	10.84
D + Simplified generator	5.21	19.47	10.41
E + Boundaries & upsampling	6.02	24.62	10.97
F + Filtered nonlinearities	6.35	30.60	10.81
G + Non-critical sampling	4.78	43.90	10.84
H + Transformed Fourier features	4.64	45.20	10.61
T + Flexible layers (Alias-Free-T)	4.62	63.01	13.12
R + Rotation equiv. (Alias-Free-R)	4.50	66.65	40.48



- **Discriminator:** Proposes several regularization for stabilizing the training
- **Enforcing Lipschitzness of Transformer discriminator**
 - **Unbounded Lipschitz constant** of conventional self-attention [Kim et al., 2021]

$$\text{Attention}_h(\mathbf{X}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}}\right)\mathbf{V},$$

- Instead utilizes “L2 attention”, which **enforces Lipschitzness**

$$\text{Attention}_h(\mathbf{X}) = \text{softmax}\left(-\frac{\overset{\text{Euclidean distance}}{d(\mathbf{X}\mathbf{W}_q, \mathbf{X}\mathbf{W}_k)}}{\sqrt{d_h}}\right)\mathbf{X}\mathbf{W}_v, \quad \text{where } \mathbf{W}_q = \mathbf{W}_k;$$

- **Improved spectral normalization (SN)**
 - Multiplies SN at “initialization” in addition to current SN
 - ... as with **small Lipschitz constant, Transformer collapse to rank 1** matrices
 - Empirically, ViTGAN without SN learns in a healthy manner

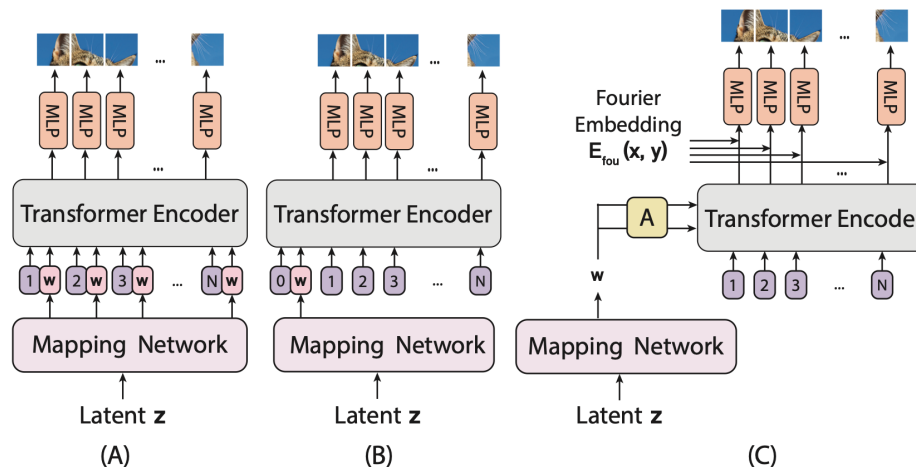
$$\bar{W}_{\text{ISN}}(\mathbf{W}) := \sigma(\mathbf{W}_{init}) \cdot \mathbf{W} / \sigma(\mathbf{W}).$$

- **Discriminator:** Proposes several regularization for stabilizing the training
- **Overlapping image patches**
 - Same pre-defined patch grid $(P \times P)$ **memorizes local cues** and prohibits training
 - **Use overlapping patch** for discriminator to mitigate the issue $((P + 2o) \times (P + 2o))$
- **Convolution projection**
 - For computing $\mathbf{Q}, \mathbf{K}, \mathbf{V}$, **apply convolutions** rather than linear layers
 - It promotes to focus on local context as well as global context

- **Generator:** Proposes a new Transformer-based generator
 - ...as baseline Transformers perform poorly compared with CNN-based generator
- **Self-modulated LayerNorm**
 - Instead of injecting latents as inputs, use to “modulate” the LayerNorm

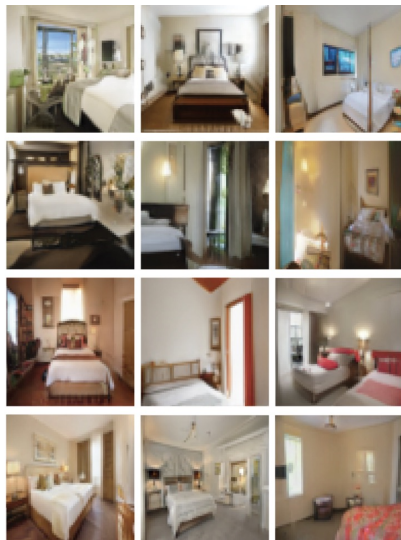
$$\text{SLN}(\mathbf{h}_\ell, \mathbf{w}) = \text{SLN}(\mathbf{h}_\ell, \text{MLP}(\mathbf{z})) = \gamma_\ell(\mathbf{w}) \odot \frac{\mathbf{h}_\ell - \boldsymbol{\mu}}{\boldsymbol{\sigma}} + \beta_\ell(\mathbf{w})$$

- **Implicit neural representations for patch generation**
 - Use [implicit neural representations](#) to map from patch embedding to pixel values
 - Implicit neural representations: MLP with sinusoidal activations (details in Lecture 8)



- **ViTGAN achieves state-of-the-art (or comparable) results to prior arts**
 - Intriguingly, “naïve” extension of ViT for GANs fails

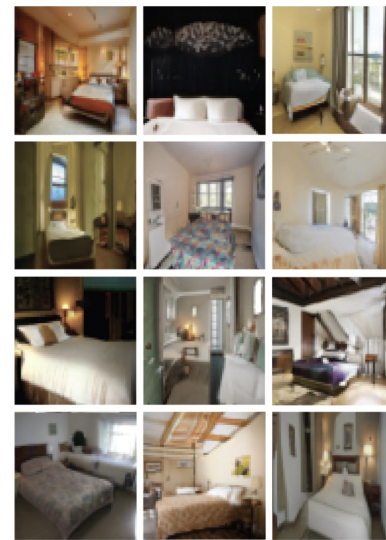
Architecture	CIFAR 10		CelebA 64x64		LSUN 64x64		LSUN 128x128	
	FID ↓	IS ↑	FID ↓	IS ↑	FID ↓	IS ↑	FID ↓	IS ↑
BigGAN+ DiffAug (Zhao et al., 2020a)	8.59*	9.25*	-	-	-	-	-	-
StyleGAN2 (Karras et al., 2020b)	5.60	9.41	3.39	3.43	2.33	2.44	3.26	2.26
TransGAN (Jiang et al., 2021)	9.02*	9.26*	-	-	-	-	-	-
Vanilla-ViT	12.7	8.40	20.2	2.57	218.1	2.20	-	-
ViTGAN (Ours)	4.92	9.69	3.74	3.21	2.40	2.28	2.48	2.26
StyleGAN2-D+ViTGAN-G (Ours)	4.57	9.89	-	-	1.49	2.46	1.87	2.32



(g) StyleGAN2 (FID = 2.33)



(h) Vanilla-ViT (FID = 218.1)



(i) ViTGAN (FID = 2.40)

1. Basics of Generative Adversarial Networks (GAN)

- Advantages and disadvantages of GAN
- Conditional GANs

2. Advanced Topics of GANs

- Loss, regularization and normalization
- Data augmentations for GANs
- GAN architectures
- GAN training with pretrained networks

3. Beyond Image Generation: More Applications

- GANs for image manipulation
- GANs for video generation
- GANs for medical diagnosis

- Sauer et al. (2021): GAN training can be improved by **projecting generated and real samples into a fixed, pretrained feature space**

$$\min_G \max_D \left(\mathbb{E}_{\mathbf{x}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))] \right)$$

Original GAN objective

$$\min_G \max_{\{D_l\}} \sum_{l \in \mathcal{L}} \left(\mathbb{E}_{\mathbf{x}}[\log D_l(\underline{P_l}(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}}[\log(1 - D_l(P_l(G(\mathbf{z}))))] \right)$$

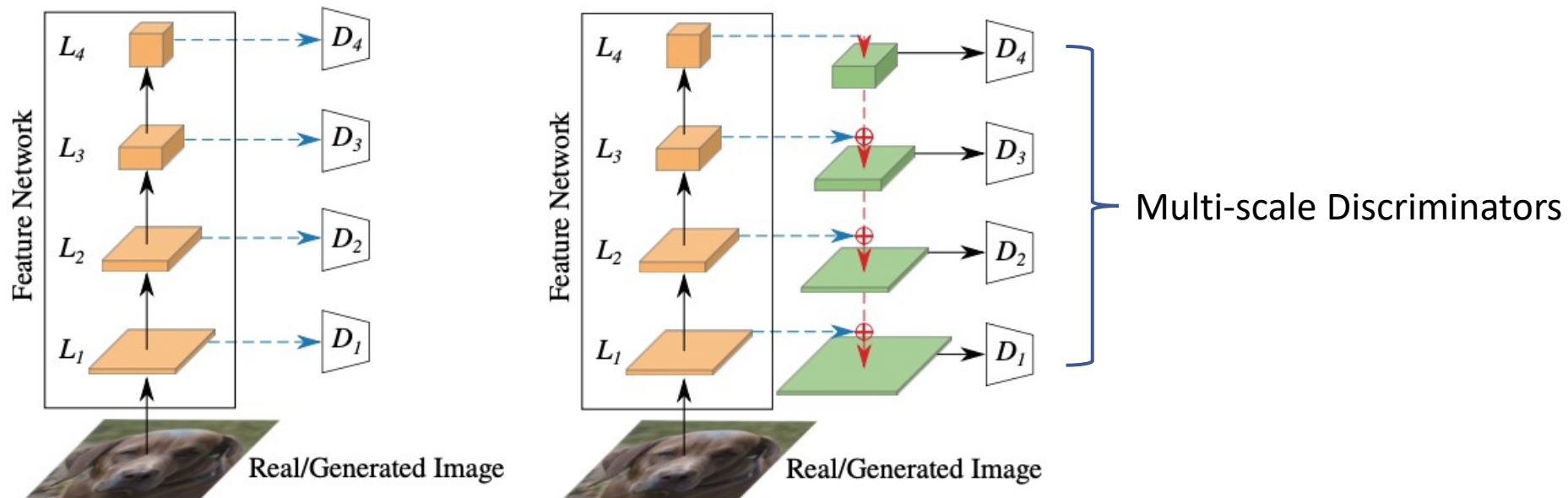
Projected GAN objective

Not optimize P

Optimize G and D

- How to effectively utilize **pretrained layers** $\{P_l\}$?
 - **Note.** Naïve application of using pretrained networks does not lead to SOTA results
 - 1. **Multi-scale Discriminators.** Enable multi-scale feedback with multiple discriminators
 - 2. **Random Projections.** Better utilize deeper layers of the pretrained network

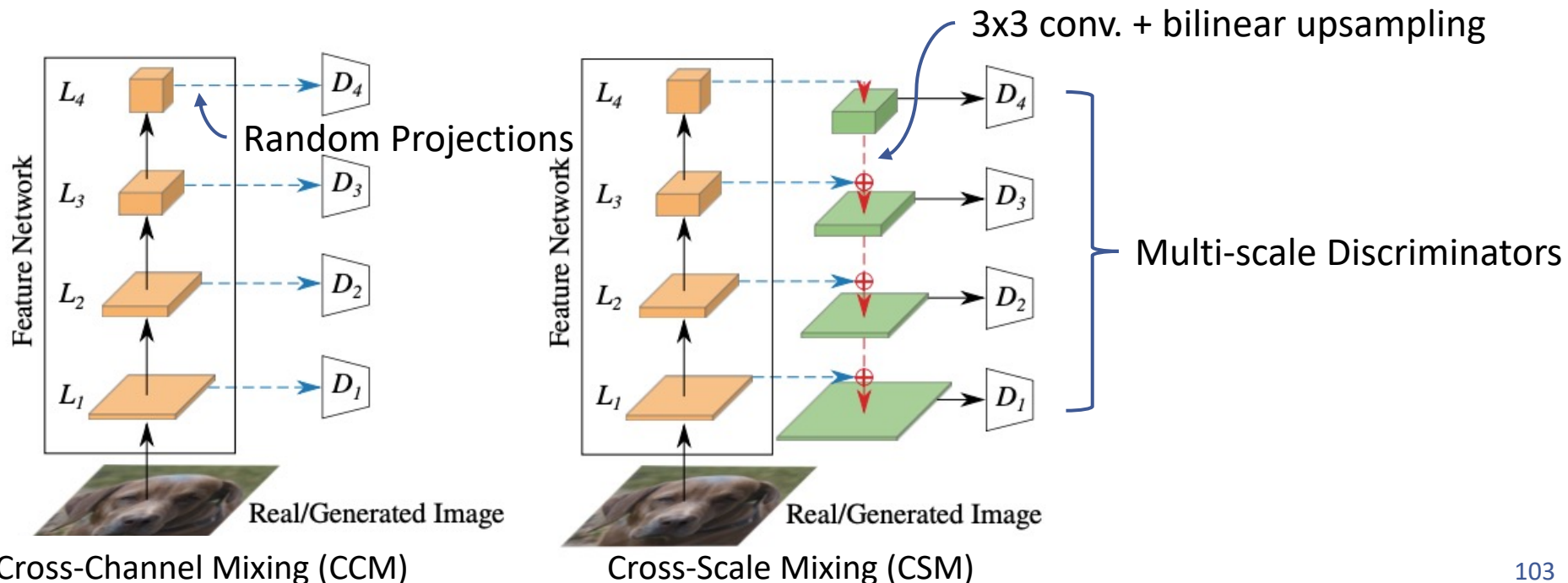
- Sauer et al. (2021): GAN training can be improved by **projecting generated and real samples into a fixed, pretrained feature space**
- **Multi-scale Discriminators**
 - Obtain features from four layers of a pretrained feature network
 - Feature resolutions: 8×8 , 16×16 , 32×32 , 64×64
 - Associate a separate discriminator D_l with the features at layer l , respectively



- Sauer et al. (2021): GAN training can be improved by **projecting generated and real samples into a fixed, pretrained feature space**

- **Random Projections**

- A discriminator can focus on a subset of the feature space
 - This problem might be especially prominent in the deeper, more semantic layers
- To dilute prominent features, mix features using random projections
 - **Note.** They are fixed after random initialization



- Sauer et al. (2021): GAN training can be improved by **projecting generated and real samples into a fixed, pretrained feature space**
- What is **the best pretrained network** for GAN training?

	EfficientNet					ResNet			Transformer	
	lite0	lite1	lite2	lite3	lite4	R18	R50	R50-CLIP	DeiT	ViT
Params (M) ↓	2.96	3.72	4.36	6.42	11.15	11.18	23.51	23.53	92.36	317.52
IN top-1 ↑	75.48	76.64	77.47	79.82	81.54	69.75	79.04	N/A	85.42	85.16
FID ↓	2.53	1.65	1.69	1.79	2.35	4.16	4.40	3.80	2.46	12.38

- No correlation with ImageNet accuracy
- On the contrary, smaller models \Rightarrow smaller FIDs (i.e., better generation)
- This indicates that **a more compact representation is beneficial** for training GANs

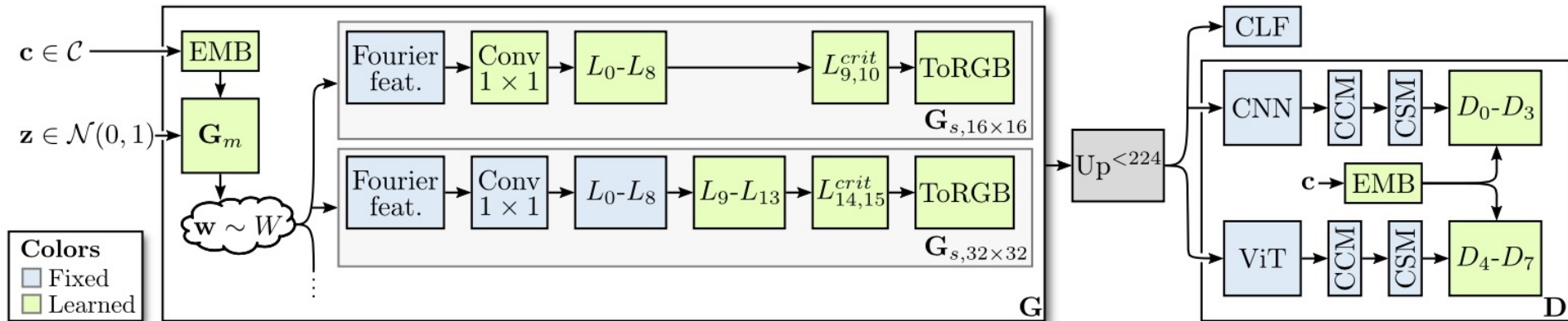
Projected GANs Converge Faster [Sauer et al., 2021]

- Sauer et al. (2021): GAN training can be improved by **projecting generated and real samples into a fixed, pretrained feature space**
- Projected GANs achieves fast convergence and improved generation quality
 - Projected GANs outperforms all baselines on small & large datasets with a resolution up to 1024^2 pixels



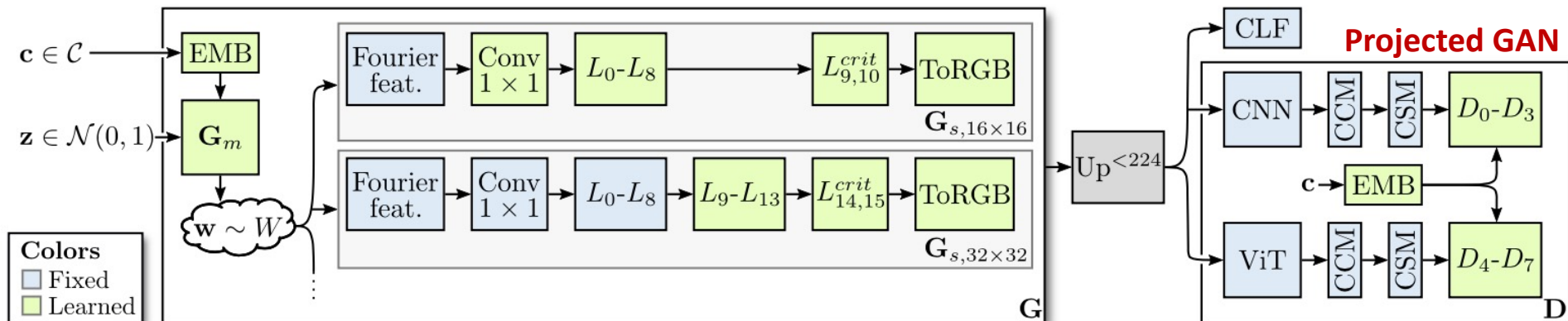
- **Scaling StyleGANs** is non-trivial
 - Why? Due to architectural constraints or some missing piece in the training strategy?
 - Leveraging the benefits of Projected GAN training might enable scaling StyleGAN
- **StyleGAN-XL** is based on StyleGAN3

Configuration	FID ↓	IS ↑
A StyleGAN3	53.57	15.30
B + Projected GAN & small \mathbf{z}	22.98	57.62
C + Pretrained embeddings	20.91	35.79
D + Progressive growing	19.51	35.74
E + ViT & CNN as $\mathbf{F}_{1,2}$	12.43	56.72
F + CLF guidance (StyleGAN-XL)	12.24	86.21



- **Scaling StyleGANs** is non-trivial
 - Why? Due to architectural constraints or some missing piece in the training strategy?
 - Leveraging the benefits of Projected GAN training might enable scaling StyleGAN
- **StyleGAN-XL** is based on StyleGAN3

Configuration	FID ↓	IS ↑
A StyleGAN3	53.57	15.30
B + Projected GAN & small z	22.98	57.62
C + Pretrained embeddings	20.91	35.79
D + Progressive growing	19.51	35.74
E + ViT & CNN as $F_{1,2}$	12.43	56.72
F + CLF guidance (StyleGAN-XL)	12.24	86.21



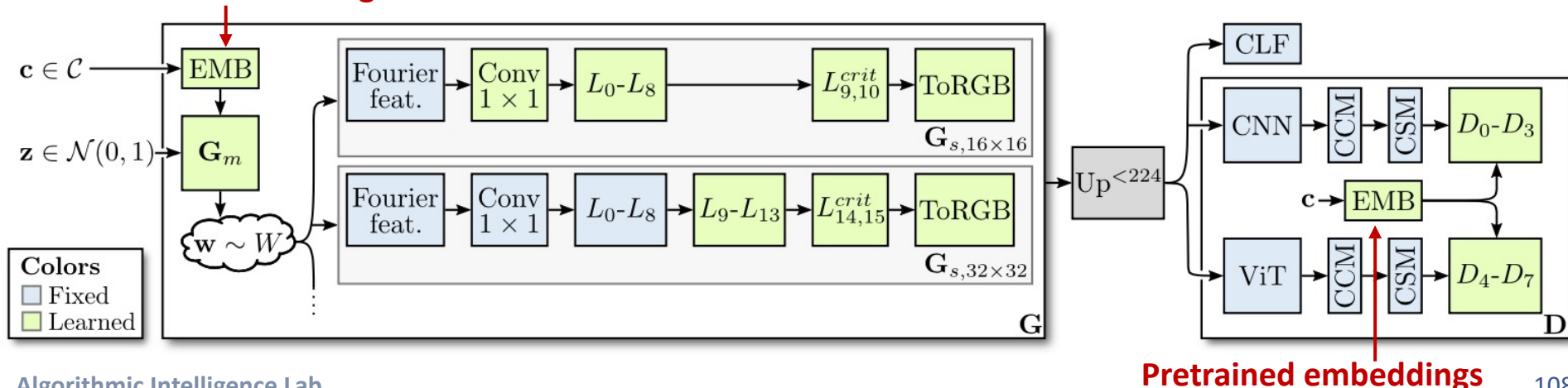
- **Scaling StyleGANs** is non-trivial

- Why? Due to architectural constraints or some missing piece in the training strategy?
- Leveraging the benefits of Projected GAN training might enable scaling StyleGAN

- **StyleGAN-XL** is based on StyleGAN3

Configuration	FID ↓	IS ↑
A StyleGAN3	53.57	15.30
B + Projected GAN & small z	22.98	57.62
C + Pretrained embeddings	20.91	35.79
D + Progressive growing	19.51	35.74
E + ViT & CNN as $F_{1,2}$	12.43	56.72
F + CLF guidance (StyleGAN-XL)	12.24	86.21

Initialized by
Pretrained embeddings

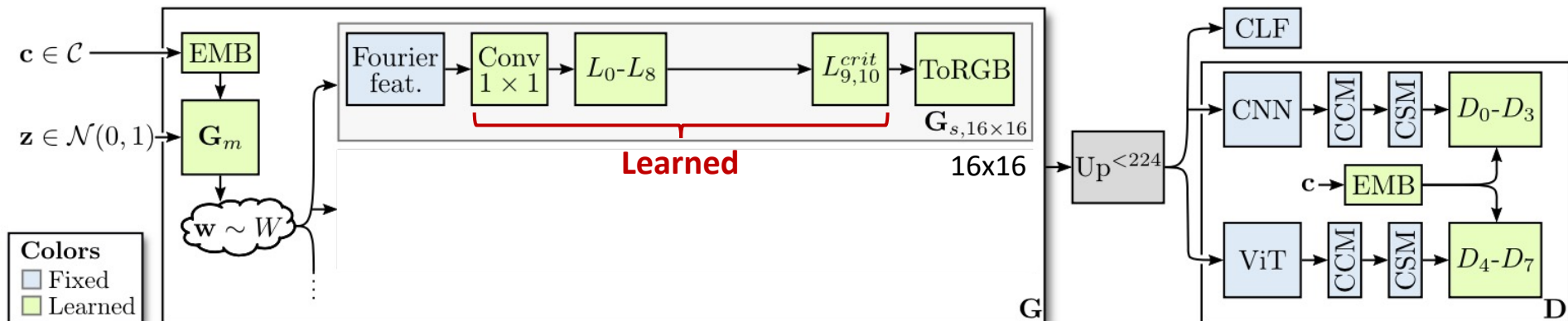


- **Scaling StyleGANs** is non-trivial

- Why? Due to architectural constraints or some missing piece in the training strategy?
- Leveraging the benefits of Projected GAN training might enable scaling StyleGAN

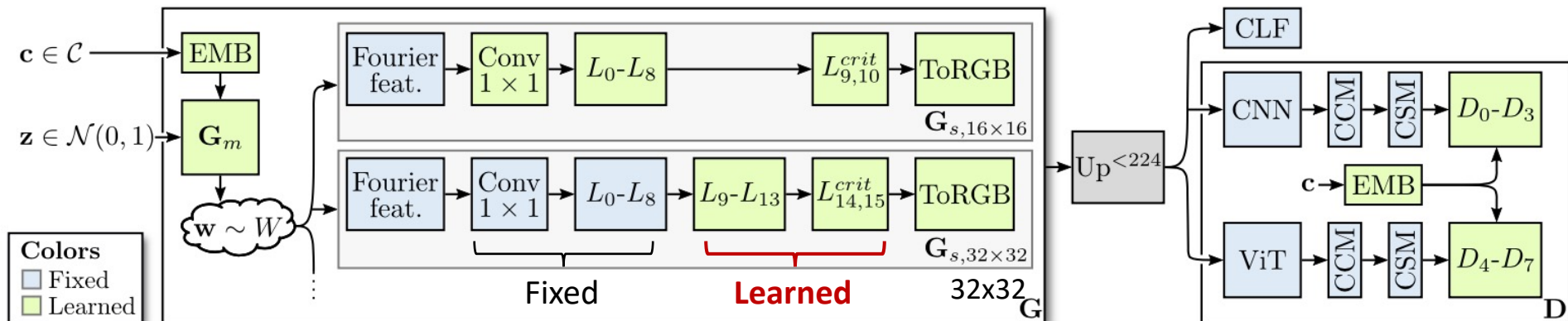
- **StyleGAN-XL** is based on StyleGAN3

Configuration	FID ↓	IS ↑
A StyleGAN3	53.57	15.30
B + Projected GAN & small z	22.98	57.62
C + Pretrained embeddings	20.91	35.79
D + Progressive growing	19.51	35.74
E + ViT & CNN as $F_{1,2}$	12.43	56.72
F + CLF guidance (StyleGAN-XL)	12.24	86.21



- **Scaling StyleGANs** is non-trivial
 - Why? Due to architectural constraints or some missing piece in the training strategy?
 - Leveraging the benefits of Projected GAN training might enable scaling StyleGAN
- **StyleGAN-XL** is based on StyleGAN3

Configuration	FID ↓	IS ↑
A StyleGAN3	53.57	15.30
B + Projected GAN & small z	22.98	57.62
C + Pretrained embeddings	20.91	35.79
D + Progressive growing	19.51	35.74
E + ViT & CNN as $F_{1,2}$	12.43	56.72
F + CLF guidance (StyleGAN-XL)	12.24	86.21

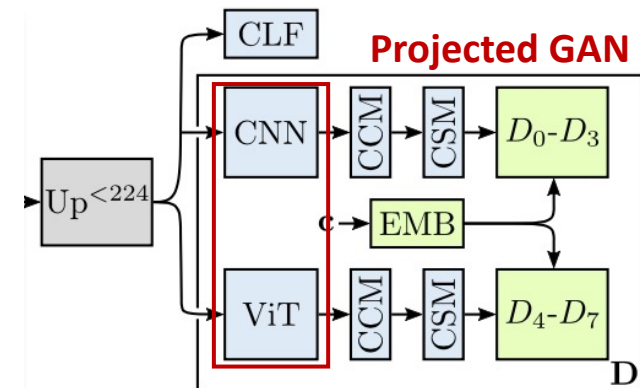


- **Scaling StyleGANs** is non-trivial
 - Why? Due to architectural constraints or some missing piece in the training strategy?
 - Leveraging the benefits of Projected GAN training might enable scaling StyleGAN
- **StyleGAN-XL** is based on StyleGAN3

Configuration		FID ↓	IS ↑
A	StyleGAN3	53.57	15.30
B	+ Projected GAN & small z	22.98	57.62
C	+ Pretrained embeddings	20.91	35.79
D	+ Progressive growing	19.51	35.74
E	+ ViT & CNN as $F_{1,2}$	12.43	56.72
F	+ CLF guidance (StyleGAN-XL)	12.24	86.21

Combining CNN and ViT improves performance significantly

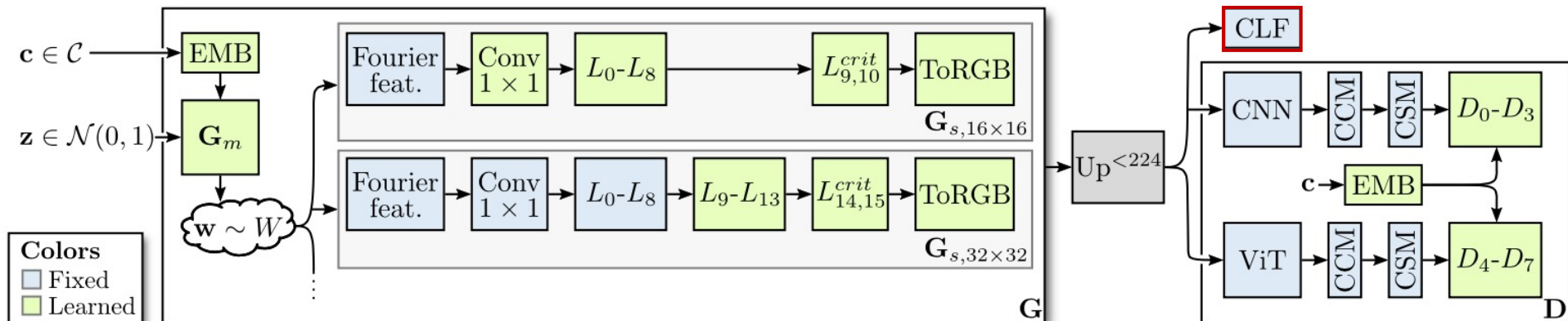
Model		Type		Objective		FID ↓	IS ↑
F_1	F_2	F_1	F_2	F_1	F_2		
EffNet		CNN		Class		19.51	35.74
EffNet	ResNet50	CNN	CNN	Class	Class	16.16	49.13
EffNet	ResNet50	CNN	CNN	Class	Self	18.53	38.26
EffNet	DeiT-M	CNN	ViT	Class	Class	12.43	56.72



- **Scaling StyleGANs** is non-trivial
 - Why? Due to architectural constraints or some missing piece in the training strategy?
 - Leveraging the benefits of Projected GAN training might enable scaling StyleGAN
- **StyleGAN-XL** is based on StyleGAN3

Configuration	FID ↓	IS ↑
A StyleGAN3	53.57	15.30
B + Projected GAN & small \mathbf{z}	22.98	57.62
C + Pretrained embeddings	20.91	35.79
D + Progressive growing	19.51	35.74
E + ViT & CNN as $\mathbf{F}_{1,2}$	12.43	56.72
F + CLF guidance (StyleGAN-XL)	12.24	86.21

When training G,
add classification loss



- **Scaling StyleGANs** is non-trivial
 - Why? Due to architectural constraints or some missing piece in the training strategy?
 - Leveraging the benefits of Projected GAN training might enable scaling StyleGAN
- **StyleGAN-XL** is based on StyleGAN3
 - It is trained with various training techniques: projected GANs, progressive growing, ...
 - It enables to demonstrate image synthesis on ImageNet-scale at resolution of 1024^2

Class-conditioned samples generated by StyleGANs trained on ImageNet at resolution 256^2



StyleGAN3



StyleGAN-XL

1. Basics of Generative Adversarial Networks (GAN)

- Advantages and disadvantages of GAN
- Conditional GANs

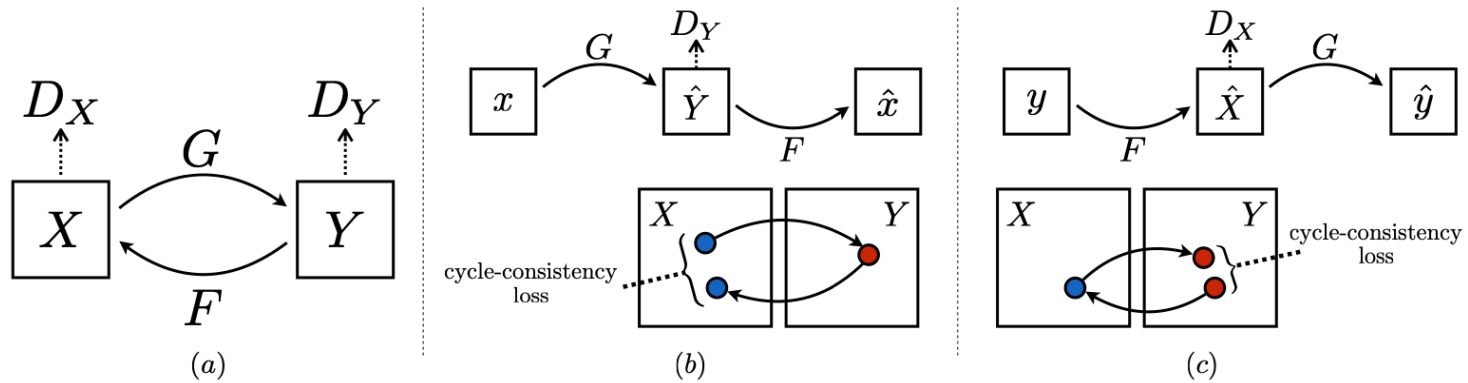
2. Advanced Topics of GANs

- Loss, regularization and normalization
- Data augmentations for GANs
- GAN architectures
- GAN training with pretrained networks

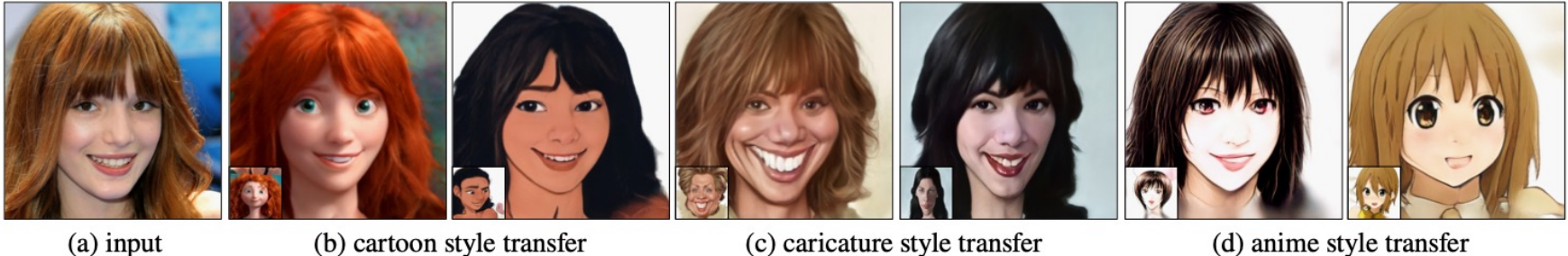
3. Beyond Image Generation: More Applications

- GANs for image manipulation
- GANs for video generation
- GANs for medical diagnosis

- **Image manipulation** (or image-to-image translation, style transfer)
 - The goal is to learn mapping functions between two domains X and Y
 - CycleGAN [Zhu et al., 2017] learn GAN ($G(X)$ vs. Y) w/ cycle consistency $F(G(X)) \approx Y$

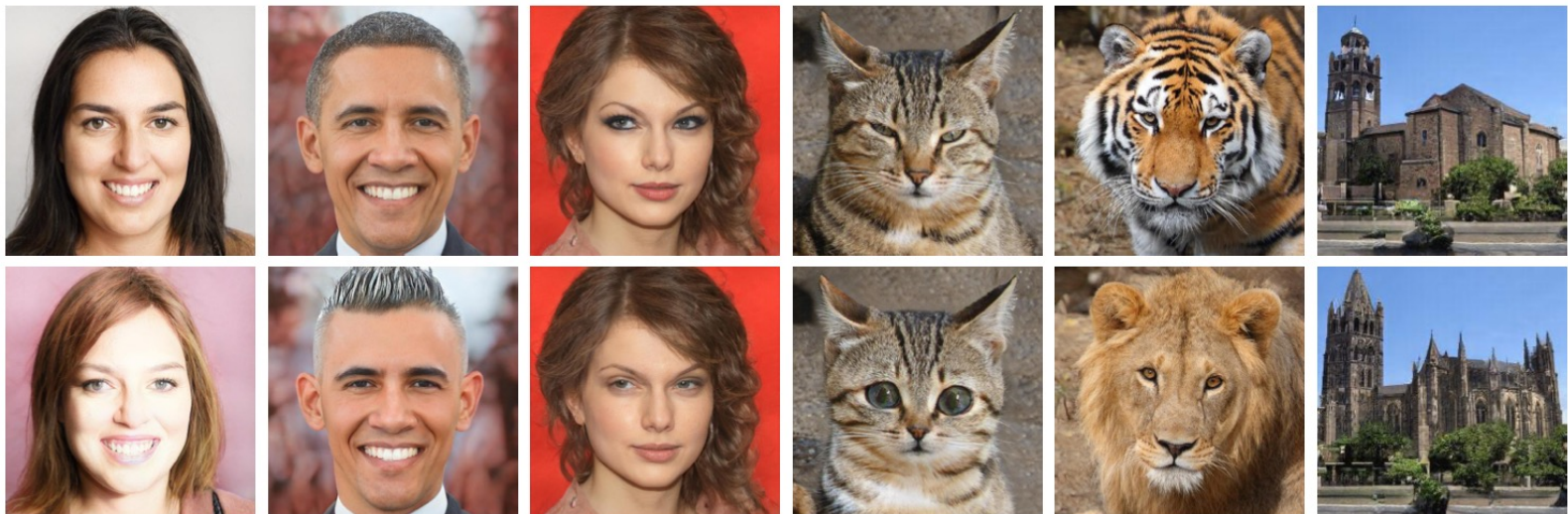
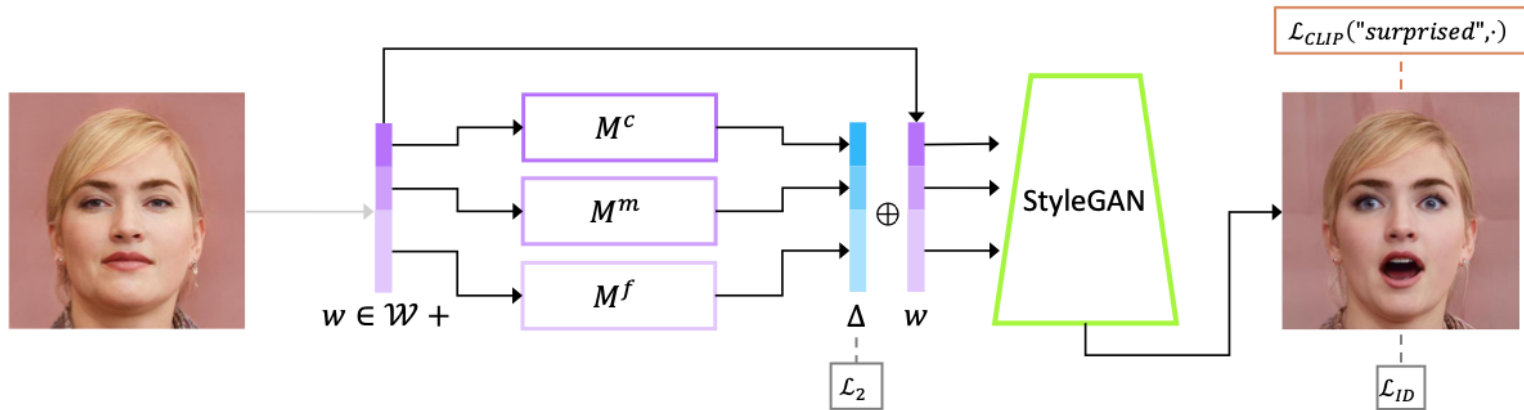


- Recent advances with StyleGAN [Yang et al., 2022]
 - Image manipulation via latent space manipulation in StyleGAN
 - Given a style image, fine-tune StyleGAN submodules



• Text-driven Image manipulation

- StyleCLIP [Patashnik et al., 2021] manipulates latent variables to satisfy that the generated image has a similar embedding vector to the given text prompt
 - **Note.** CLIP [Radford et al., 2021] learns a multi-modal (image & text) embedding space



"Emma Stone"

"Mohawk hairstyle"

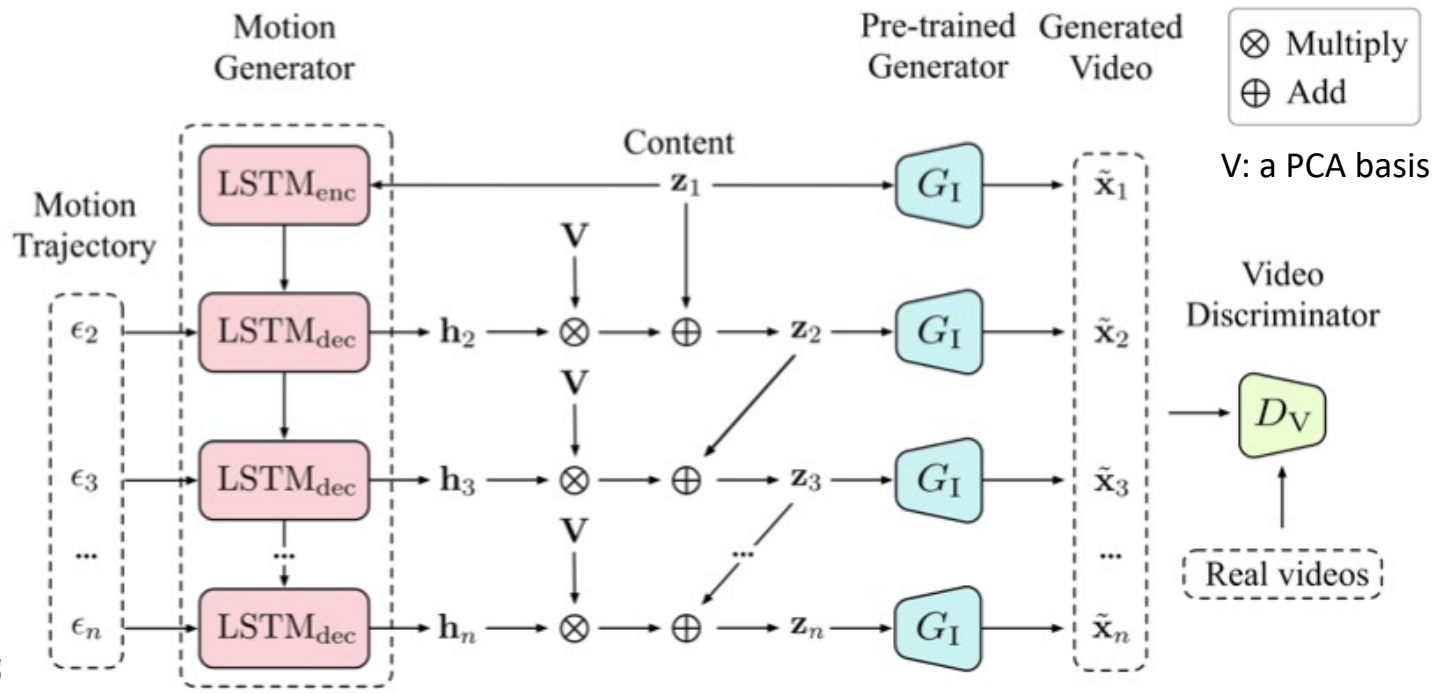
"Without makeup"

"Cute cat"

"Lion"

"Gothic church"

- **Video generation** has emerged as the next challenge of deep generative models
 - Why difficult? It is continuously correlated across spatial-temporal directions
 - A large gap still exists between generated results and large-scale real-world videos
- Motion and Content decomposed GAN (**MoCoGAN**) [Tian et al., 2021]
 - For high quality for each individual frame, use a **pre-trained** and **fixed image** generator
 - For temporal consistency, generates a sequence of latent codes
 - **Note.** Each latent code is decomposed by motion & content vectors

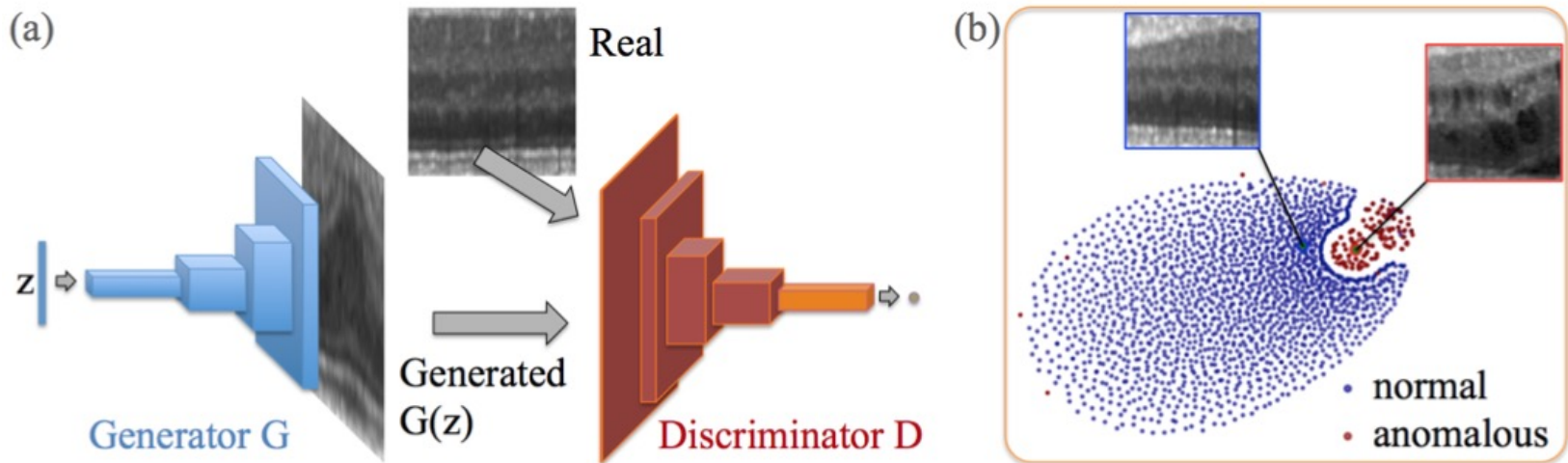


- **Video generation** has emerged as the next challenge of deep generative models
 - Why difficult? It is continuously correlated across spatial-temporal directions
 - A large gap still exists between generated results and large-scale real-world videos
- Motion and Content decomposed GAN (**MoCoGAN**) [Tian et al., 2021]

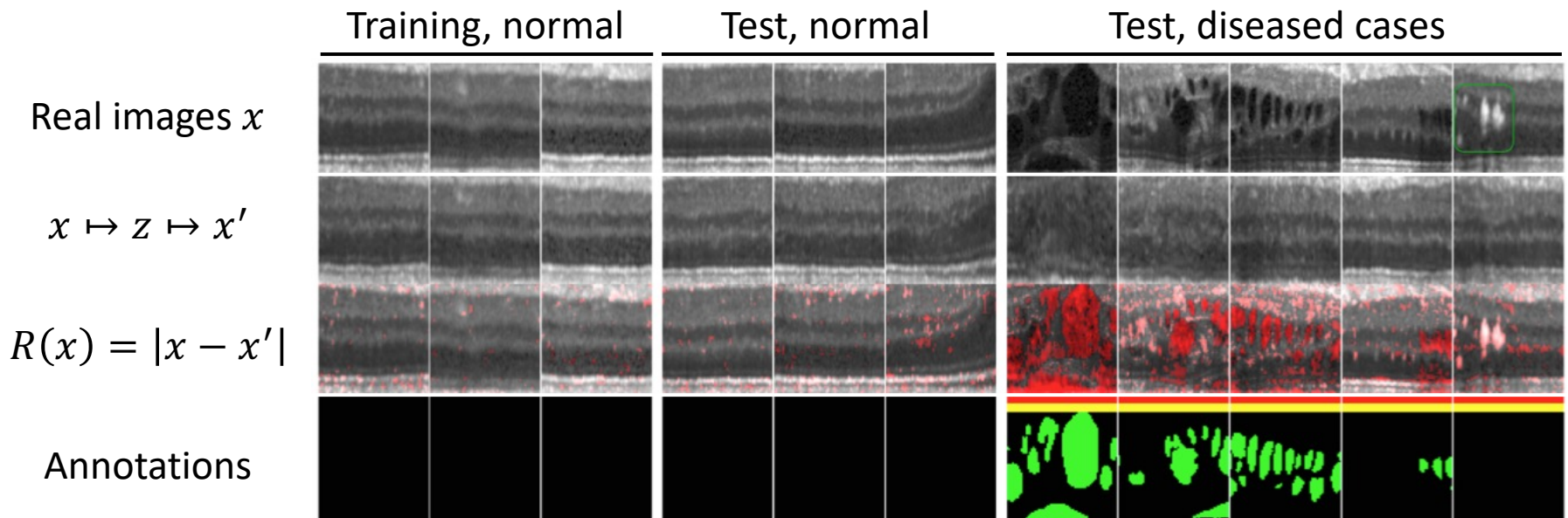


- A better latent space for videos?
 - **Implicit neural representations (INRs)** is a new paradigm for representing continuous signals, e.g., videos. We will learn INR in Lecture 8!

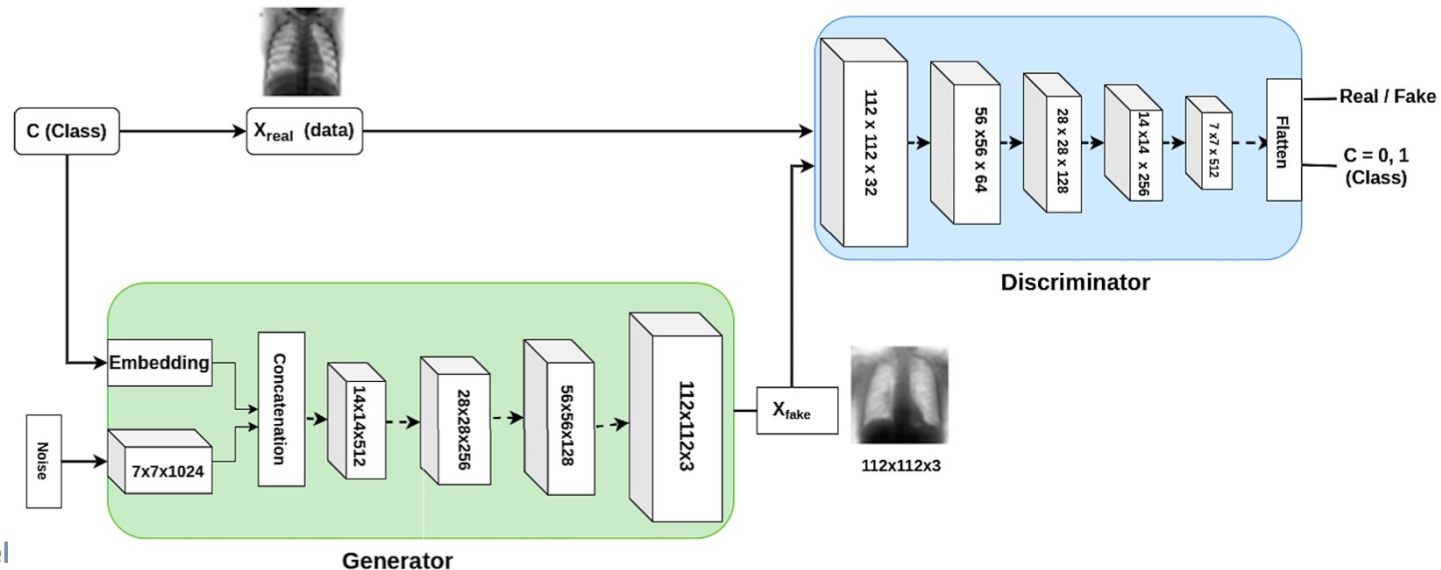
- Difficulties of training DNNs for **medical diagnosis**
 - It requires expert knowledge to collect labels of medical images
 - Designing an augmentation strategy for medical images is non-trivial
- GAN frameworks have several advantages
 - GANs can **discriminate between normal and anomalous** samples
 - AnoGAN [Schlegl et al., 2017] learns $x \mapsto z$ via minimizing $R(x) = \min_z |x - G(z)|$
 - Detection score = $(1 - \lambda)R(x) + \lambda D(x)$



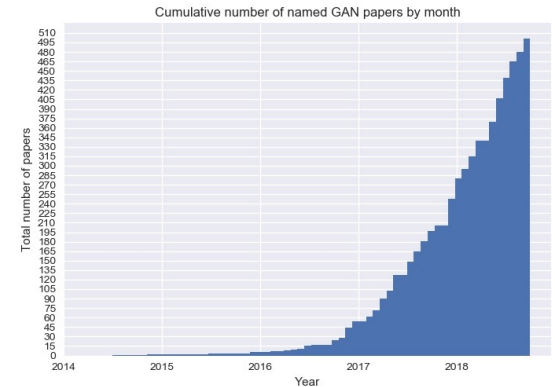
- Difficulties of training DNNs for **medical diagnosis**
 - It requires expert knowledge to collect labels of medical images
 - Designing an augmentation strategy for medical images is non-trivial
- GAN frameworks have several advantages
 - GANs can **discriminate between normal and anomalous** samples
 - AnoGAN [Schlegl et al., 2017] learns $x \mapsto z$ via minimizing $R(x) = \min_z |x - G(z)|$
 - Detection score = $(1 - \lambda)R(x) + \lambda D(x)$



- Difficulties of training DNNs for **medical diagnosis**
 - It requires expert knowledge to collect labels of medical images
 - Designing an augmentation strategy for medical images is non-trivial
- GAN frameworks have several advantages
 - GANs can **discriminate between normal and anomalous** samples
 - AnoGAN [Schlegl et al., 2017] learns $x \mapsto z$ via minimizing $R(x) = \min_z |x - G(z)|$
 - Detection score = $(1 - \lambda)R(x) + \lambda D(x)$
 - GANs can **create new, realistic samples** useful for representation learning
 - CovidGAN [Waheed et al., 2020] uses ACGAN for generating chest X-ray images



- **GAN has been one of the prominent topic in deep learning since 2014**
- Thousands of papers about GAN:
 - Theoretical aspects of GANs
 - Stabilizing GAN training dynamics
 - Applications of GAN to various AI tasks
 - ... and many more
- **GANs are especially good at generating “high-precision” samples**
 - Achieving “high-recall”, however, is still challenging
 - Lots of improvement in loss, regularization, and architecture have been made
 - Some large-scale studies have revealed sober views on them, nevertheless
- **Recently, the GAN framework has been incorporated with other techniques**
 - Better generation with diffusion models (we’ll learn in Lecture 6)
 - Better representation learning (= self-supervised learning) w/ strong augmentations (we’ll learn in Lecture 7)
 - Better latent space, implicit neural representations (INR; we’ll learn in Lecture 8)



References

[Shannon, 1949] Communication in the presence of noise, Proc. Institute of Radio Engineers, 37(1):10–21, 1949

[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>

[Villani, 2009] Optimal transport: old and new, Grundlehren der mathematischen wissenschaften 2009

link: <http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf>

[Goodfellow, et. al., 2014] Generative adversarial nets, NIPS 2014

link: <http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf>

[Mirza et al., 2014] Conditional Generative Adversarial Nets

link: <https://arxiv.org/pdf/1411.1784.pdf>

[Huszar 2015] How (not) to train your generative model: scheduled sampling, likelihood, adversary?

link: <https://arxiv.org/pdf/1511.05101.pdf>

[Ioffe & Szegedy, 2015] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, ICML 2015

link: <https://arxiv.org/abs/1502.03167>

[Radford, et. al., 2015] Unsupervised representation learning with deep convolutional generative adversarial networks, 2015

link: <https://arxiv.org/pdf/1511.06434.pdf>

[Ba et al., 2016] Layer Normalization, 2016

link: <https://arxiv.org/abs/1607.06450>

References

[Reed, et. al., 2016] Generative adversarial text to image synthesis, ICML 2016

link: <https://arxiv.org/pdf/1605.05396.pdf>

[Salimans, et. al., 2016] Improved techniques for training GANS, NIPS 2016

link: <https://arxiv.org/abs/1606.03498>

[Theis, et. al., 2016] A note on the evaluation of generative models, ICLR 2016

link: <http://bethgelab.org/media/publications/1511.01844v1.pdf>

[Arjovsky, et. al., 2017] Wasserstein GAN, ICML 2017

link: <https://arxiv.org/pdf/1701.07875.pdf>

[Arjovsdy and Bottou, 2017] Towards principled methods for training generative adversarial networks, ICLR 2017

link: <https://arxiv.org/pdf/1701.04862.pdf>

[Basart et. al., 2017] Analysis of Generative Adversarial Models

link: https://newtraell.cs.uchicago.edu/files/ms_paper/xksteven.pdf

[DeVries, et. al., 2017] Modulating early visual processing by language, NIPS 2017

link: <https://arxiv.org/pdf/1707.00683.pdf>

[Dumoulin, et. al., 2017] A Learned Representation for Artistic Style, ICLR 2017

link: <https://arxiv.org/pdf/1610.07629.pdf>

[Gulrajani, et. al., 2017] Improved training of Wasserstein GANs, NIPS 2017

link: <https://arxiv.org/pdf/1704.00028.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>

References

[Kodali et al., 2017] On Convergence and Stability of GANs, 2018

link: <https://arxiv.org/abs/1705.07215v5>

[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

[Mao et al., 2017] Least Squares Generative Adversarial Networks, 2016

link: <https://arxiv.org/abs/1611.04076>

[Odena, et. al., 2017] Conditional Image Synthesis with Auxiliary Classifier GANs, ICML 2017

link: <https://arxiv.org/pdf/1610.09585.pdf>

[Schlegl et al., 2017] Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery, IPMI 2017

link: <https://arxiv.org/abs/1703.05921>

[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>

[Lucic et al., 2018] Are GANs Created Equal? A Large-Scale Study, NIPS 2018

link: <https://arxiv.org/abs/1711.10337>

[Miyato, et. al., 2018] cGANs with Projection Discriminator, ICLR 2018

link: <https://arxiv.org/pdf/1802.05637.pdf>

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

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

References

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

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

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

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

[Kurach et al., 2019] A Large-Scale Study on Regularization and Normalization in GANs, ICML 2019

link: <http://proceedings.mlr.press/v97/kurach19a.html>

[Wang et al., 2019] Self-Attention Generative Adversarial Networks, ICML 2019

link: <https://arxiv.org/pdf/1805.08318.pdf>

[Zhang, et. al., 2019] Self-Attention Generative Adversarial Networks, ICML 2019

link: <https://arxiv.org/pdf/1805.08318.pdf>

[Zhang et al., 2020] Consistency Regularization for Generative Adversarial Networks, ICLR 2020

link: <https://arxiv.org/abs/1910.12027>

[Karras et al., 2020a] Analyzing and Improving the Image Quality of StyleGAN, CVPR 2020

link: <https://arxiv.org/abs/1912.04958>

[Karras et al., 2020b] Training Generative Adversarial Networks with Limited Data, NeurIPS 2020

link: <https://arxiv.org/abs/2006.06676>

References

[Waheed et al., 2020] CovidGAN: Data Augmentation Using Auxiliary Classifier GAN for Improved Covid-19 Detection, IEEE Access

link: <https://ieeexplore.ieee.org/document/9093842>

[Zhao et al., 2020a] Improved Consistency Regularization for GANs, 2020

link: <https://arxiv.org/abs/2002.04724>

[Zhao et al., 2020b] Differentiable Augmentation for Data-Efficient GAN Training, NeurIPS 2020

link: <https://arxiv.org/abs/2006.10738>

[Karras et al., 2021] Alias-Free Generative Adversarial Networks, NeurIPS 2021

link: <https://arxiv.org/pdf/2106.12423.pdf>

[Kim et al., 2021] The Lipschitz Constant of Self-Attention, ICML 2021

link: <https://arxiv.org/pdf/2006.04710.pdf>

[Patashnik et al., 2021] StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery, ICCV 2021

link: <https://arxiv.org/abs/2103.17249>

[Radford et al., 2021] Learning transferable visual models from natural language supervision, 2021

link: <https://arxiv.org/abs/2103.00020>

[Sauer et al., 2021] Projected GANs Converge Faster, NeurIPS 2021

link: <https://arxiv.org/abs/2111.01007>

[Tian et al., 2021] A Good Image Generator Is What You Need for High-Resolution Video Synthesis, ICLR 2021

link: <https://arxiv.org/abs/2104.15069>

[Lee et al., 2022] ViTGAN: Training GANs with Vision Transformers, ICLR 2022

link: <https://openreview.net/pdf?id=dwg5rXg1WS>

References

[Yang et al., 2022] Pastiche Master: Exemplar-Based High-Resolution Portrait Style Transfer, CVPR 2022

link: <https://arxiv.org/abs/2203.13248>

[Sauer et al., 2022] StyleGAN-XL: Scaling StyleGAN to Large Diverse Datasets, 2022

link: <https://arxiv.org/abs/2202.00273>