Generative Adversarial Network

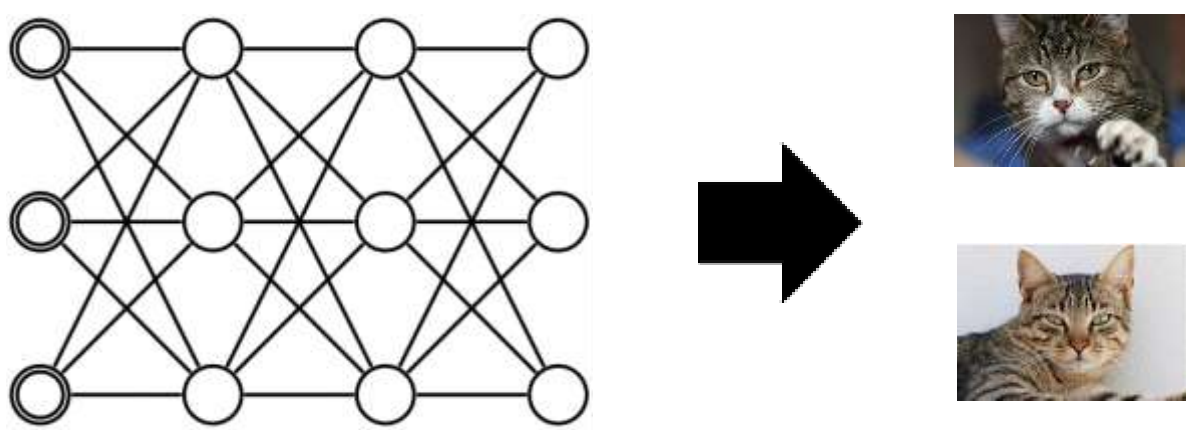
성균관대학교 소프트웨어학과
이 지 형

Why Generative Models?

- We've only seen discriminative models so far
  - Given an image X, predict a label Y
  - Estimates P(Y|X)
- Discriminative models have several key limitations
  - Can't model P(X), i.e. the probability of seeing a certain image
  - Thus, can't sample from P(X), i.e. can't generate new images
- Generative models (in general) cope with all of above
  - Can model P(X)
  - Can generate new images

Why Generative Models?

- Generative models model a full probability distribution of given data
- $P(x)$ enables us to generate new data similar to existing (training) data
- Sampling methods are required for generation



Why Generative Model?

- **Generate new samples from the same distribution with training data**
  - **Vision: super-resolution, style transfer, image inpainting, etc.**
  - **Audio: synthesizing audio, speech generation, voice conversion, etc.**

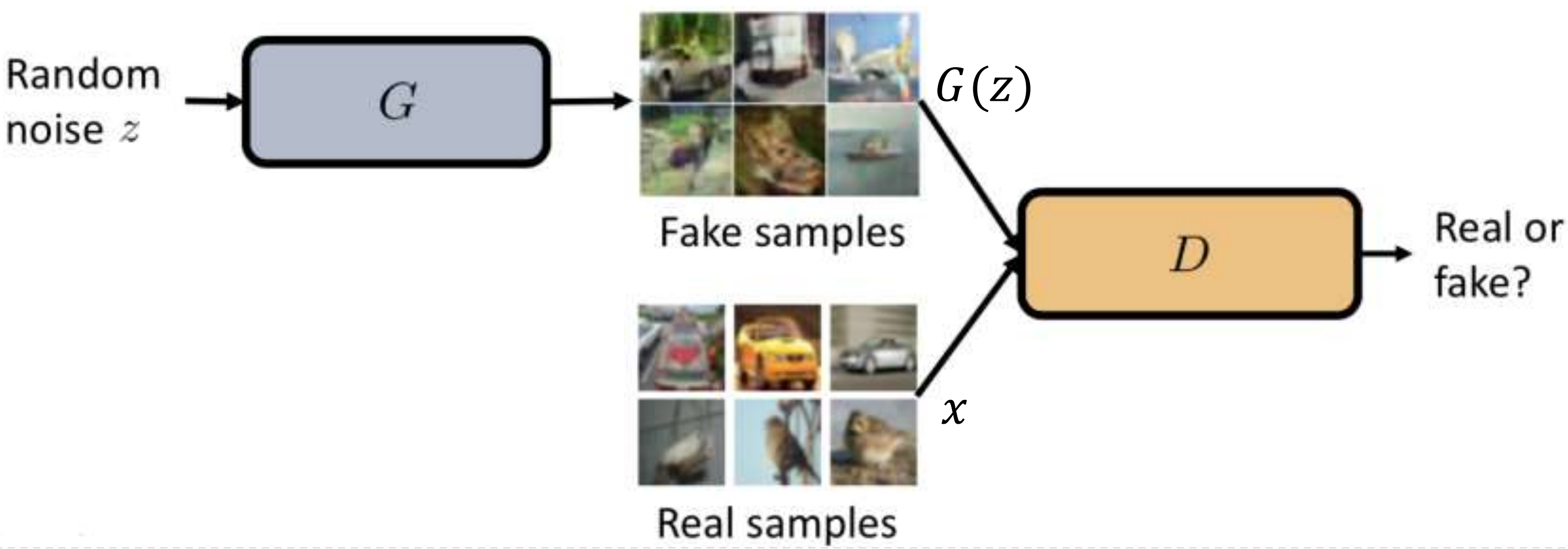
Super-resolution [Ledig, et. al., 2017]

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

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

GAN

- **Two player game between discriminator network and generator network**
  - *D* tries to distinguish real data and samples generated by (fake samples)
  - *G* tries to fool the *D* by generating real-looking images

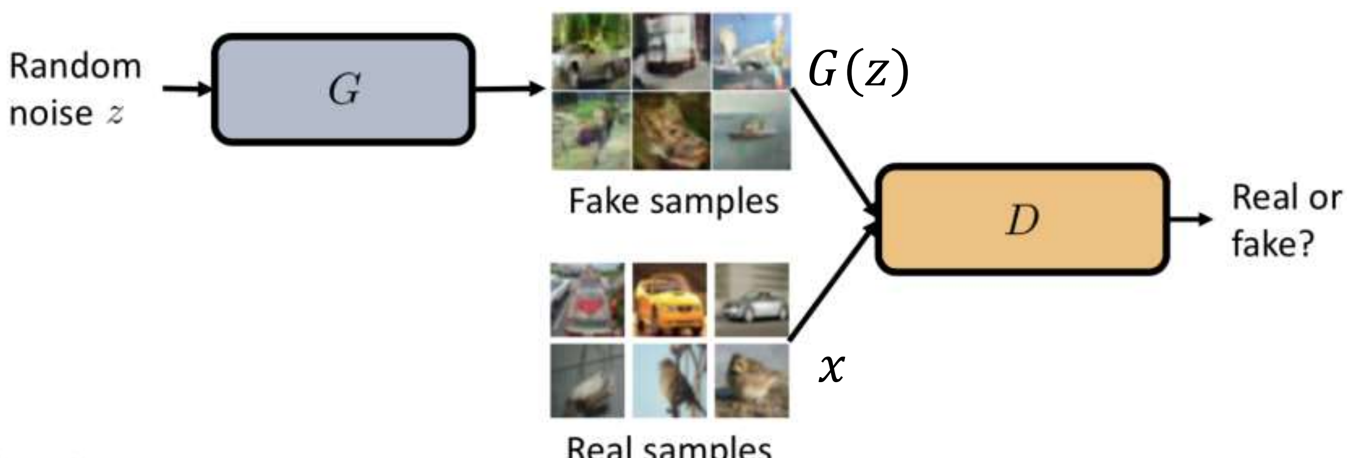


GAN

Objective Function

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

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



GAN

- **Training**

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

  - **Alternative training between *D* and *G***
    - For D

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

    - For G

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

GAN

- **Optimal Strategy of Discriminator**
  - **For fixed *G*, the D minimizes:**

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

  - **Optimal discriminator**

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

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

GAN

Training algorithm

for number of training iterations **do**

for k steps **do**

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

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

end for

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

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

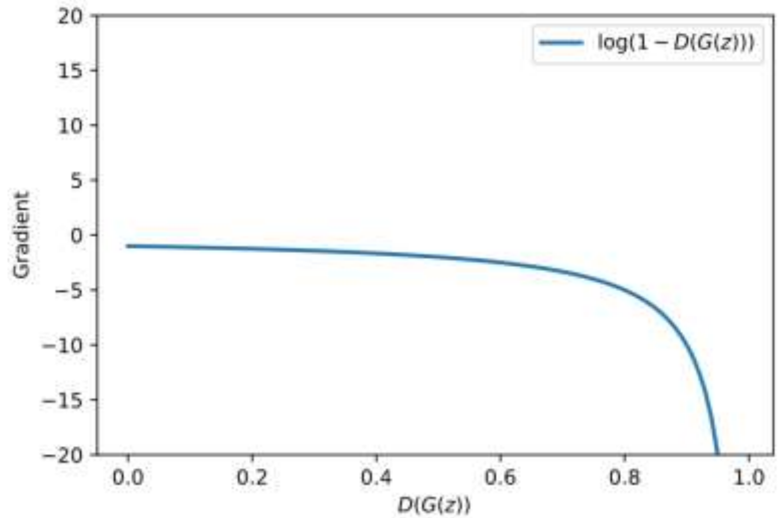
end for

GAN

- **Problem in Training Generator**

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

  - **Optimizing generator objective does not work well**
  - **When generated sample looks bad (at the beginning of training) gradient is relatively flat**

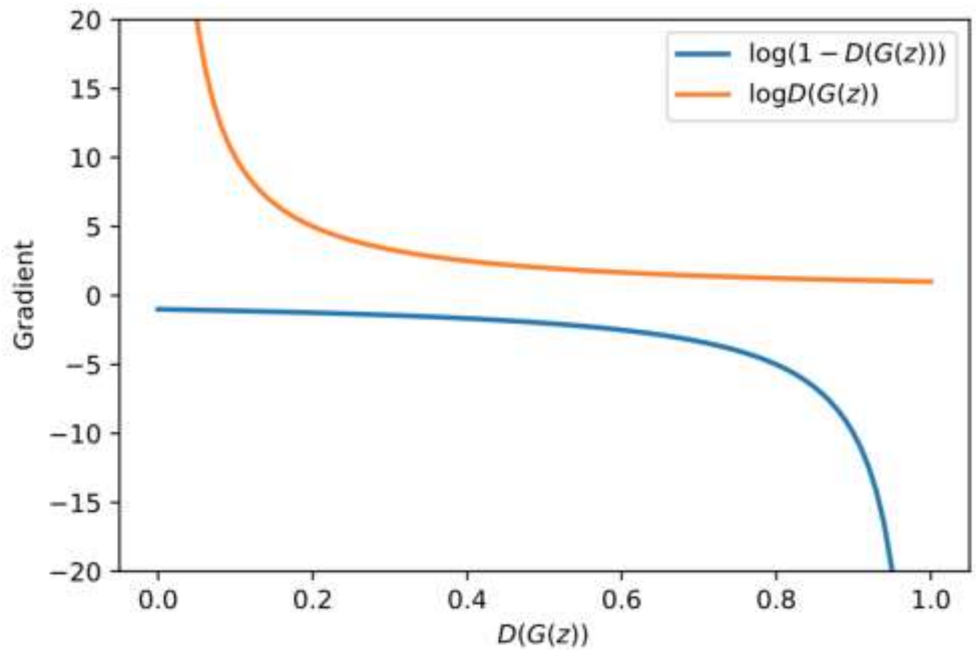


성균관대학교

GAN

- **Problem in Training Generator**

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



Advantage of GAN

- Sampling (or generation) is straightforward.
- Training doesn't involve Maximum Likelihood estimation.
- Robust to Overfitting since Generator never sees the training data.
- Empirically, GANs are good at capturing the modes of the distribution.

Issue of GAN

- **Hard to achieve Nash equilibrium (Optimal Solution)**
  - **GANs involve two (or more) players**
    - Discriminator is trying to maximize its reward.
    - Generator is trying to minimize Discriminator's reward.

$$\min_G \max_D V(D, G)$$

  - **SGD was not designed to find the Nash equilibrium of a game**
  - **Problem:**
    - We might not converge to the Nash equilibrium at all.

Issue of GAN

- Hard to achieve Nash equilibrium (Optimal Solution)
  - Each model updates its own objective function
  - Modification θ_d that reduces D's objective can increase G's and vice versa

- Mode collapse
  - Generator is easy to produce the same outputs
  - It is one of easiest way to fool the discriminator

Examples of mode collapse in GAN.

Issue of GAN

- **Hard to achieve Nash equilibrium (Optimal Solution)**

$$\min_{x} \max_{y} V(x, y)$$

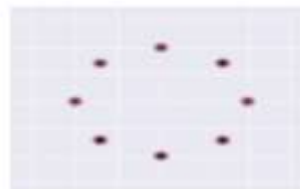
$$\text{Let} \quad V(x, y) = xy$$

State 1:	x > 0	y > 0	V > 0		Increase y	Decrease x
State 2:	x < 0	y > 0	V < 0		Decrease y	Decrease x
State 3:	x < 0	y < 0	V > 0		Decrease y	Increase x
State 4:	x > 0	y < 0	V < 0		Increase y	Increase x
State 5:	x > 0	y > 0	V > 0	== State 1	Increase y	Decrease x

Issue of GAN

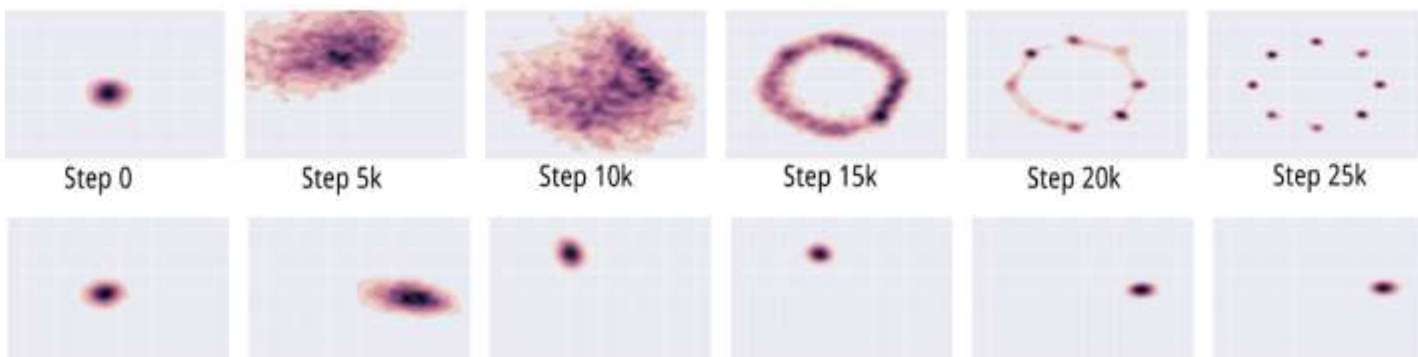
- Mode-Collapse
  - Generator fails to output diverse samples

Target



Expected

Output



Issue of GAN

- **Mode-Collapse**
  - Objective of Generator

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

  - For optimal discriminator

$$\mathbb{E}_{z\sim p_z}\left[-\log(D_{\theta_d^*}(G_{\theta_g}(z)))\right]$$

$$= \underline{KL(p_g \parallel p_{\text{data}}) - 2JS(p_{\text{data}} \parallel p_g)}$$

Issue of GAN

- **KL divergence**

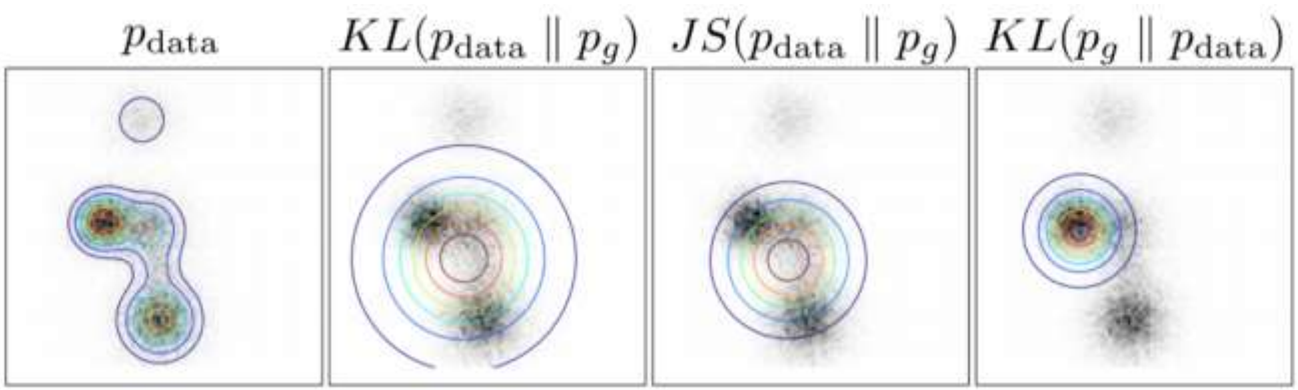
$$KL(p_{\text{data}} \| p_g) = \int_x p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_g(x)} dx$$

- **Jensen-Shannon divergence**

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

Help to generate sharp image

Allow mode-collapse



Improving Techniques

- **Feature matching**
  - **Instead of directly maximizing the output of the *D*, make *G* to generate data that matches features of the real data**
  - **Loss of generator becomes:**

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

$$\Rightarrow \quad \min_{\theta_g} \|\mathbb{E}_{x\sim p_{\text{data}}} f(x) - \mathbb{E}_{z\sim p_z} f(G(z))\|$$

    - where *f* is activations of an intermediate layer of *D*
  - ***D*'s loss remains the same with original GAN's discriminator loss**
  - **Effective when the GAN model is unstable during training.**

Improving Techniques

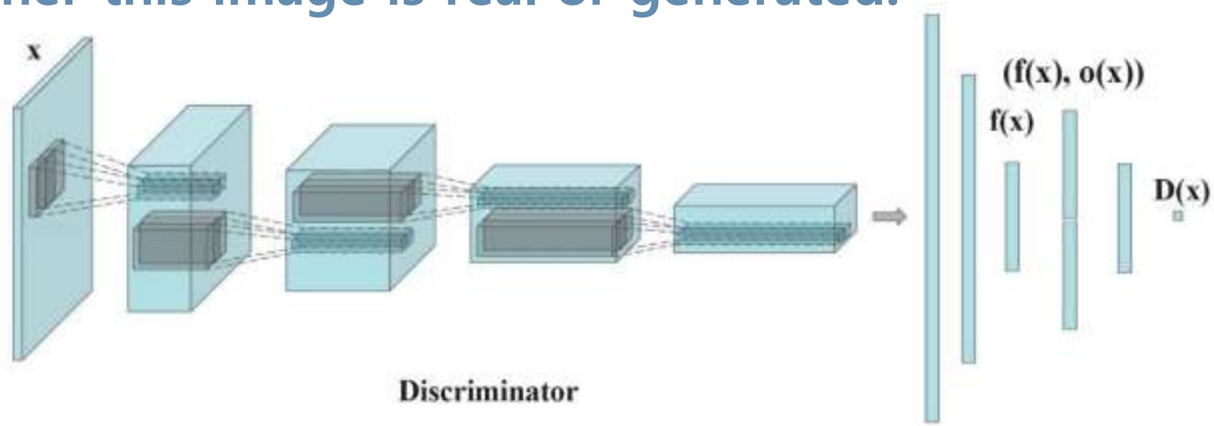
- **Historical averaging**
  - Keep track of the model parameters for the last *t* models.
  - Add additional loss term to penalize model different from the historical average.

$$\left\| \theta - \frac{1}{t} \sum_{i=1}^{t} \theta_i \right\|^2$$

  - For GANs with non-convex object function, historical averaging may stop models circle around the equilibrium point and act as a damping force to converge the model.

Improving Techniques

- **Minibatch Discrimination**
  - When mode collapses, all images created looks similar.
  - Feed real images and generated images into the discriminator separately in different batches
  - Compute the similarity of the image *x* with images in the same batch.
  - Append the similarity *o(x)* in the discriminator to classify whether this image is real or generated.

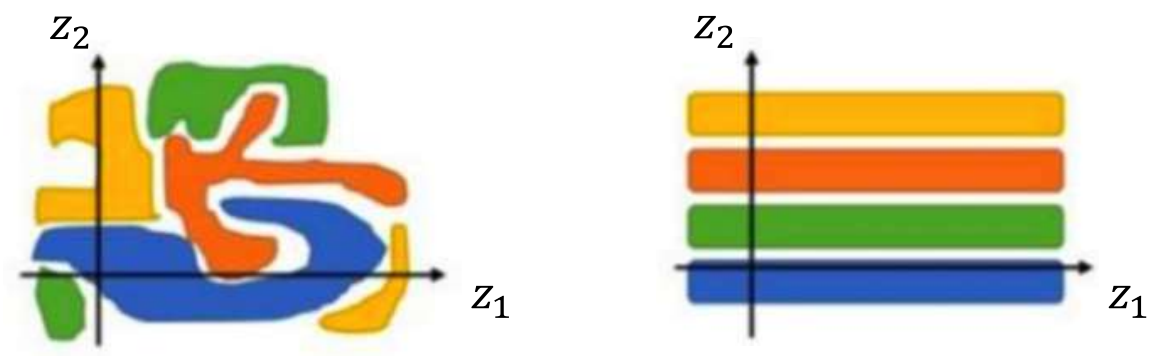


https://towardsdatascience.com/gan-ways-to-improve-gan-performance-acf37f9f59b

Various Types of GAN

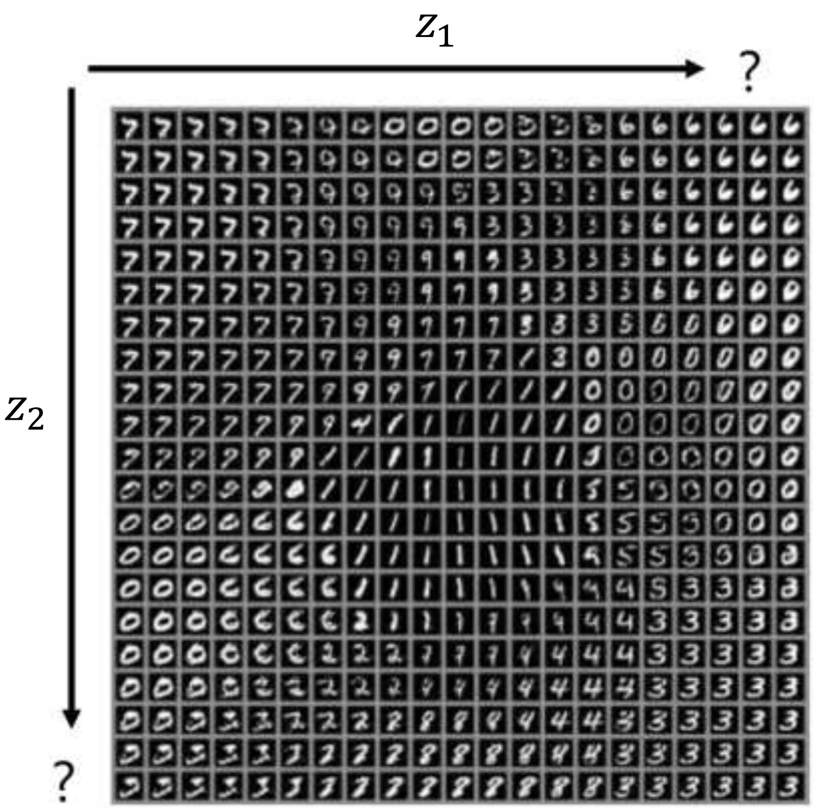
InfoGAN

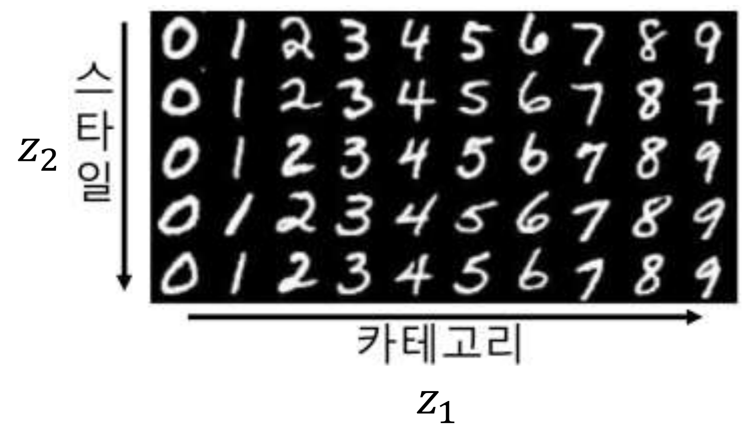
- **Entangled vs. Disentangled**
  - Can we interpret the meaning of z?
  - If we continuously change z,
    does the generated image semantically continuously changes?



InfoGAN

- **Entangled vs. Disentangled**



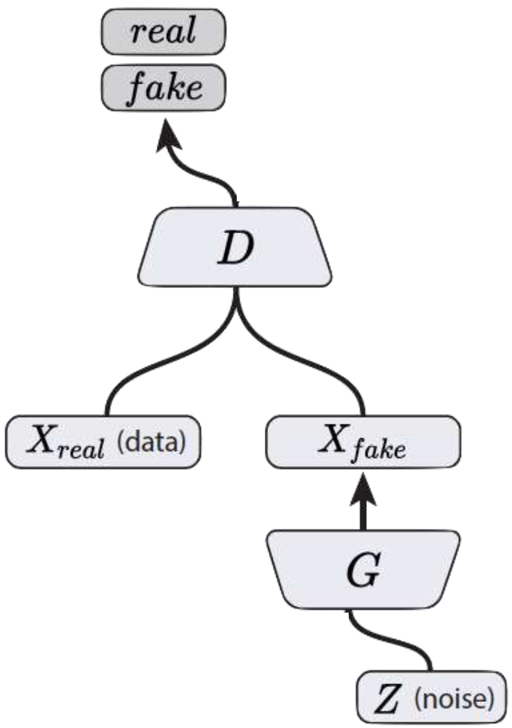


성균관대학교

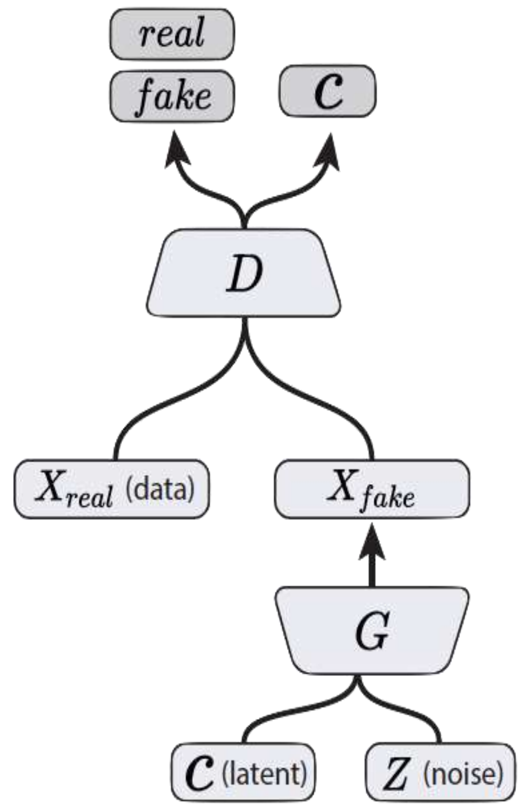
InfoGAN

- **How to reward Disentanglement?**
  - **Disentanglement means individual dimensions independently capturing key attributes of the image**
  - **Let's partition the noise vector into 2 parts :-**
    - z vector will capture slight variations in the image
    - c vector will capture the main attributes of the image
    - For e.g. Digit, Angle and Thickness of images in MNIST
  - **If c vector captures the key variations in the image, Will c and $\boldsymbol{x\_fake}$ be highly correlated or weakly correlated?**

InfoGAN



real
fake
D
X_{real} (data)
X_{fake}
G
Z (noise)



real
fake
C
D
X_{real} (data)
X_{fake}
G
C (latent)
Z (noise)

InfoGAN

- **We want to maximize the mutual information I between c and $\mathbf{x}=G(z,c)$**

$$I(X;Y) = \sum_{x,y} p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right)$$

$$I(X;Y) = H(X) - H(X|Y) = \mathrm{H(Y)} - \mathrm{H(Y|X)}$$

- **Incorporate in the value function of the minimax game.**

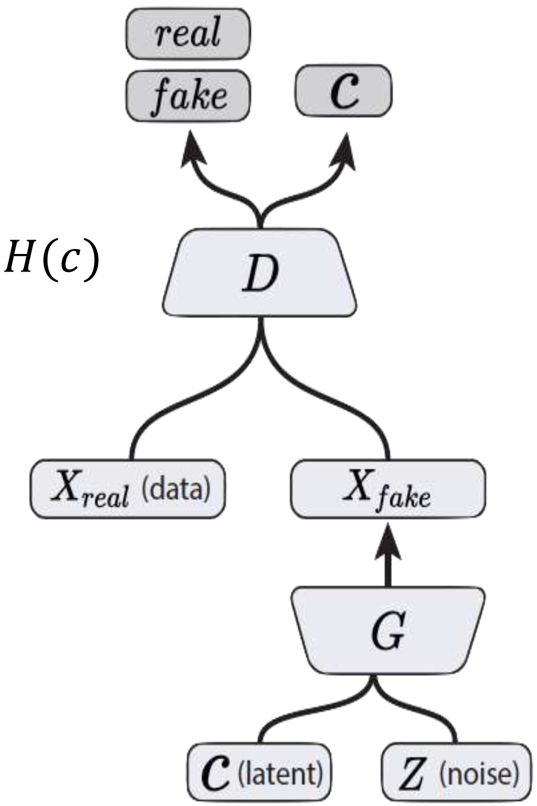
$$\min_G \max_D V_I(D,G) = V(D,G) \boxed{- \lambda\, I\big(c; G(z,c)\big)}$$

InfoGAN

Mutual Information's Variational Lower bound

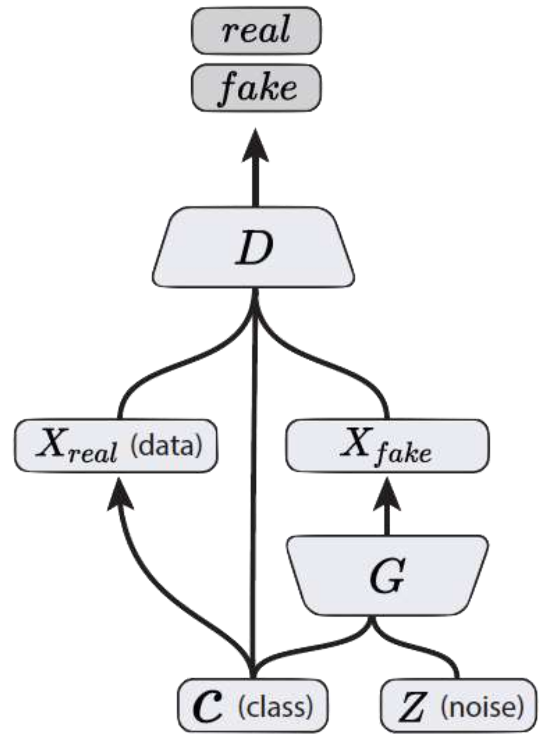
$$I\big(c; G(z,c)\big) = H(c) - H(c|G(z,c))$$
$$= \mathbb{E}_{x \sim G(z,c)}\Big[\mathbb{E}_{c' \sim P(c|x)}[\log P(c'|x)]\Big] + H(c)$$
$$= \mathbb{E}_{x \sim G(z,c)}\Big[D_{KL}(P||Q) + \mathbb{E}_{c' \sim P(c|x)}[\log Q(c'|x)]\Big] + H(c)$$
$$\geq \mathbb{E}_{x \sim G(z,c)}\Big[\mathbb{E}_{c' \sim P(c|x)}[\log Q(c'|x)]\Big] + H(c)$$

$$\geq \mathbb{E}_{c \sim P(c),\ x \sim G(z,c)}[\log \boldsymbol{Q}(\boldsymbol{c}|\boldsymbol{x})] + H(c)$$



Conditional GAN

- Simple modification to the original GAN framework that conditions the model on additional information for better multi-modal learning.
- Lends to many practical applications of GANs when we have explicit supervision available.
- Provide an effective way to handle many complex domains without worrying about designing structured loss functions explicitly.



Conditional GAN
(Mirza & Osindero, 2014)

Image-to-Image Translation

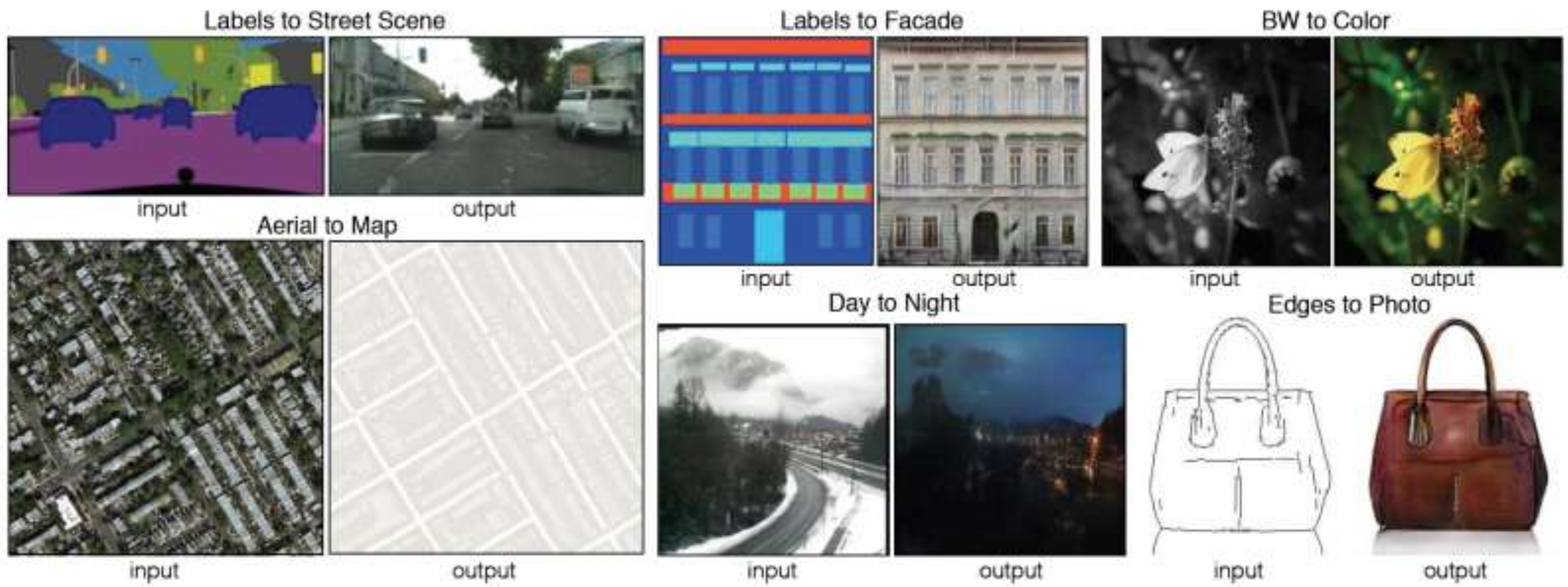
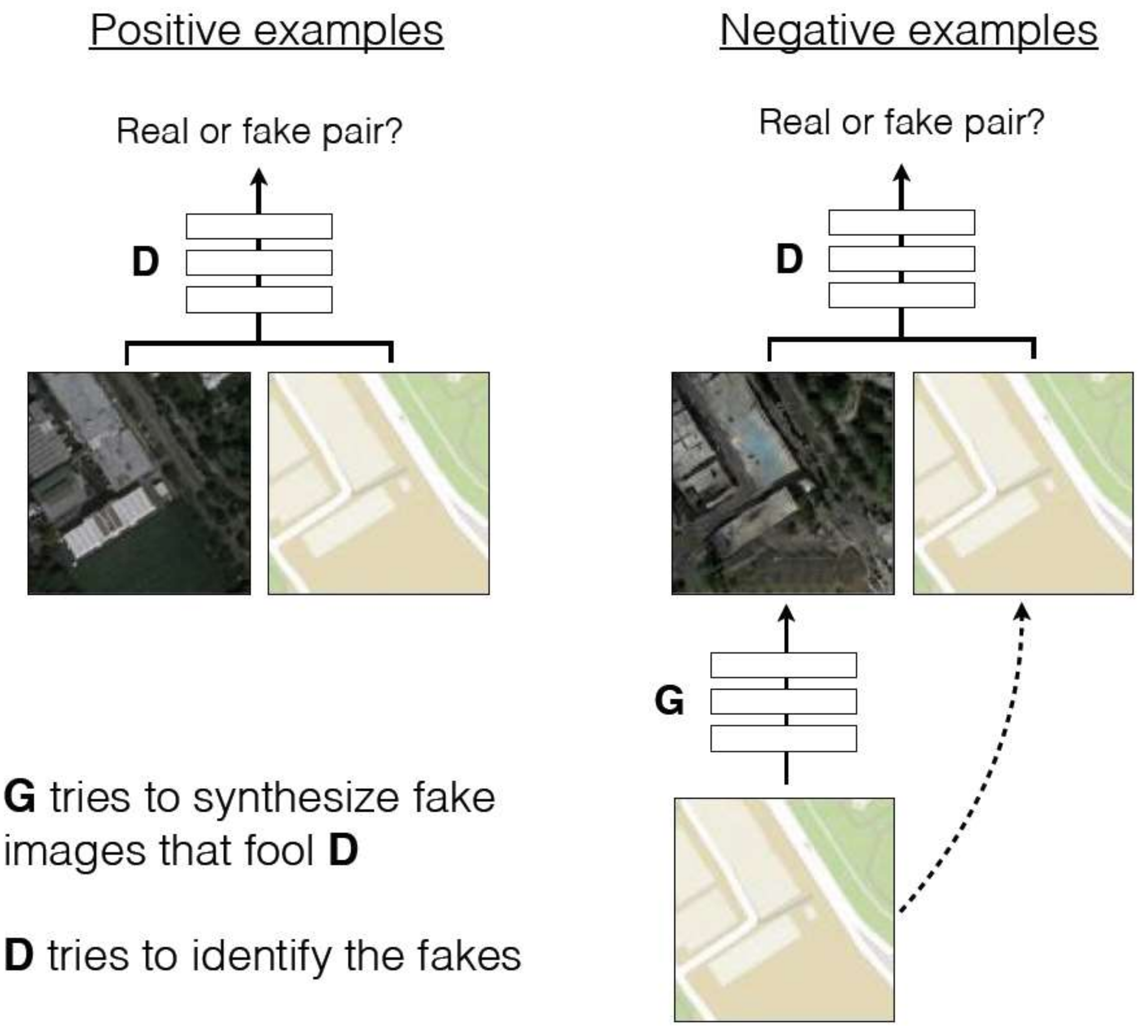


Image-to-Image Translation

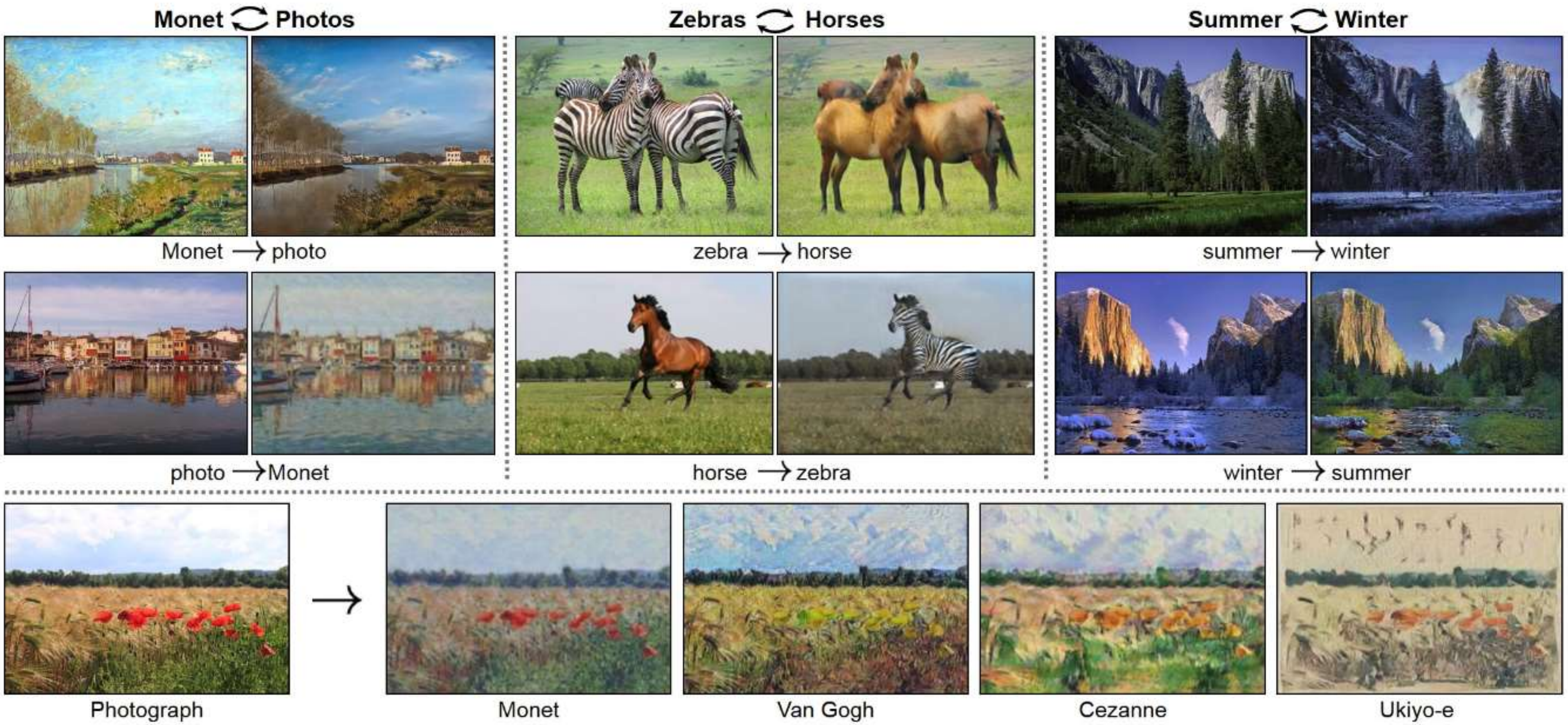


G tries to synthesize fake images that fool **D**

D tries to identify the fakes

CycleGAN

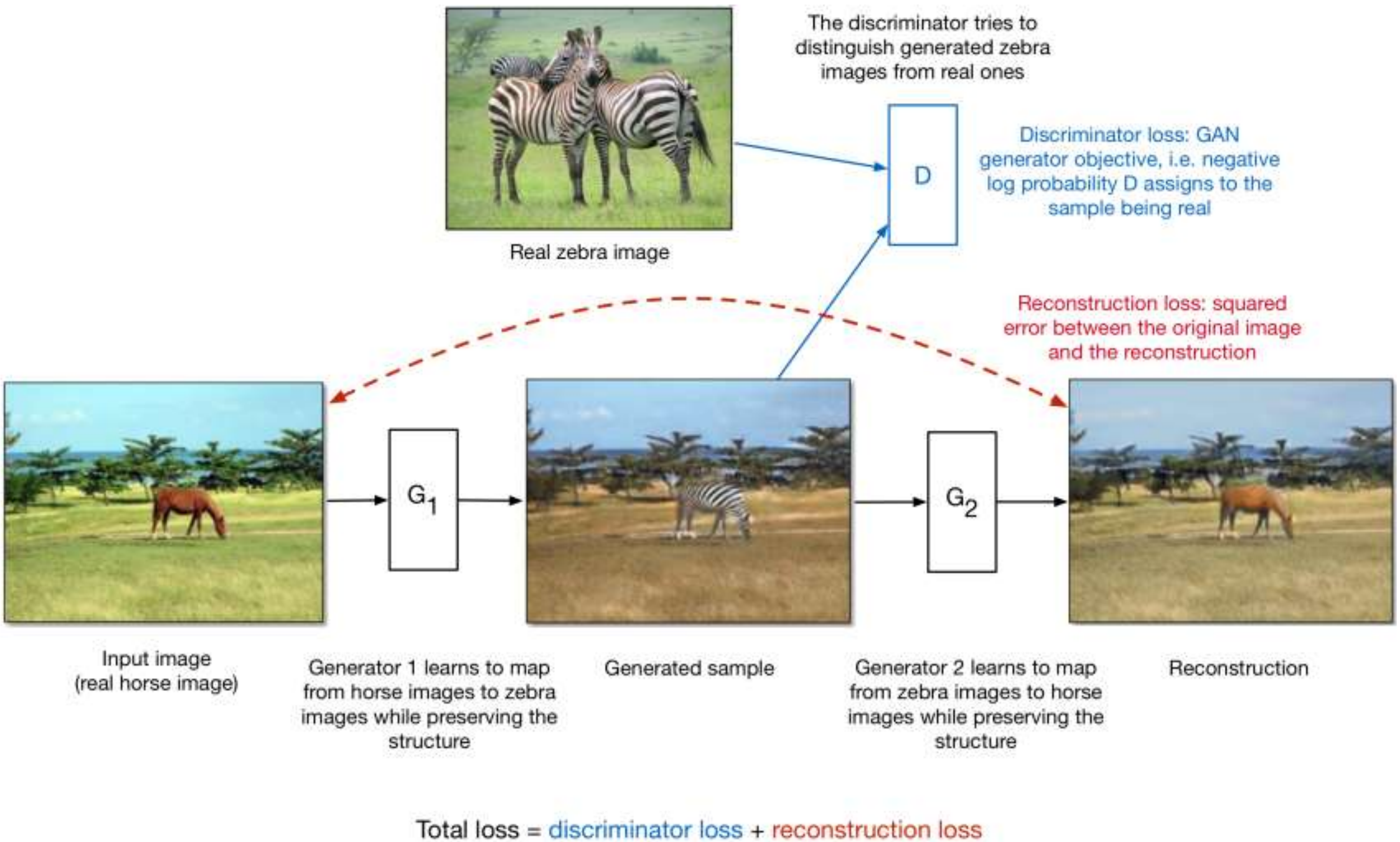
- **Style transfer problem**
  - change the style of an image while preserving the content.



CycleGAN

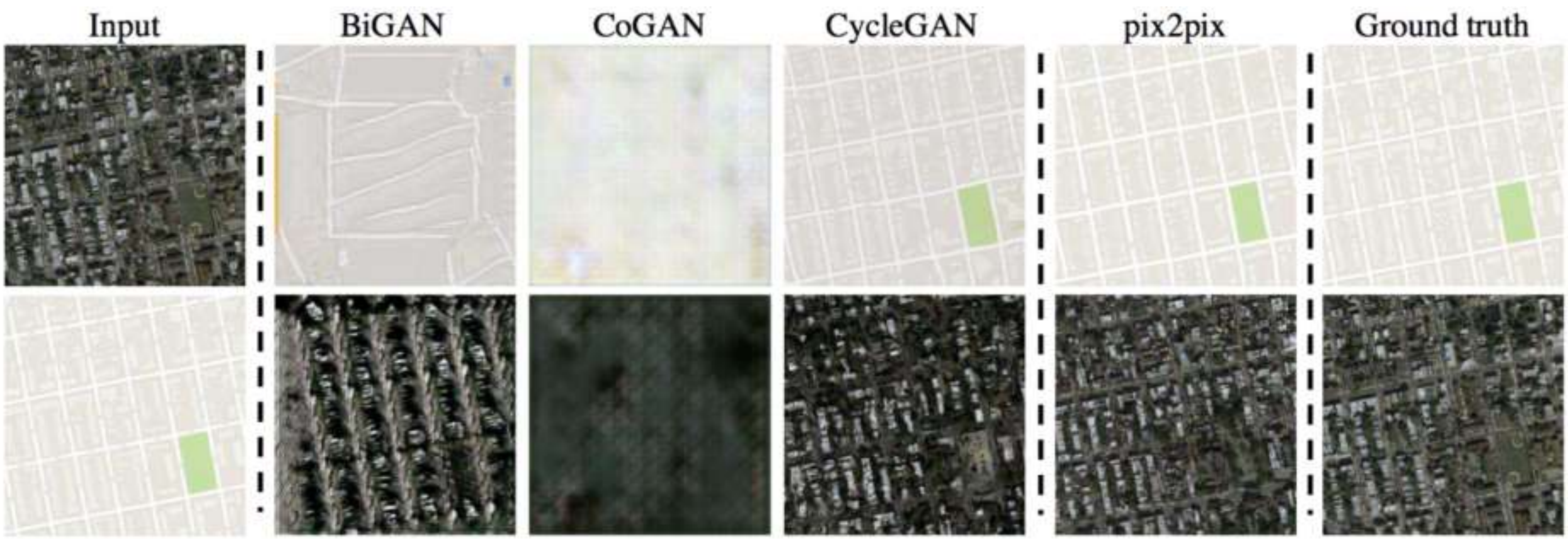
- If we had paired data (same content in both styles), this would be a supervised learning problem. But this is hard to find.
- The CycleGAN architecture learns to do it from unpaired data.
  - Train two different generator nets to go from style 1 to style 2, and vice versa.
  - Make sure the generated samples of style 2 are indistinguishable from real images by a discriminator net.
  - Make sure the generators are cycle-consistent: mapping from style 1 to style 2 and back again should give you almost the original image.

CycleGAN


The discriminator tries to distinguish generated zebra images from real ones
D
Discriminator loss: GAN generator objective, i.e. negative log probability D assigns to the sample being real
Real zebra image
Reconstruction loss: squared error between the original image and the reconstruction
G1
G2
Input image (real horse image)
Generator 1 learns to map from horse images to zebra images while preserving the structure
Generated sample
Generator 2 learns to map from zebra images to horse images while preserving the structure
Reconstruction
Total loss = discriminator loss + reconstruction loss

CycleGAN

- **Style transfer between aerial photos and maps**



Text-to-Image Synthesis

- **Motivation**
  - Given a text description, generate images closely associated.
  - Uses a conditional GAN with the generator and discriminator being condition on "dense" text embedding.

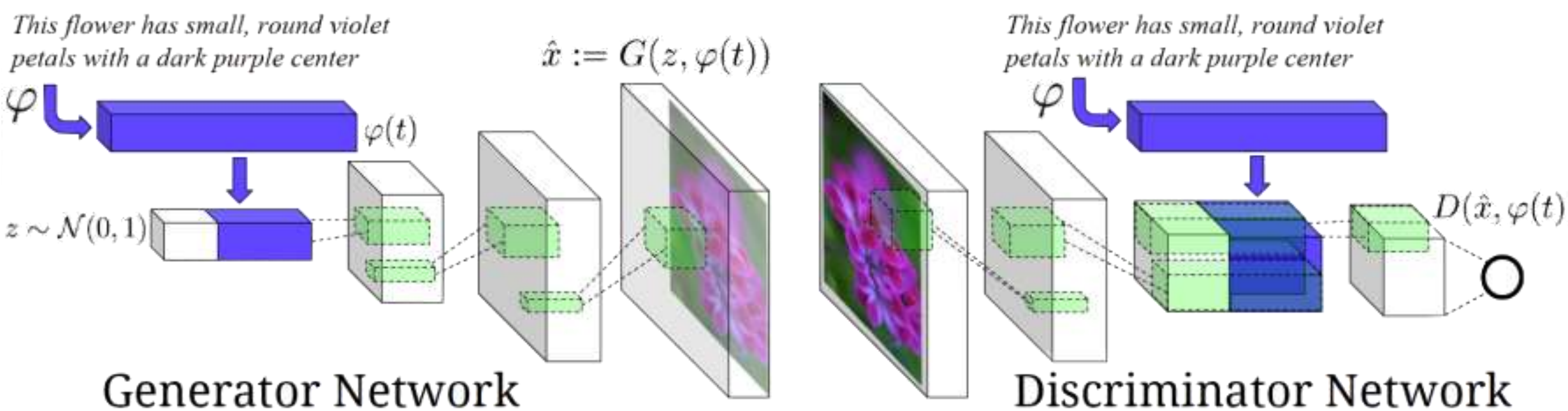
this small bird has a pink breast and crown, and black primaries and secondaries.

this magnificent fellow is almost all black with a red crest, and white cheek patch.

the flower has petals that are bright pinkish purple with white stigma

this white and yellow flower have thin white petals and a round yellow stamen

Text-to-Image Synthesis



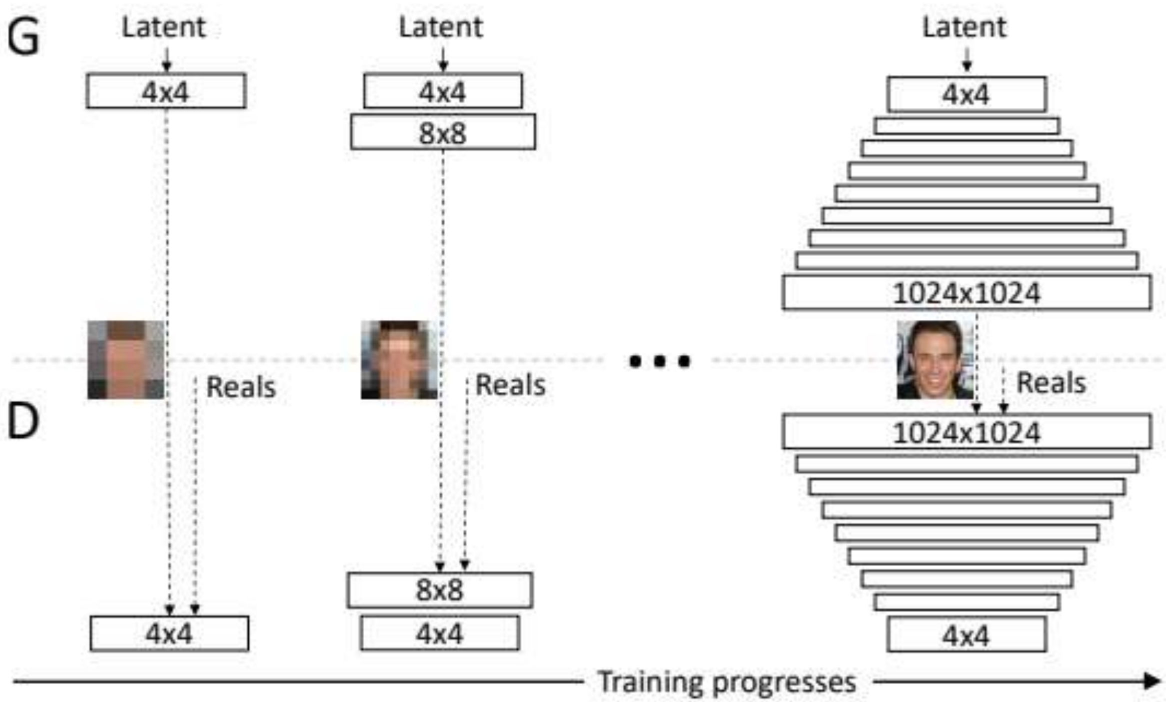
Positive Example:
(Real Image, Right Text)

Negative Examples:
(Real Image, Wrong Text)
(Fake Image, Right Text)

Progressive GAN (High-Res. Image Gen.)

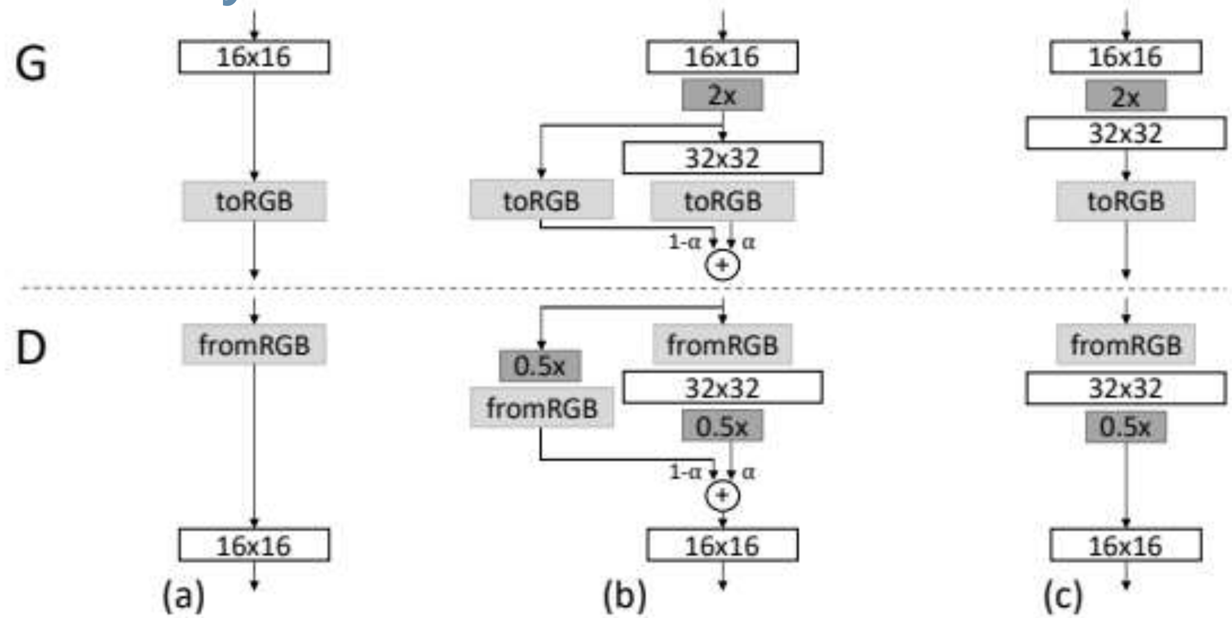
- GANs produce sharp images
  - But only in fairly small resolutions and with somewhat limited variation
- Training continues to be unstable despite recent progress
- Generating high resolution image is difficult
  - It is easier to tell the generated images from training images in high-res images [Karras,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

Progressive GAN (High-Res. Image Gen.)



Progressive GAN (High-Res. Image Gen.)

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



Transition from 16 × 16 images **(a)** to 32 × 32 images **(c)**. During the transition **(b)** we treat the layers that operate on the higher resolution like a residual block, whose weight α increases linearly from 0 to 1

Progressive GAN (High-Res. Image Gen.)

1024x1024 images generated using the CELEBA-HQ dataset

Why GAN?

- Can be trained using back-propagation for Neural Network based Generator/Discriminator functions.
- Sharper images can be generated.
- Faster to sample from the model distribution: single forward pass generates a single sample.

Summary

- GANs are generative models that are implemented using two stochastic neural network modules: Generator and Discriminator.
- Generator tries to generate samples from random noise as input
- Discriminator tries to distinguish the samples from Generator and samples from the real data distribution.
- Both networks are trained adversarially (in tandem) to fool the other component. In this process, both models become better at their respective tasks.

Question and Answer