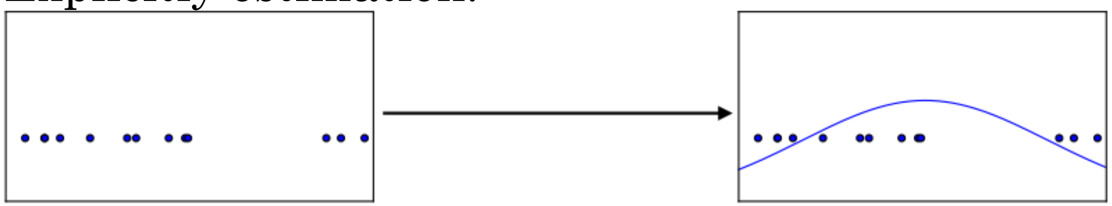
Generative Adversarial Networks (GAN)

Presented by Omer Stein and Moran Rubin

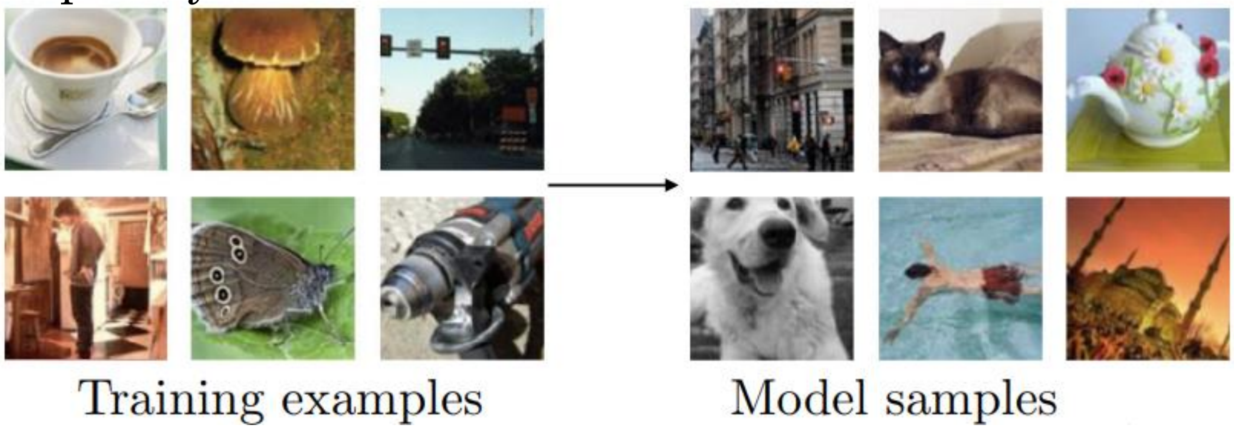
Generative Model

- Given a training dataset, x, try to estimate the distribution , Pdata(x)
- Explicitly or Implicitly (GAN)

Explicitly estimation:



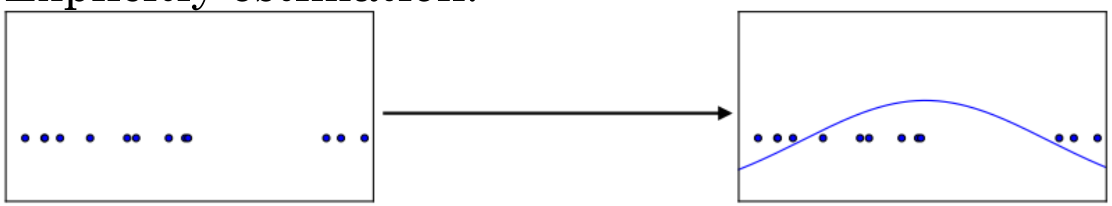
Implicitly estimation:



Generative Model

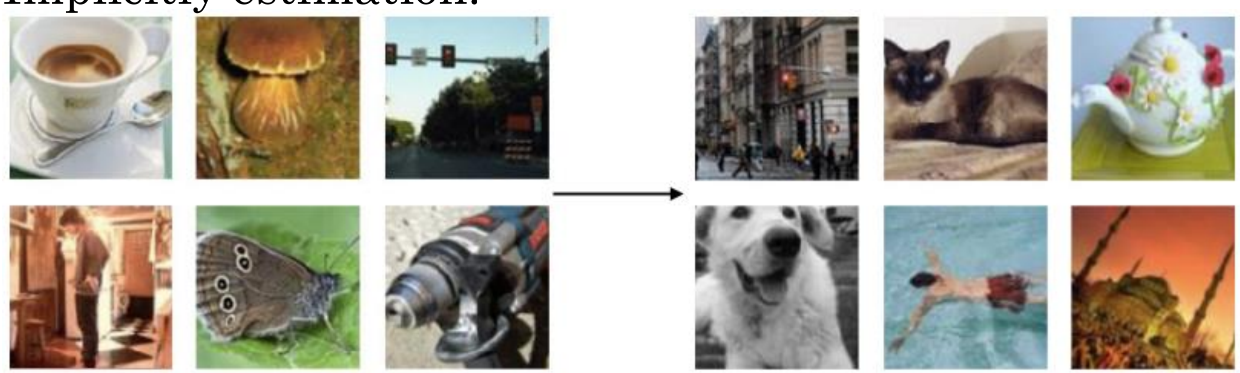
- Given a training dataset, x, try to estimate the distribution , Pdata(x)
- Explicitly or Implicitly (GAN)

Explicitly estimation:



Variational Autoencoders (VAE)

Implicitly estimation:

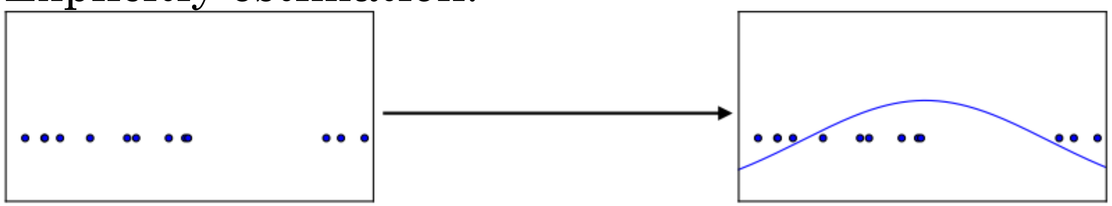


Training examples   Model samples

Generative Model

- Given a training dataset, x, try to estimate the distribution , Pdata(x)
- Explicitly or Implicitly (GAN)

Explicitly estimation:



Variational Autoencoders (VAE)

Implicitly estimation:

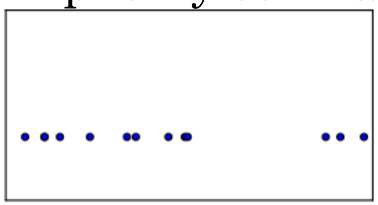
Generative Adversarial Networks (GAN)

Training examples Model samples

Generative Model

- Given a training dataset, x, try to estimate the distribution , Pdata(x)
- Explicitly or Implicitly (GAN)

Explicitly estimation:



GAN's : Don't work with any explicit density function!

Instead, learn to generate from training distribution through 2 players game

...ers (VAE)

Implicitly estimation:

Generative Adversarial Networks (GAN)

Training examples   Model samples

Why Generative Model?

- Realistic samples for artwork, colorization, etc.

- Training GM can enable inference of latent representation that can be useful as general features
- GM can be trained with missing data and can provide predictions on inputs that are missing data
- GM enable machine learning to work with multi-model outputs (produce multiple different correct answers)

Generative Adversarial Networks (GAN)

- Problem:
  - Want to sample from complex, high dimensional training distribution.
  - No direct way to do this!

- Solution:
  - Sample from a simple distribution e.g. random noise.
  - Learn complex transformation to training distribution.

Generative Adversarial Networks (GAN)

- Problem:
  - Want to sample from complex, high dimensional training distribution.
  - No direct way to do this!

- Solution:
  - Sample from a simple distribution e.g. random noise.
  - Learn complex transformation to training distribution.

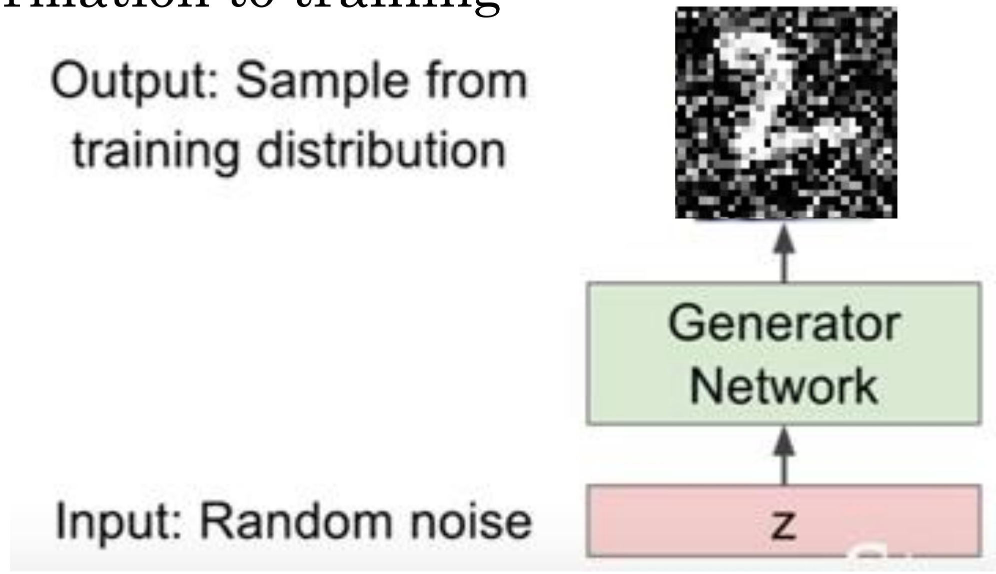
Complex transformation = neural network!

Generative Adversarial Networks (GAN)

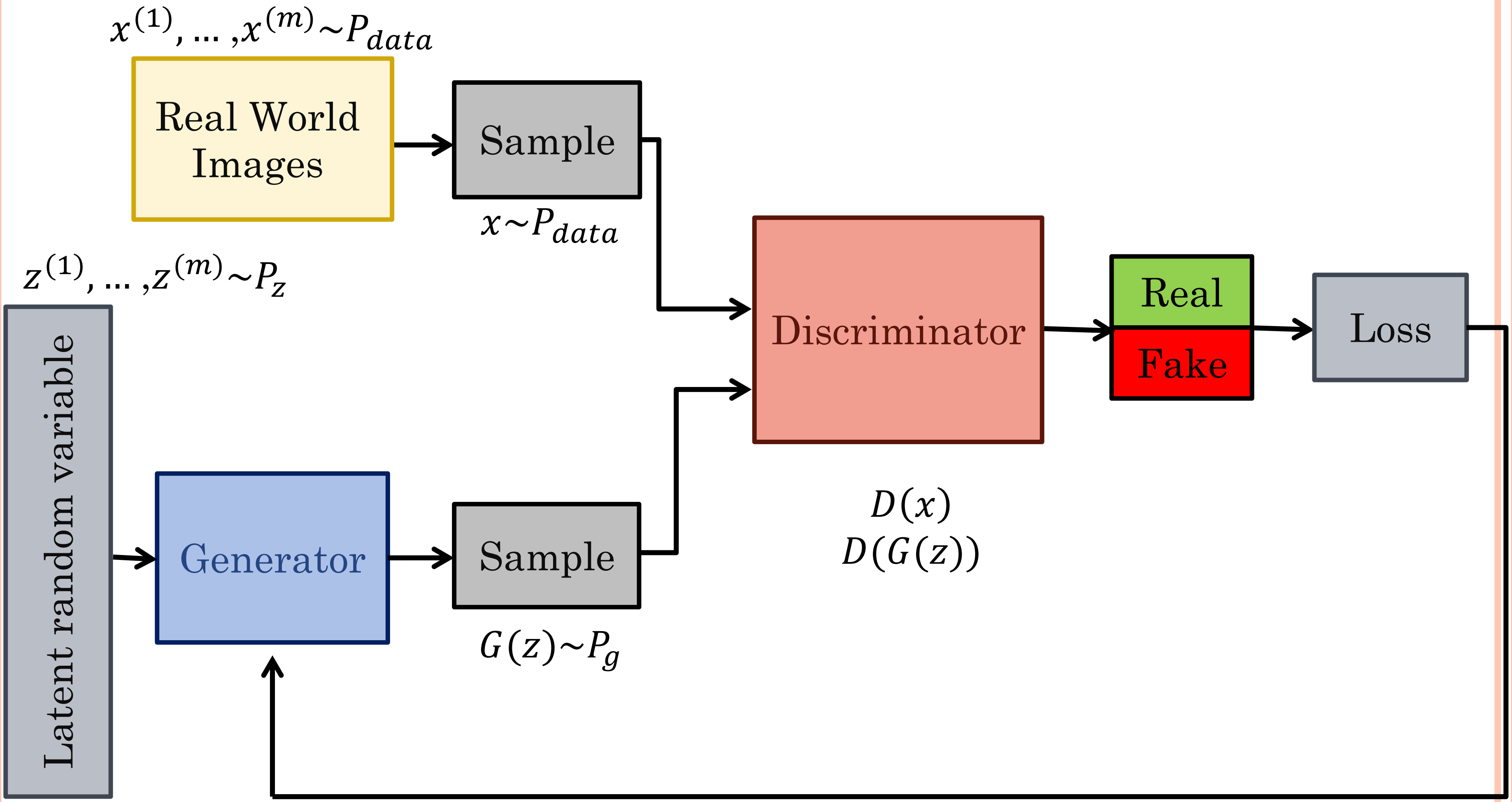
- Problem:
  - Want to sample from complex, high dimensional training distribution.
  - No direct way to do this!

- Solution:
  - Sample from a simple distribution e.g. random noise.
  - Learn complex transformation to training distribution.

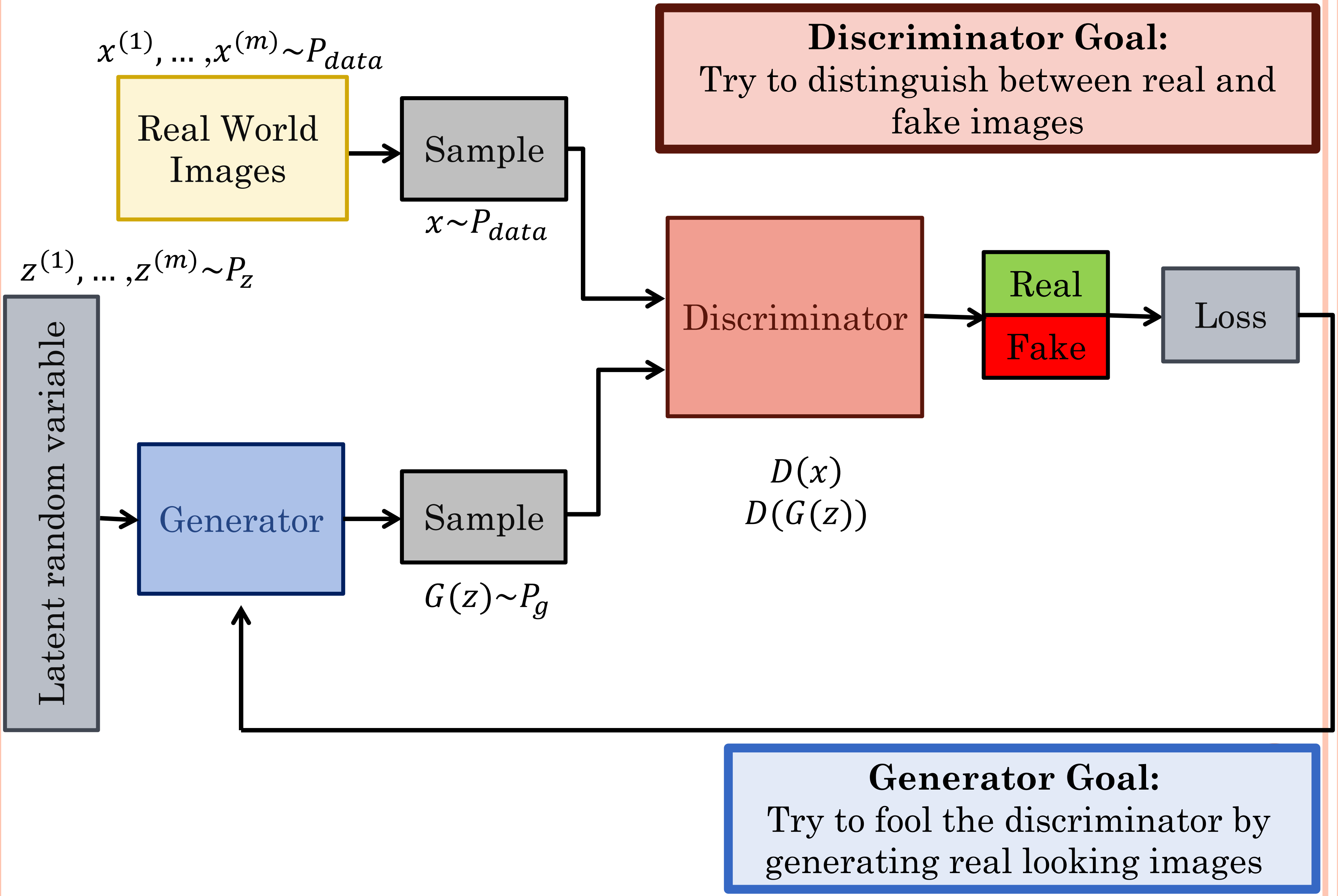
Complex transformation = neural network!



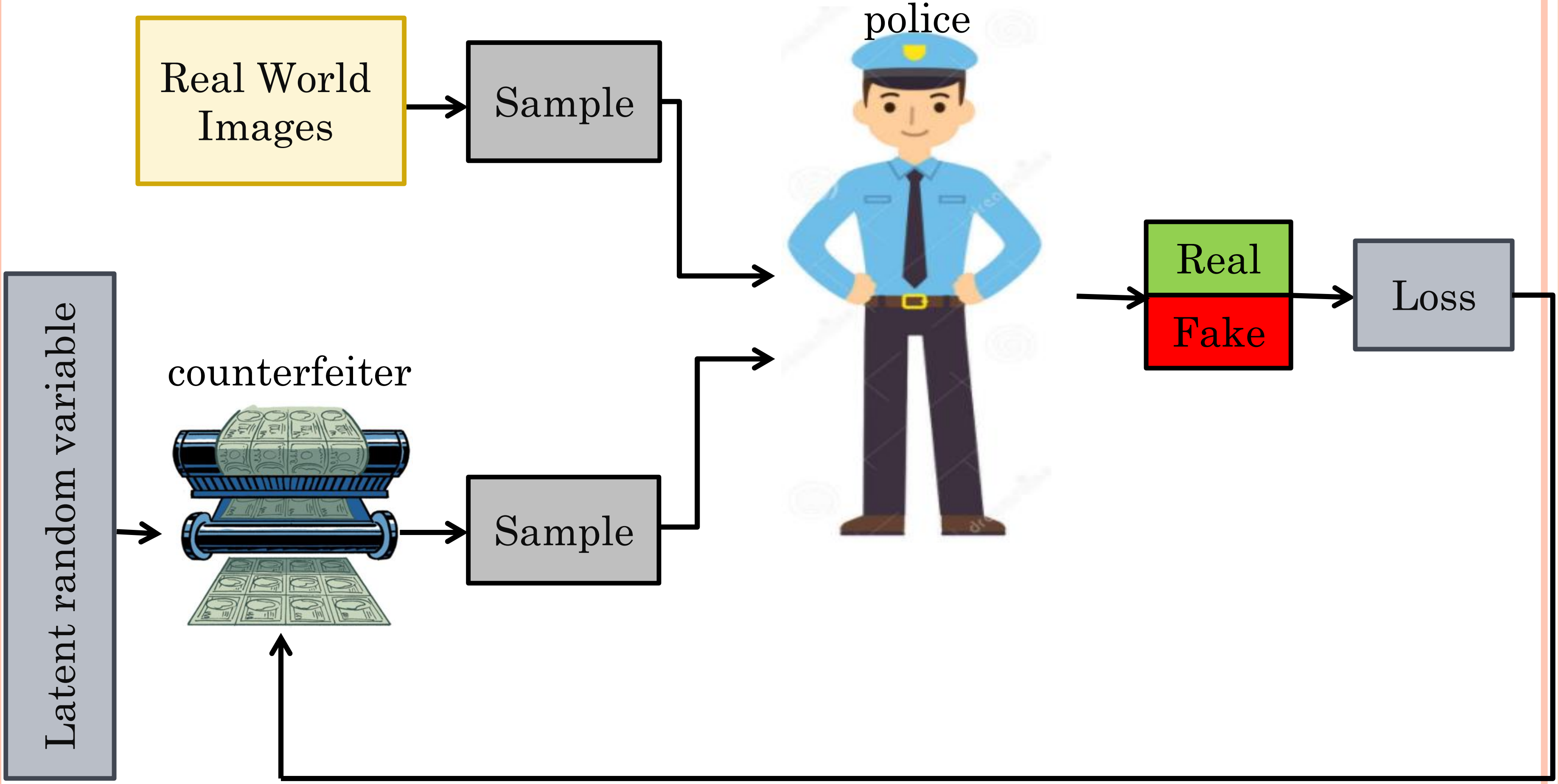
Training GAN : Two player game



Training GAN : Two player game

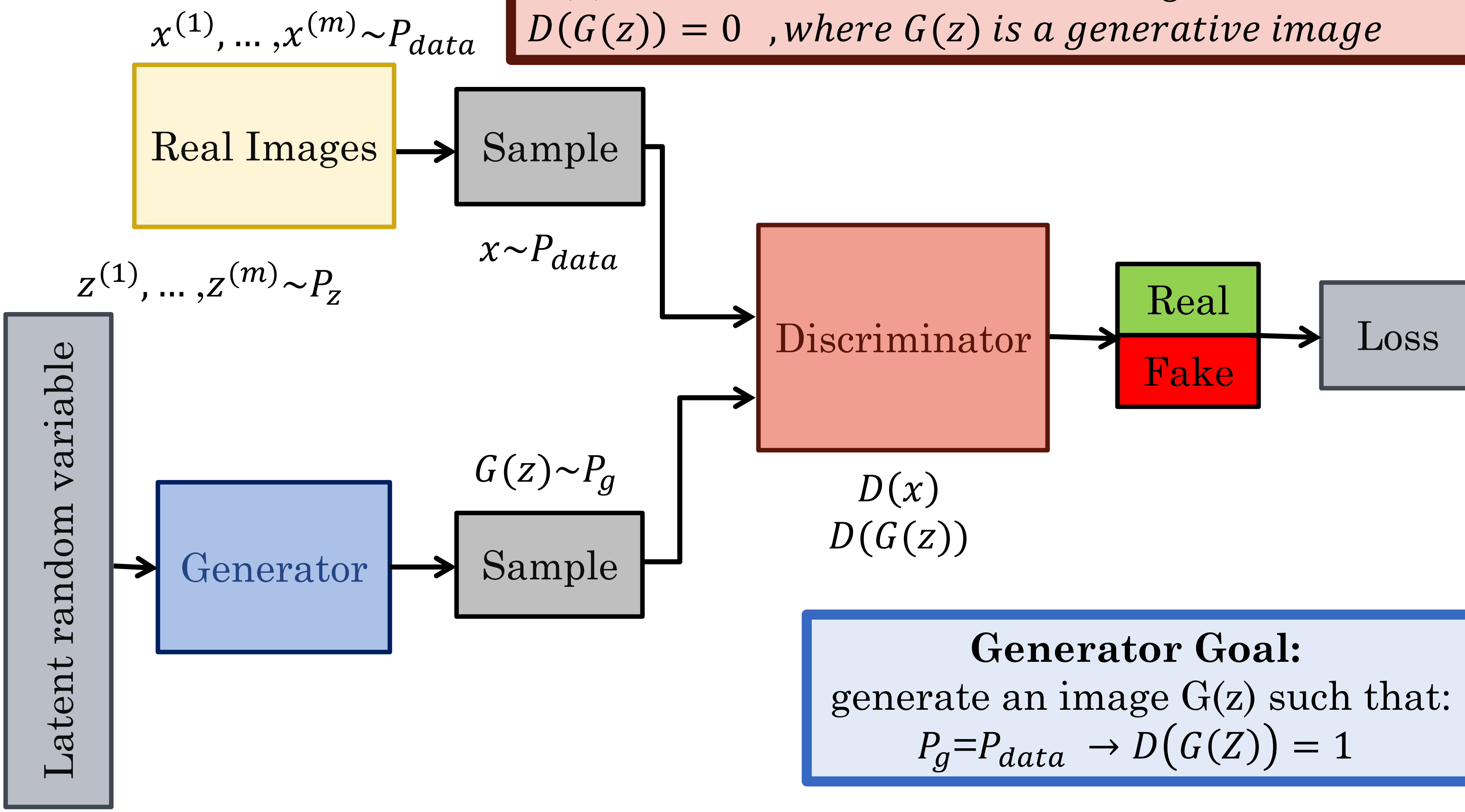


Training GAN : Two player game



Training GAN : Two player game

Discriminator Goal:
Discriminate between real and generated images:
$D(x) = 1 \quad , where\ x\ is\ a\ real\ image$
$D(G(z)) = 0 \quad , where\ G(z)\ is\ a\ generative\ image$



Generator Goal:
generate an image G(z) such that:
$P_g = P_{data} \rightarrow D\big(G(Z)\big) = 1$

Training GAN : The MinMax Game

- G(z) – Generator network.
  - Output in the dimensions of x.
- D(x) – Discriminator network.
  - Output in [0,1] where 1 means x is from $\boldsymbol{p_{data}}$. ($\mathbf{log}\{\boldsymbol{D(x)}\}$ [-inf,0])

$$\boldsymbol{min_G max_D V(D,G) = E_{x\sim p_{data}(x)}}\underbrace{[\mathbf{log}\{\boldsymbol{D(x)}\}]}_{\text{Discriminator output for real data}}+ \boldsymbol{E_{z\sim p_z(z)}}\underbrace{[\mathbf{log}\{\mathbf{1} - \boldsymbol{D\big(G(z)\big)}\}]}_{\text{Discriminator for output generated fake data G(z)}}$$

Training GAN : The MinMax Game

$$\boldsymbol{min_G max_D V(D,G) = E_{x\sim p_{data}(x)}[\log\{D(x)\}] + E_{z\sim p_z(z)}[\log\{1 - D(G(z))\}]}$$

Alternate between:

- **Gradient ascent** on discriminator:

$$\boldsymbol{max_D[\, E_{x\sim p_{data}(x)}[\log\{D(x)\}] + E_{z\sim p_z(z)}[\log\{1 - D(G(z))\}]]}$$

- **Gradient descent** on generator:

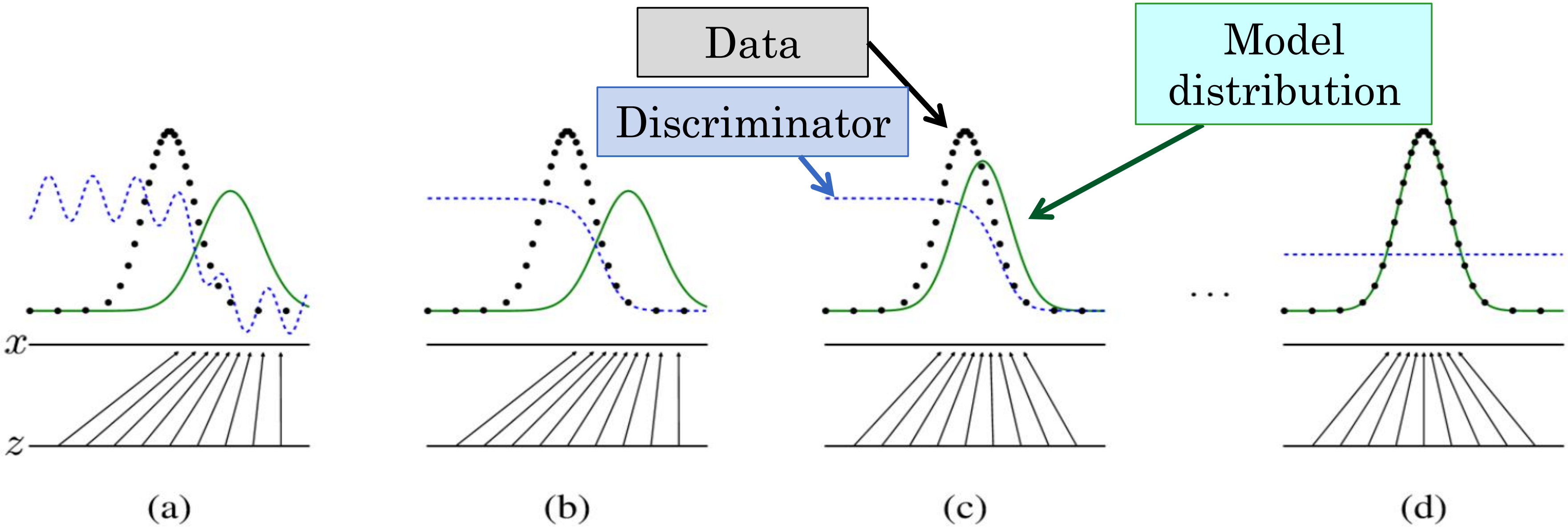
$$\boldsymbol{min_G[E_{z\sim p_z(z)}[\log\{1 - D(G(z))\}]]}$$

TRAINING GAN : THE MINMAX GAME

- Equilibrium is a saddle point of the discriminator loss

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \xrightarrow{p_g(x)=p_{data}(x)} 0.5$$

- Estimating this ratio using supervised learning is the key approximation mechanism used by GAN



TRAINING GAN : THE MINMAX GAME

$$\boldsymbol{min_G max_D V(D,G) = E_{x\sim p_{data}(x)}[\log\{D(x)\}] + E_{z\sim p_z(z)}[\log\{1 - D(G(z))\}]}$$

Alternate between:

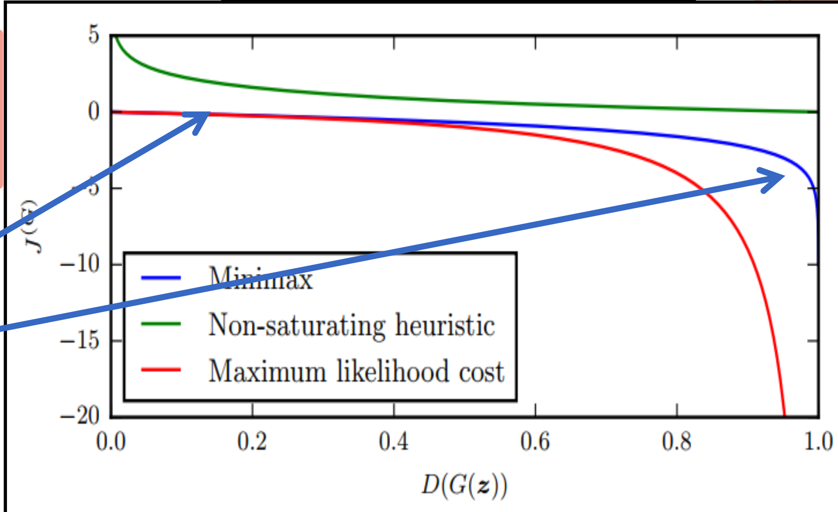
- **Gradient ascent** on discriminator:

$$\boldsymbol{max_D[\, E_{x\sim p_{data}(x)}[\log\{D(x)\}] + E_{z\sim p_z(z)}[\log\{1 - D(G(z))\}]]}$$

- **Gradient descent** on generator:

$$\boldsymbol{min_G[E_{z\sim p_z(z)}[\log\{1 - D(G(z))\}]]}$$

Generator Loss



In practice, optimizing this generator objective not work well!

Training GAN : The MinMax Game

$$min_G max_D V(D,G) = E_{x\sim p_{data}(x)}[\log\{D(x)\}] + E_{z\sim p_z(z)}[\log\{1 - D(G(z))\}]$$

Alternate between:

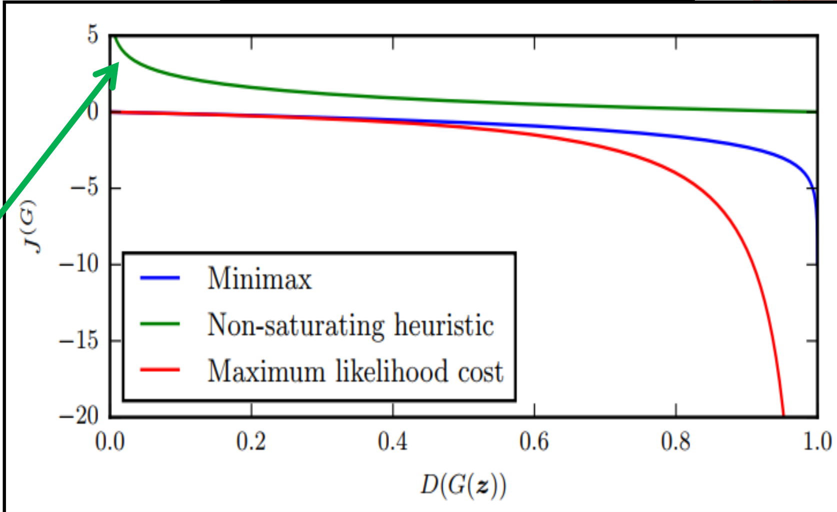
- **Gradient ascent** on discriminator:

$$max_D[\ E_{x\sim p_{data}(x)}[\log\{D(x)\}] + E_{z\sim p_z(z)}[\log\{1 - D(G(z))\}]]$$

- **Gradient ascent** on generator:

$$max_G[E_{z\sim p_z(z)}[\log\{D(G(z))\}]]$$

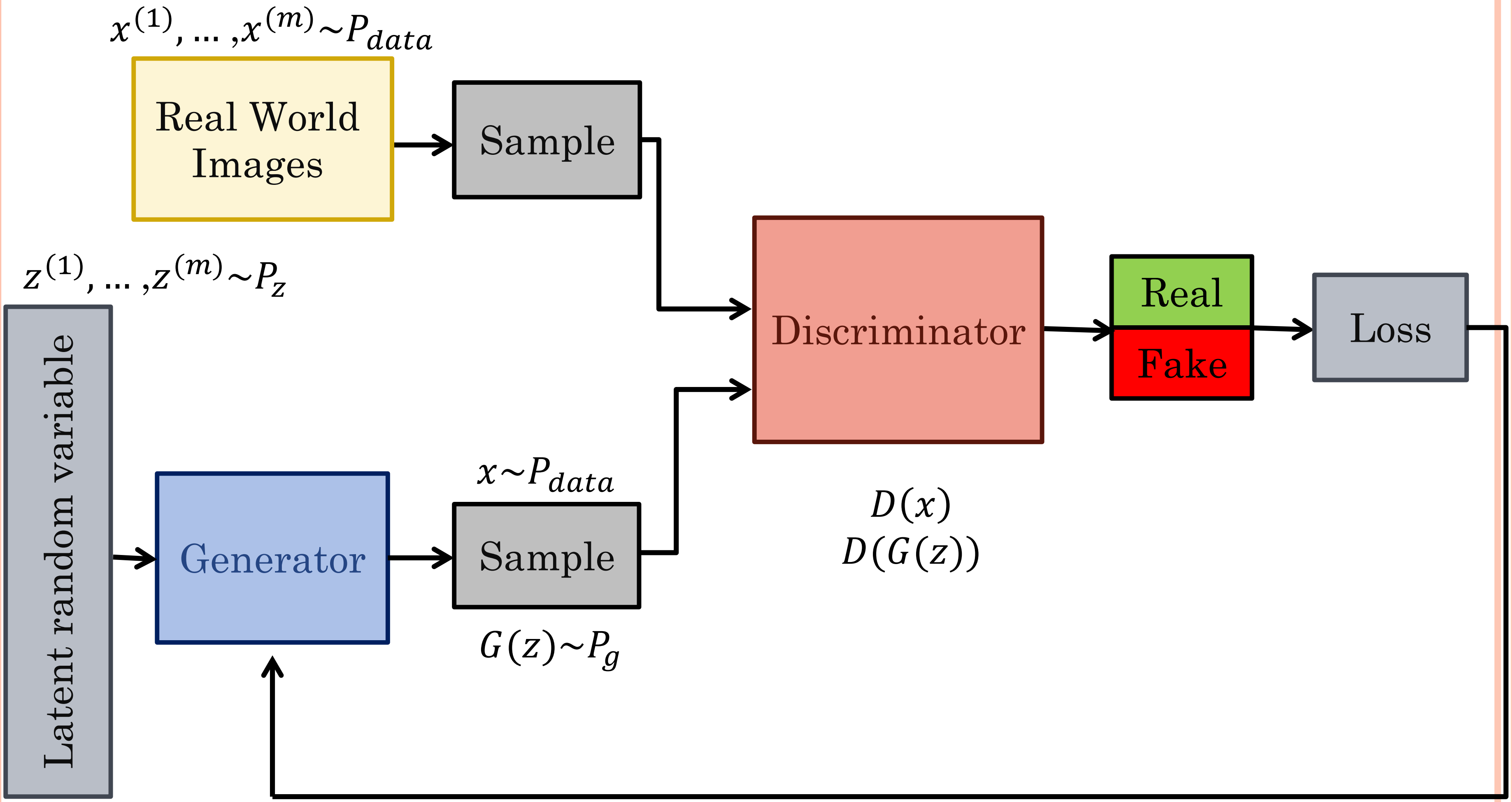
Generator Loss



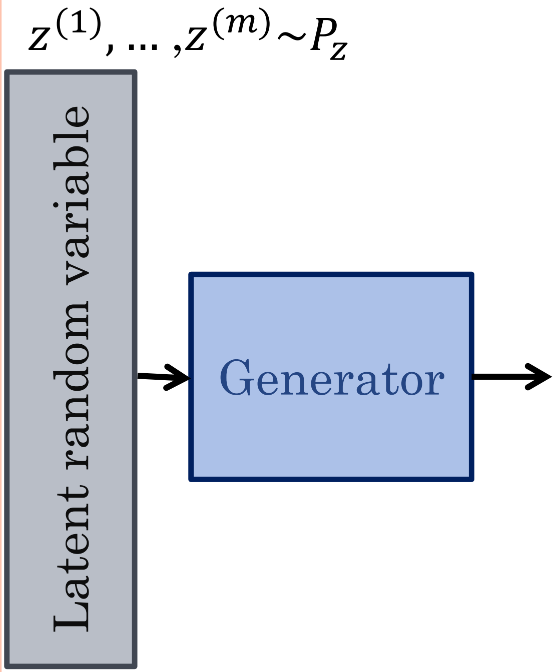
Maximizing likelihood of discriminator being wrong!

Higher gradient for bad samples

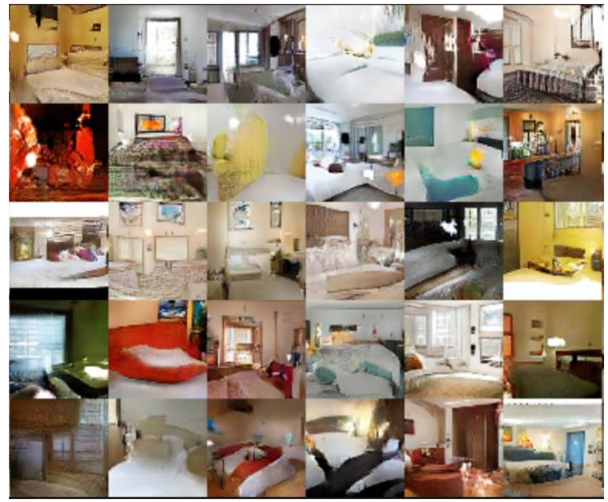
TRAINING GAN : TWO PLAYER GAME



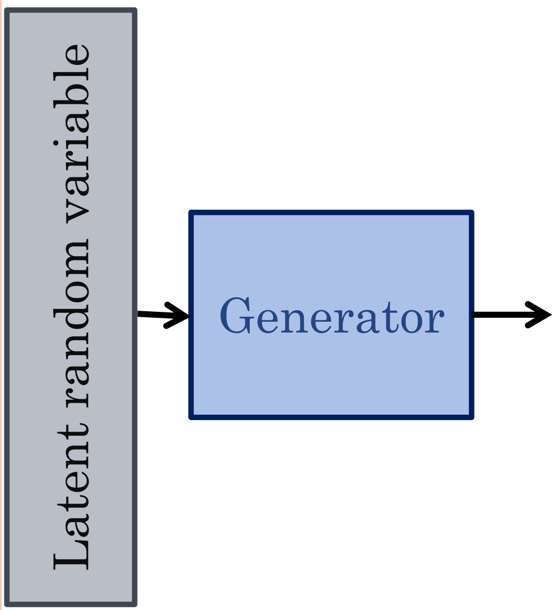
Training GAN : Two player game



Training GAN : Two player game

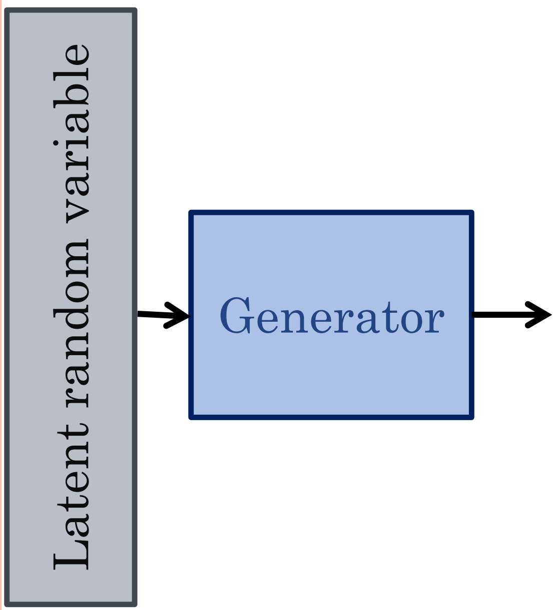


$z^{(1)}, \dots, z^{(m)} \sim P_z$



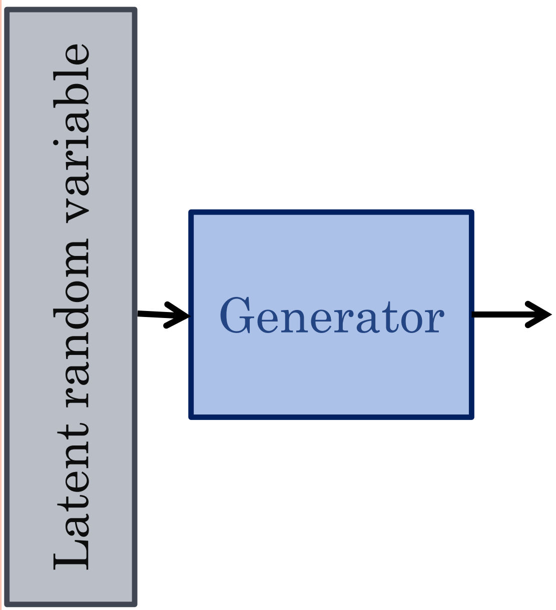
Training GAN : Two player game

$z^{(1)}, \dots, z^{(m)} \sim P_z$



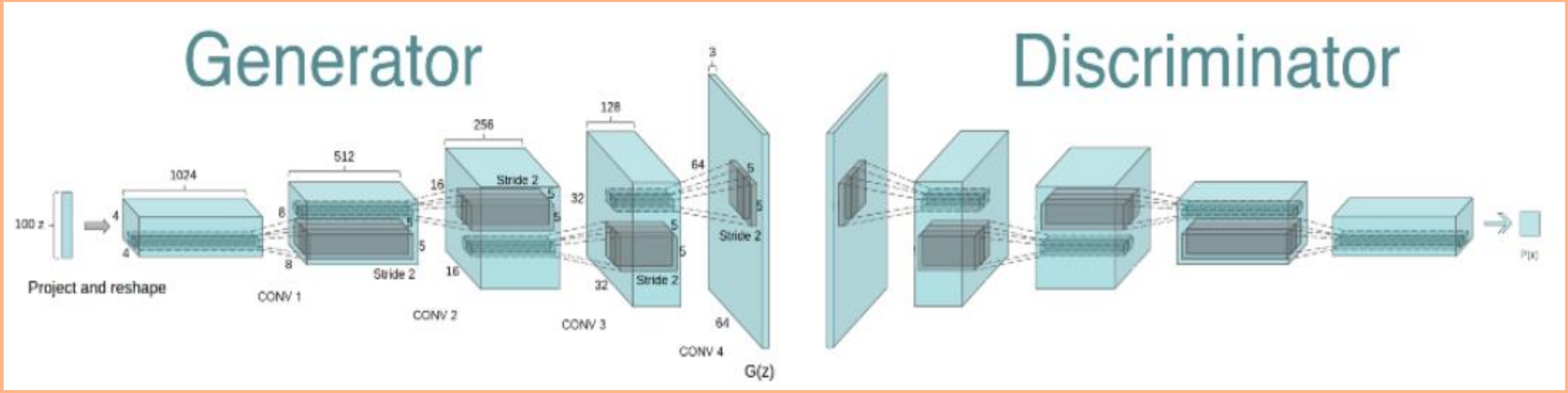
TRAINING GAN : TWO PLAYER GAME

$z^{(1)}, \dots, z^{(m)} \sim P_z$



DEEP CONVOLUTION GAN (DCGAN)

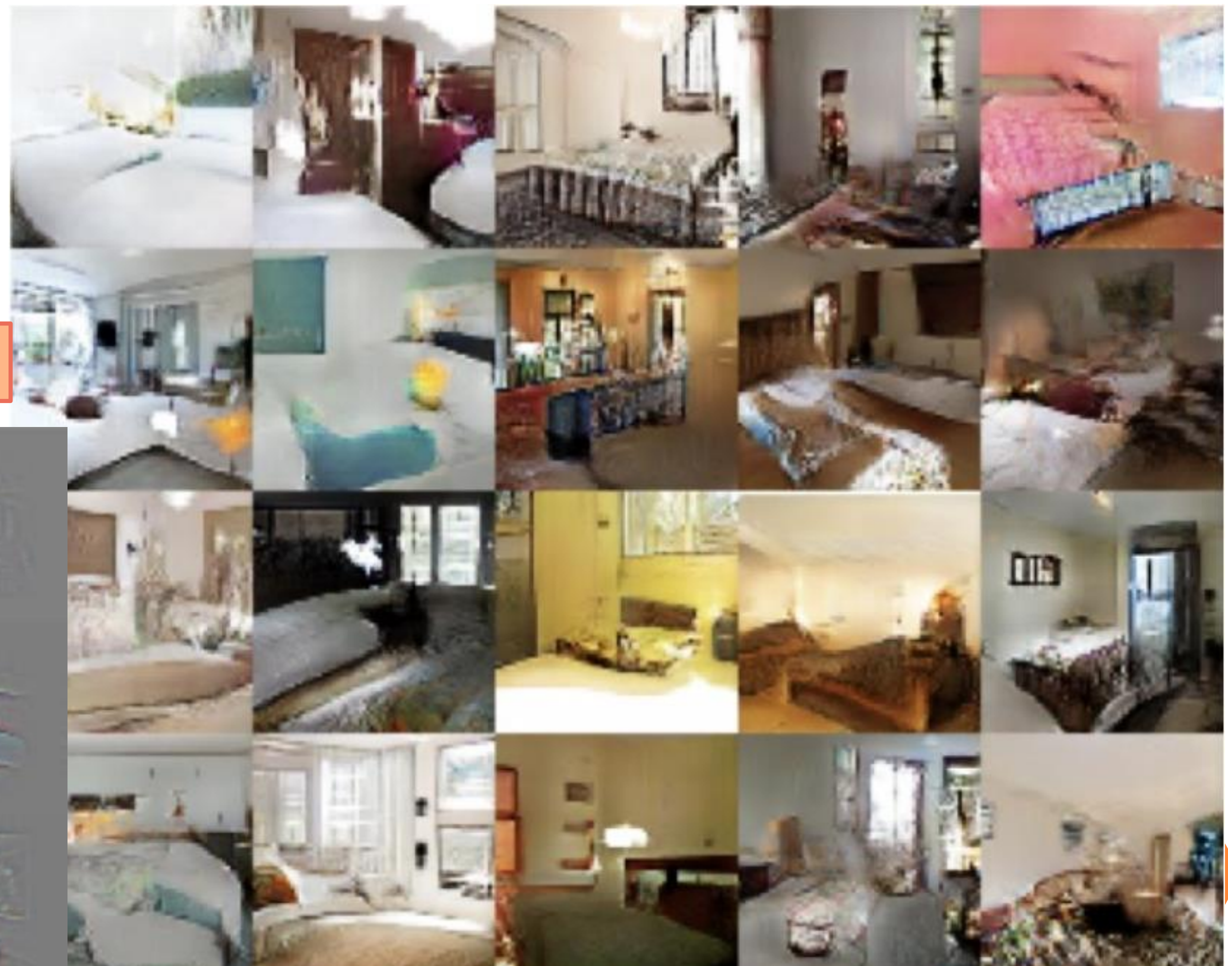
- Architecture key insights:
  - Use batch normalization layers in both discriminator and generator.
  - Use convolution with a stride greater than 1 instead of pooling layer.
  - Use Adam optimizer instead of SGD.
  - Use ReLU activation in generator for all layers except for the output, which uses Tanh.
  - Use LeakyReLU activation in the discriminator for all layers.



DEEP CONVOLUTION GAN (DCGAN)

- Results:

Images of bedrooms generated by a DCGAN



Trained filters

Deep Convolution GAN (DCGAN)

- Results:

Vector space arithmetic

Samples from the model

man
with glasses

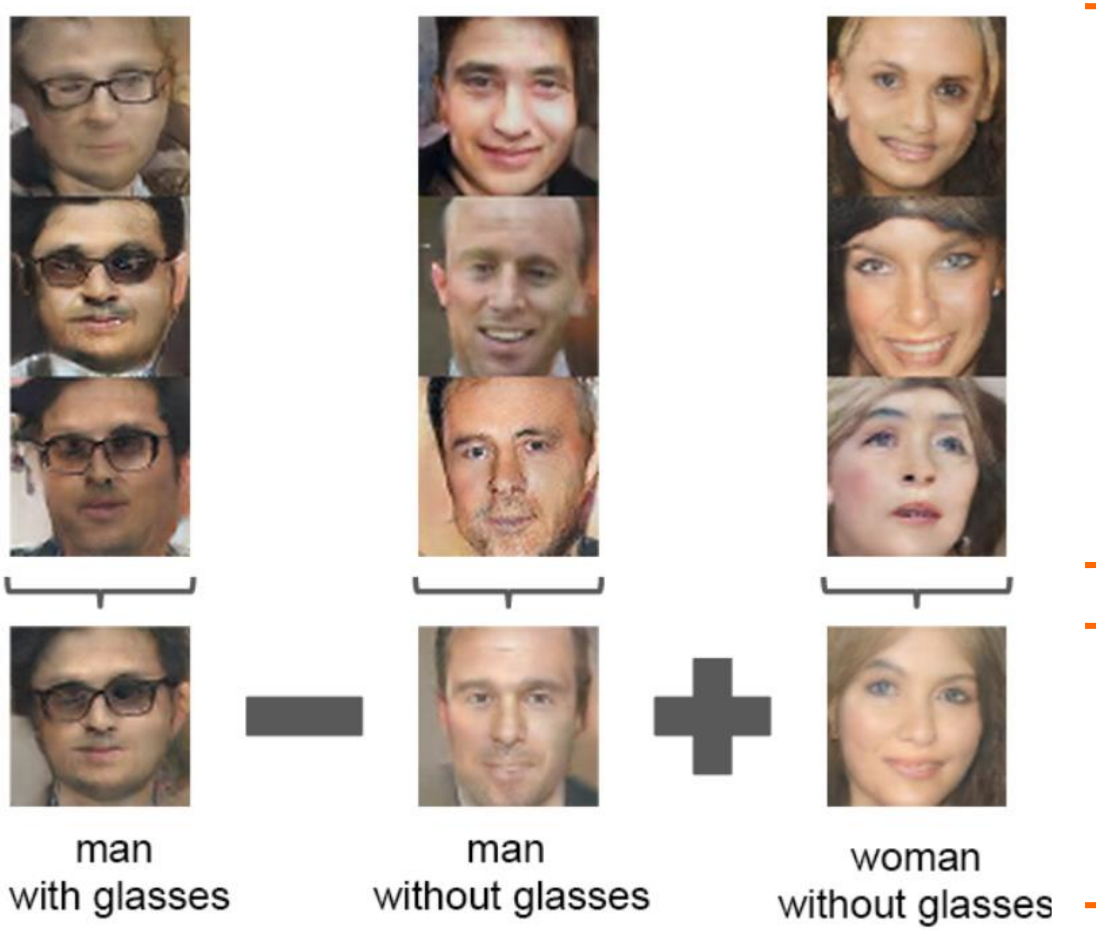
man
without glasses

woman
without glasses

DEEP CONVOLUTION GAN (DCGAN)

- Results:

Vector space arithmetic



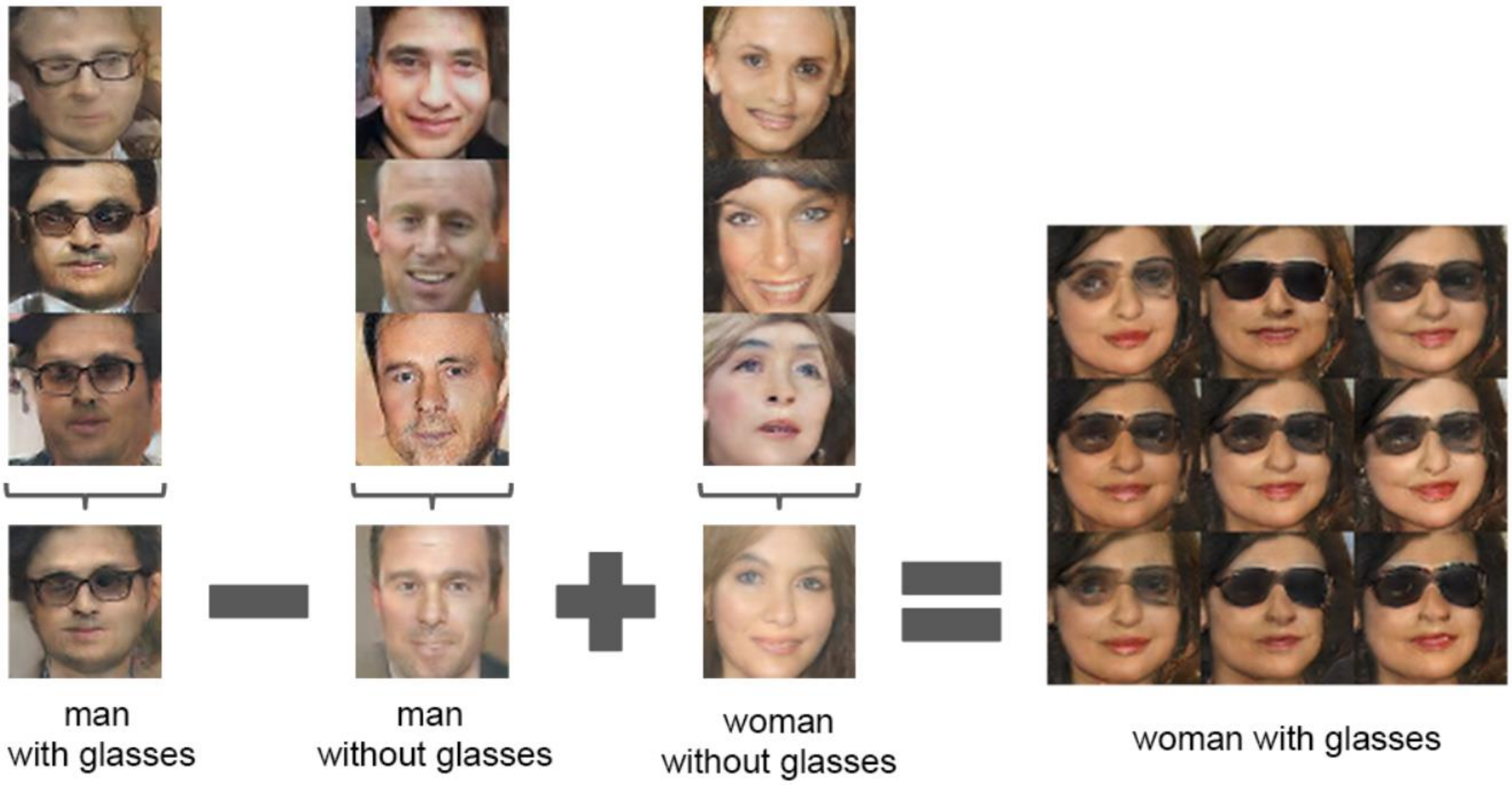
Samples from the model

Average Z vectors, do arithmetic

Deep Convolution GAN (DCGAN)

- Results:

Vector space arithmetic

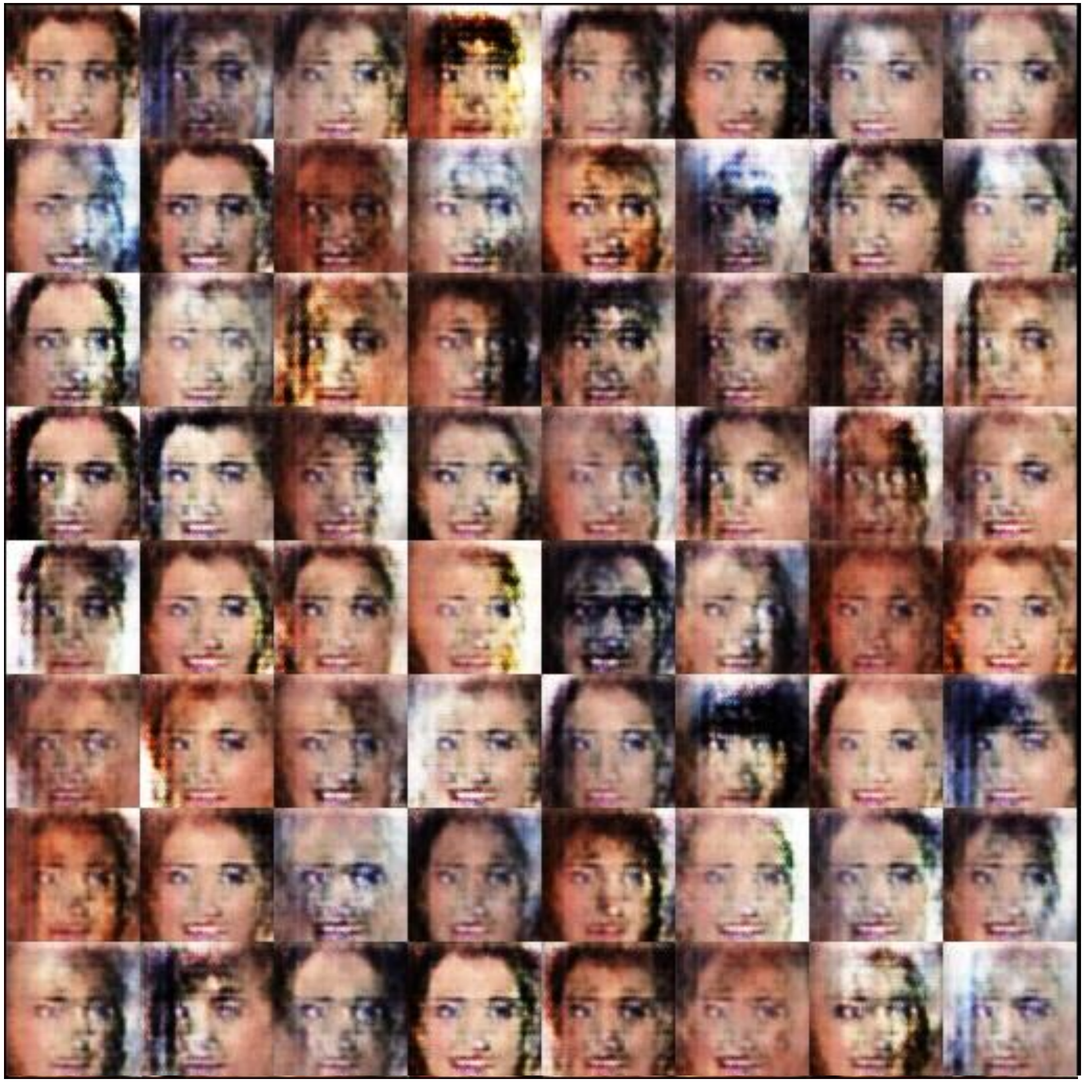


Deep Convolution GAN (DCGAN)

- Our Results:

Deep Convolution GAN (DCGAN)

- Our Results:

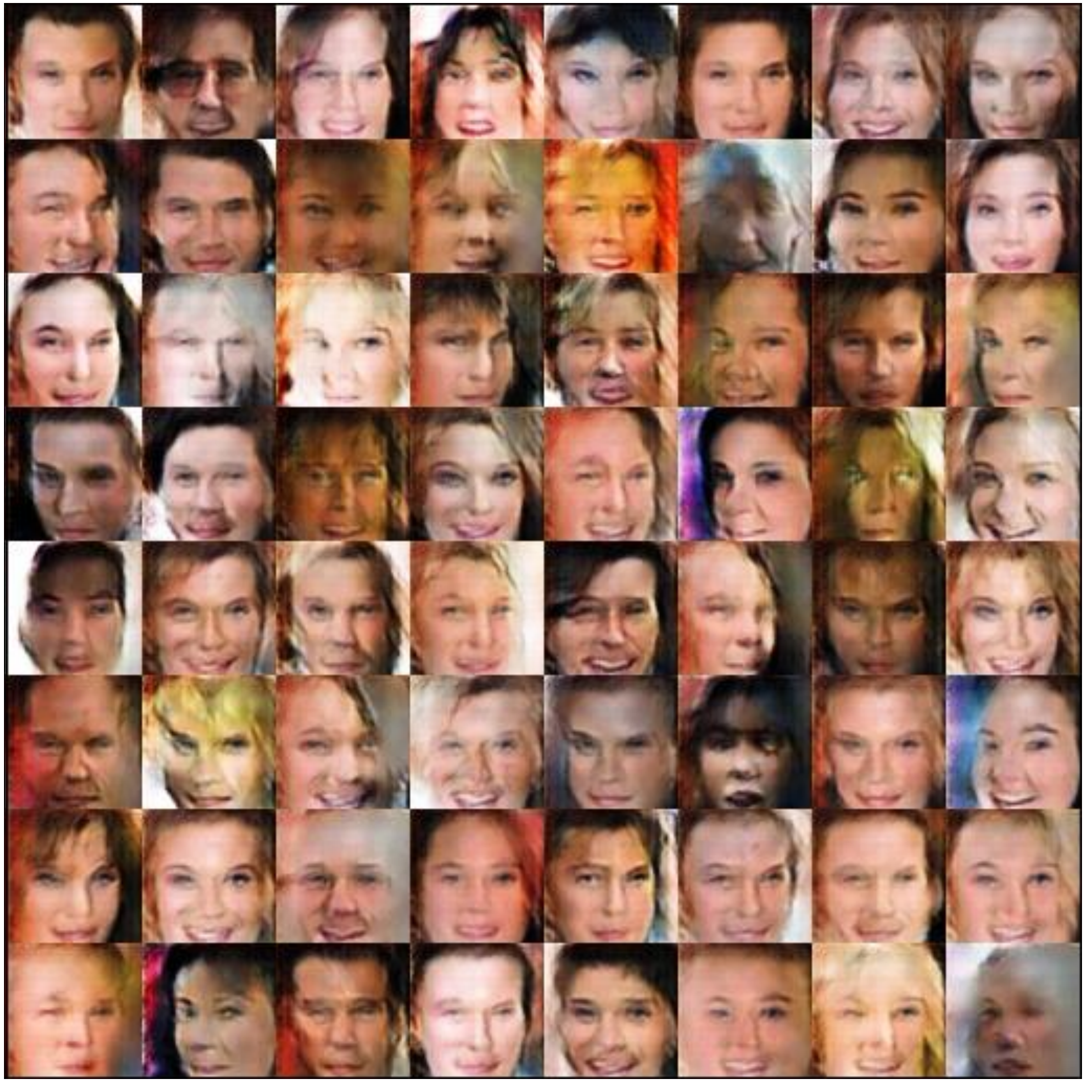


Deep Convolution GAN (DCGAN)

- Our Results:

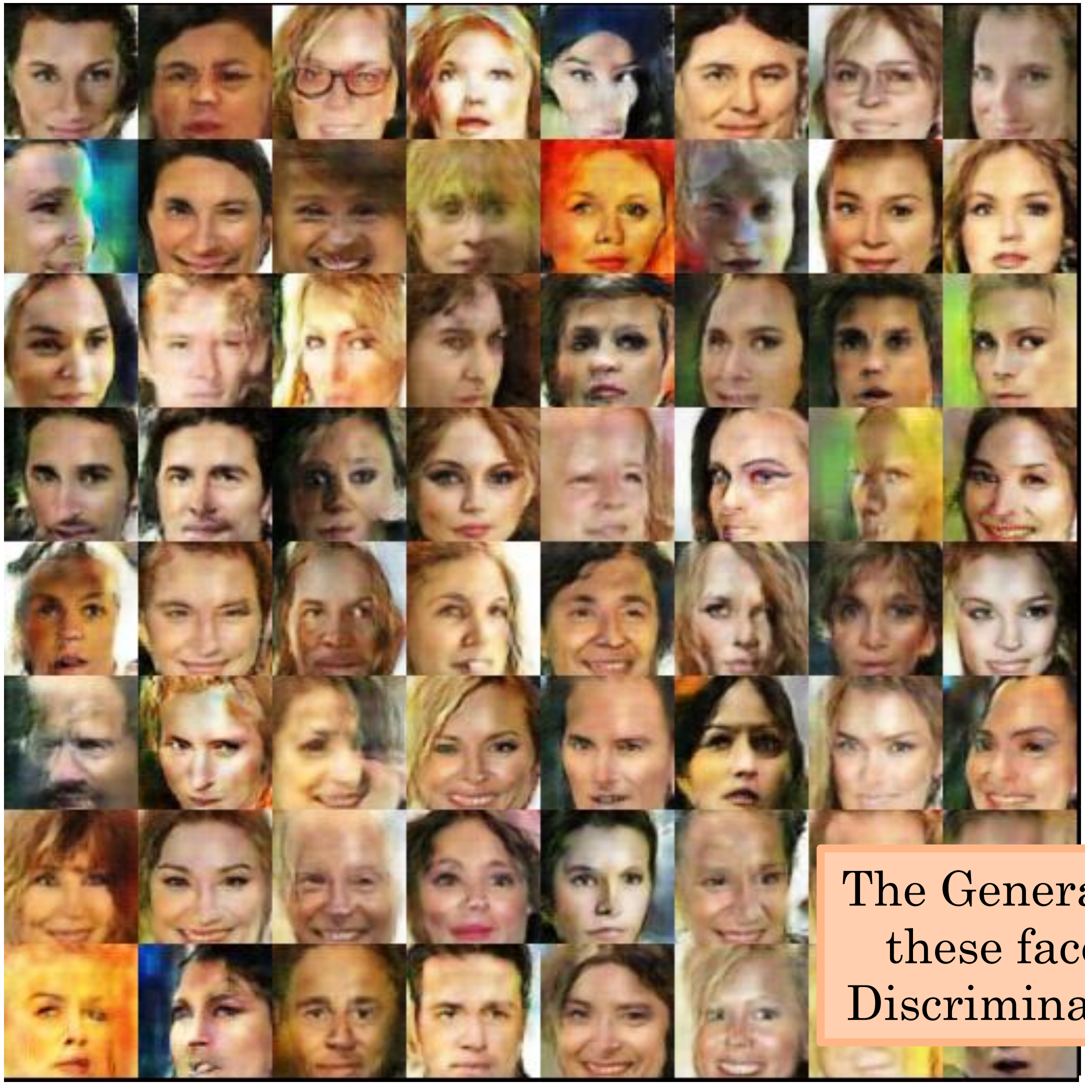
Deep Convolution GAN (DCGAN)

- Our Results:



Deep Convolution GAN (DCGAN)

- Our Results:



The Generator never saw these faces ! Only the Discriminative network !

GAN : Applications

- Single image super-resolution
- Image to image translation
- Text to image
- Style transfer
- Video generation
- Semi-supervised learning

And more…

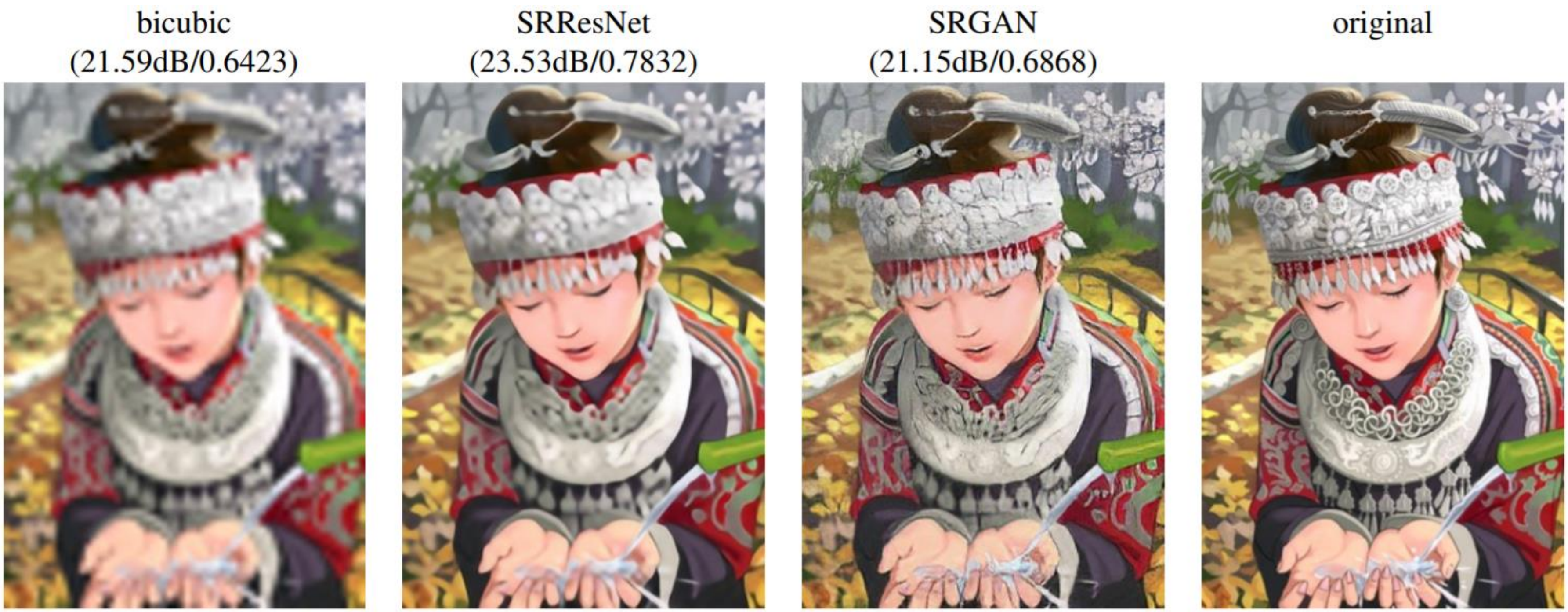
GAN : Applications

- Single image super-resolution
- Image to image translation
- Text to image
- Style transfer
- Video generation
- Semi-supervised learning

And more…

GAN : Applications

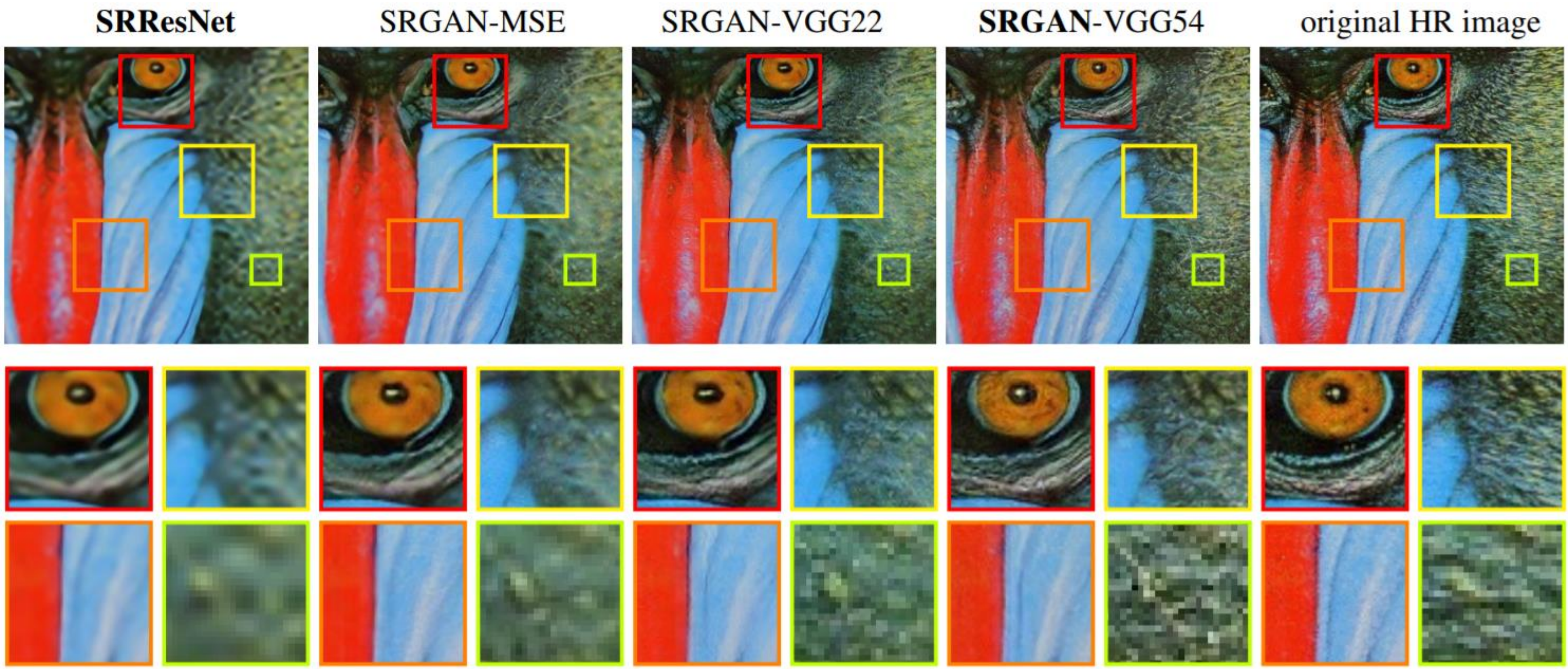
Single image super-resolution



From the paper “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network“

GAN : Applications

Single image super-resolution



From the paper "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network"

GAN : Applications

- Single image super-resolution
- Image to image translation
- Text to image
- Style transfer
- Video generation
- Semi-supervised learning

And more…

Image to image translation

Image to image translation

- A class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image

Image to image translation

- We will focus on 2 papers from 2017 dealing with unsupervised image to image translation
  - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks (UC Berkeley) - cycleGAN
  - Unsupervised Image-to-Image Translation Networks (NVIDIA) – UNIT network

Image to image translation

Supervised approach

- Data $\{x_i, y_i\}$
- Goal: Learn mapping from $x_i \rightarrow y_i$ across all pairs

Unsupervised approach

- Data $(\{x\}, \{y\})$
- Goal: learn mapping from group $\{x\}$ to group $\{y\}$ by learning underlying structure

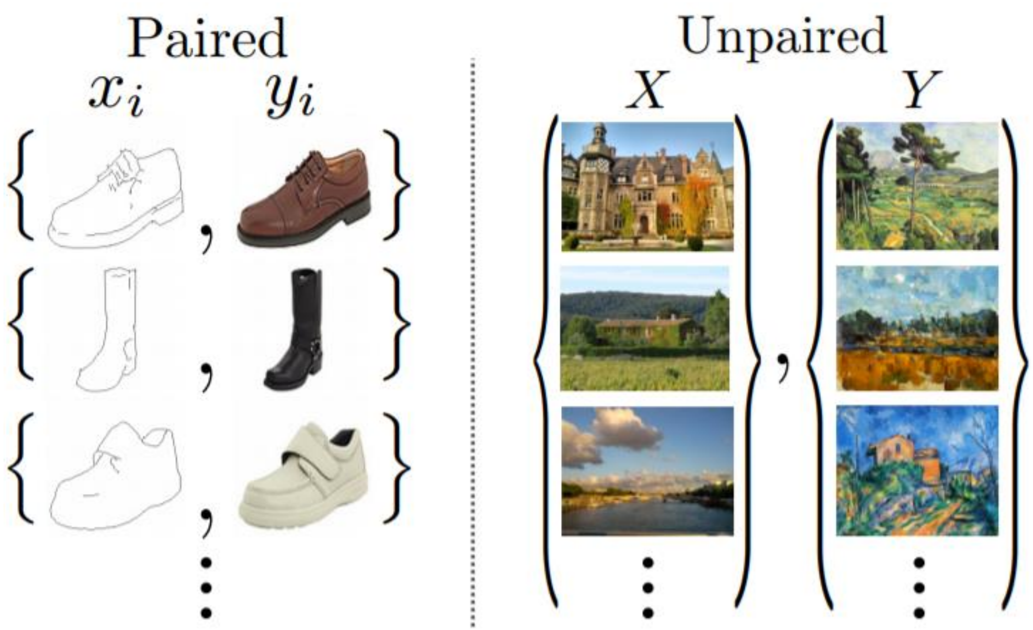
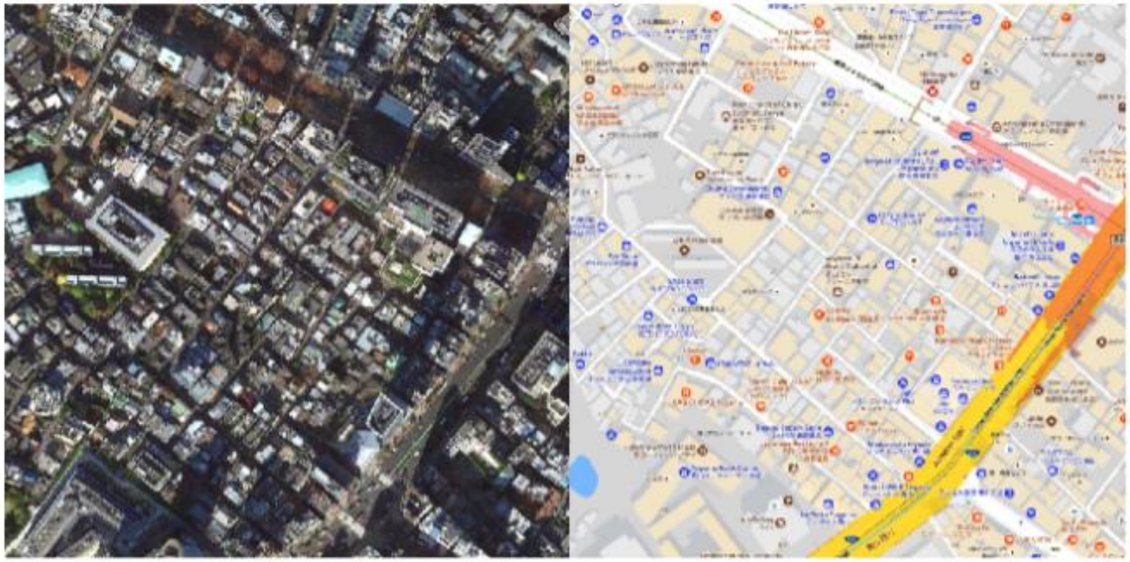


Image to image translation

Examples:

SUPERVISED LEARNING



Corresponding pairs of images are available

Image to image translation

Examples:

UNSUPERVISED LEARNING

Harder to solve, but data collection is easier

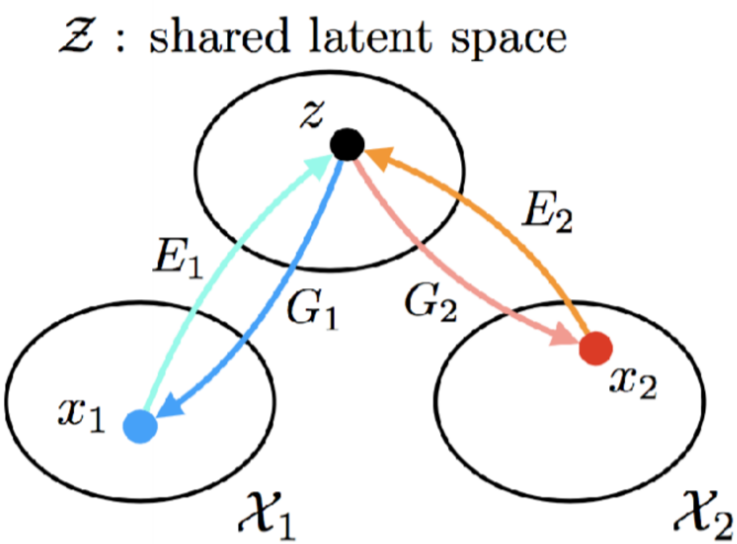
Image to image translation

- Papers use similar approach to solving this problem:
  - Use GAN generator – discriminator model
  - Use VAE latent space encoding as part generator architecture (though only UNIT trains VAE explicitly)
  - Learn translation in both directions at once

Image to image translation

UNIT

- assumes a shared latent space: pair of corresponding images can be mapped to same latent representation



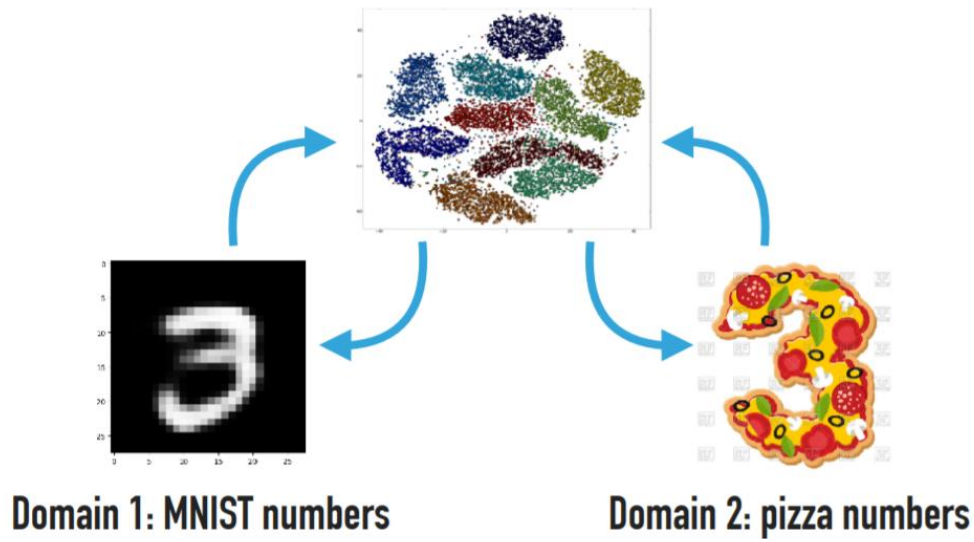


Image to image translation

UNIT

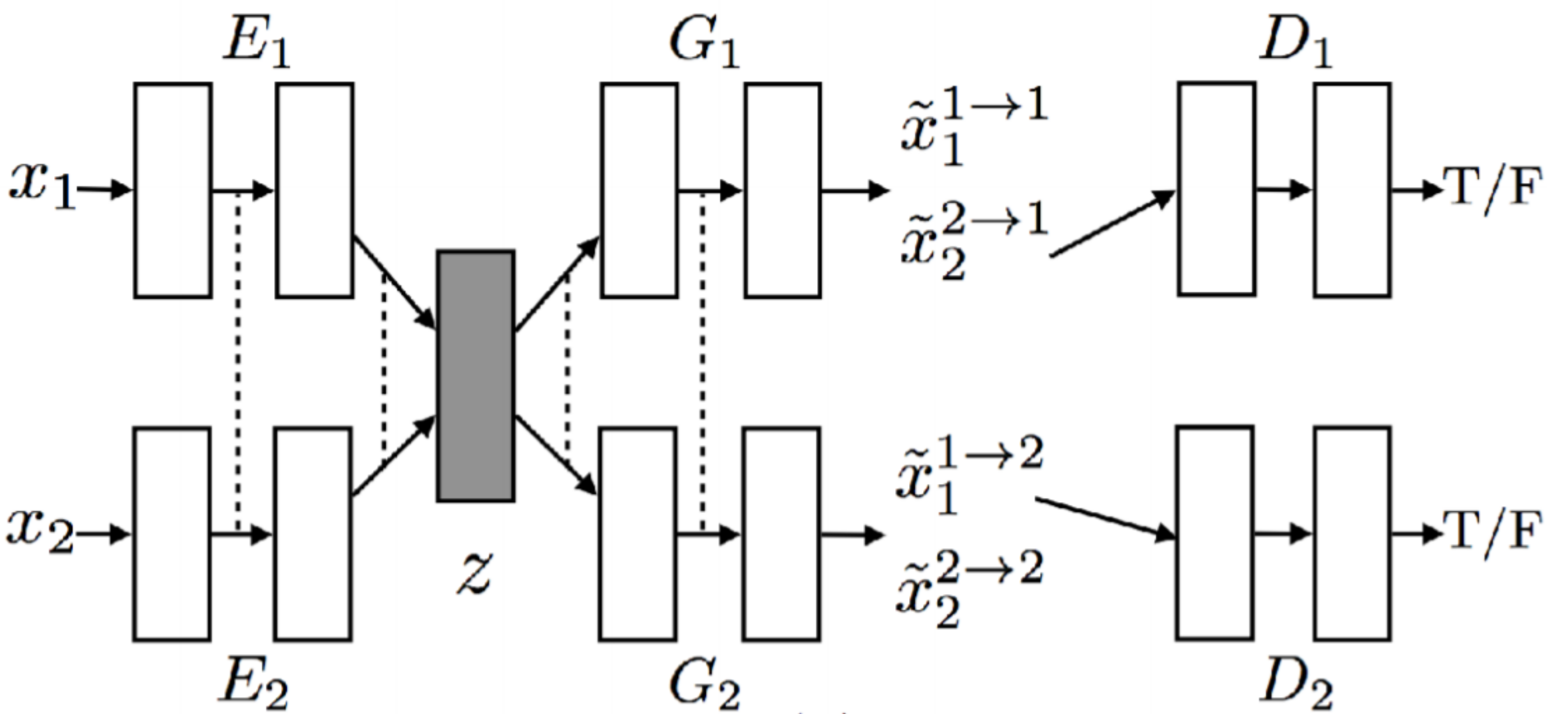


Table 1: Interpretation of the roles of the subnetworks in the proposed framework.

Networks	$\{E_1, G_1\}$	$\{E_1, G_2\}$	$\{G_1, D_1\}$	$\{E_1, G_1, D_1\}$	$\{G_1, G_2, D_1, D_2\}$
Roles	VAE for $\mathcal{X}_1$	Image Translator $\mathcal{X}_1 \to \mathcal{X}_2$	GAN for $\mathcal{X}_1$	VAE-GAN [14]	CoGAN [17]

Image to image translation

UNIT

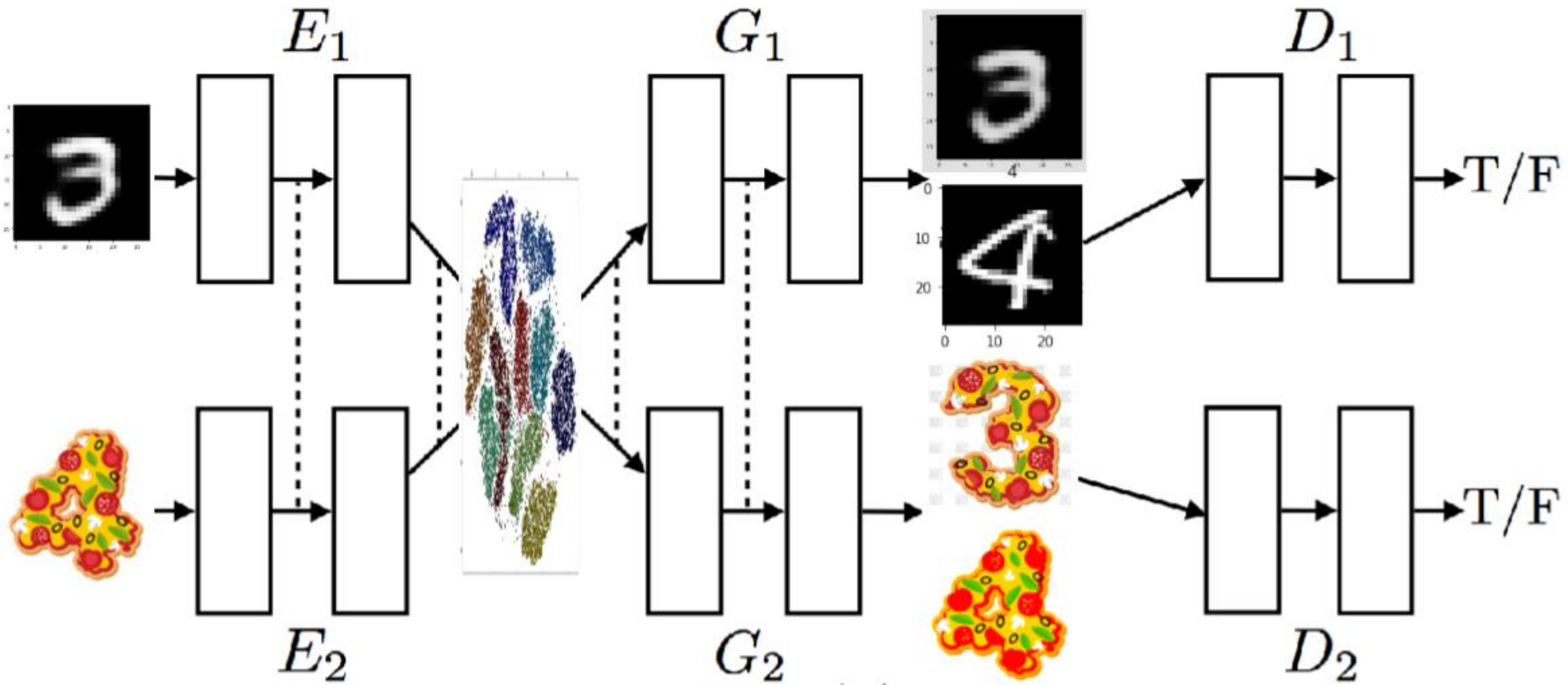


Table 1: Interpretation of the roles of the subnetworks in the proposed framework.

Networks	$\{E_1, G_1\}$	$\{E_1, G_2\}$	$\{G_1, D_1\}$	$\{E_1, G_1, D_1\}$	$\{G_1, G_2, D_1, D_2\}$
Roles	VAE for $\mathcal{X}_1$	Image Translator $\mathcal{X}_1 \rightarrow \mathcal{X}_2$	GAN for $\mathcal{X}_1$	VAE-GAN [14]	CoGAN [17]

IMAGE TO IMAGE TRANSLATION

UNIT

- Encoder-generator pair is a VAE
- E1(E2) encodes image x_1 (x_2) to a latent vector z
- G1 (G2) decodes a randomly perturbed version of the code z
- VAE1 and VAE2 share weights: high level layers of E1 and E2, and same for G1 and G2
- VAE training ensure the reconstructed image and the original image are similar

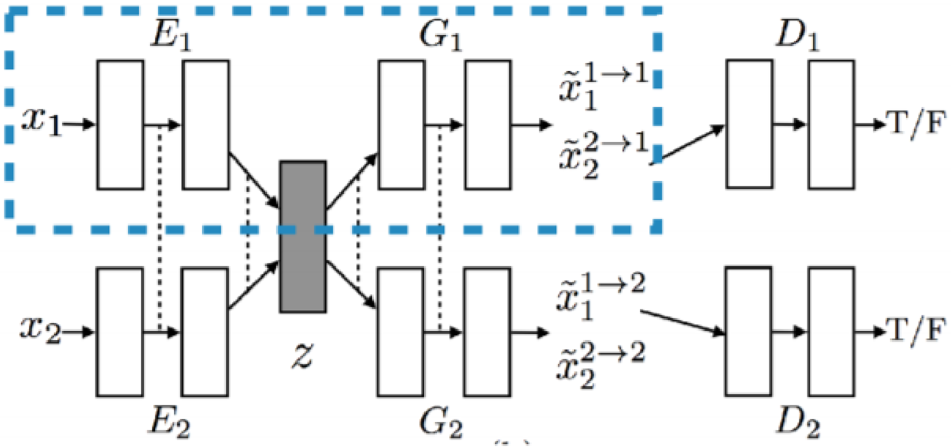


IMAGE TO IMAGE TRANSLATION

UNIT

- Generator-discriminator pair is a GAN
- Reconstruction is already trained by VAE, so D is only applied to translated images
- G1(G2) generates images for domain 1 (2) from a latent vector z
- D1 is trained to output True for x_1, and False for $\tilde{x}_2^{2\to 1}$ (opposite for D2)
- GAN training ensures translated images resemble images in the target domain

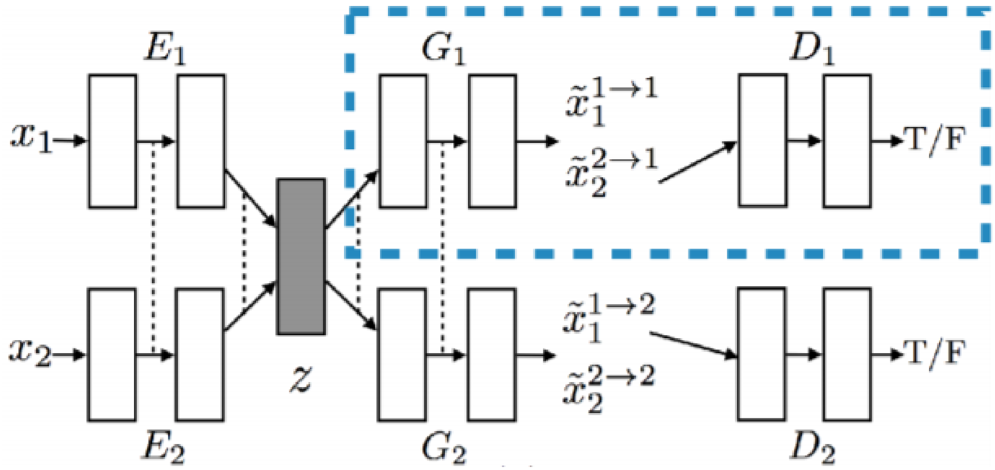


IMAGE TO IMAGE TRANSLATION

UNIT

CYCLE CONSISTENCY

- Shared-latent space implies cycle consistency
- Added weight sharing to regularize training
- Forward translation: x*2 = G2(E1(x1))
- Backward translation: x**1 = G1(E2(x*2))
- Impose that original image x1 and cycle-translated image x*1 are equal: x1 = x**1 (and similarly: x2 = x**2)

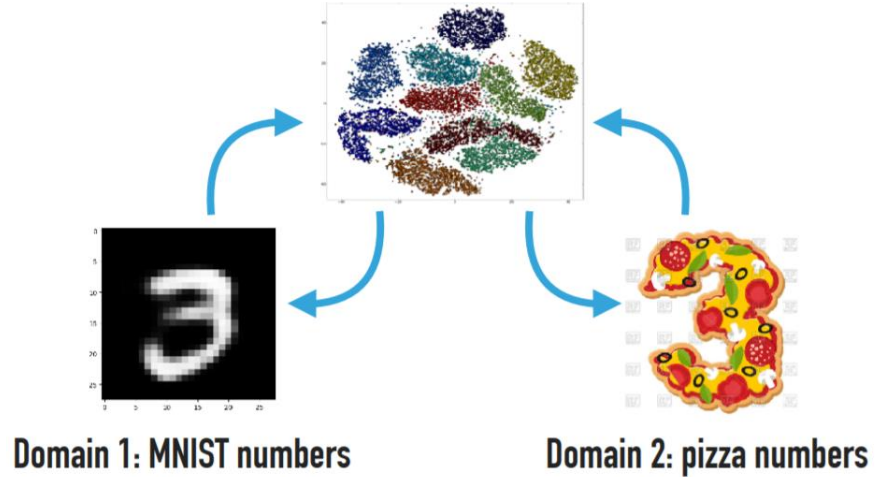


Image to image translation

UNIT

- Explicit loss function:

$$\min_{E_1,E_2,G_1,G_2}\max_{D_1,D_2} \mathcal{L}_{\text{VAE}_1}(E_1,G_1)+\mathcal{L}_{\text{GAN}_1}(E_1,G_1,D_1)+\mathcal{L}_{\text{CC}_1}(E_1,G_1,E_2,G_2)$$
$$\mathcal{L}_{\text{VAE}_2}(E_2,G_2)+\mathcal{L}_{\text{GAN}_2}(E_2,G_2,D_2)+\mathcal{L}_{\text{CC}_2}(E_2,G_2,E_1,G_1).$$

$$\mathcal{L}_{\text{VAE}_1}(E_1,G_1)=\lambda_1\text{KL}(q_1(z_1|x_1)||p_\eta(z))-\lambda_2\mathbb{E}_{z_1\sim q_1(z_1|x_1)}[\log p_{G_1}(x_1|z_1)]$$
$$\mathcal{L}_{\text{VAE}_2}(E_2,G_2)=\lambda_1\text{KL}(q_2(z_2|x_2)||p_\eta(z))-\lambda_2\mathbb{E}_{z_2\sim q_2(z_2|x_2)}[\log p_{G_2}(x_2|z_2)].$$

$$\mathcal{L}_{\text{GAN}_1}(E_1,G_1,D_1)=\lambda_0\mathbb{E}_{x_1\sim P_{\mathcal{X}_1}}[\log D_1(x_1)]+\lambda_0\mathbb{E}_{z_2\sim q_2(z_2|x_2)}[\log(1-D_1(G_1(z_2)))]$$
$$\mathcal{L}_{\text{GAN}_2}(E_2,G_2,D_2)=\lambda_0\mathbb{E}_{x_2\sim P_{\mathcal{X}_2}}[\log D_2(x_2)]+\lambda_0\mathbb{E}_{z_1\sim q_1(z_1|x_1)}[\log(1-D_2(G_2(z_1)))].$$

$$\mathcal{L}_{\text{CC}_1}(E_1,G_1,E_2,G_2)=\lambda_3\text{KL}(q_1(z_1|x_1)||p_\eta(z))+\lambda_3\text{KL}(q_2(z_2|x_1^{1\to 2}))||p_\eta(z))-$$
$$\lambda_4\mathbb{E}_{z_2\sim q_2(z_2|x_1^{1\to 2})}[\log p_{G_1}(x_1|z_2)]$$
$$\mathcal{L}_{\text{CC}_2}(E_2,G_2,E_1,G_1)=\lambda_3\text{KL}(q_2(z_2|x_2)||p_\eta(z))+\lambda_3\text{KL}(q_1(z_1|x_2^{2\to 1}))||p_\eta(z))-$$
$$\lambda_4\mathbb{E}_{z_1\sim q_1(z_1|x_2^{2\to 1})}[\log p_{G_2}(x_2|z_1)].$$

Image to image translation

UNIT

- Explicit loss function:

$$
\min_{E_1,E_2,G_1,G_2} \max_{D_1,D_2} \mathcal{L}_{\text{VAE}_1}(E_1,G_1) + \mathcal{L}_{\text{GAN}_1}(E_1,G_1,D_1) + \mathcal{L}_{\text{CC}_1}(E_1,G_1,E_2,G_2)
$$
$$
\mathcal{L}_{\text{VAE}_2}(E_2,G_2) + \mathcal{L}_{\text{GAN}_2}(E_2,G_2,D_2) + \mathcal{L}_{\text{CC}_2}(E_2,G_2,E_1,G_1).
$$

$$
\begin{aligned}
\mathcal{L}_{\text{VAE}_1}(E_1,G_1) &= \lambda_1 \text{KL}(q_1(z_1|x_1)||p_\eta(z)) - \lambda_2 \mathbb{E}_{z_1\sim q_1(z_1|x_1)}[\log p_{G_1}(x_1|z_1)] \\
\mathcal{L}_{\text{VAE}_2}(E_2,G_2) &= \lambda_1 \text{KL}(q_2(z_2|x_2)||p_\eta(z)) - \lambda_2 \mathbb{E}_{z_2\sim q_2(z_2|x_2)}[\log p_{G_2}(x_2|z_2)].
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{L}_{\text{GAN}_1}(E_1,G_1,D_1) &= \lambda_0 \mathbb{E}_{x_1\sim P_{\mathcal{X}_1}}[\log D_1(x_1)] + \lambda_0 \mathbb{E}_{z_2\sim q_2(z_2|x_2)}[\log(1 - D_1(G_1(z_2)))] \\
\mathcal{L}_{\text{GAN}_2}(E_2,G_2,D_2) &= \lambda_0 \mathbb{E}_{x_2\sim P_{\mathcal{X}_2}}[\log D_2(x_2)] + \lambda_0 \mathbb{E}_{z_1\sim q_1(z_1|x_1)}[\log(1 - D_2(G_2(z_1)))].
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{L}_{\text{CC}_1}(E_1,G_1,E_2,G_2) &= \lambda_3 \text{KL}(q_1(z_1|x_1)||p_\eta(z)) + \lambda_3 \text{KL}(q_2(z_2|x_1^{1\to 2}))||p_\eta(z)) - \\
&\quad \lambda_4 \mathbb{E}_{z_2\sim q_2(z_2|x_1^{1\to 2})}[\log p_{G_1}(x_1|z_2)] \\
\mathcal{L}_{\text{CC}_2}(E_2,G_2,E_1,G_1) &= \lambda_3 \text{KL}(q_2(z_2|x_2)||p_\eta(z)) + \lambda_3 \text{KL}(q_1(z_1|x_2^{2\to 1}))||p_\eta(z)) - \\
&\quad \lambda_4 \mathbb{E}_{z_1\sim q_1(z_1|x_2^{2\to 1})}[\log p_{G_2}(x_2|z_1)].
\end{aligned}
$$

Image to image translation

CycleGAN

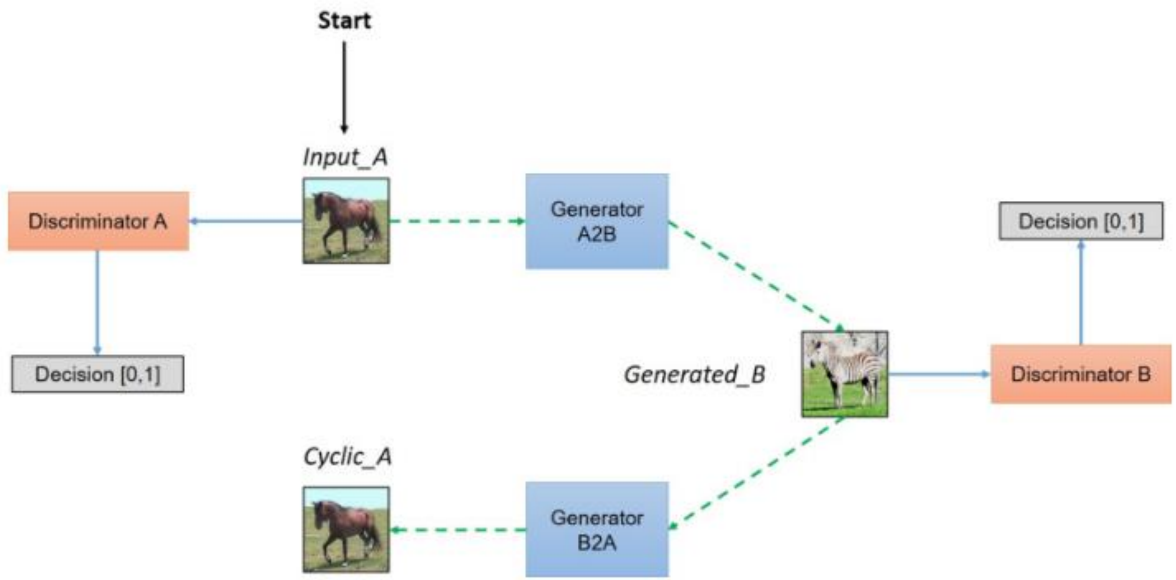


Image to image translation

CycleGAN

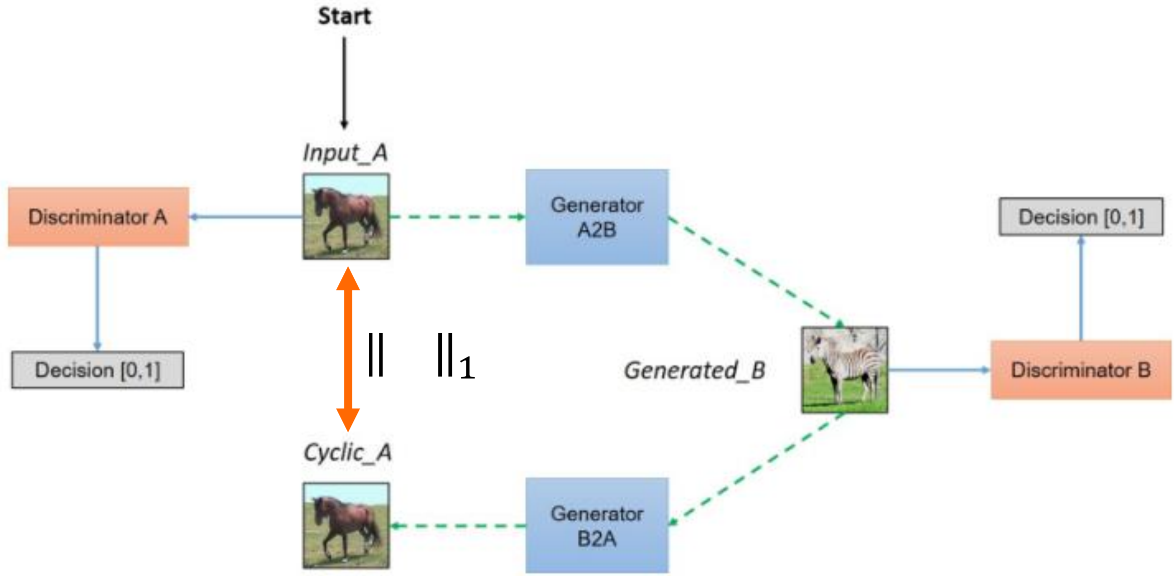


Image to image translation

CycleGAN

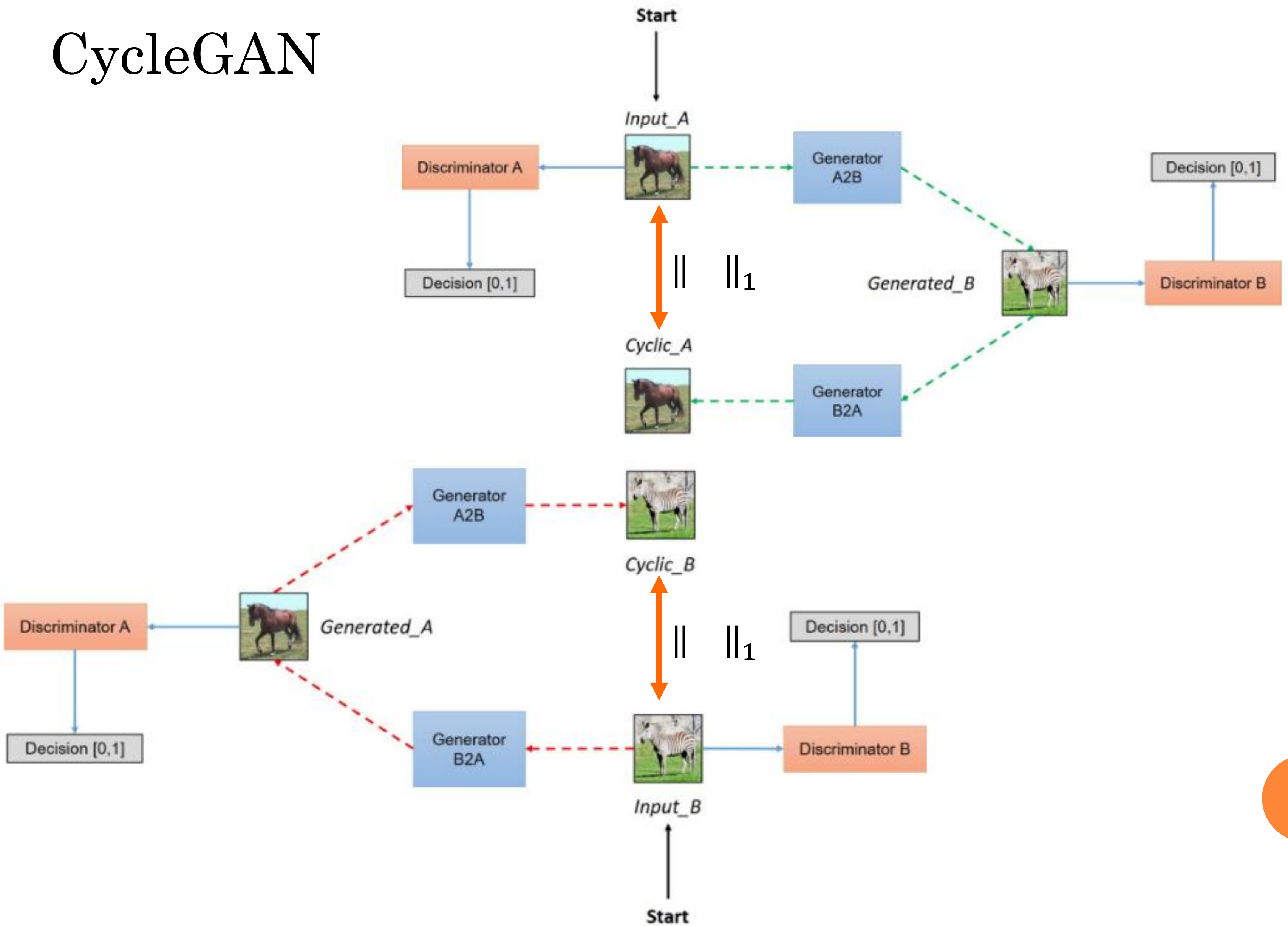


Image to image translation

CycleGAN

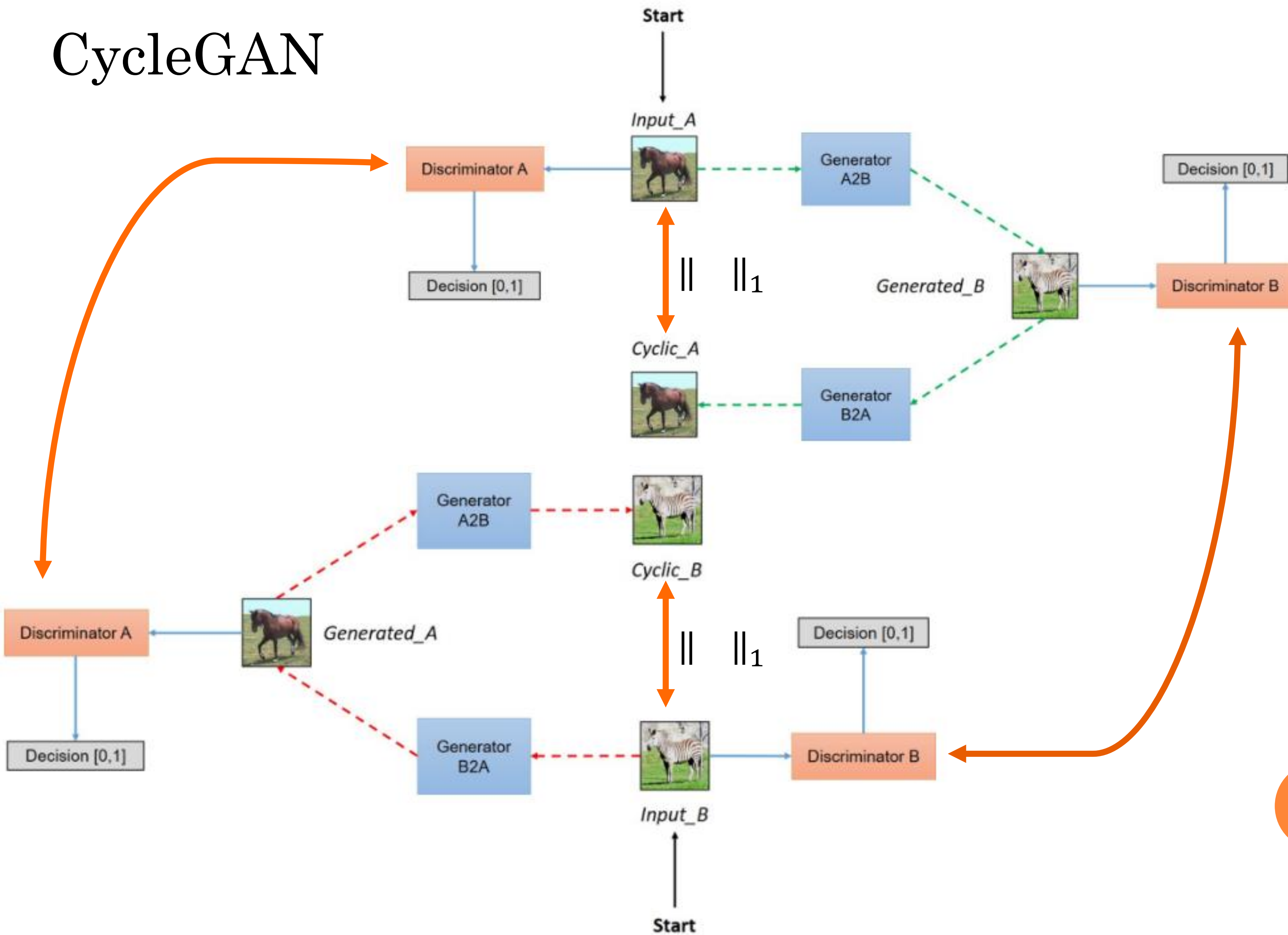


Image to image translation

CycleGAN

- Explicit loss function:

$$
\begin{aligned}
\mathcal{L}(G, F, D_X, D_Y) =&\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\
&+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\
&+ \lambda\mathcal{L}_{\text{cyc}}(G, F),
\end{aligned}
$$

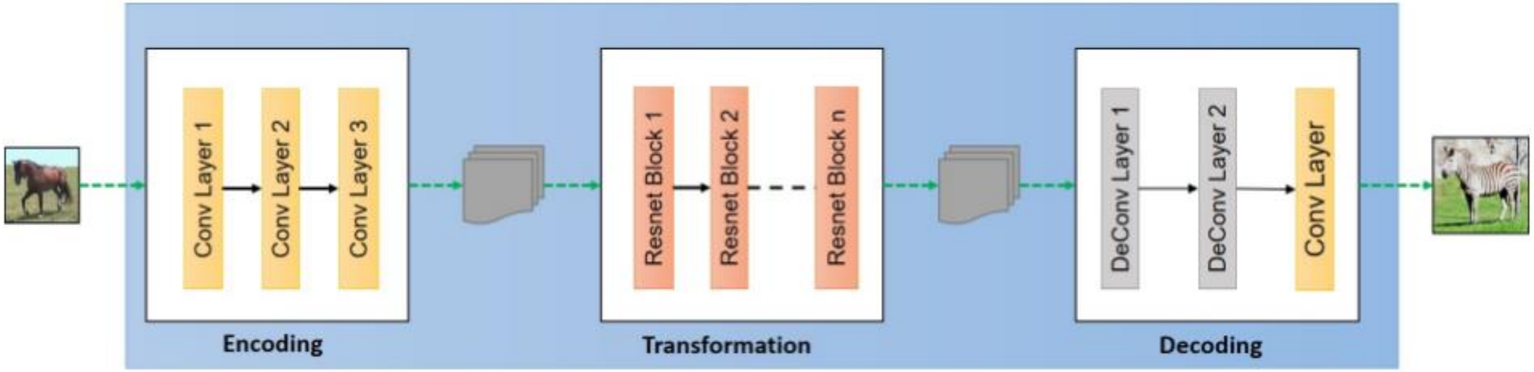
$$
\begin{aligned}
\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) =&\mathbb{E}_{y\sim p_{\text{data}}(y)}[\log D_Y(y)] \\
&+\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]
\end{aligned}
$$

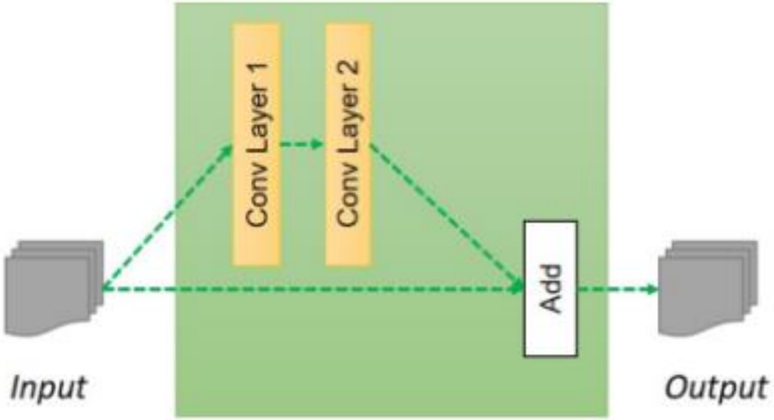
$$
\begin{aligned}
\mathcal{L}_{\text{cyc}}(G, F) =&\mathbb{E}_{x\sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\
&+\mathbb{E}_{y\sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].
\end{aligned}
$$

Image to image translation

CycleGAN

- Generator architecture:





Resnet Block

Image to image translation

CycleGAN

- Discriminator architecture

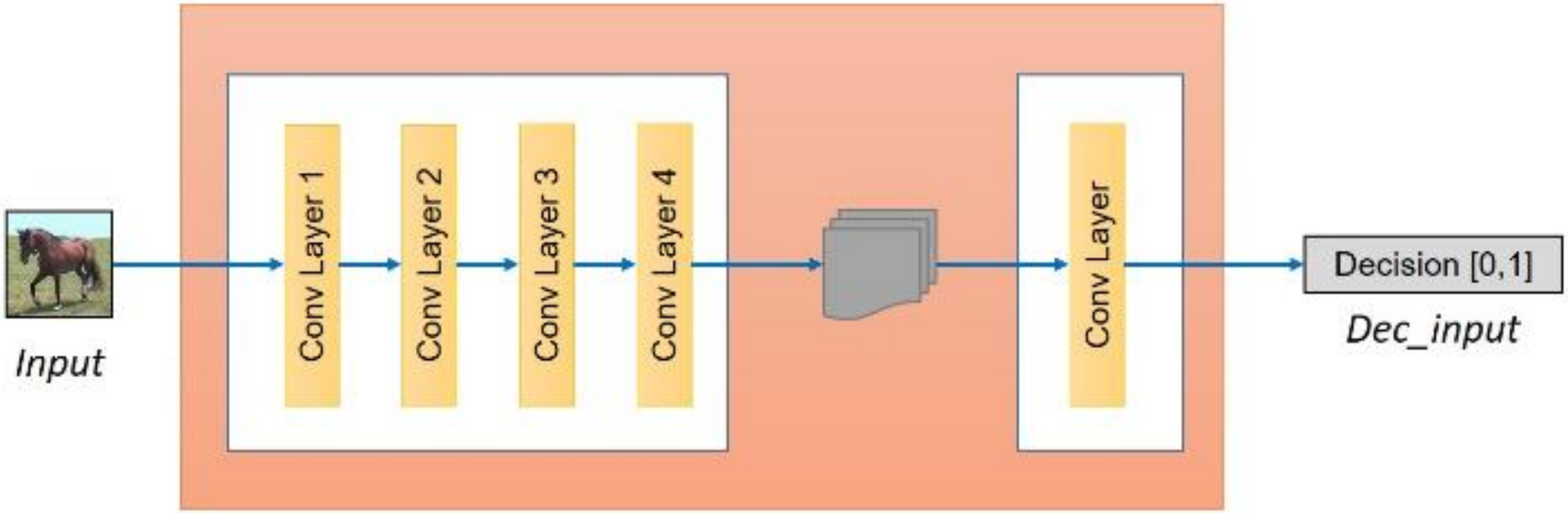


Image to image translation

CycleGAN

- Results:

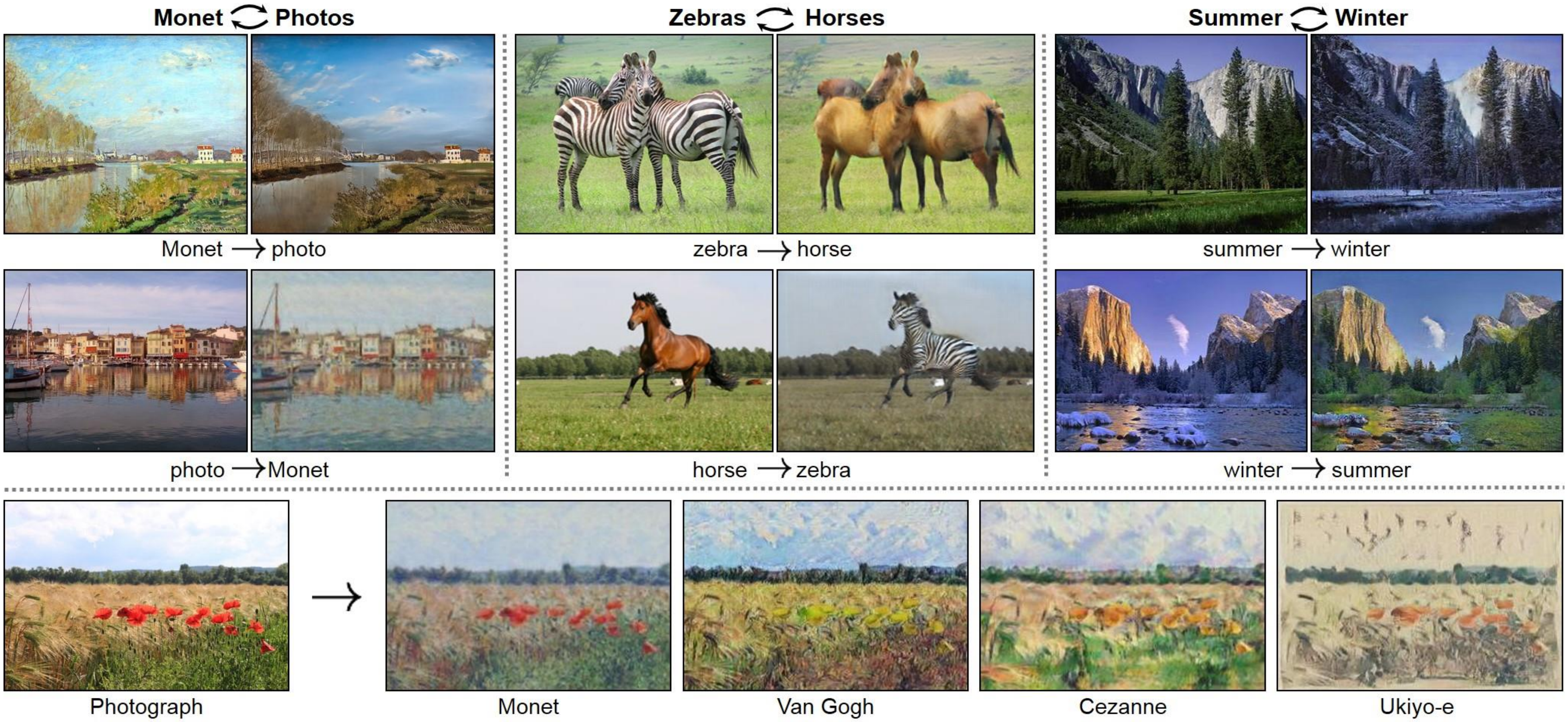


Image to image translation

CycleGAN

- Results: video

Image to imagne translation

UNIT

- Results:

night ↔ *day*

rain ↔ sunshine

IMAGE TO IMAGNE TRANSLATION

UNIT

- Results:

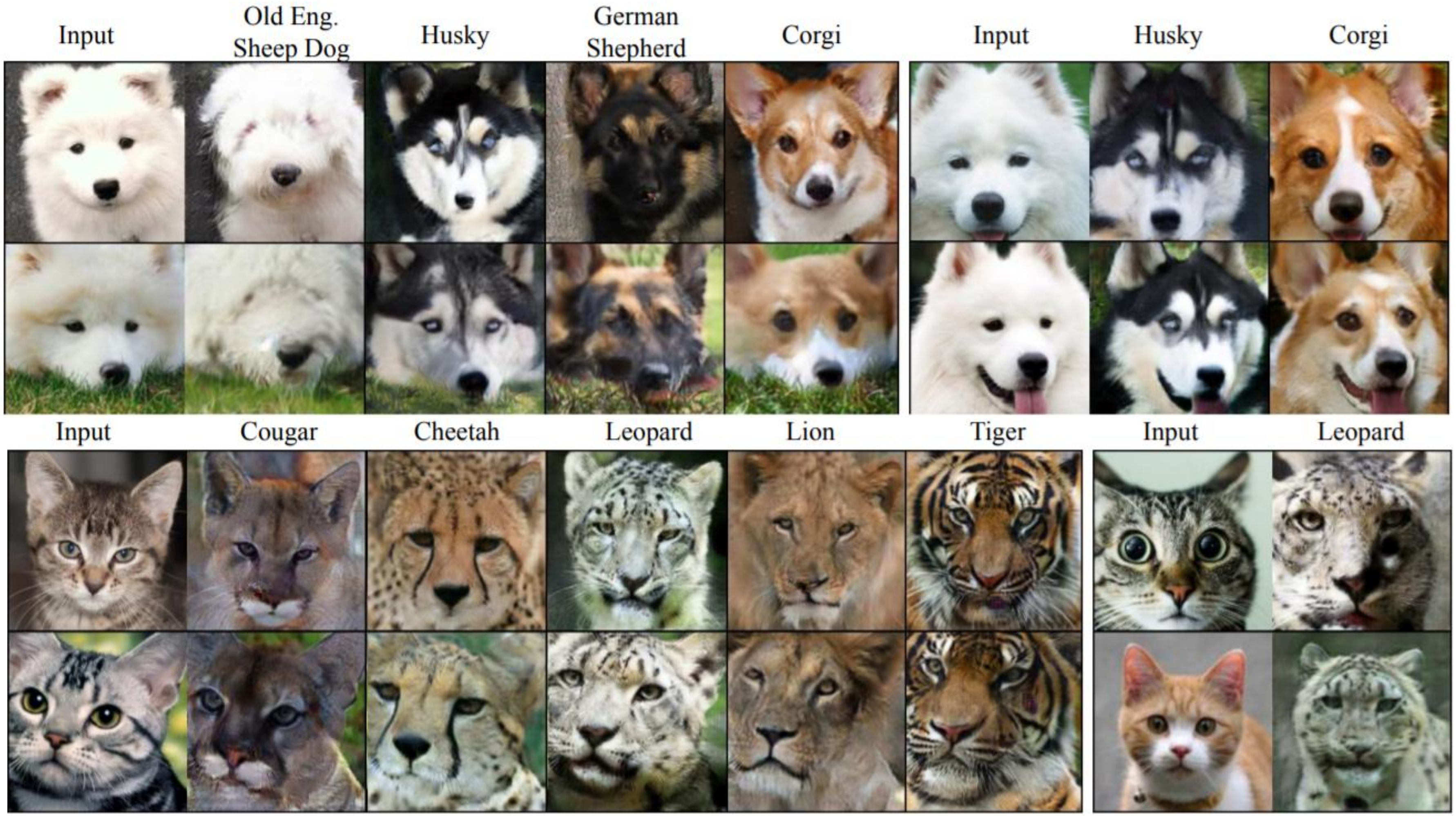


Image to imagne translation

UNIT

- Results:

Some more results CycleGAN

References

- Ian Goodfellow. “NIPS 2016 Tutorial: Generative Adversarial Networks” (2016)
- Jun-Yan Zhu, Tasung Park. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks” (2017)
- Ming-Yu Liu, Thomas Breuel. “Unsupervised Image-to-Image Translation Networks” (2017)
- Ian Goodfellow. “Generative adversarial nets” (2014)
- Christian Ledig. “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network“ (2017)

Thanks for listening

Tips And Tricks

- Labels improve subjective sample quality: (Denton et al 2015)
  - Learning a conditional model p(y|x) often gives much better samples from all classes than learning p(x) does.
- One-sided label smoothing: (Salimans et al 2016)

D_cost = cross_entropy(1., D(data)) + cross_entropy(0., D(samples))

D*_cost = cross_entropy(.9, D(data)) + cross_entropy(0., D(samples))

  - Prevent extreme extrapolation behavior in the discriminator.
  - Excellent regularizer that doesn't encourage misclassification.