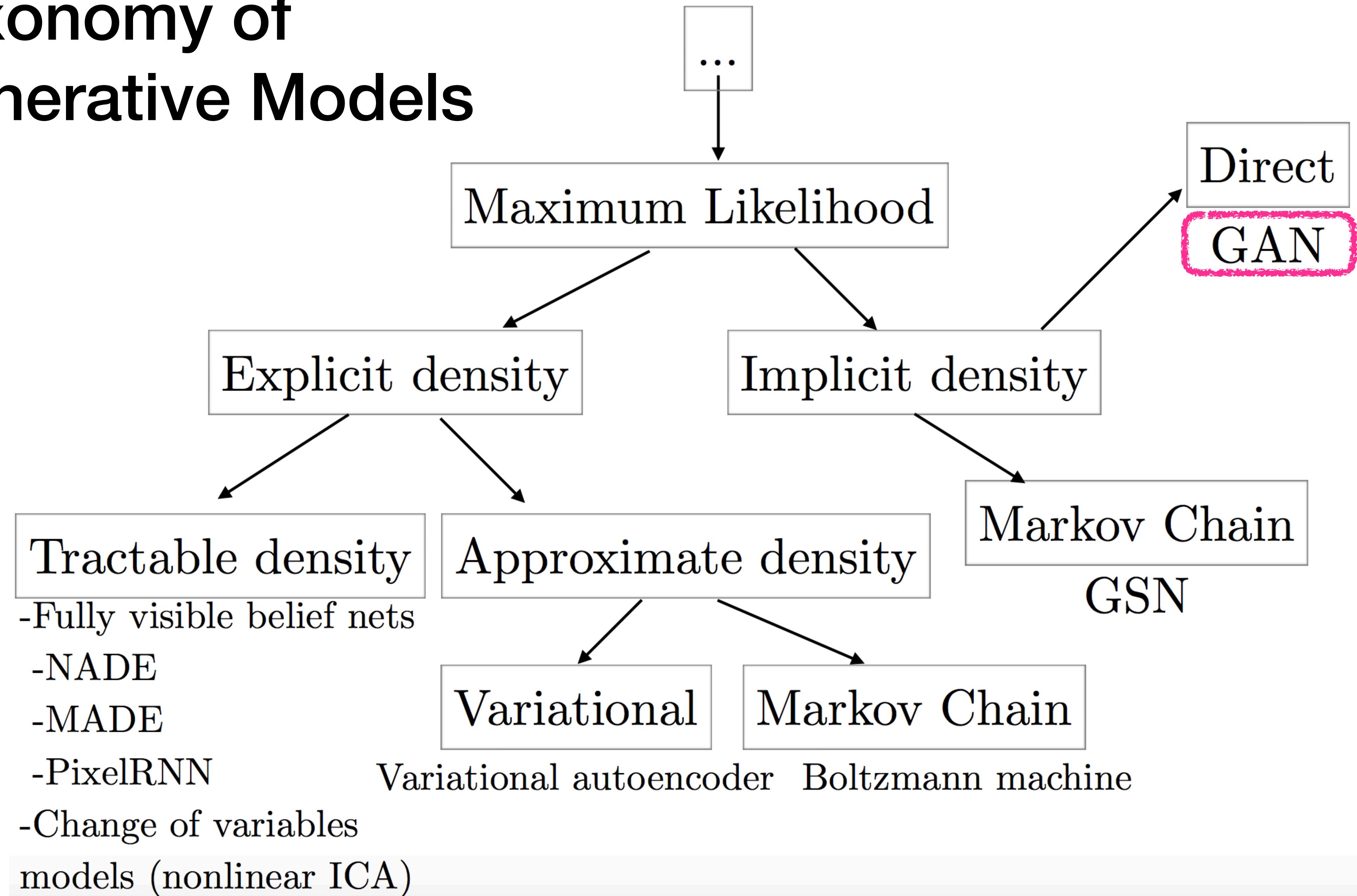


# Generative Adversarial Networks - 1

2021  
Multicampus

**Il Gu Yi**  
ModuLabs, Research Scientist  
**Soochul Park**  
GaudioLab, Research Scientist

# Taxonomy of Generative Models



# Generative Adversarial Networks (**GANs**)

UDACITY TALKS

---

# Yann LeCun

Director of AI Research

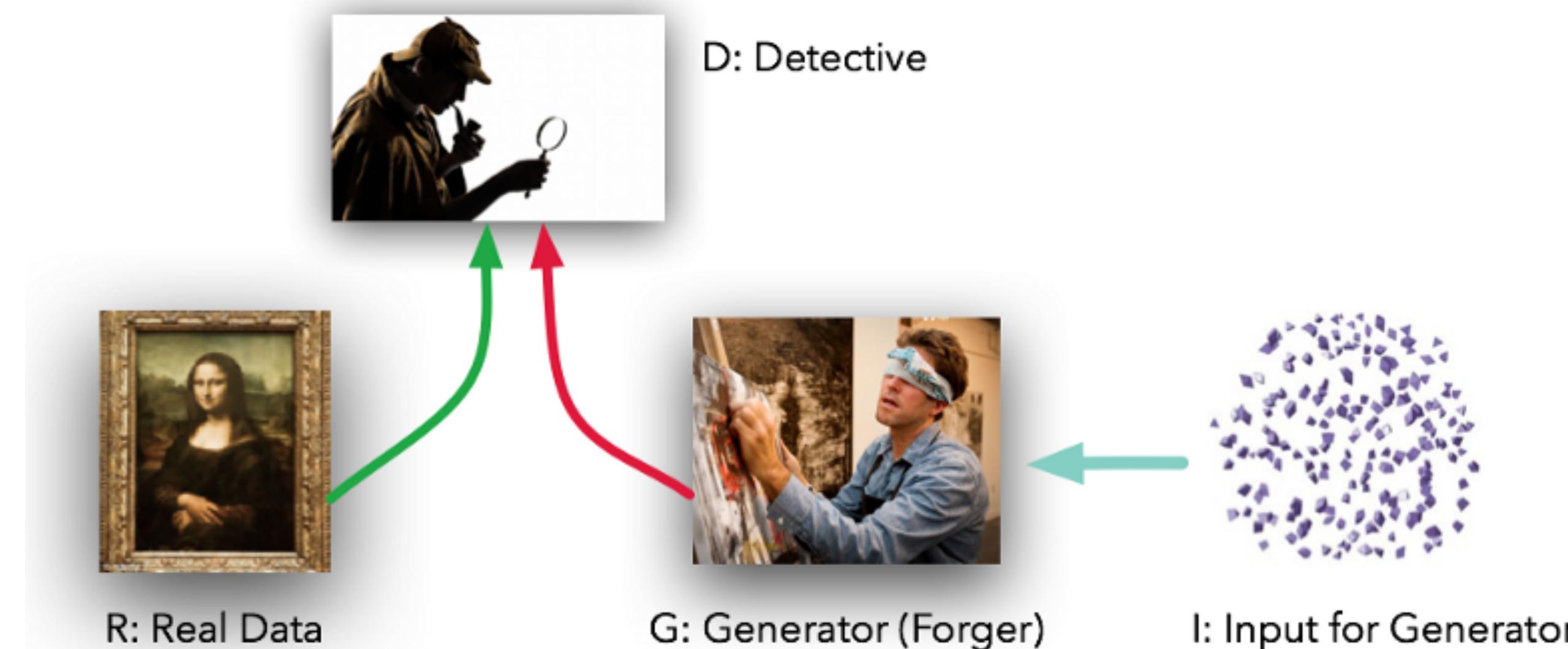
facebook



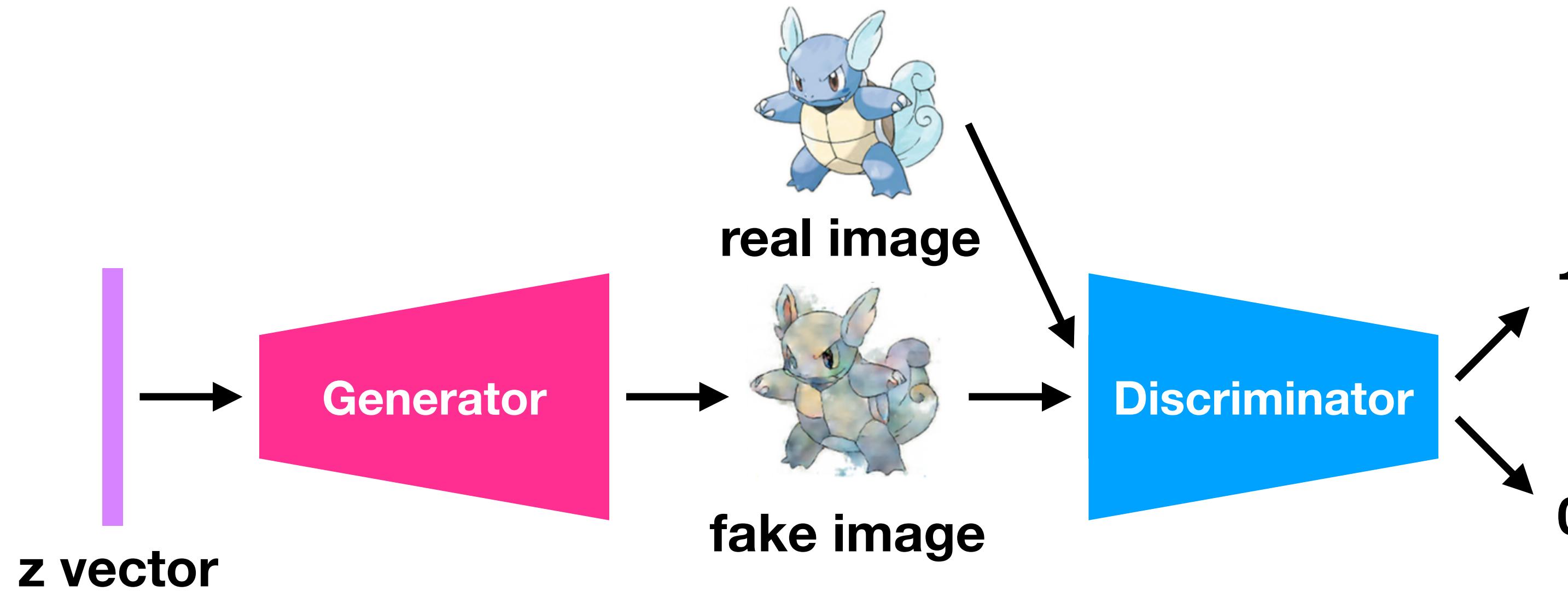
**“The most important idea in Machine Learning in  
the last 20 years”**

# Generative Adversarial Networks Framework

- GAN은 Generative Adversarial Networks의 약자로 두 개의 네트워크가 서로 독립적으로 경쟁하며 트레이닝이 진행됩니다.
- 이는 눈을 가린채 그림을 그리는 화가와 그림을 판별하는 탐정에 비유할 수 있습니다.
- 화가는 진품과 모방한 모조품을 그리는 역할을 합니다. 하지만 진품을 직접 볼 수는 없고, 오직 탐정이 가려내는 결과만을 들을 수 있습니다. 많은 시행착오 끝에 결국 탐정이 진품과 모조품을 구별할 수 있도록 만듭니다.
- 탐정은 화가가 그리는 모조품과 진품을 받고서 서로 구별하는 능력을 키웁니다. 그리고 구별한 결과를 화가에게 알려줍니다.



# Generative Adversarial Networks Framework



- 전체 네트워크는 그림을 생성하는 네트워크인 generator와 그림이 진짜인지 가짜인지 판별하는 discriminator로 이루어져 있습니다.
- Generator는 임의의 random distribution으로부터 샘플링한 z vector를 사용해 그림을 생성하며, 이를 fake image라 칭합니다.
- Discriminator는 generator가 생성한 fake image와 trainset으로 준비된 real image를 입력받아 각각 0과 1을 출력하도록 학습합니다.

# GANs Framework

- The **Discriminator**( $\mathcal{D}$ ): minimize  $J_D(\Theta_D; \Theta_G)$  using only  $\Theta_D$
- The **Generator**( $\mathcal{G}$ ): minimize  $J_G(\Theta_G; \Theta_D)$  using only  $\Theta_G$
- Each player's cost depends on the other player's parameters
  - But each player cannot control the other player's parameters
- Describe as the game rather than the optimization problem
  - The solution to a game is a Nash equilibrium



# GAN Formulation

- Generator: a function from latent space(also called representation space) to real space
  - $\mathcal{G} : \mathcal{G}(z) \rightarrow \mathbb{R}^{|x|}, z \in \mathbb{R}^{|z|}$  is a sample in latent space
- Discriminator: a function from input space to the probability space
  - $\mathcal{D} : \mathcal{D}(x) \rightarrow [0, 1]$
- When the real data distribution and distribution generated by generator are the exactly same, discriminator cannot distinguish between each other
  - It means the optimal solution



# Notation

- $p_{\text{data}}(\mathbf{x})$  : probability density function of real data
- $p_g(\mathbf{x})$  : probability density function of the samples generated by the generator
- $\mathcal{G}$ : generator networks,  $\mathcal{D}$  : discriminator networks
- $\Theta_G, \Theta_D$  : parameters of  $\mathcal{G}$  and  $\mathcal{D}$ , respectively
- Cost (objective) function
  - $J_G(\Theta_G; \Theta_D)$  ,  $J_D(\Theta_D; \Theta_G)$
  - Two player game frameworks



# Cost Functions

- Discriminator cost,  $J^{(D)}$

$$J^{(D)}(\Theta_D; \Theta_G) = -\frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] - \frac{1}{2} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} [\log(1 - D(G(\mathbf{z})))]$$

- Generator cost,  $J^{(G)}$

- Minimax game

$$J^{(G)} = -J^{(D)}$$

- Non-saturating game

$$J^{(G)} = -\frac{1}{2} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \log D(G(\mathbf{z}))$$



# Cost Functions

- Discriminator cost,  $J^{(D)}$

$$J^{(D)}(\Theta_D; \Theta_G) = -\frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] - \frac{1}{2} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} [\log(1 - D(G(\mathbf{z})))]$$

- Cost function is derived from binary cross entropy

$$\begin{aligned} & \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [H[Ber(y; \mu = 1), Ber(y; \mu = D(x))]] \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [-Ber(1; \mu = 1)\log Ber(1; \mu = D(x)) - Ber(0; \mu = 1)\log Ber(0; \mu = D(x))] \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [-1 \log D(x) - 0 \log(1 - D(x))] \\ &= -\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(x)] \end{aligned}$$

Cross-Entropy

$$\begin{aligned} H(p, q) &= -\mathbb{E}_p [\log q] \\ &= -\sum_{x \in X} p(x) \log q(x), \text{ for discrete} \end{aligned}$$

$$\begin{aligned} & \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [H[Ber(y; \mu = 0), Ber(y; \mu = D(G(\mathbf{z})))]] \\ &= \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [-Ber(1; \mu = 0)\log Ber(1; \mu = D(G(\mathbf{z}))) - Ber(0; \mu = 0)\log Ber(0; \mu = D(G(\mathbf{z})))] \\ &= \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [-0 \log D(G(\mathbf{z})) - 1 \log(1 - D(G(\mathbf{z})))] \\ &= -\mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] \end{aligned}$$

# Minimax Game

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

- Good for theoretical analysis
- Equilibrium is a saddle point of the discriminator loss
- Resembles Jensen-Shannon divergence
- Generator minimizes the log-probability of the discriminator being correct



# Minimax Game

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

- Minimax game
  - Theoretical analysis (good) but In practice (bad)
  - Discriminator minimize but Generator maximize the same cross-entropy
  - When  $D(G(\mathbf{z})) = 0$ ,  $G$ 's gradient vanishes



# Non-Saturating Game

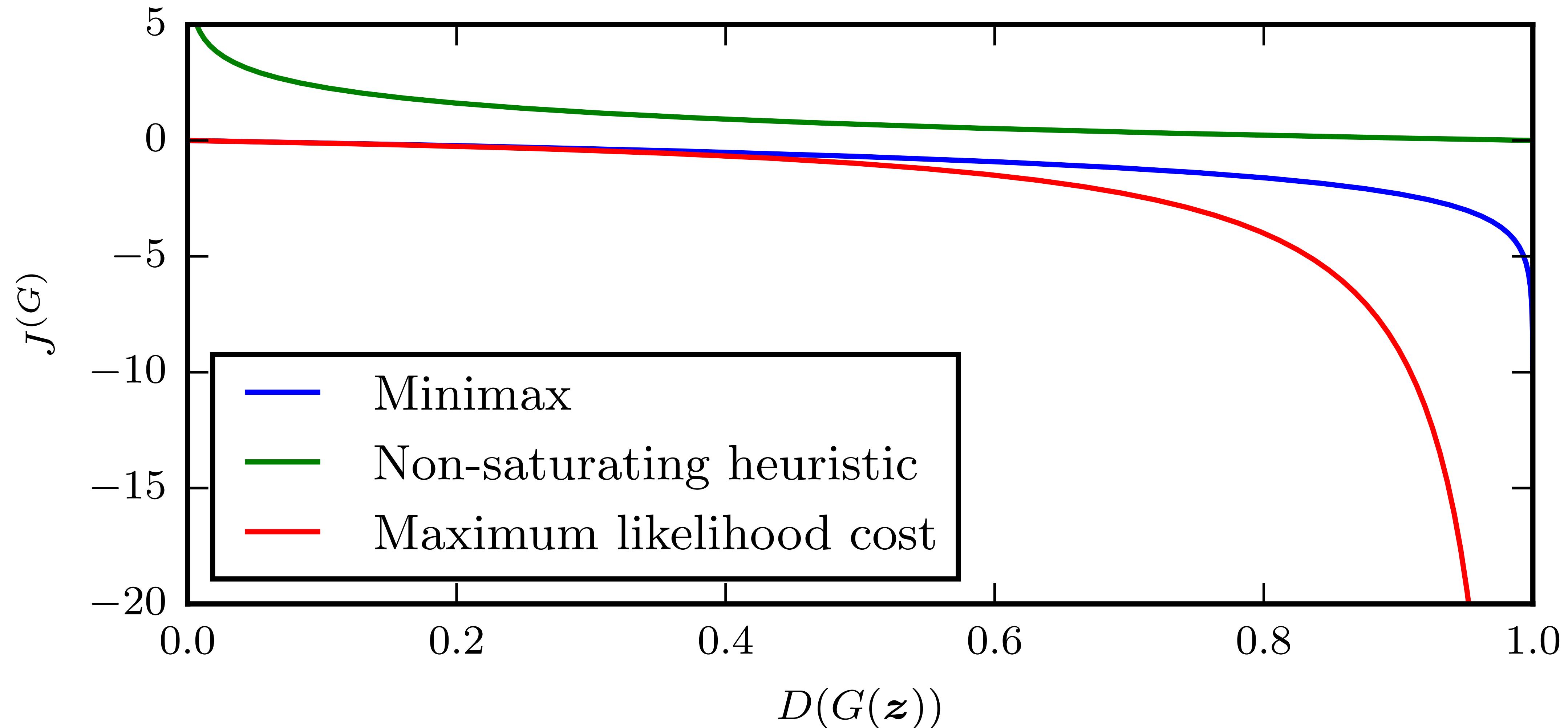
$$J^{(D)}(\Theta_D; \Theta_G) = -\frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] - \frac{1}{2} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} [\log(1 - D(G(\mathbf{z})))]$$

$$J^{(G)} = -\frac{1}{2} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \log D(G(\mathbf{z}))$$

- Equilibrium no longer describable with a single loss
- Generator maximizes the log-probability of the discriminator being mistaken
- Heuristically motivated; generator can still learn even when discriminator successfully rejects all generator samples



# Comparison of Generator Losses

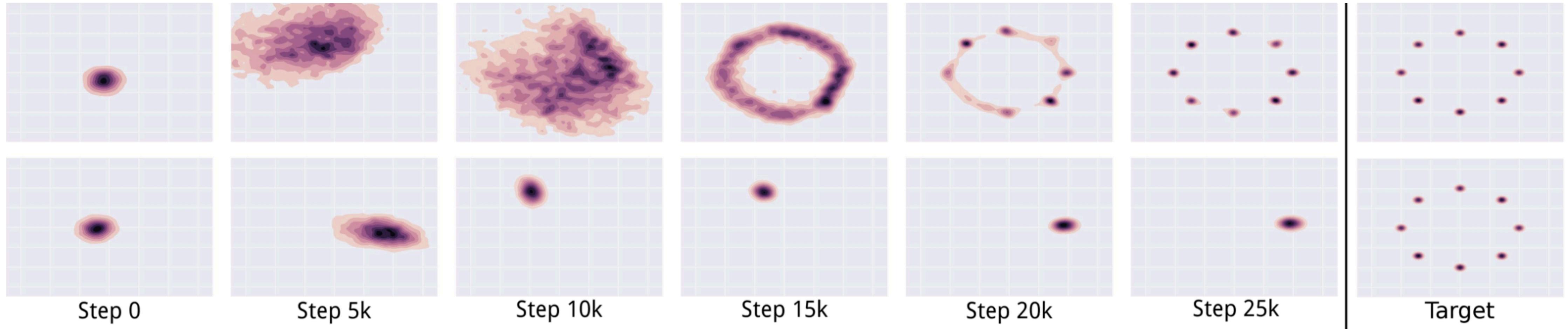


# Difficulties in GAN Training

- Difficulties in getting the pair of models to converge
- Mode collapsing
  - To generate very similar samples for different inputs
- The discriminator loss converging quickly to zero
  - No gradient updates to the generator



# Mode Collapse



The generator distribution keeps oscillating between different modes.  
It never converges to a fixed distribution.



# Mode Collapse



Standard GAN procedure:  
Generator is an MLP with 4 hidden layers of 512 units  
Discriminator is a DCGAN



# Evolution of GANs

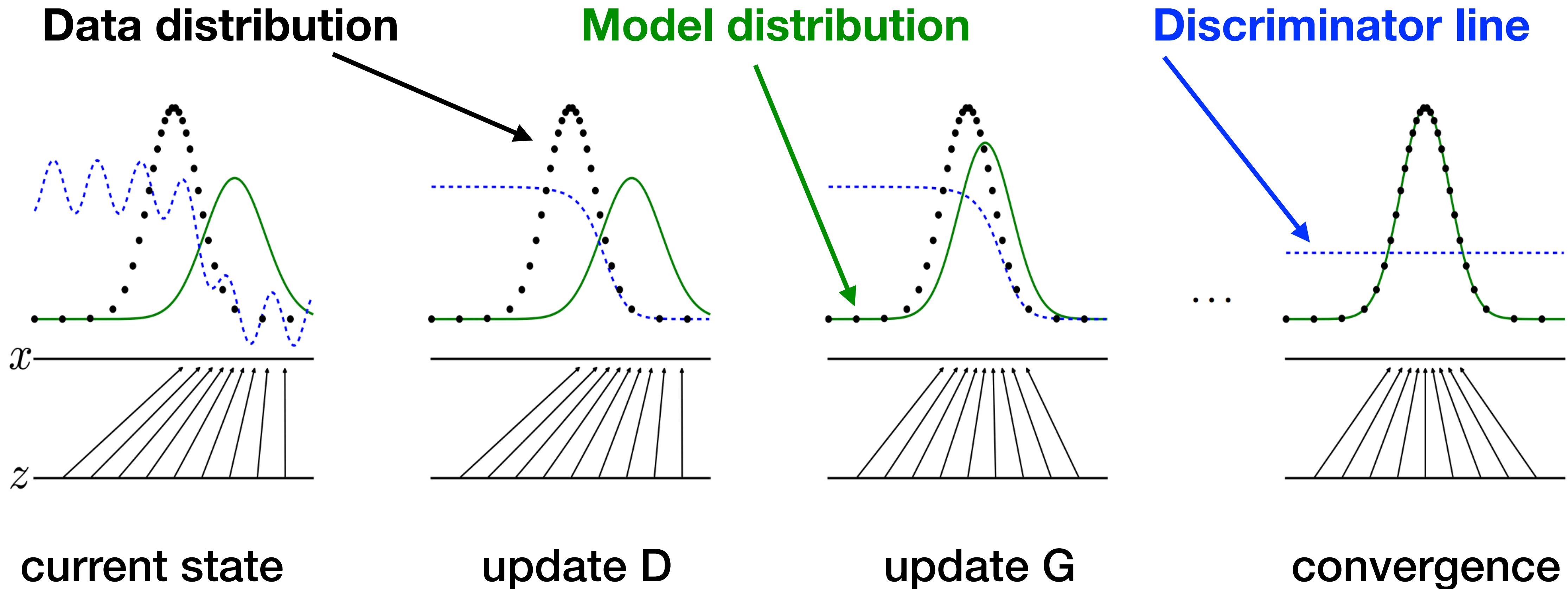
# **Generative Adversarial Net (GAN)**

# Original GANs

- Ian Goodfellow proposed GAN framework first
- This time  $\mathcal{G}, \mathcal{D}$  are **fully connected neural networks**
- MNIST, CIFAR-10, Toronto Face Dataset(TFD)



# Training of GANs



# Training Algorithm of GANs

---

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

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

**end for**

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

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

**end for**

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

---



# Optimal Solution for the Discriminator

## 4.1 Global Optimality of $p_g = p_{\text{data}}$

We first consider the optimal discriminator  $D$  for any given generator  $G$ .

**Proposition 1.** *For  $G$  fixed, the optimal discriminator  $D$  is*

$$D_G^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \quad (2)$$

*Proof.* The training criterion for the discriminator  $D$ , given any generator  $G$ , is to maximize the quantity  $V(G, D)$

$$\begin{aligned} V(G, D) &= \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log(D(\mathbf{x})) d\mathbf{x} + \int_z p_{\mathbf{z}}(\mathbf{z}) \log(1 - D(g(\mathbf{z}))) dz \\ &= \int_{\mathbf{x}} p_{\text{data}}(\mathbf{x}) \log(D(\mathbf{x})) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) d\mathbf{x} \end{aligned} \quad (3)$$

For any  $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$ , the function  $y \rightarrow a \log(y) + b \log(1 - y)$  achieves its maximum in  $[0, 1]$  at  $\frac{a}{a+b}$ . The discriminator does not need to be defined outside of  $\text{Supp}(p_{\text{data}}) \cup \text{Supp}(p_g)$ , concluding the proof.  $\square$



# Jensen-Shannon Divergence

**Theorem 1.** *The global minimum of the virtual training criterion  $C(G)$  is achieved if and only if  $p_g = p_{\text{data}}$ . At that point,  $C(G)$  achieves the value  $-\log 4$ .*

*Proof.* For  $p_g = p_{\text{data}}$ ,  $D_G^*(\mathbf{x}) = \frac{1}{2}$ , (consider Eq. 2). Hence, by inspecting Eq. 4 at  $D_G^*(\mathbf{x}) = \frac{1}{2}$ , we find  $C(G) = \log \frac{1}{2} + \log \frac{1}{2} = -\log 4$ . To see that this is the best possible value of  $C(G)$ , reached only for  $p_g = p_{\text{data}}$ , observe that

$$\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [-\log 2] + \mathbb{E}_{\mathbf{x} \sim p_g} [-\log 2] = -\log 4$$

and that by subtracting this expression from  $C(G) = V(D_G^*, G)$ , we obtain:

$$C(G) = -\log(4) + KL \left( p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2} \right) + KL \left( p_g \middle\| \frac{p_{\text{data}} + p_g}{2} \right) \quad (5)$$

where KL is the Kullback–Leibler divergence. We recognize in the previous expression the Jensen–Shannon divergence between the model’s distribution and the data generating process:

$$C(G) = -\log(4) + 2 \cdot JSD(p_{\text{data}} \| p_g) \quad (6)$$

Since the Jensen–Shannon divergence between two distributions is always non-negative and zero only when they are equal, we have shown that  $C^* = -\log(4)$  is the global minimum of  $C(G)$  and that the only solution is  $p_g = p_{\text{data}}$ , i.e., the generative model perfectly replicating the data generating process.  $\square$



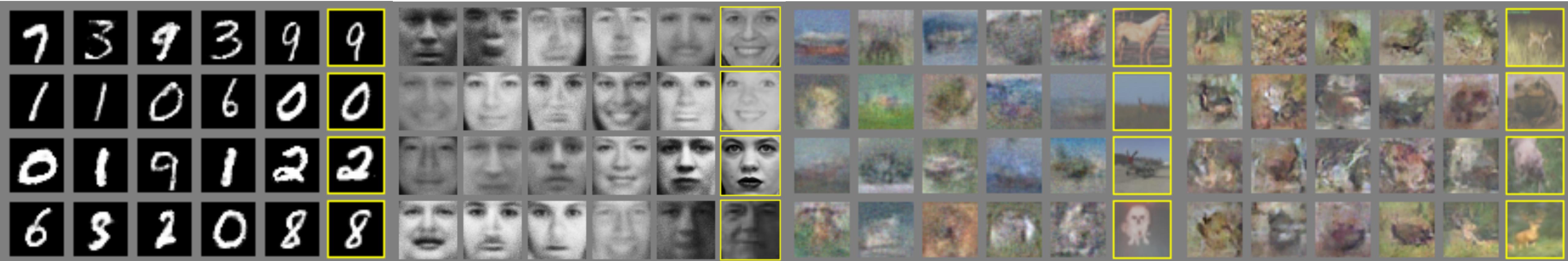
# Jensen-Shannon Divergence

$$\min_G \max_D V(D, G) = \min_G V(D^*, G) \quad \text{for fixed optimal } \mathcal{D}$$

$$\begin{aligned} V(D^*, G) &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D^*(\mathbf{x})] + \mathbb{E}_{\mathbf{x} \sim p_g} [\log(1 - D^*(\mathbf{x}))] \\ &= \int_{\mathbf{x}} d\mathbf{x} \ p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} + \int_{\mathbf{x}} d\mathbf{x} \ p_g(\mathbf{x}) \log \frac{p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \\ &= -2 \log 2 + 2 \log 2 + \int_{\mathbf{x}} d\mathbf{x} \ p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} + \int_{\mathbf{x}} d\mathbf{x} \ p_g(\mathbf{x}) \log \frac{p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \\ &= -\log 4 + \int_{\mathbf{x}} d\mathbf{x} \ p_{\text{data}}(\mathbf{x}) \log \frac{2 \cdot p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} + \int_{\mathbf{x}} d\mathbf{x} \ p_g(\mathbf{x}) \log \frac{2 \cdot p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})} \\ &= -\log 4 + KL \left( p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2} \right) + KL \left( p_g \middle\| \frac{p_{\text{data}} + p_g}{2} \right) \\ &= -\log 4 + 2 \cdot JSD(p_{\text{data}} || p_g) \end{aligned}$$



# GAN Results



MNIST data

TFD data

CIFAR10 data  
(fully connected)

CIFAR10 data  
(convolutional)

# **Deep Convolutional GAN**

## **(DCGAN)**

# Guidelines for Stable DCGAN

-DCGAN은 GAN에 deep convolutional networks를 적용한 모델로써 GAN이 활발히 연구되게 한 시작점입니다.

- GAN의 트레이닝은 매우 불안정하여 많은 데이터를 오래 트레이닝한다고 해서 성공적인 결과를 낳는 것이 아닙니다. 모델의 구성요소들 즉, layer의 종류, 층 수, 채널 수, 커널의 크기, activation, normalization의 종류와 유무, optimizer의 종류와 learning rate 모든 것이 잘 맞아떨어져야 제대로 트레이닝됩니다.
- 어떤 구성요소가 왜 성공적으로 트레이닝을 이끄는지 그 이유에 대해서는 명확하게 밝혀내지 못합니다. 단지 많은 실험에 의해 어떤 구성요소들이 좋은 결과를 낳는지 알아내고 있습니다.

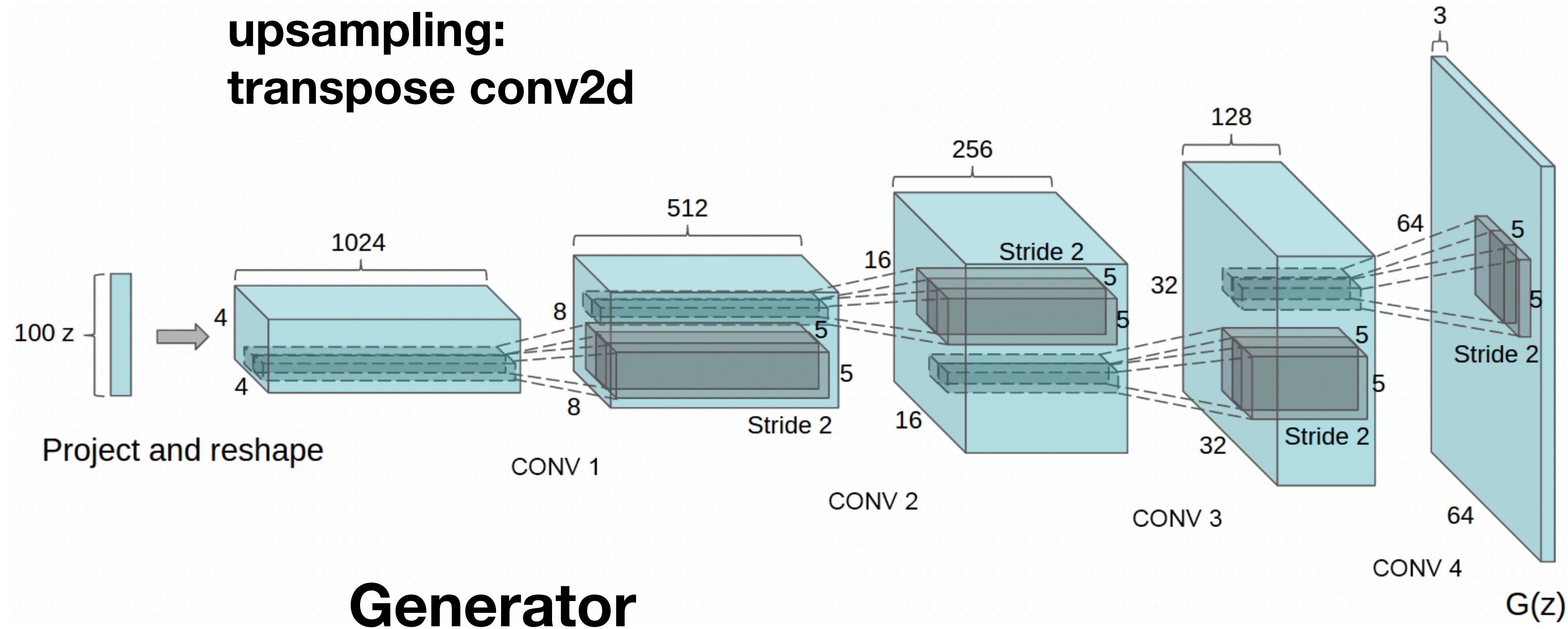
## Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

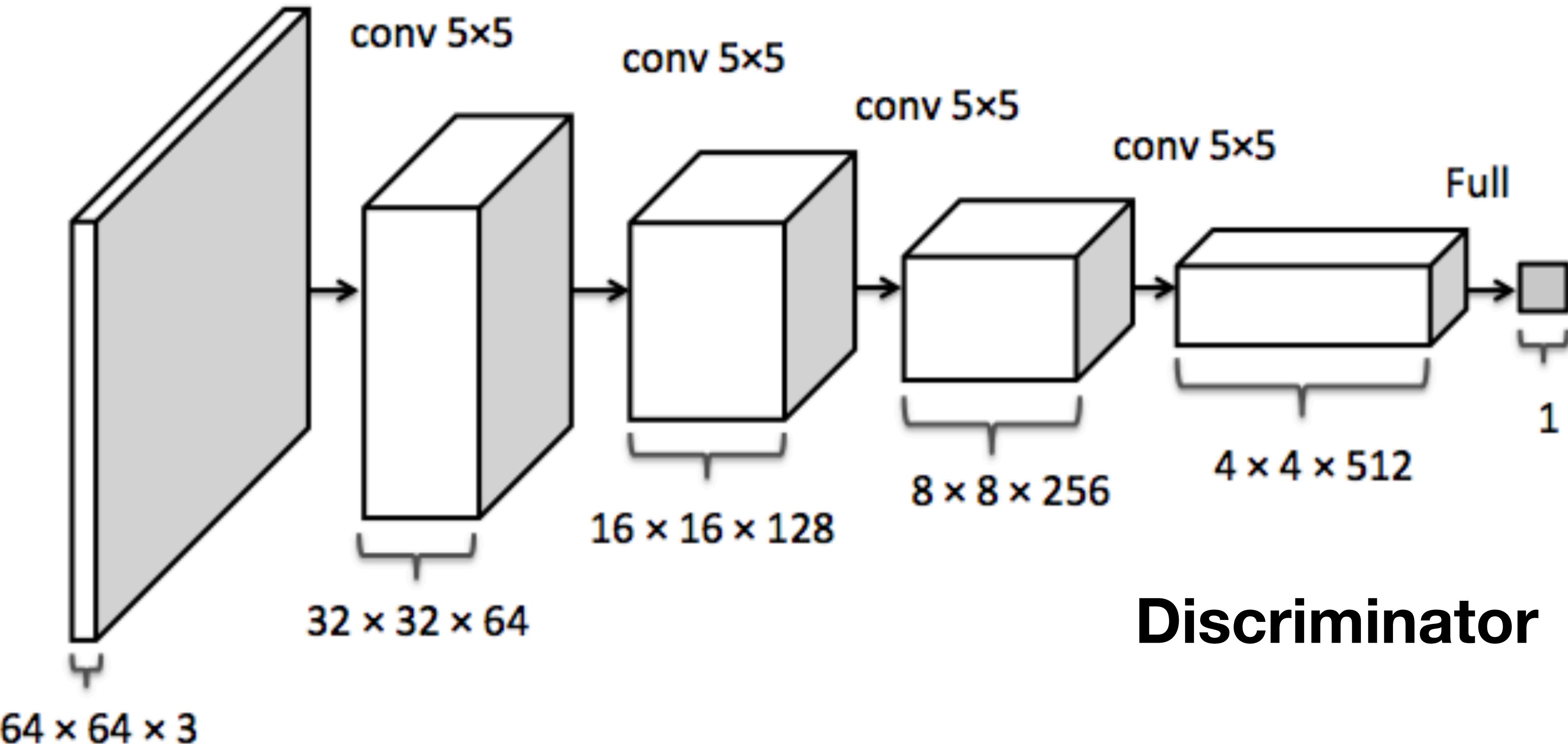


# DCGAN Architectures

**upsampling:**  
**transpose conv2d**



# DCGAN Architectures



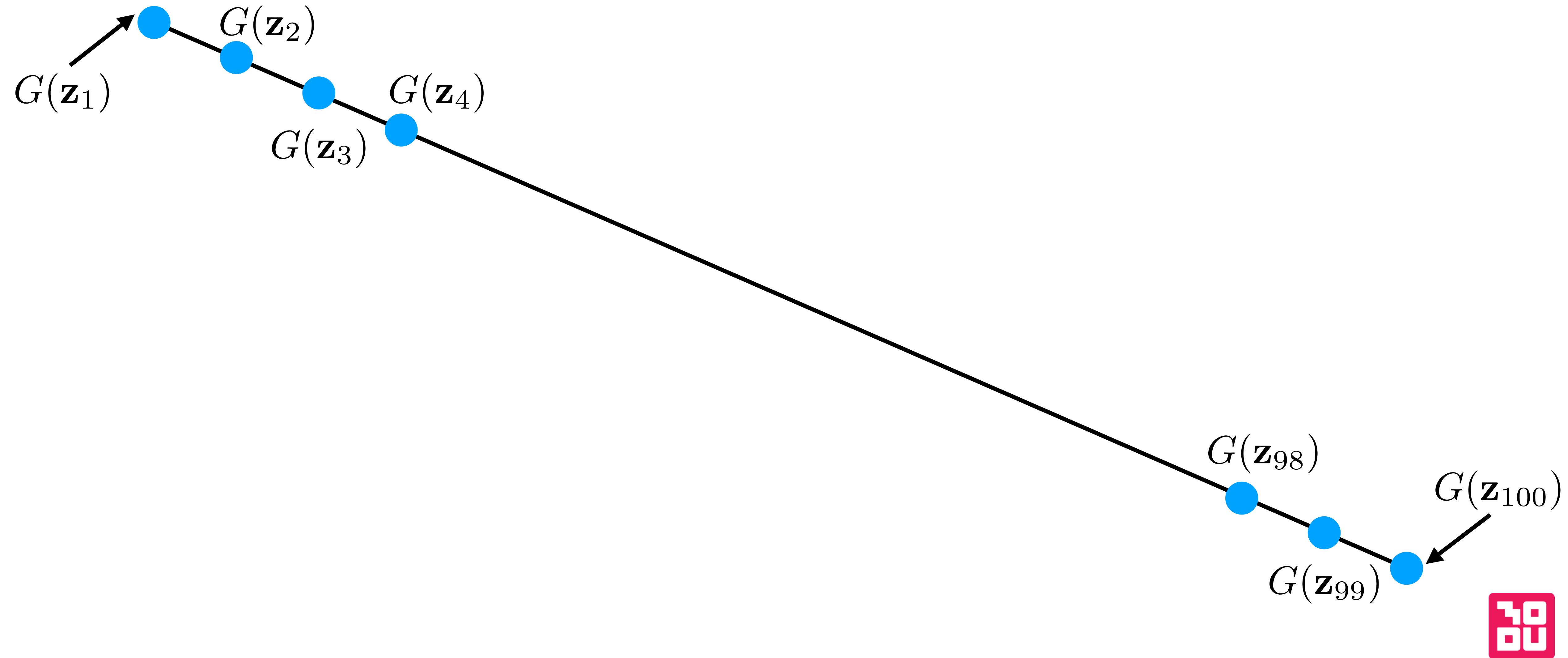
# DCGAN Results



# DCGAN Results (Interpolation)



# DCGAN Results (Interpolation)



# DCGAN Results (Vector Arithmetic in Latent Space)

$$\begin{matrix} \text{smiling} \\ \text{woman} \end{matrix} - \begin{matrix} \text{neutral} \\ \text{woman} \end{matrix} + \begin{matrix} \text{neutral} \\ \text{man} \end{matrix} =$$



smiling man



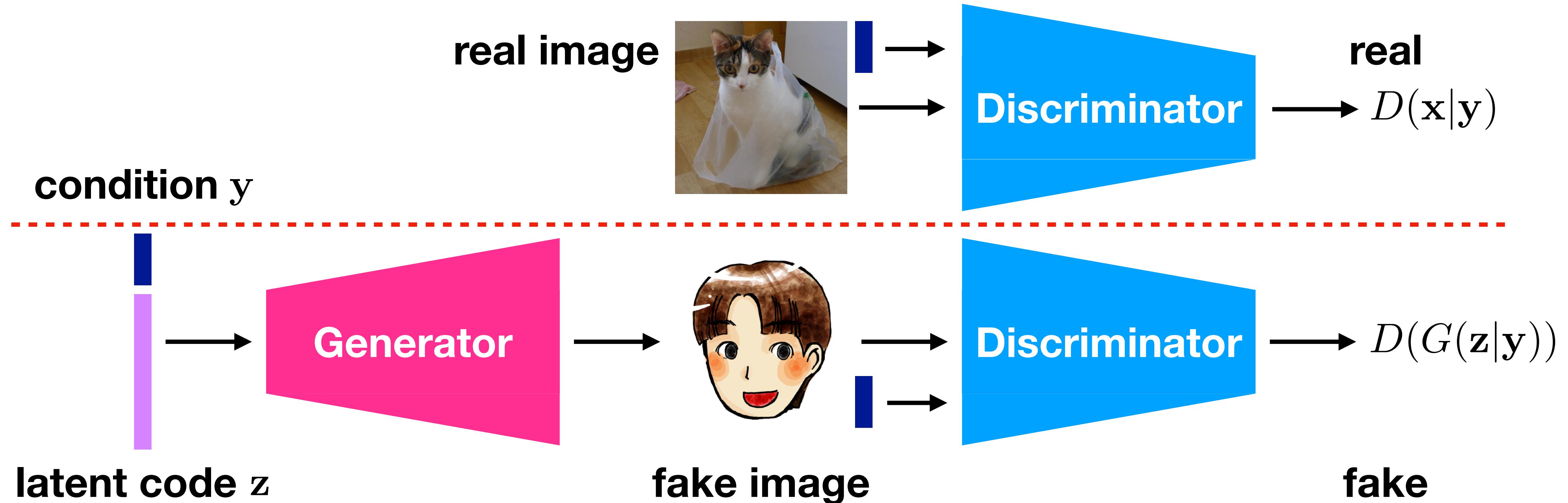
# Conditional GAN (cGAN)

# Conditional GAN

- Add a class conditional structure into Generator and Discriminator
- Conditional GAN can learn multi-modal structures from labels

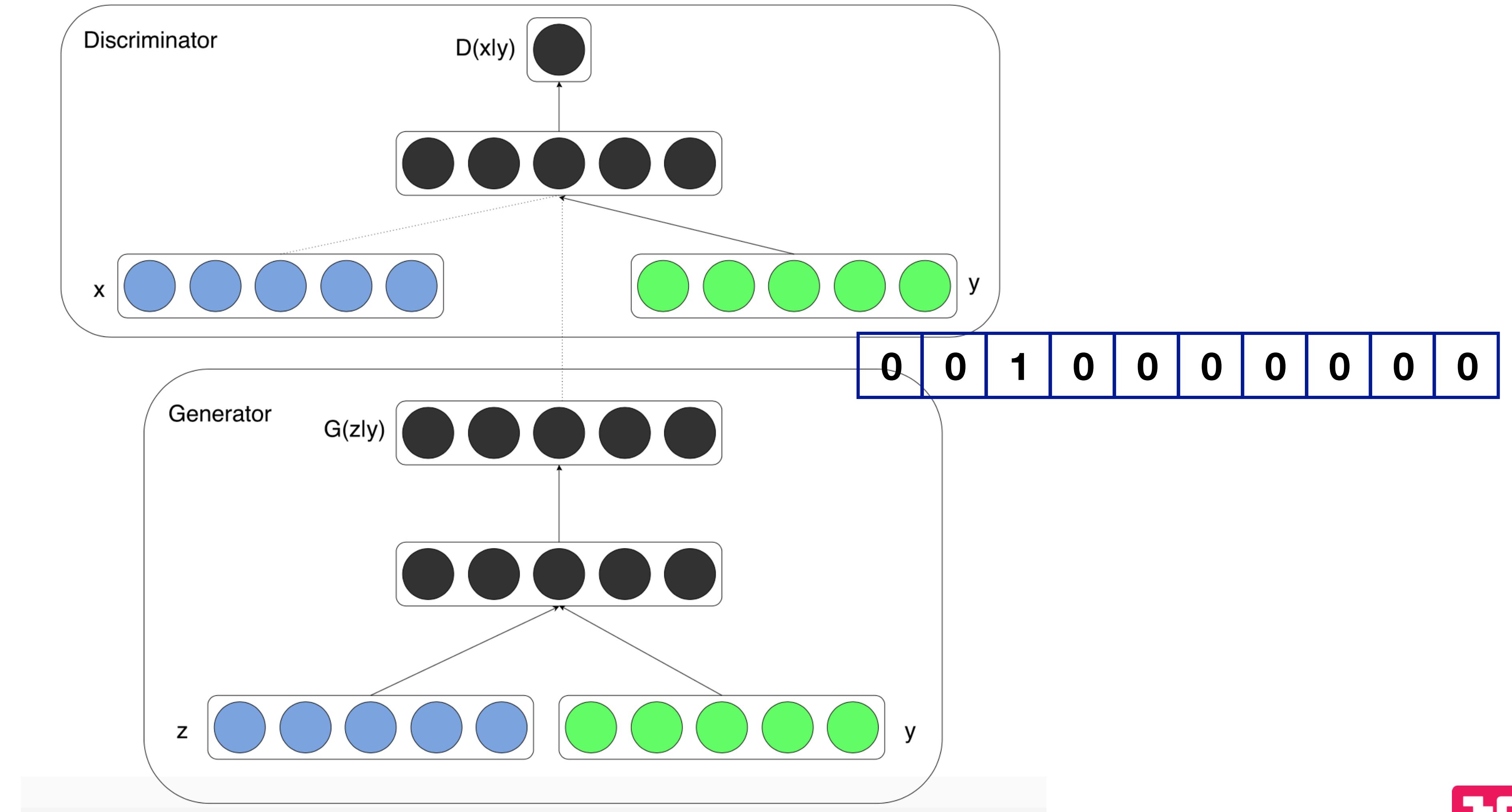


# Conditional GAN Architectures

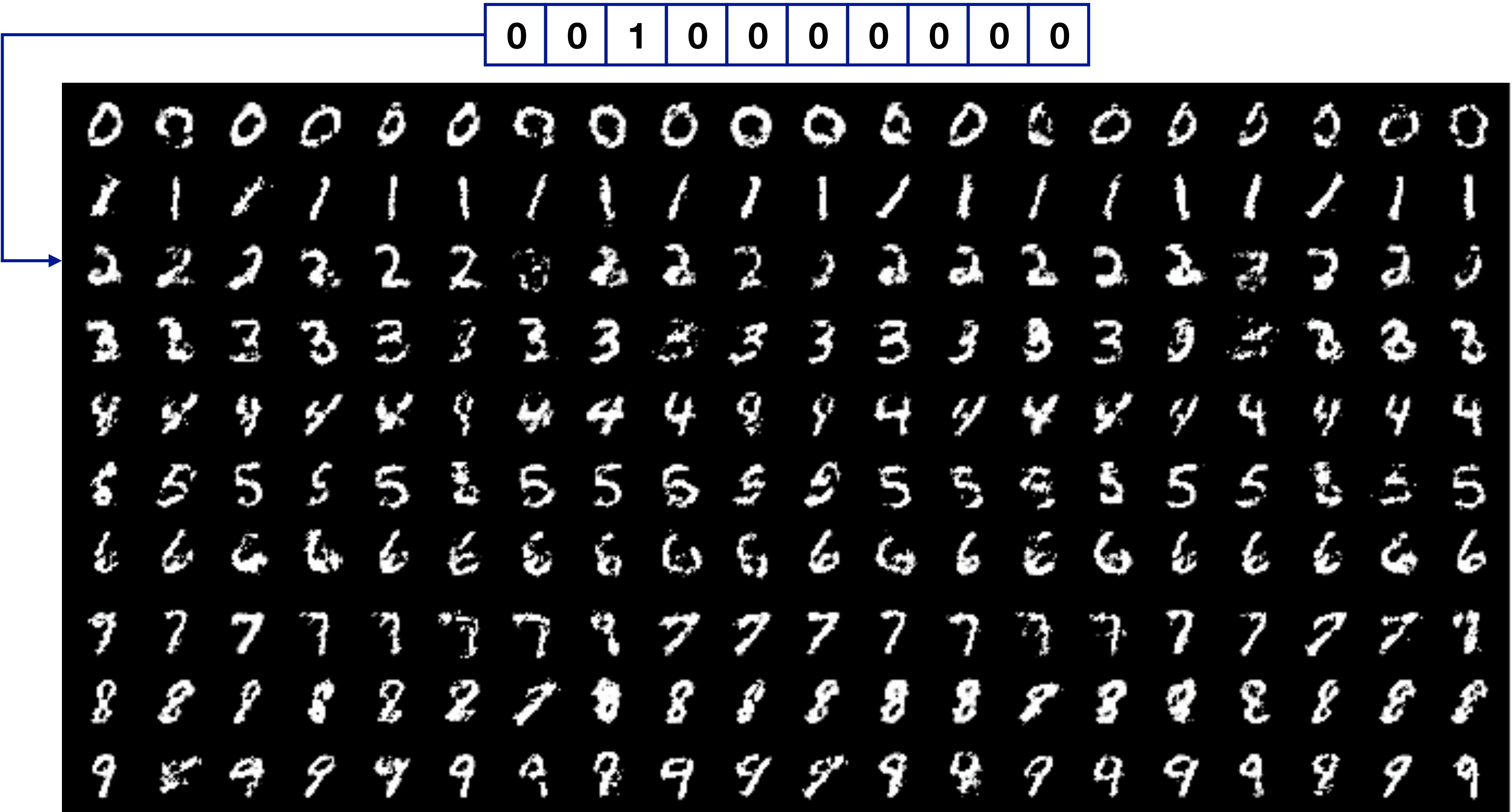


$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} [\log(1 - D(G(\mathbf{z}|\mathbf{y})))]$$

# Conditional GAN Architectures



# Conditional GAN Results



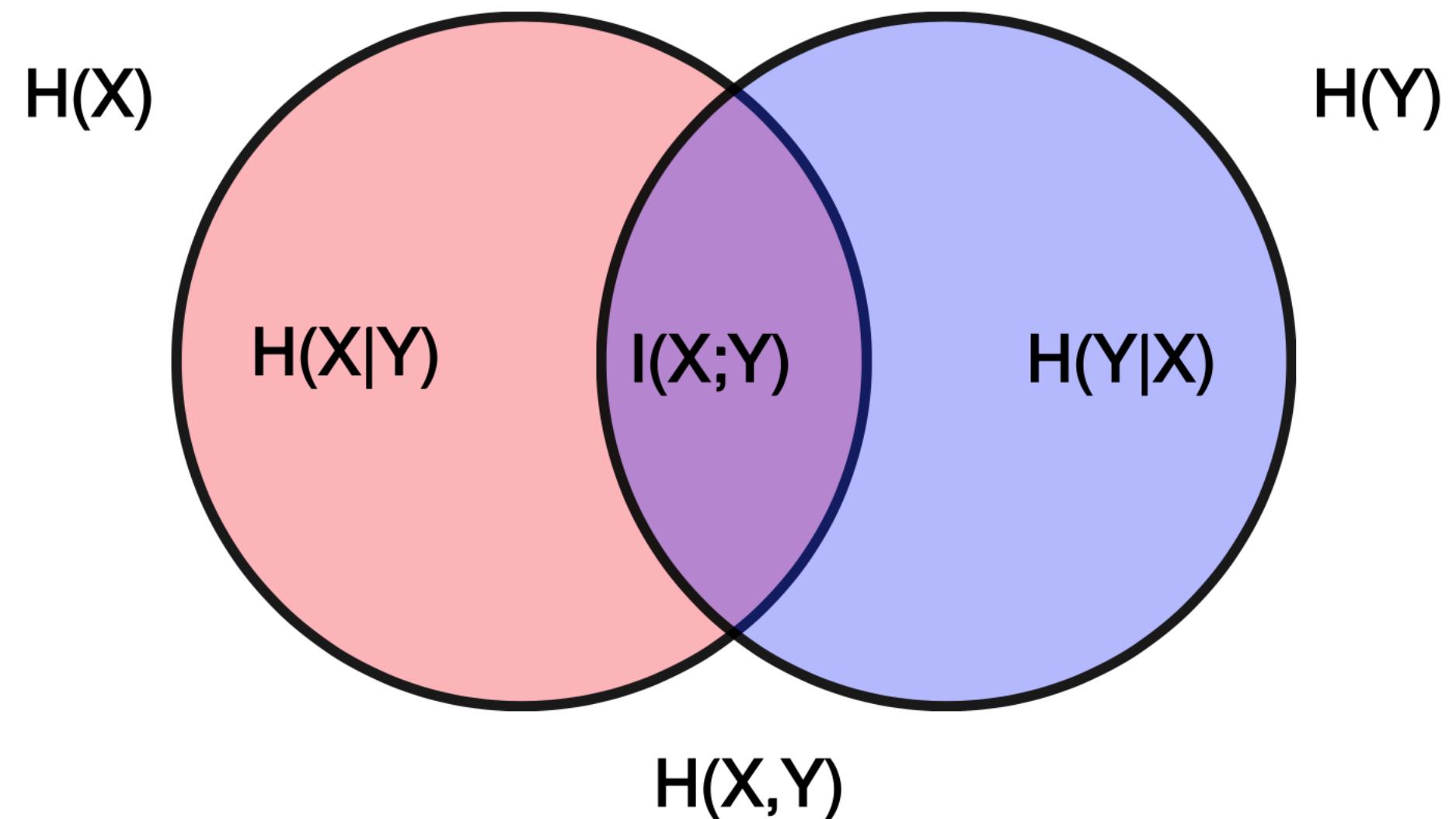
**InfoGAN**

# InfoGAN

- Learn **disentangled representations** in a completely unsupervised manner
- Maximizes the **mutual information** between a small subset of the latent codes  $c$  (condition) and the observation (generator distribution  $G(z, c)$ )
- cf) difference of cGAN
  - cGAN: learn through explicit condition
  - infoGAN: learn through mutual information between condition and generator distribution



# Mutual Information



$$\begin{aligned}I(X;Y) &= \int_Y \int_X p(x,y) \log \left( \frac{p(x,y)}{p(x)p(y)} \right) dx dy \\&= H(X) - H(X|Y) \\&= H(Y) - H(Y|X)\end{aligned}$$

**Entropy:** Uncertainty of Random Variables

**Mutual Information:**

Uncertainty of X - Uncertainty of X given Y

= Removed Uncertainty of X given Y

= Information of X given Y

# Cost Function of InfoGAN

## original GAN loss

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} [\log(1 - D(G(\mathbf{z})))]$$

## InfoGAN loss

$$\begin{aligned} \min_G \max_D V_I(D, G) &= V(D, G) - \lambda I(\mathbf{c}; G(\mathbf{z}, \mathbf{c})) \\ &= V(D, G) - \lambda [H(\mathbf{c}) - H(\mathbf{c}|G(\mathbf{z}, \mathbf{c}))] \end{aligned}$$



# Variational Mutual Information

$$\begin{aligned} I(c; G(z, c)) &= H(c) - H(c|G(z, c)) \\ &= \mathbb{E}_{x \sim G(z, c)} [\mathbb{E}_{c' \sim P(c|x)} [\log P(c'|x)]] + H(c) \\ &= \mathbb{E}_{x \sim G(z, c)} [\underbrace{D_{\text{KL}}(P(\cdot|x) \parallel Q(\cdot|x))}_{\geq 0} + \mathbb{E}_{c' \sim P(c|x)} [\log Q(c'|x)]] + H(c) \\ &\geq \mathbb{E}_{x \sim G(z, c)} [\mathbb{E}_{c' \sim P(c|x)} [\log Q(c'|x)]] + H(c) = L_I(G, Q) \end{aligned}$$

## InfoGAN loss

$$\min_{G, Q} \max_D V_{\text{InfoGAN}}(D, G, Q) = V(D, G) - \lambda L_I(G, Q)$$



# InfoGAN Results



(a) Varying  $c_1$  on InfoGAN (Digit type)



(b) Varying  $c_1$  on regular GAN (No clear meaning)



(c) Varying  $c_2$  from  $-2$  to  $2$  on InfoGAN (Rotation)



(d) Varying  $c_3$  from  $-2$  to  $2$  on InfoGAN (Width)

# InfoGAN Results



(a) Azimuth (pose)

(b) Elevation



(c) Lighting

(d) Wide or Narrow



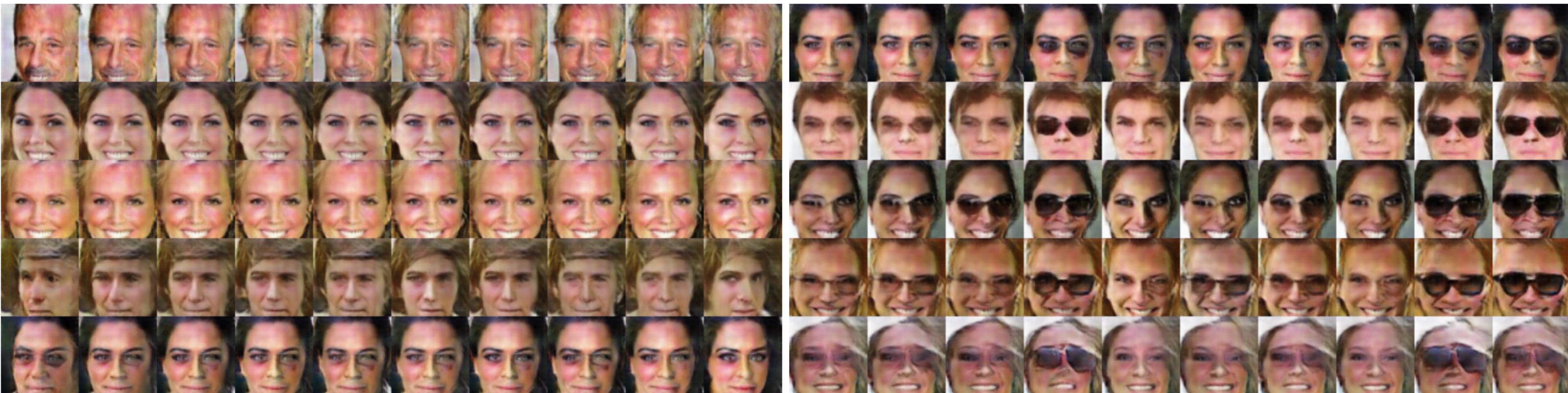
# InfoGAN Results



(a) Rotation

(b) Width

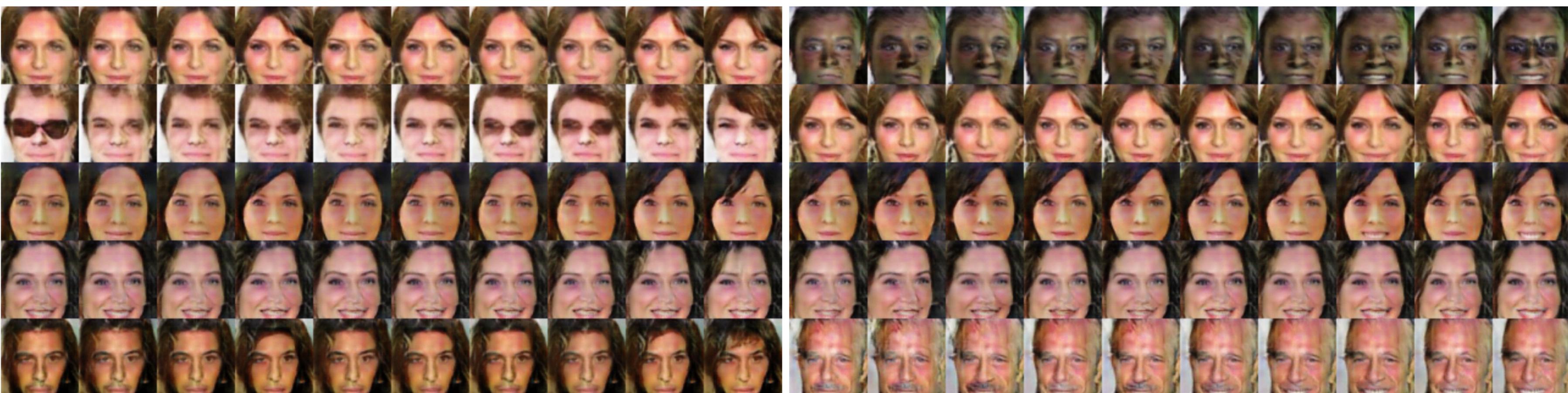
# InfoGAN Results



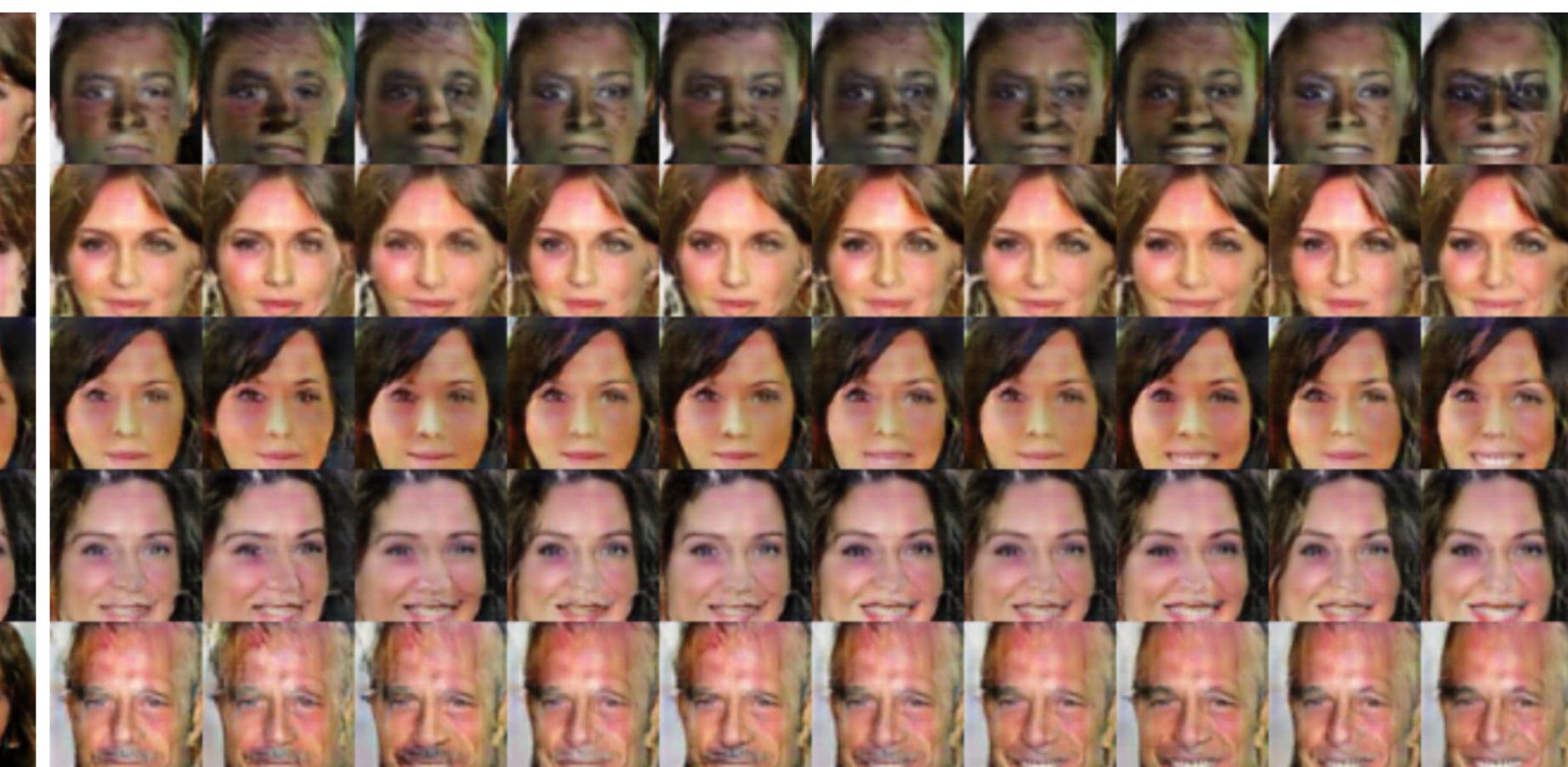
(a) Azimuth (pose)



(b) Presence or absence of glasses



(c) Hair style



(d) Emotion



# **Bidirectional GAN (BiGAN)**

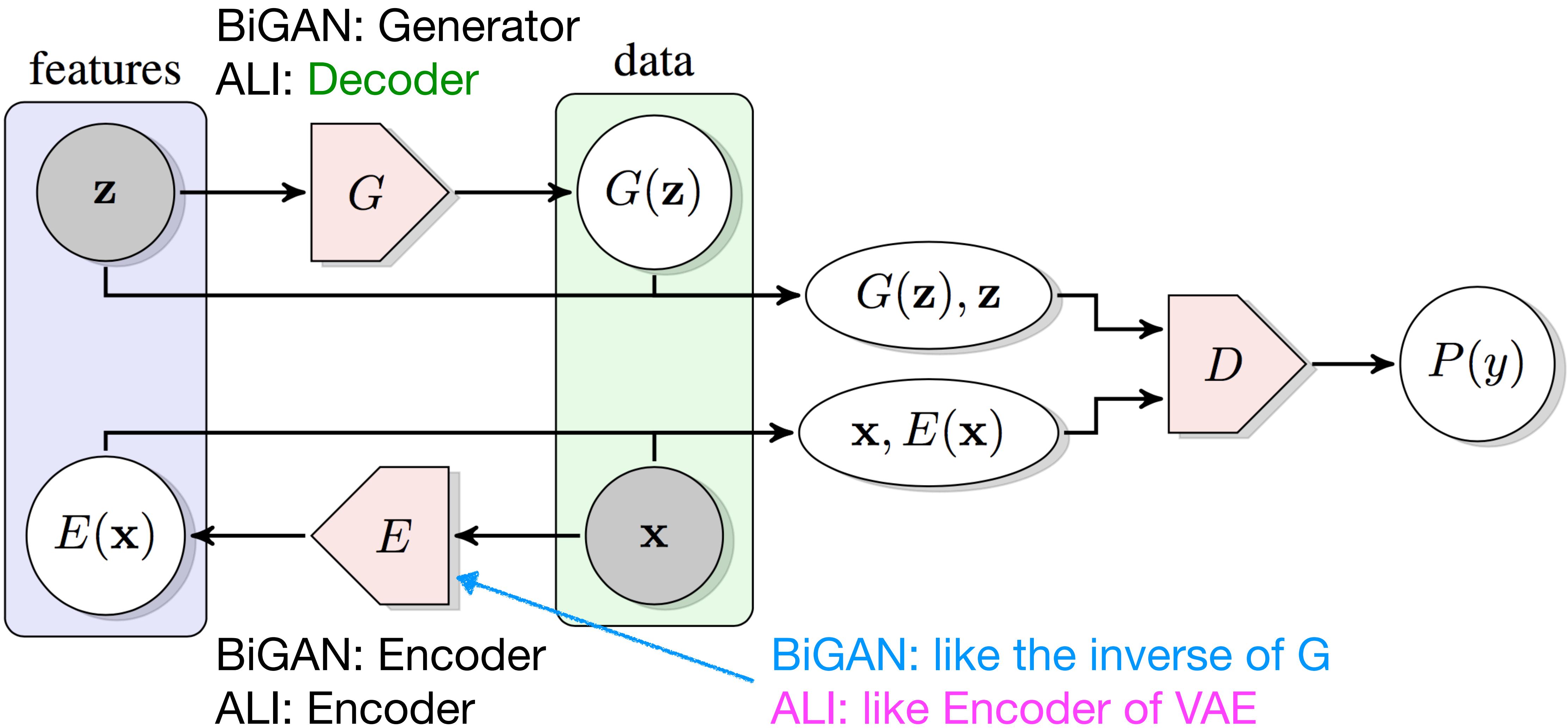
# **Adversarial Learned Inference (ALI)**

# GANs with Inference Models

- Generator in original framework: latent space -> real space
- Conversely, can we find a latent vector corresponding to real data?
  - Bidirectional GANs
  - Adversarially Learned Inference (ALI)
- Generator: two networks
  - Encoder (inference network), Decoder (original generator)
  - Learning ( $x, z$ ) pair



# BiGAN and ALI Architectures



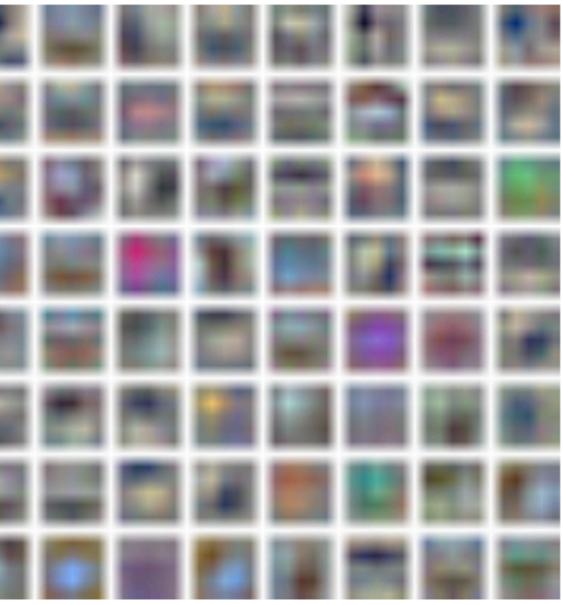
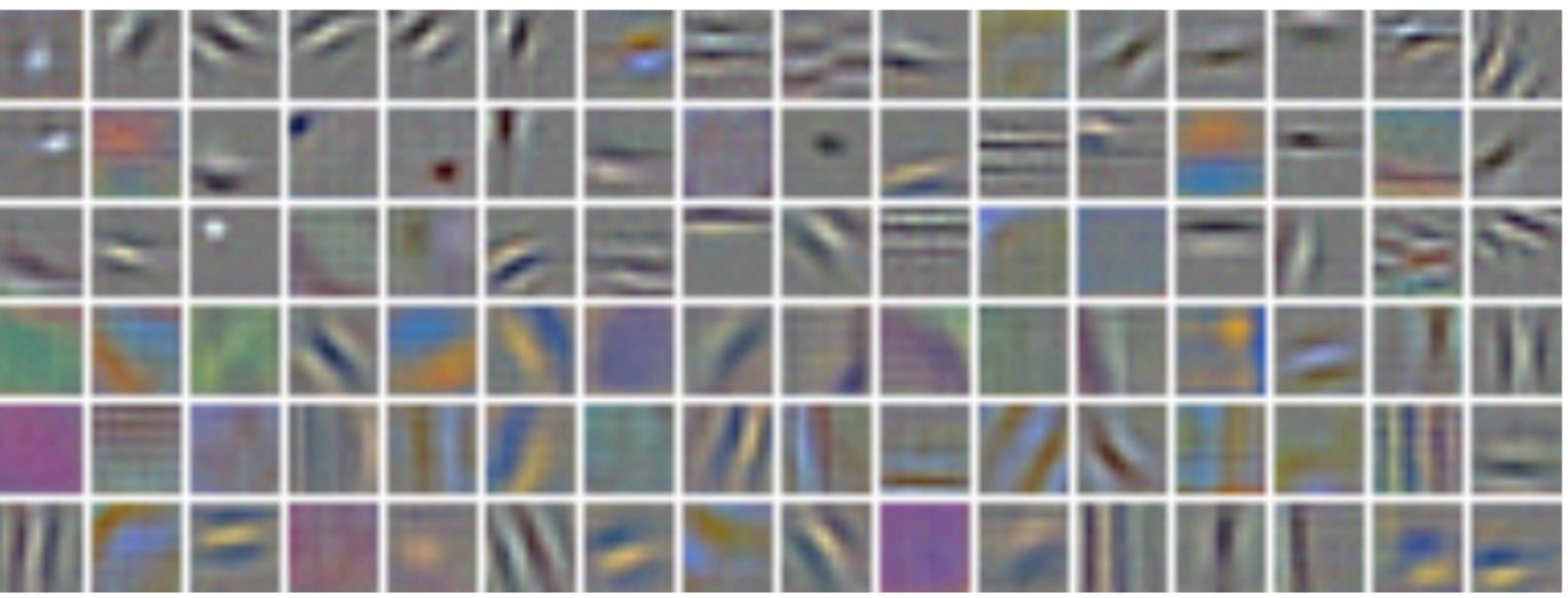
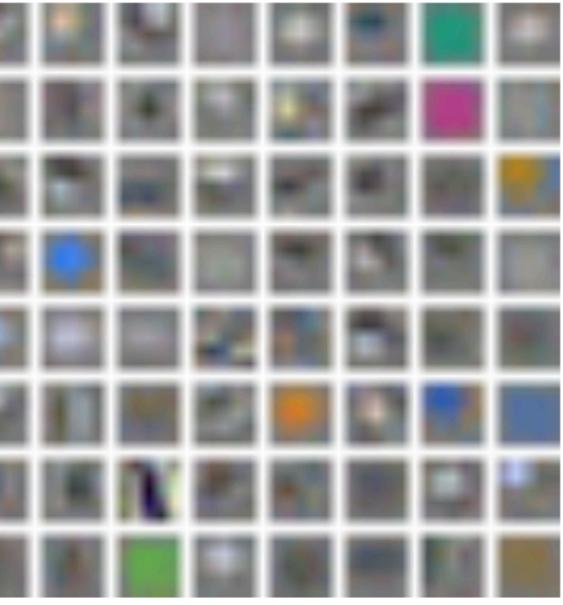
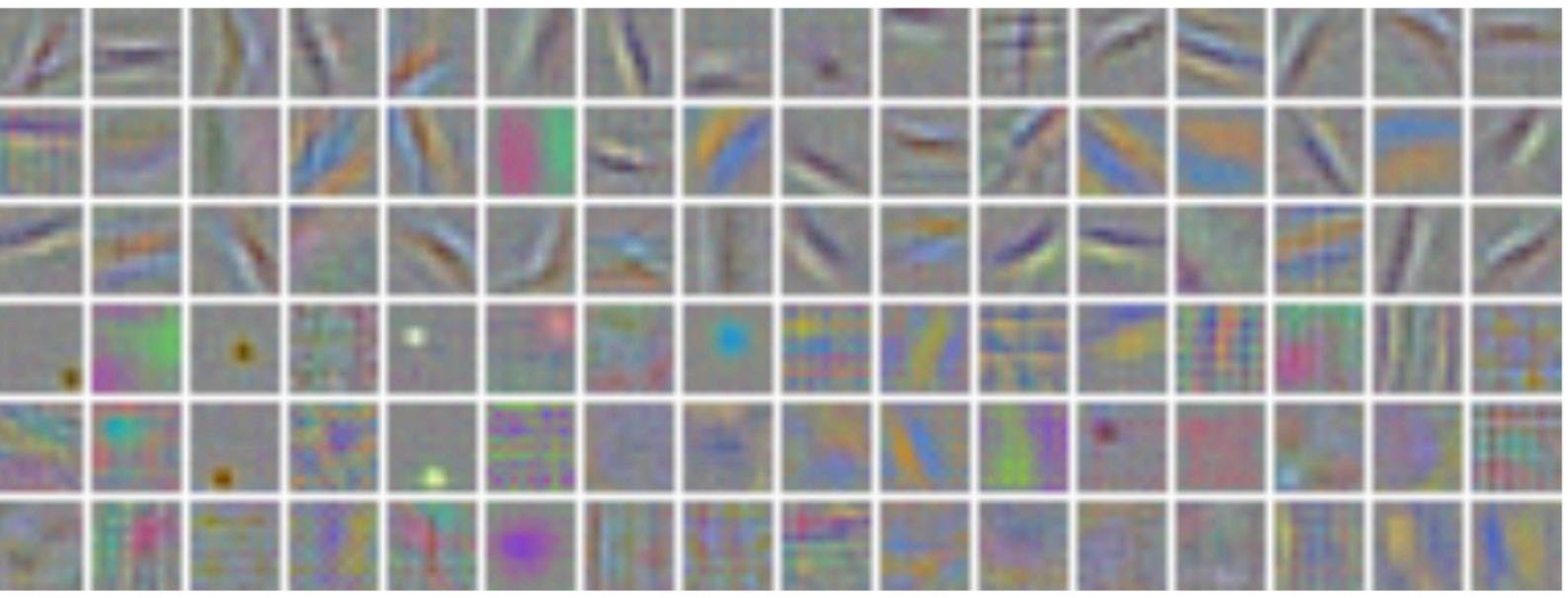
# BiGAN Objective Function

$$\min_{G, E} \max_D V(D, E, G)$$

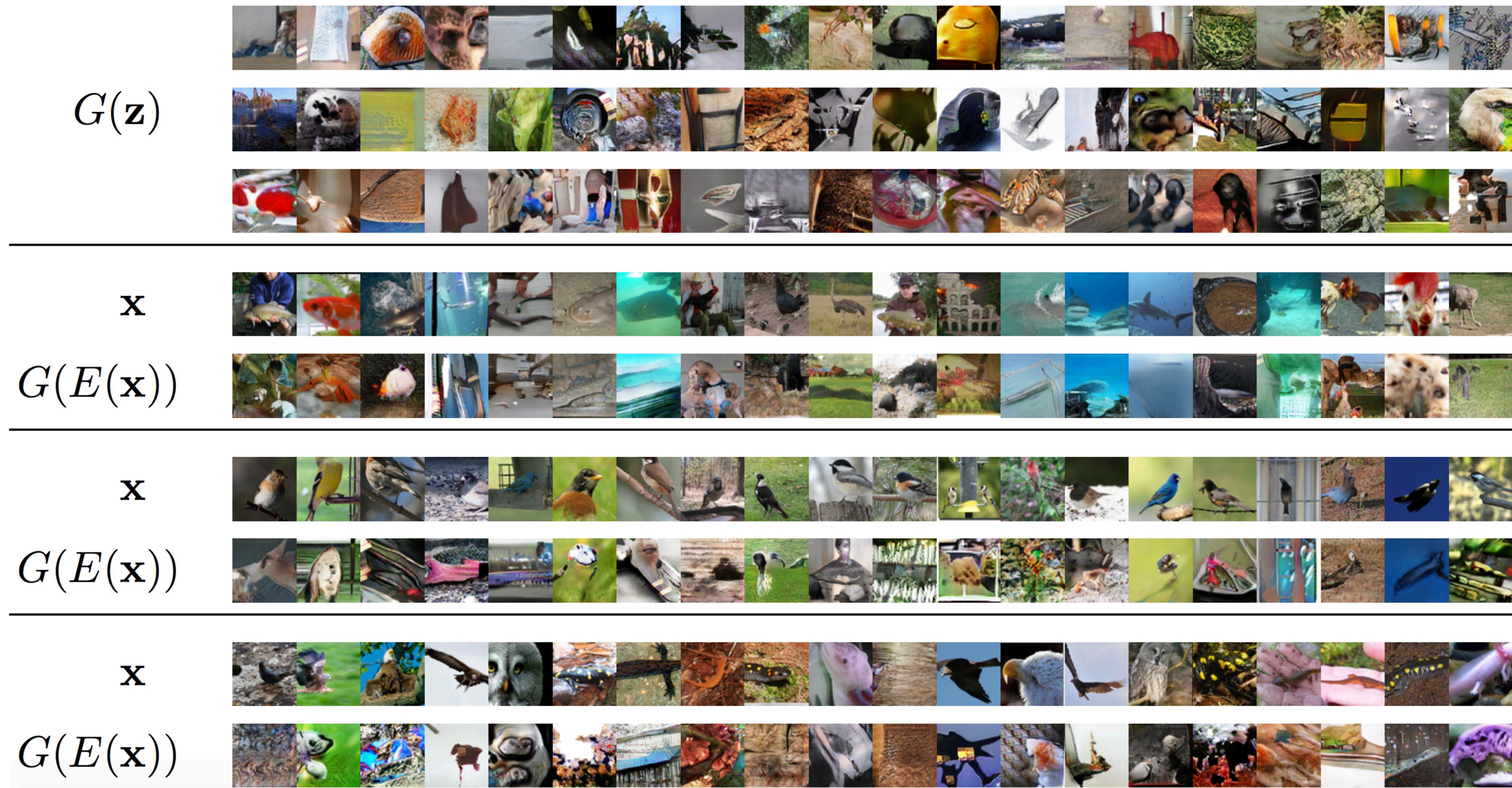
$$\begin{aligned} V(D, E, G) &:= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[ \underbrace{\mathbb{E}_{\mathbf{z} \sim p_E(\cdot | \mathbf{x})} [\log D(\mathbf{x}, \mathbf{z})]}_{\log D(\mathbf{x}, E(\mathbf{x}))} \right] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} \left[ \underbrace{\mathbb{E}_{\mathbf{x} \sim p_G(\cdot | \mathbf{z})} [\log(1 - D(\mathbf{x}, \mathbf{z}))]}_{\log(1 - D(G(\mathbf{z}), \mathbf{z}))} \right] \\ &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x}, E(\mathbf{x}))] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} [\log(1 - D(G(\mathbf{z}), \mathbf{z}))] \end{aligned}$$



# BiGAN Results

$G(\mathbf{z})$	
$\mathbf{x}$	
$G(E(\mathbf{x}))$	
$D$	
$E$	
$G$	
AlexNet-based $D$	
<b>Noroozi &amp; Favaro (2016)</b>	
<b>Krizhevsky et al. (2012)</b>	

# BiGAN Results



# ALI Results (Interpolation)

