Generative Adversarial Networks - 1

2022
Multicampus

Il Gu Yi
ModuLabs, Research Scientist
Soochul Park
Vivestudios, Research Scientist

v.1.3

Taxonomy of Generative Models

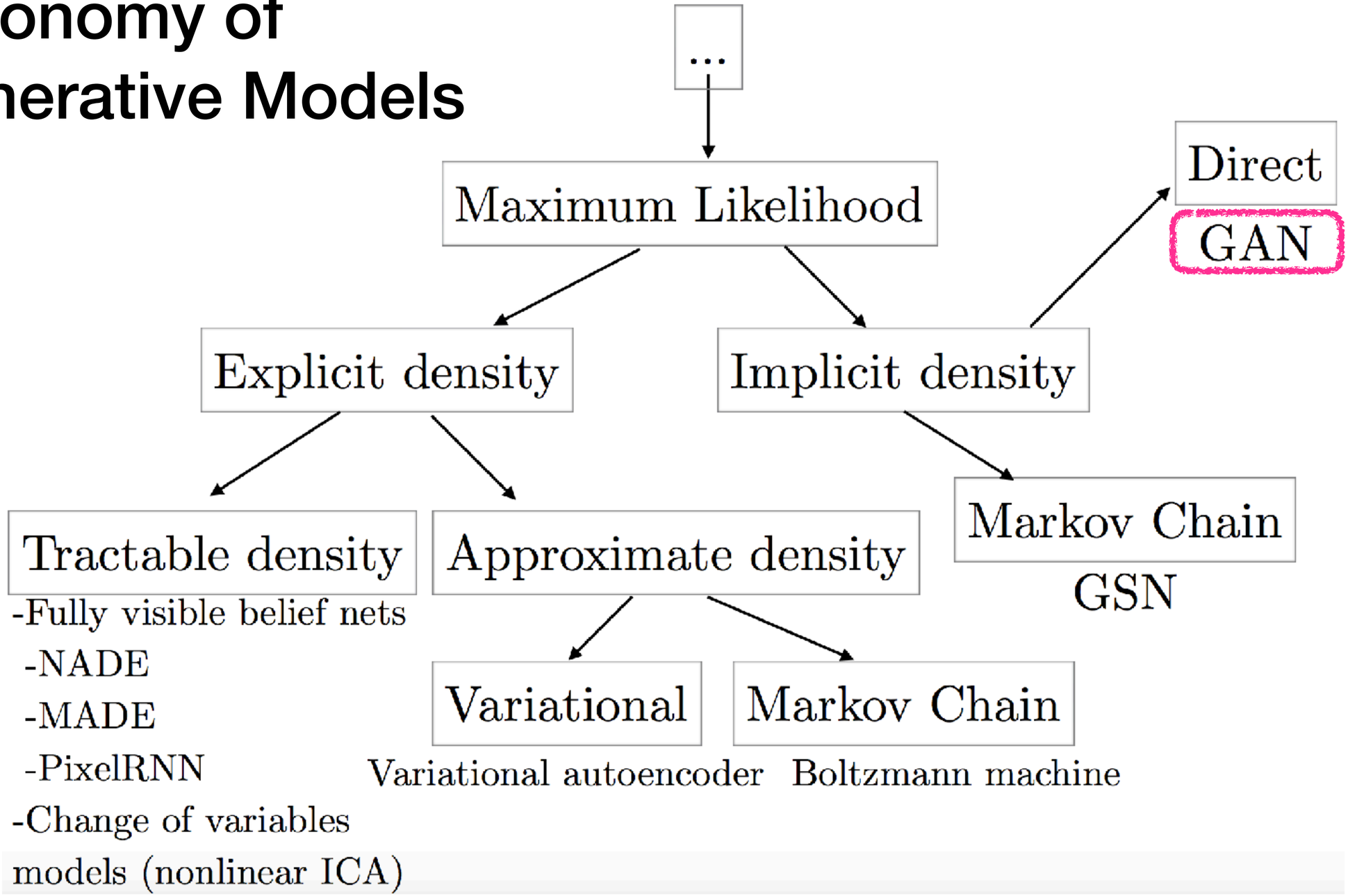


Figure credit: Ian Goodfellow, NIPS 2016 Tutorial: Generative Adversarial Networks

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 시행 착오 끝에 결국 탐정이 진품과 모조품을 구별할 수 없도록 만듭니다.
- 탐정은 화가가 그리는 모조품과 진품을 받고서 서로 구별하는 능력을 키웁니다. 그리고 구별한 결과를 화가에게 알려줍니다.

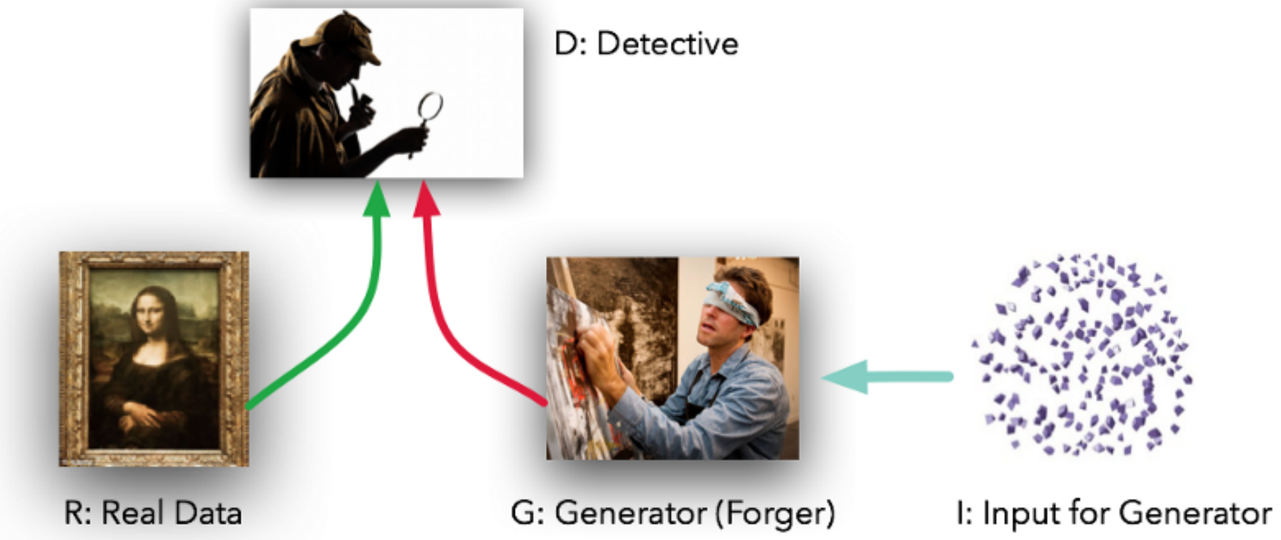
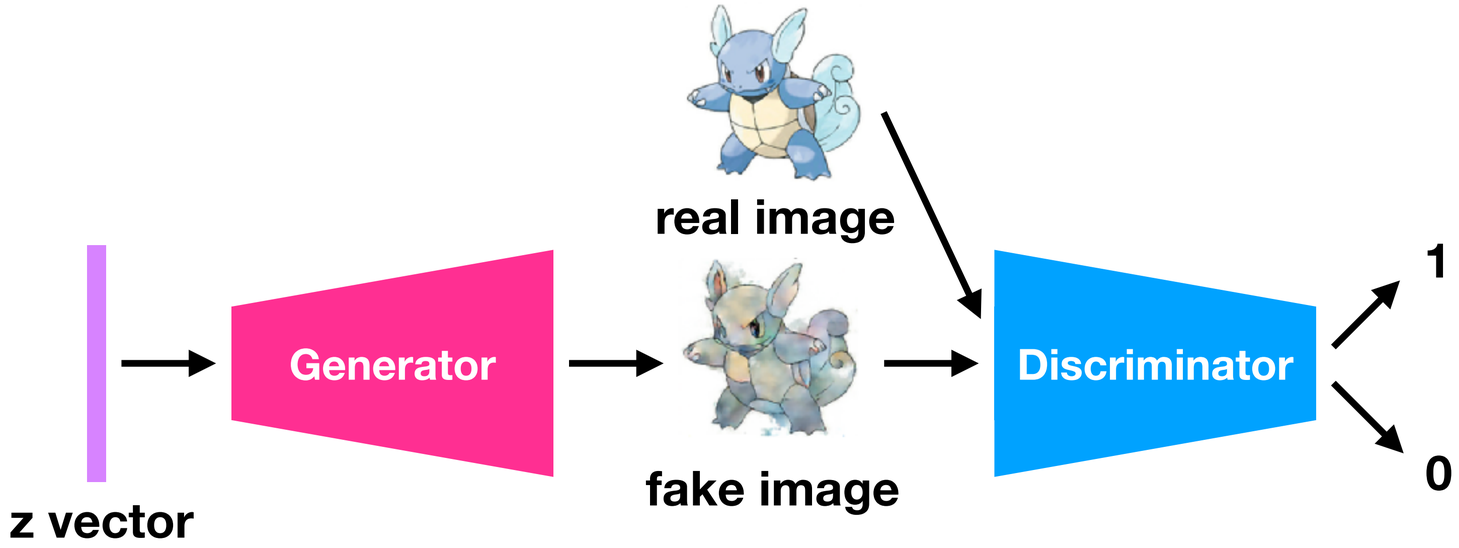


Figure credit: Blog. Generative Adversarial Networks (GANs) in 50 lines of code (PyTorch)

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 $J_D(\boldsymbol{\Theta}_D; \boldsymbol{\Theta}_G)$ using only $\boldsymbol{\Theta}_D$
- The **Generator**($\mathcal{G}$): minimize $J_G(\boldsymbol{\Theta}_G; \boldsymbol{\Theta}_D)$ using only $\boldsymbol{\Theta}_G$
- 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
  - $\mathcal{G} : \mathcal{G}(\mathbf{z}) \rightarrow \mathbb{R}^{|\mathbf{x}|}, \mathbf{z} \in \mathbb{R}^{|\mathbf{z}|}$ is a sample in latent space
- Discriminator: a function from input space to the probability space
  - $\mathcal{D} : \mathcal{D}(\mathbf{x}) \rightarrow [0, 1]$
- 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$\mathbf{x}$
- $p_g(\mathbf{x})$: probability density function of the samples generated by the generator
- $\mathcal{G}$: generator networks, $\mathcal{D}$: discriminator networks
- $\boldsymbol{\Theta}_G, \boldsymbol{\Theta}_D$: parameters of $\mathcal{G}$ and $\mathcal{D}$, respectively
- Cost (objective) function
  - $J_G(\boldsymbol{\Theta}_G; \boldsymbol{\Theta}_D)$, $J_D(\boldsymbol{\Theta}_D; \boldsymbol{\Theta}_G)$
  - Two player game frameworks

Cost Functions

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

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

- 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, $J^{(D)}$

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

- Cost function is derived from binary cross entropy

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

Cross-Entropy

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

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

Minimax Game

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

- 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

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

- 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

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

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

- 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

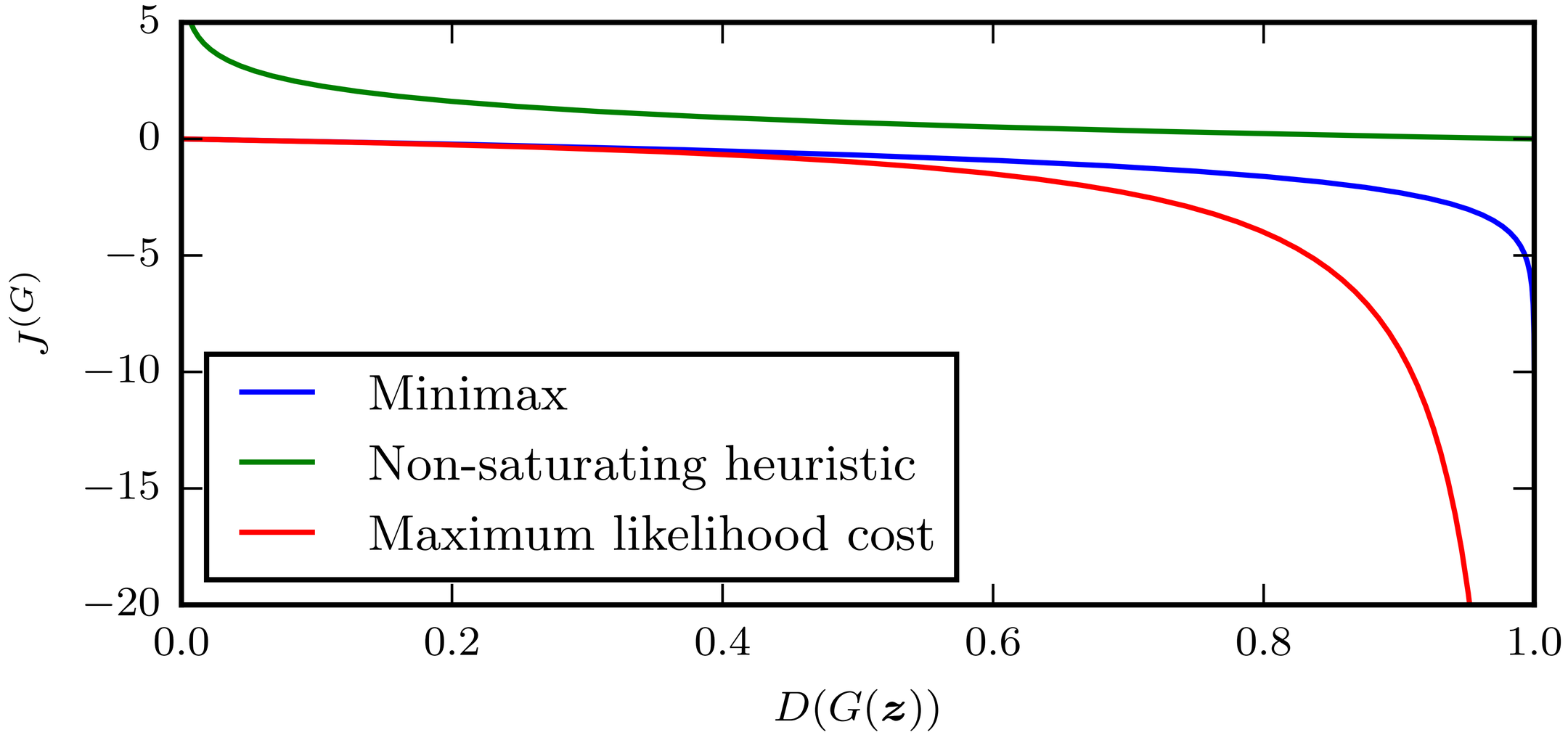
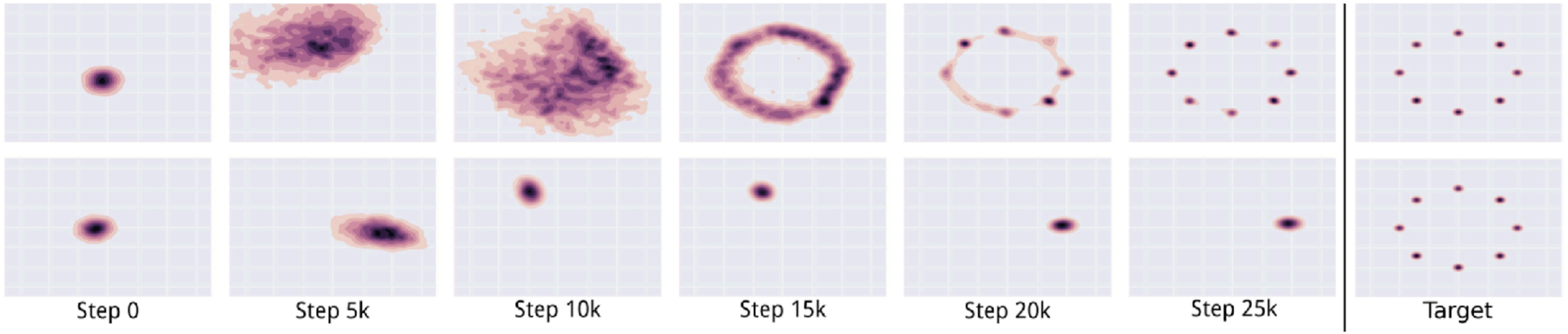


Figure credit: Ian Goodfellow, NIPS 2016 Tutorial: Generative Adversarial Networks

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.

Figure credit: L. Metz, et. al., Unrolled Generative Adversarial Networks

Mode Collapse

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

Figure credit: M. Arjovsky, et. al., Wasserstein GAN

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)

Ref.: Ian Goodfellow, et al., Generative Adversarial Nets

Training of GANs

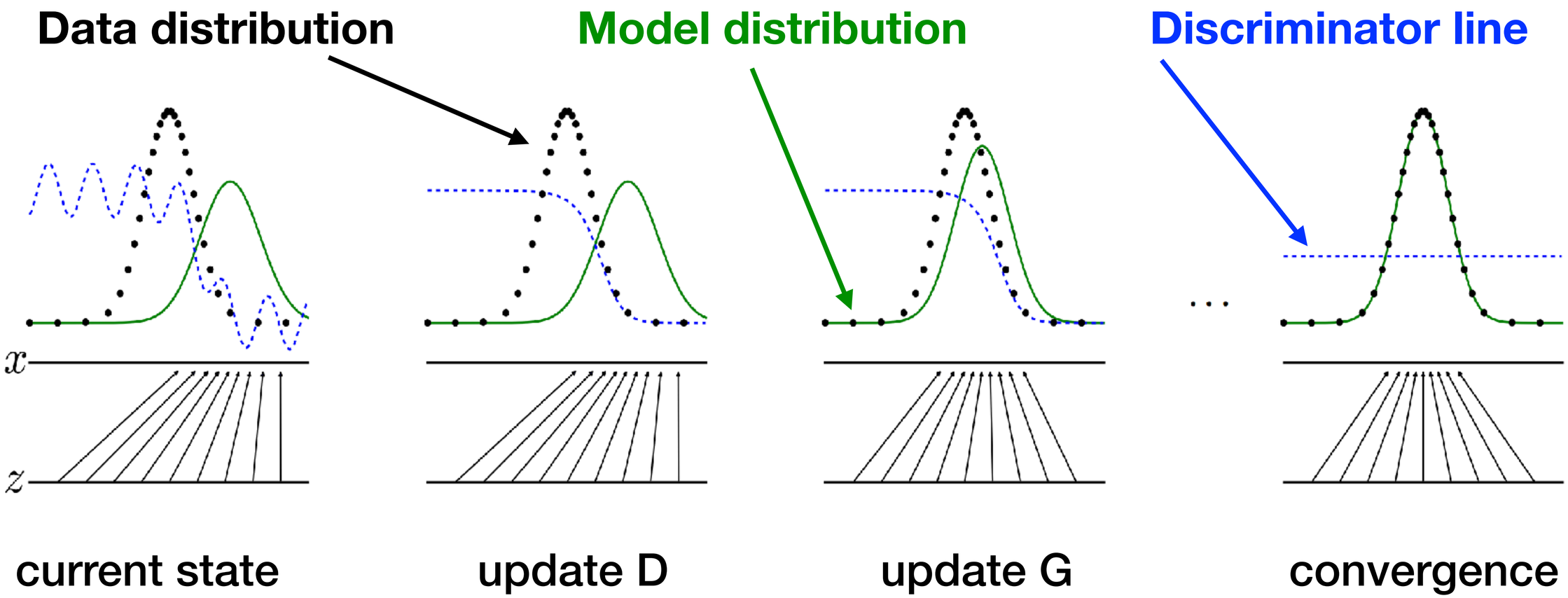


Figure credit: Ian Goodfellow, et al., Generative Adversarial Nets

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

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

  end for

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

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

end for

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

Ref.: Ian Goodfellow, et al., Generative Adversarial Nets

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*

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{2}$$

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

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

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

Ref.: Ian Goodfellow, et al., Generative Adversarial Nets

Jensen-Shannon Divergence

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

Proof. For $p_g = p_{\text{data}}$, $D^*_G(\boldsymbol{x}) = \frac{1}{2}$, (consider Eq. 2). Hence, by inspecting Eq. 4 at $D^*_G(\boldsymbol{x}) = \frac{1}{2}$, 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

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

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

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

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:

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

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. □

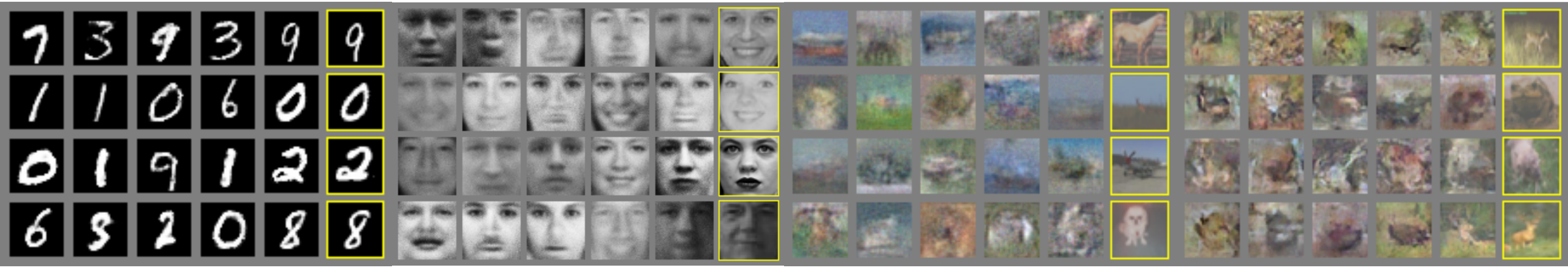
Ref.: Ian Goodfellow, et al., Generative Adversarial Nets

Jensen-Shannon Divergence

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

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

GAN Results



MNIST data

TFD data

CIFAR10 data
(fully connected)

CIFAR10 data
(convolutional)

Figure credit: Ian Goodfellow, et al., Generative Adversarial Nets

Deep Convolutional GAN (DCGAN)

Guidelines for Stable DCGAN

-DCGAN은 GAN에 deep convolutional networks를 적용한 모델로써 GAN이 활발히 연구되게한 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.

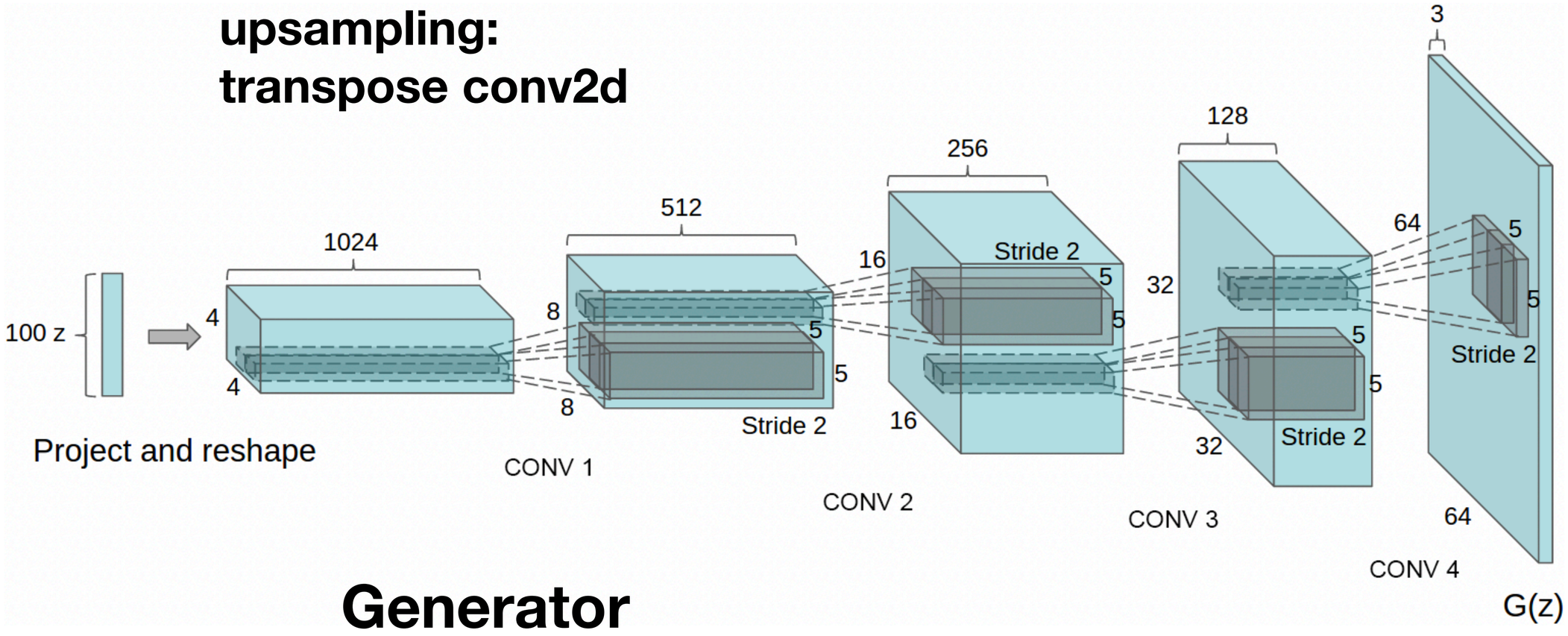
- 어떤 구성요소가 왜 성공적으로 트레이닝을 이끄는지 그 이유에 대해서는 명확하게 밝혀내지 못합니다. 단지 많은 실험에 의해 어떤 구성요소들이 좋은 결과를 낳는지 알아내고 있습니다.

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.

Ref.: A. Radford, et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

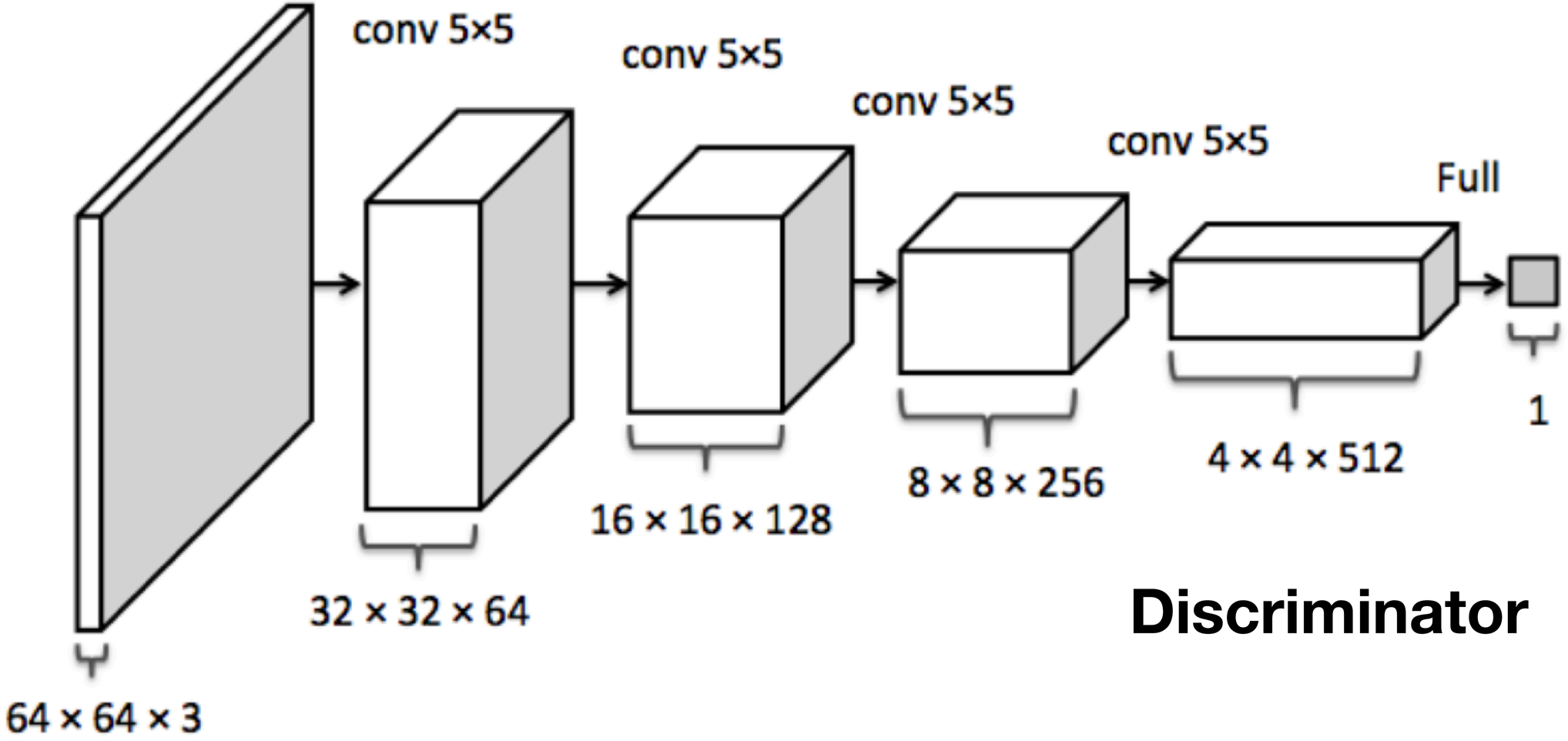
DCGAN Architectures



Generator

Figure credit: A. Radford, et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

DCGAN Architectures



Discriminator

Figure credit: R. Yeh, et al., Semantic Image Inpainting with Perceptual and Contextual Losses

DCGAN Results

Figure credit: A. Radford, et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

DCGAN Results (Interpolation)

Figure credit: A. Radford, et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

DCGAN Results (Interpolation)

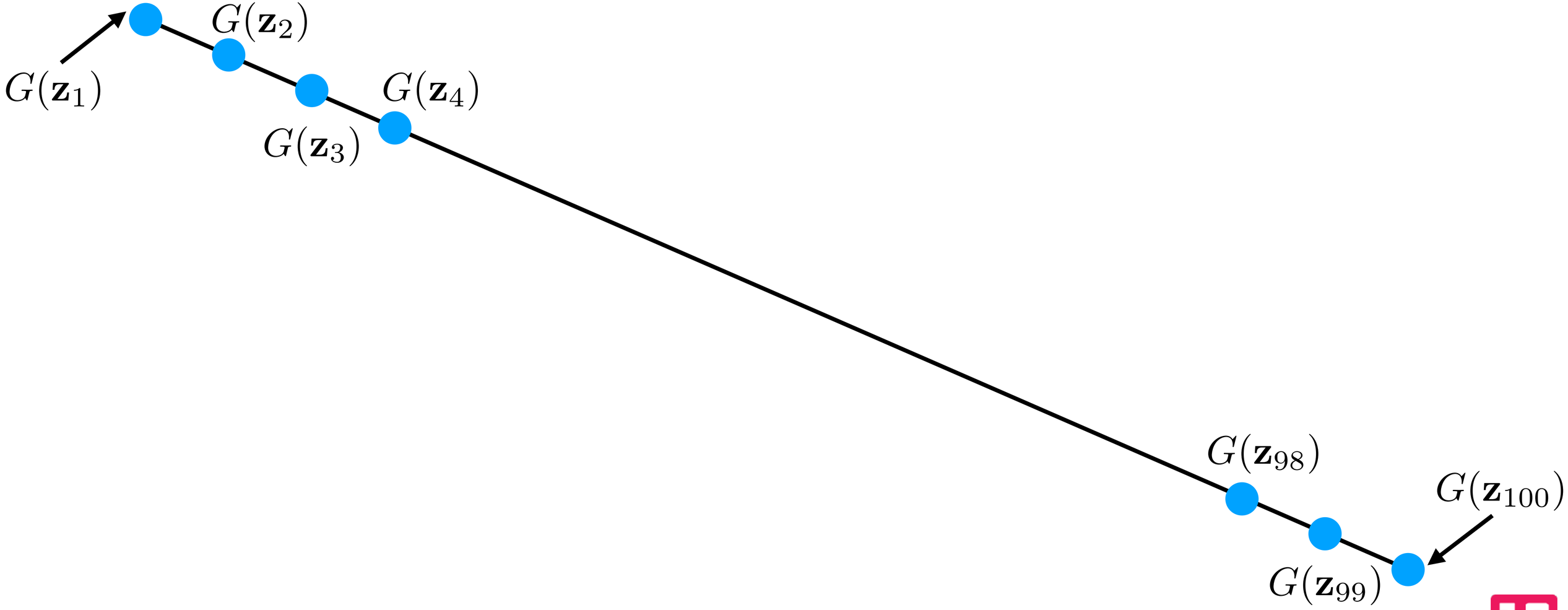


Figure credit: A. Radford, et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

DCGAN Results (Vector Arithmetic in Latent Space)

smiling man

Figure credit: A. Radford, et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

conditional GAN
(cGAN)

Conditional GAN

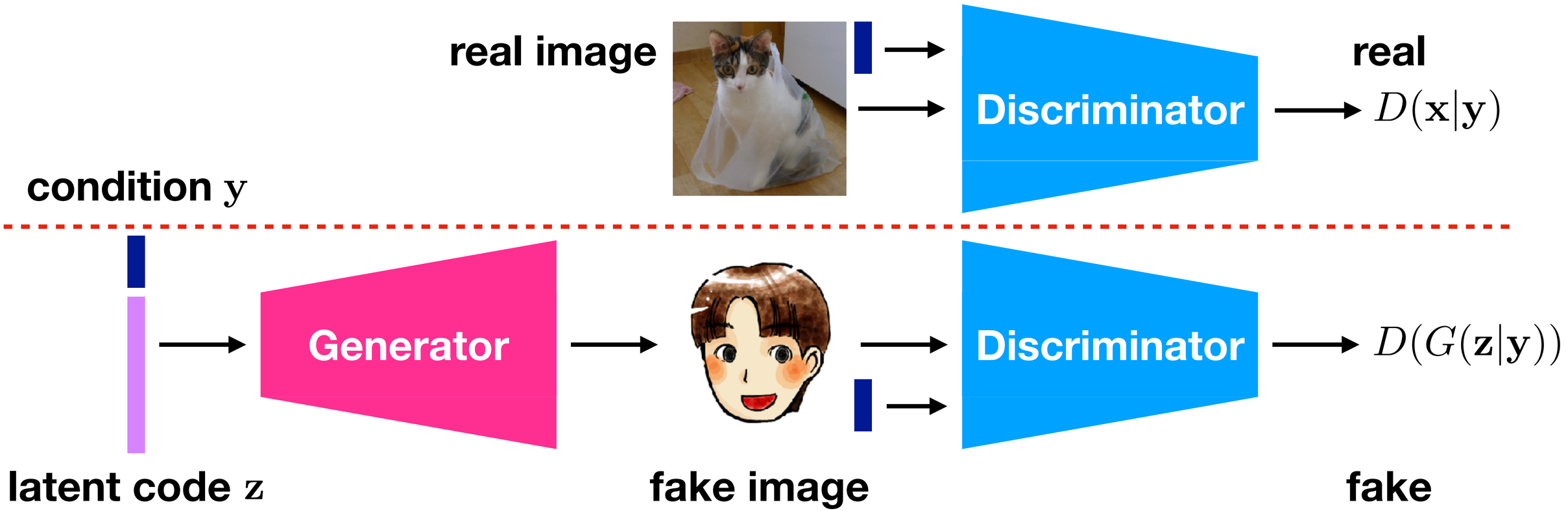
-CGAN은 Conditional Generative Adversarial Nets 논문에서 제안된 모델입니다.

-GAN은 unconditional generative model이고 어떠한 이미지를 생성할지 결정할 수 없습니다. 그런데 반해 CGAN은 어떠한 이미지를 생성할지 미리 결정할 수 있습니다.

-이를 위해 CGAN에서는 generator에 z vector와 함께 condition vector를 함께 넣어줍니다. 마찬가지로 discriminator에는 image와 함께 condition vector를 함께 넣어줍니다.

-Condition vector는 categorical information인 경우 one-hot vector로써 나타낼 수 있습니다.

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 Results

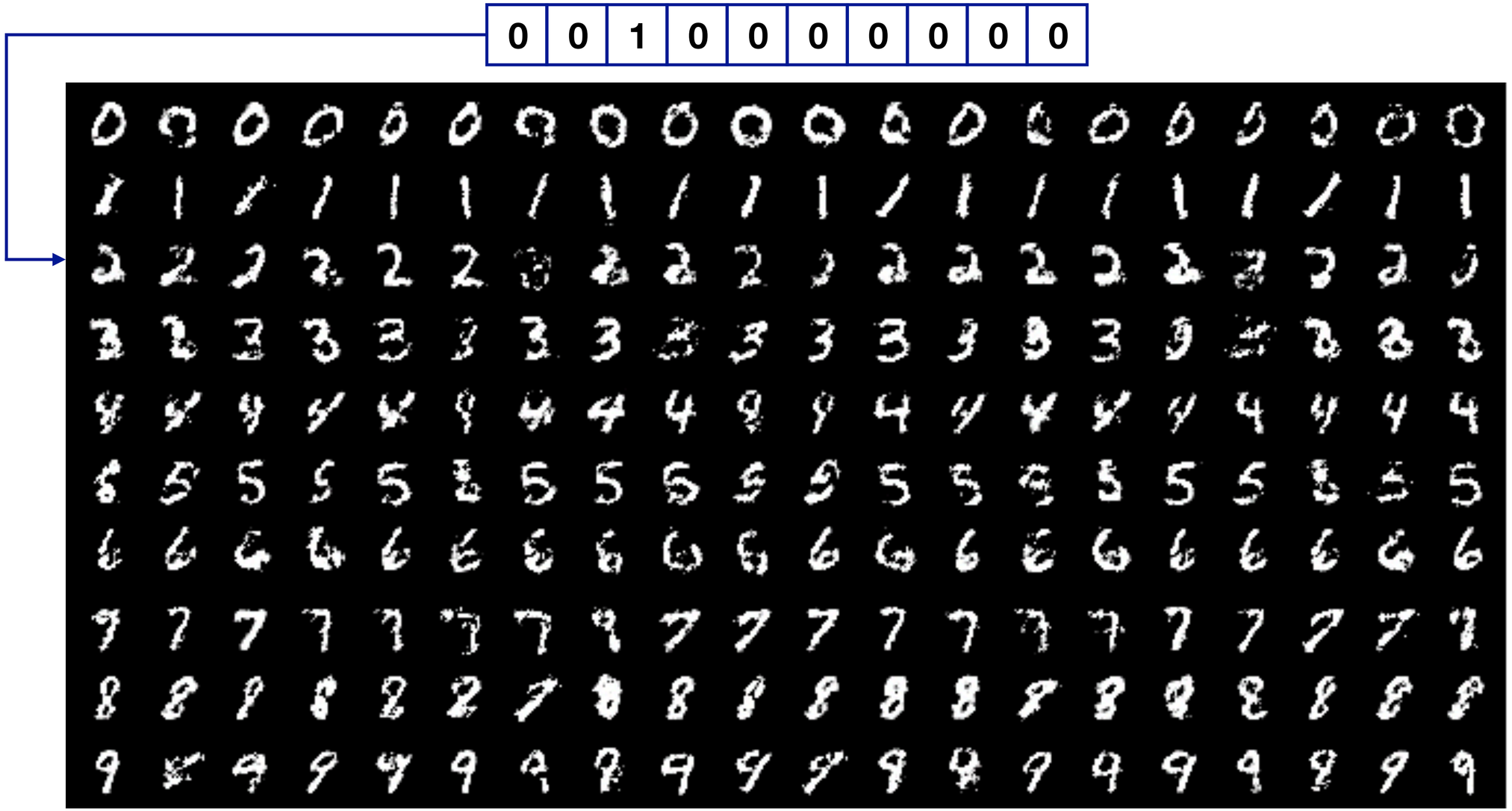


Figure credit: M. Mirza, et al., Conditional Generative Adversarial Nets

InfoGAN

InfoGAN

-InfoGAN은 Infogan: Interpretable representation learning by information maximizing generative adversarial nets 논문에서 제안되었습니다.

-CGAN은 annotation이 되어있는 condition의 경우에만 적용시킬 수 있다는 단점이 있습니다.

-즉, 사람 얼굴의 방향이나, 헤어 스타일 같이 annotation을 구하기 힘든 데이터들에 대한 condition은 반영할 수 없습니다.

-InfoGAN은 unsupervised learning으로 이러한 문제를 해결해줍니다.

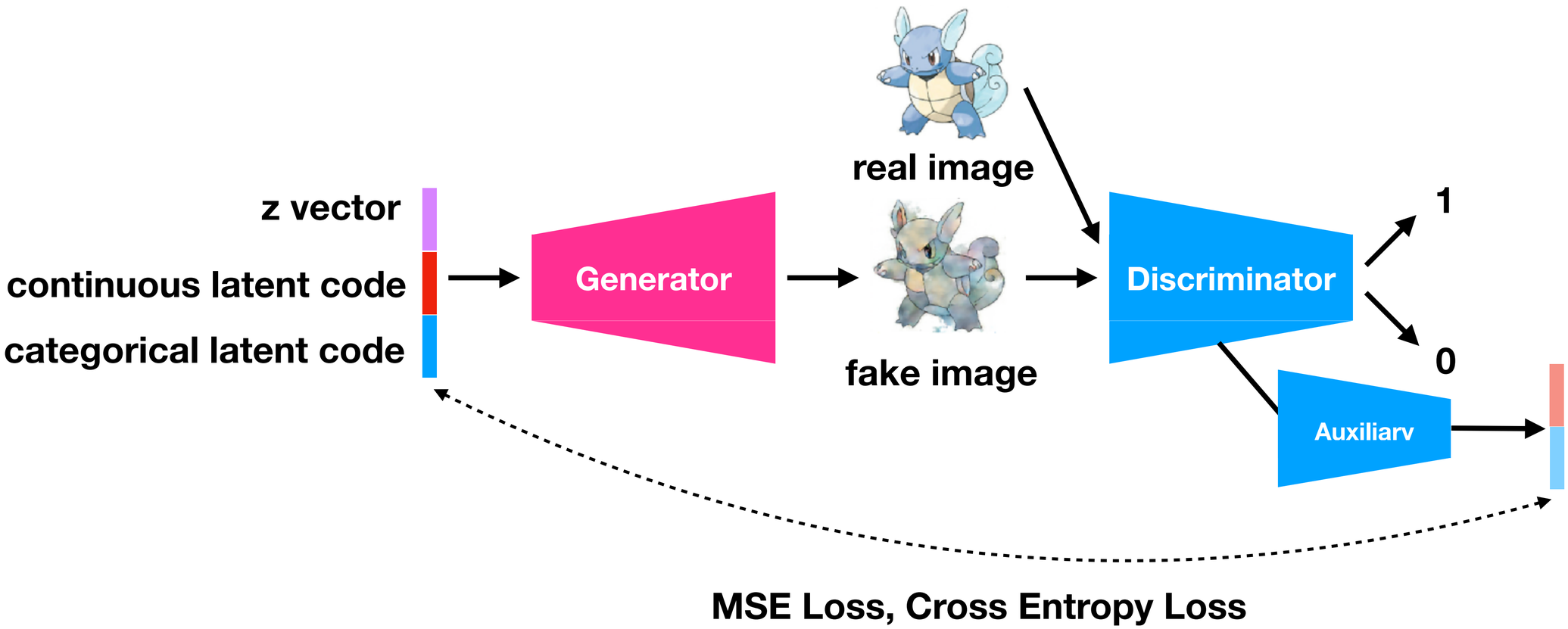
-InfoGAN은 noise z vector 뿐만 아니라 continuous latent code와 categorical latent code를 추가적인 input으로 generator에 입력합니다.

-Generator와 discriminator 뿐만 아니라 별도의 auxiliary network를 두고, continuous latent code와 latent code에 대한 prediction을 출력합니다.

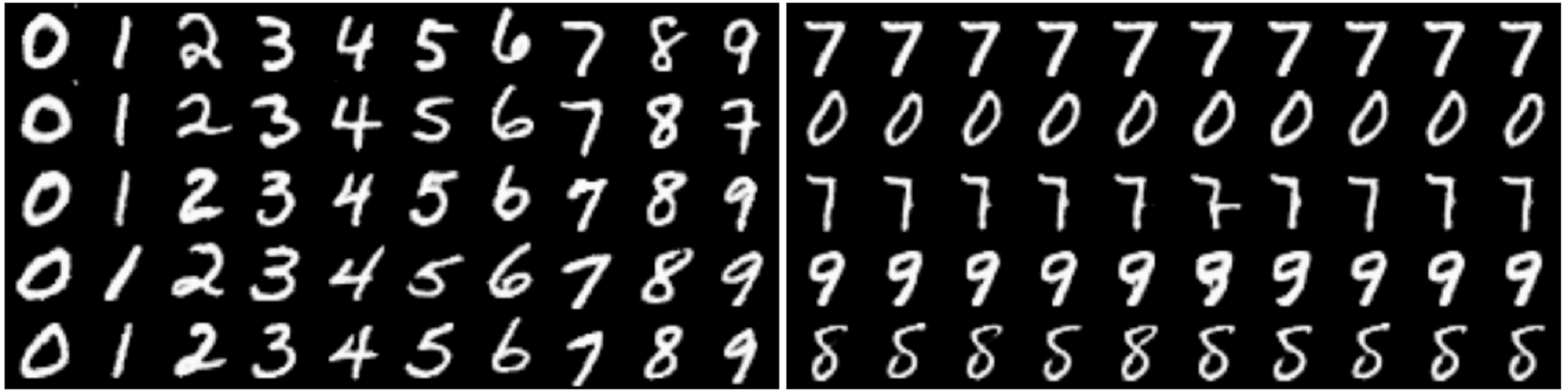
-이 출력과 원본 latent code들 간에 loss를 잡고, prediction이 잘 수행되도록 트레이닝하면 generator의 결과가 latent code의 정보를 담도록 트레이닝 됩니다.

Chen, Xi, et al. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." Proceedings of the 30th International Conference on Neural Information Processing Systems. 2016.

InfoGAN

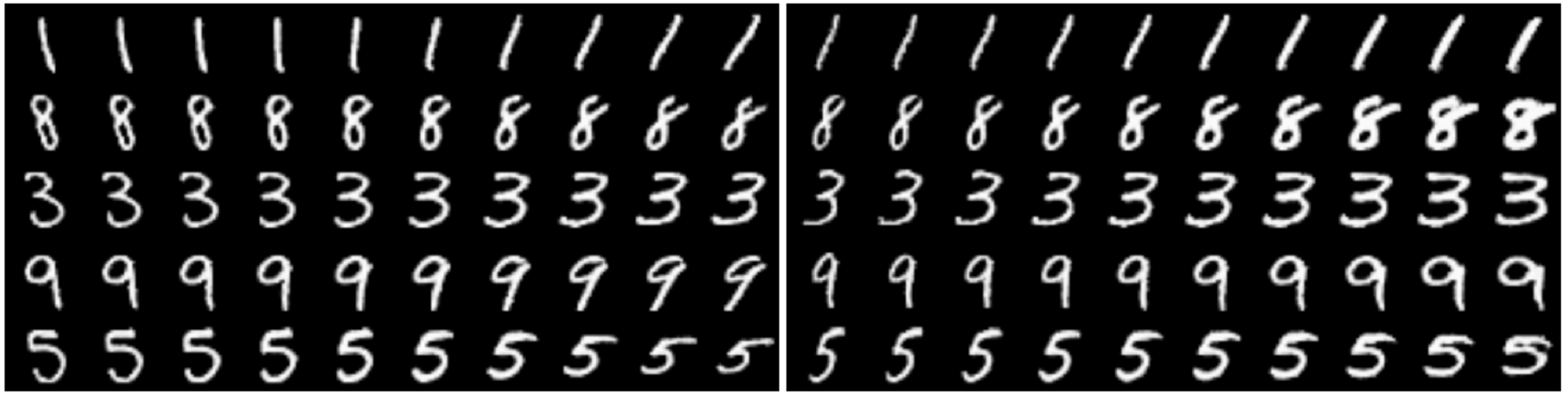


InfoGAN Results



(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)

Figure credit: X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

InfoGAN Results

(a) Rotation (b) Width

Figure credit: X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

InfoGAN Results

(a) Azimuth (pose)

(b) Presence or absence of glasses

(c) Hair style

(d) Emotion

Figure credit: X. Chen, et al., InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets

Bidirectional GAN (BiGAN)
Adversarial Learned Inference (ALI)

GANs with Inference Models

- Generator in original framework: latent space -> real space
- Conversely, can not we find a latent vector corresponding to real data?
  - Bidirectional GANs
  - Adversarially Learned Inference (ALI)
- Generator: two networks
  - Encoder (inference network), Decoder (original generator)
  - Leraning $(\mathbf{x}, \mathbf{z})$ pair

BiGAN and ALI Architectures

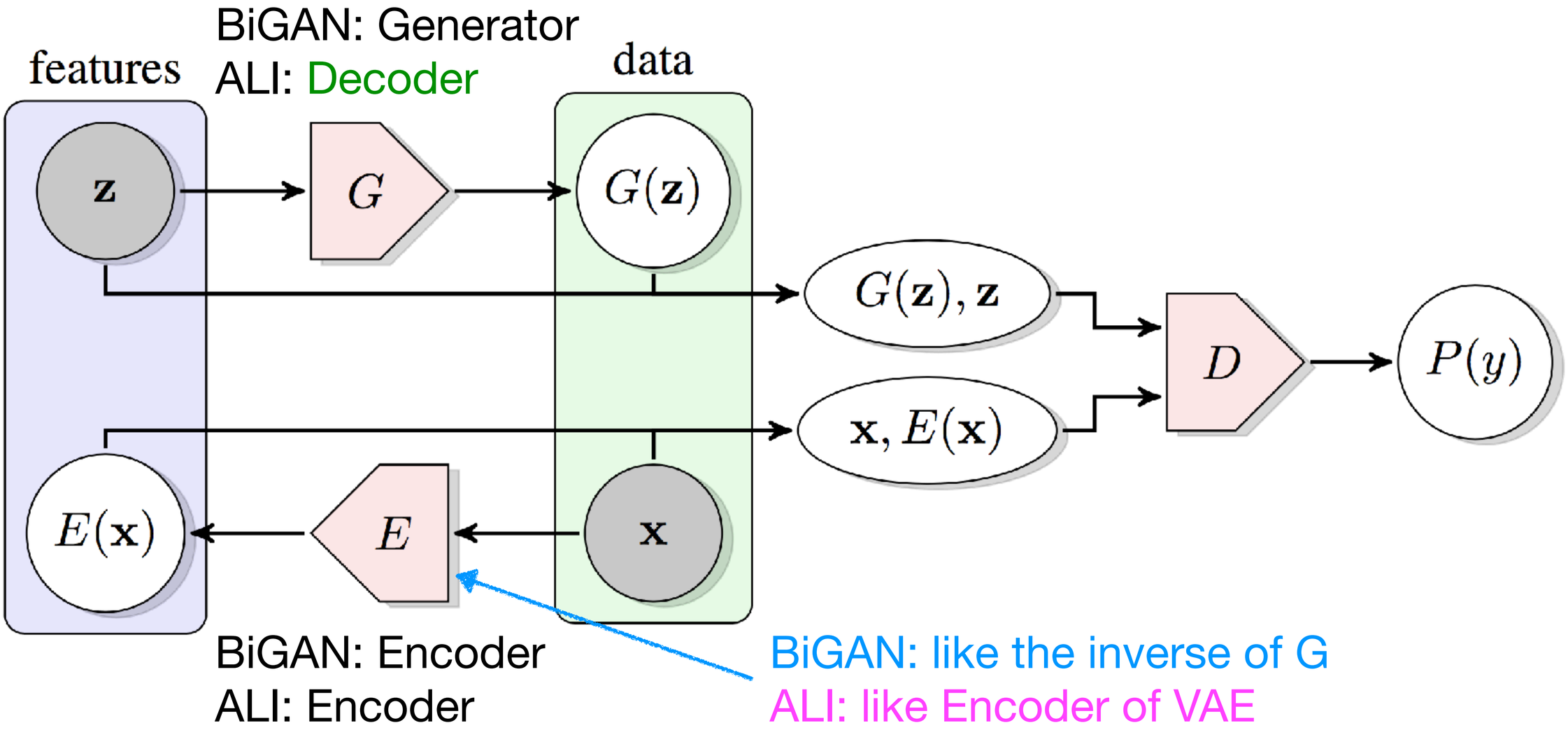


Figure credit: Jeff Donahue, et al., Adversarial Feature Learning

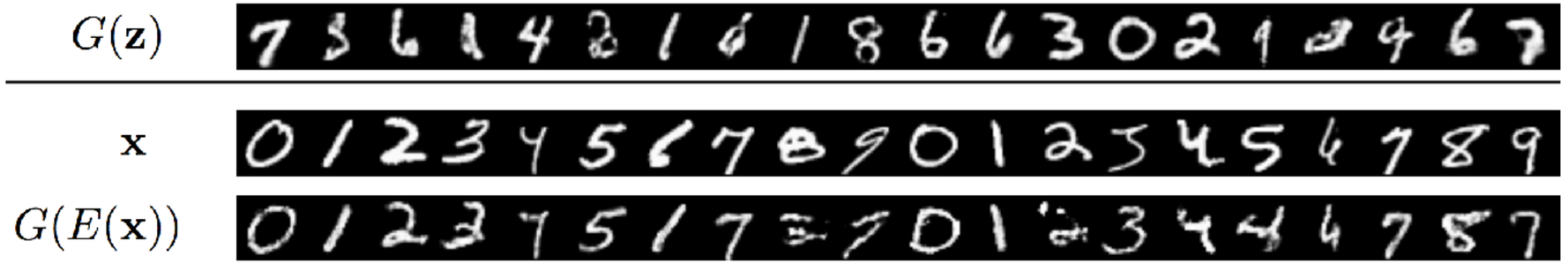
BiGAN Objective Function

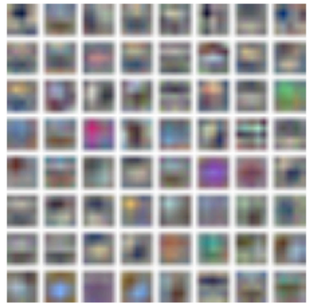
$$\min_{G,E} \max_{D} V(D, E, G)$$

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

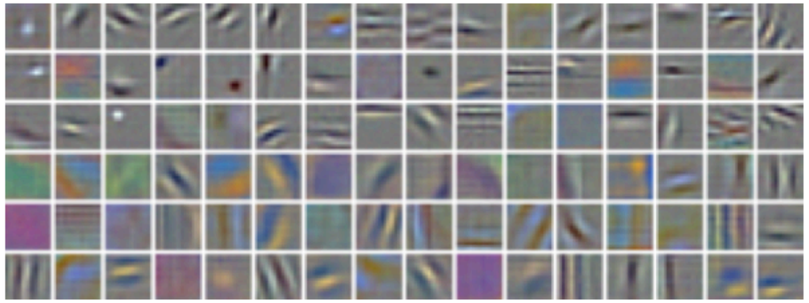
$$= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x}, E(\mathbf{x}))] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} [\log(1 - D(G(\mathbf{z}), \mathbf{z}))]$$

BiGAN Results

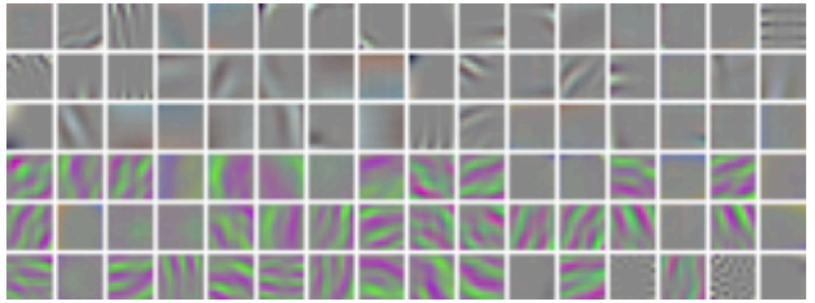




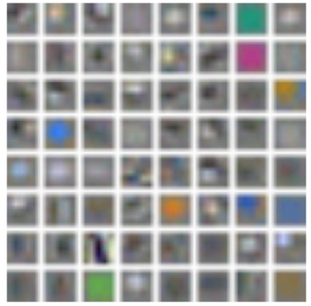
D



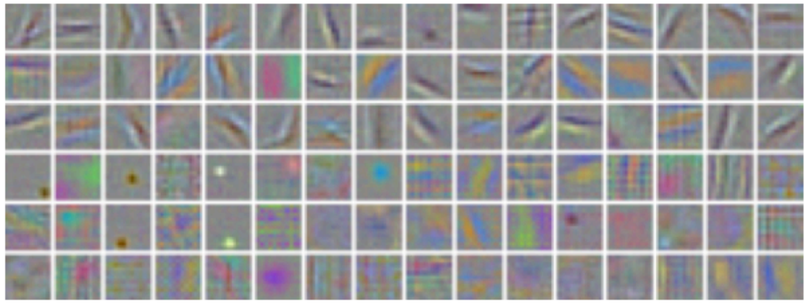
E



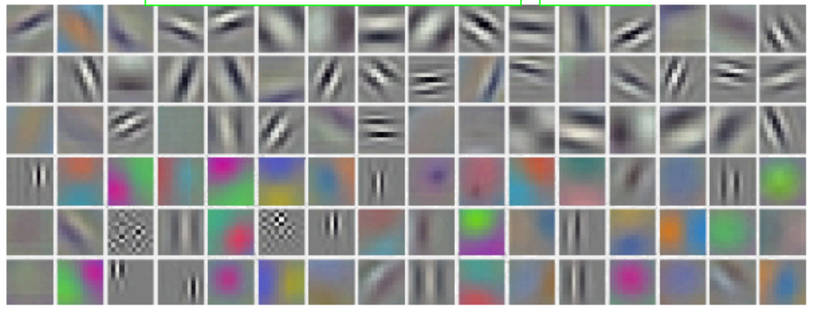
Noroozi & Favaro (2016)



G



AlexNet-based D



Krizhevsky et al. (2012)

Figure credit: Jeff Donahue, et al., Adversarial Feature Learning

BiGAN Results

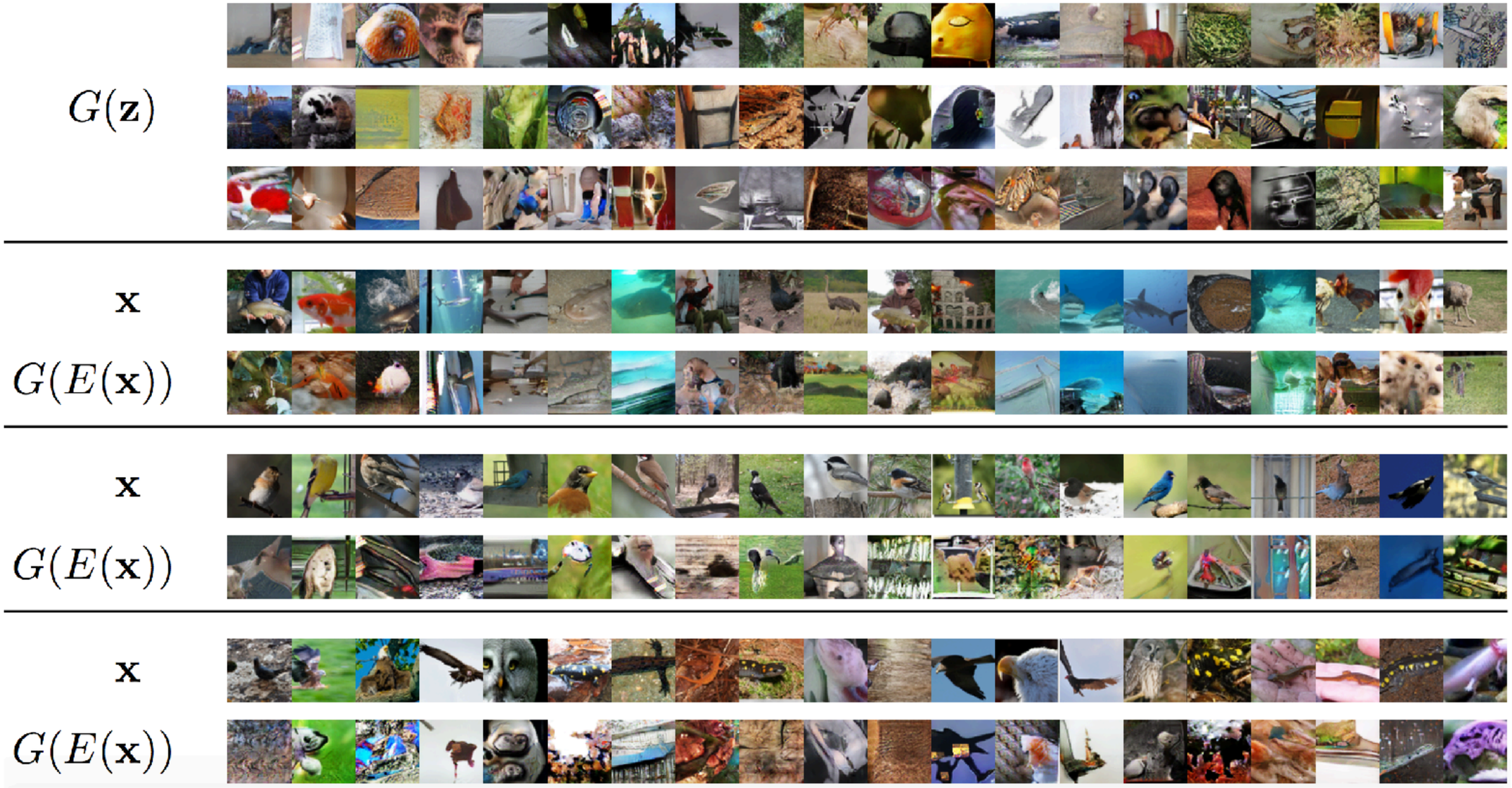


Figure credit: Jeff Donahue, et al., Adversarial Feature Learning

ALI Results (Interpolation)

Figure credit: Vincent Dumoulin, et al., Adversarially Learned Inference