Generative models

Deep Learning for Computer Vision

Valeriya Strizhkova

16 November 2021

About myself

Valeriya Strizhkova

1st year PhD student @ Inria, STARS team

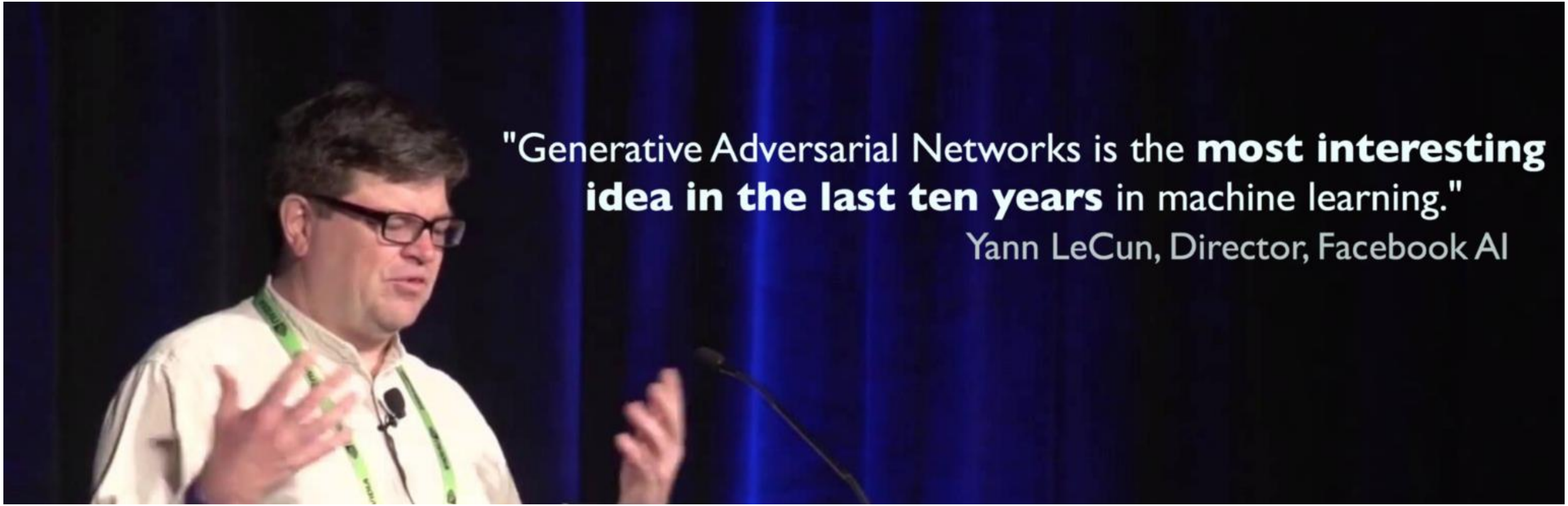
https://scholar.google.ru/citations?user=6n5PrUAAAAAJ&hl

https://github.com/valerystrizh

Part 1

Outline

- Basic idea of GAN
- Image generation
- Video Generation


"Generative Adversarial Networks is the most interesting idea in the last ten years in machine learning."
Yann LeCun, Director, Facebook AI

Generative Adversarial Networks

- Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. Want to sample from p_{data}.

Generative Adversarial Networks

- Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. Want to sample from p_{data}.
- Idea: Introduce a latent variable z with simple prior $p(z)$.

Generative Adversarial Networks

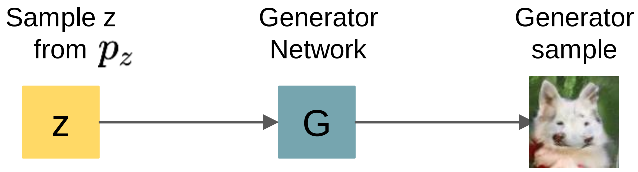
- Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. Want to sample from p_{data}.
- Idea: Introduce a latent variable z with simple prior $p(z)$.
- Sample $z \sim p(z)$ and pass to a Generator Network $x = G(z)$

Generative Adversarial Networks

- Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. Want to sample from p_{data}.
- Idea: Introduce a latent variable z with simple prior $p(z)$.
- Sample $z \sim p(z)$ and pass to a Generator Network $x = G(z)$
- Then x is a sample from the Generator distribution p_G. Want $p_G = p_{data}$

Generative Adversarial Networks

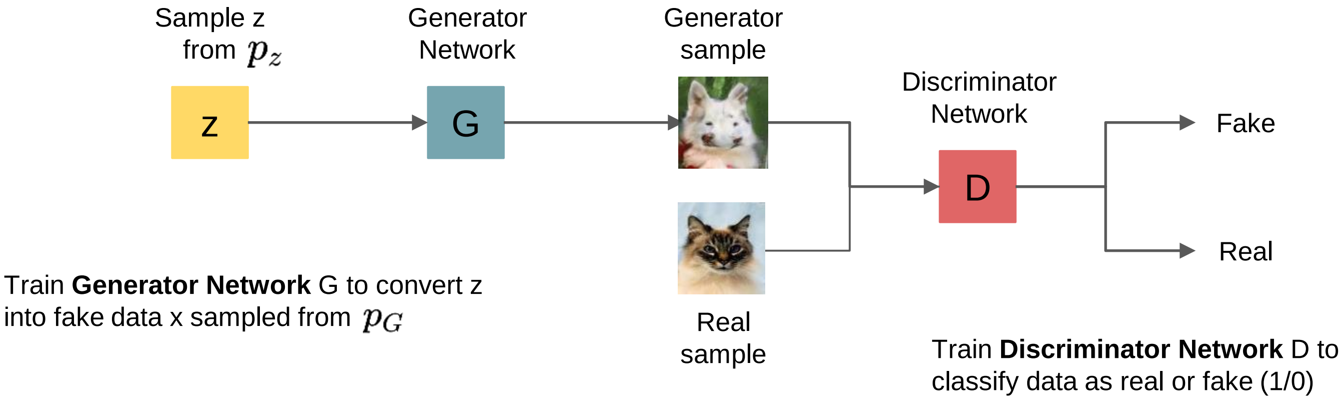
- Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. Want to sample from p_{data}.
- Idea: Introduce a latent variable z with simple prior $p(z)$.
- Sample $z \sim p(z)$ and pass to a Generator Network $x = G(z)$
- Then x is a sample from the Generator distribution p_G. Want $p_G = p_{data}$



Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks

- Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. Want to sample from p_{data}.
- Idea: Introduce a latent variable z with simple prior $p(z)$.
- Sample $z \sim p(z)$ and pass to a Generator Network $x = G(z)$
- Then x is a sample from the Generator distribution p_G. Want $p_G = p_{data}$

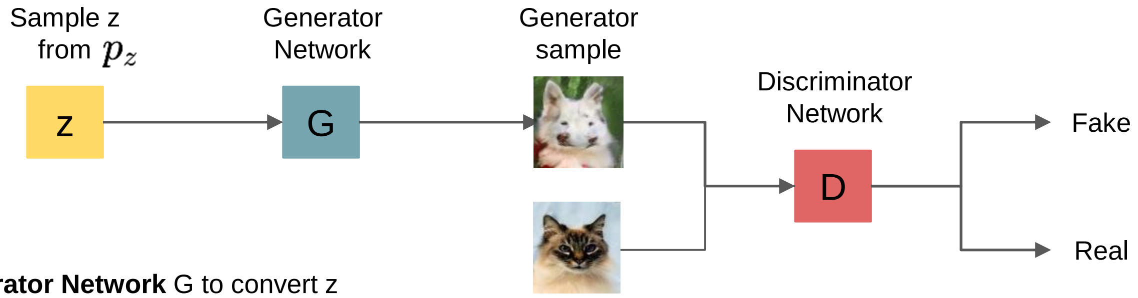


Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Training Objective

Jointly train generator G and discriminator D with a **minimax game**

$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} [\log(1 - D(G(z)))] \right)$$



Train **Generator Network** G to convert z into fake data x sampled from p_G **by fooling the Discriminator D**

Train **Discriminator Network** D to classify data as real or fake (1/0)

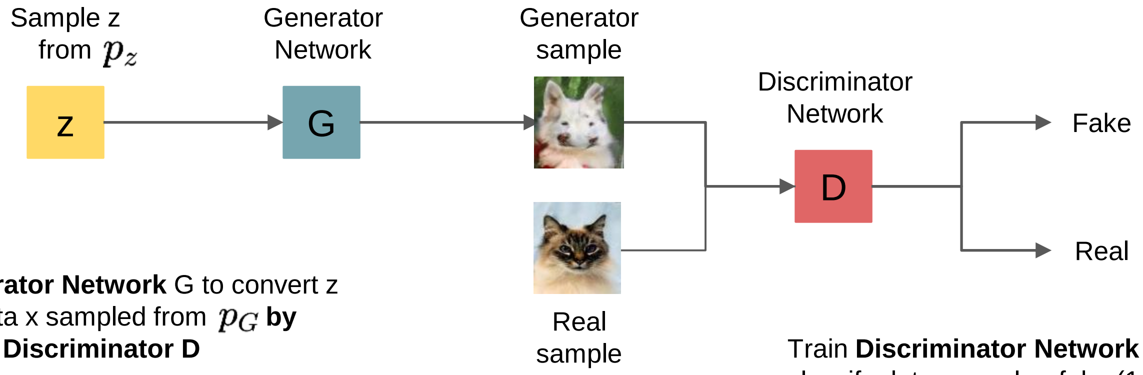
Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Training Objective

Jointly train generator G and discriminator D with a **minimax game**

Discriminator wants D(x)=1 for real data

$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} [\log(1 - D(G(z)))] \right)$$



Train **Generator Network** G to convert z into fake data x sampled from p_G **by fooling the Discriminator D**

Train **Discriminator Network** D to classify data as real or fake (1/0)

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Training Objective

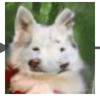
Jointly train generator G and discriminator D with a **minimax game**

Discriminator wants D(x)=0 for fake data

$$\min_{G}\max_{D}\left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}[\log(1 - D(G(z)))]\right)$$

Sample z from p_z

z → G (Generator Network) → Generator sample



Real sample

Discriminator Network

D → Fake / Real

Train **Generator Network** G to convert z into fake data x sampled from p_G **by fooling the Discriminator D**

Train **Discriminator Network** D to classify data as real or fake (1/0)

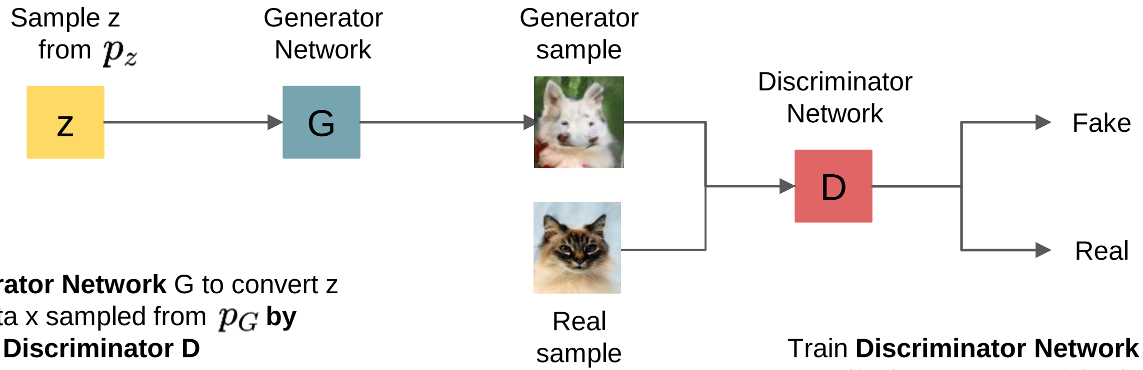
Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Training Objective

Jointly train generator G and discriminator D with a **minimax game**

$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} [\log(1 - D(G(z)))] \right)$$

Generator wants D(x)=1 for fake data



Train **Generator Network** G to convert z into fake data x sampled from p_G **by fooling the Discriminator D**

Train **Discriminator Network** D to classify data as real or fake (1/0)

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Training Objective

Jointly train generator G and discriminator D with a **minimax game**

$$\min_{\mathbf{G}} \max_{\mathbf{D}} \left(E_{x \sim p_{data}} [\log \mathbf{D}(x)] + E_{\mathbf{z} \sim p(\mathbf{z})} [\log(1 - \mathbf{D}(\mathbf{G}(\mathbf{z})))] \right)$$

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

Train G and D using alternating gradient updates:

1. Update $\mathbf{D} = \mathbf{D} + \alpha_{\mathbf{D}} \frac{\delta \mathbf{V}}{\delta \mathbf{D}}$
2. Update $\mathbf{G} = \mathbf{G} + \alpha_{\mathbf{G}} \frac{\delta \mathbf{V}}{\delta \mathbf{G}}$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

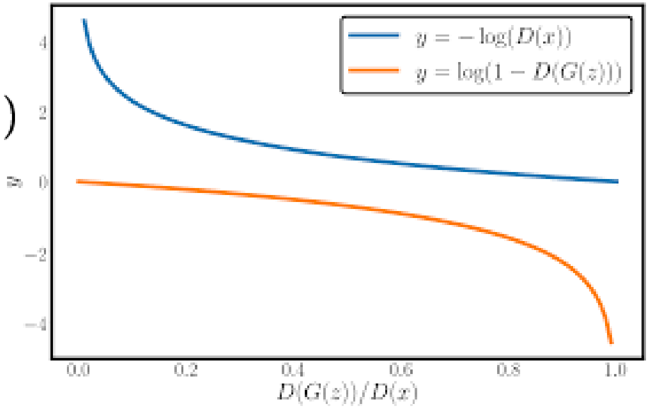
Generative Adversarial Networks: vanishing gradient

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

$$\nabla_{\Theta_G} V(G,D) = \nabla_{\Theta_G} E_{z\sim q(z)}[log(1-D(G(z)))]$$

$$\nabla_a \log(1-\sigma(a)) = \frac{-\nabla_a \sigma(a)}{1-\sigma(a)} = \frac{-\sigma(a)(1-\sigma(a))}{1-\sigma(a)} = -\sigma(a) = -D(G(z))$$

- Gradient goes to 0 if D is confident, i.e. $D(G(z)) \to 0$



Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

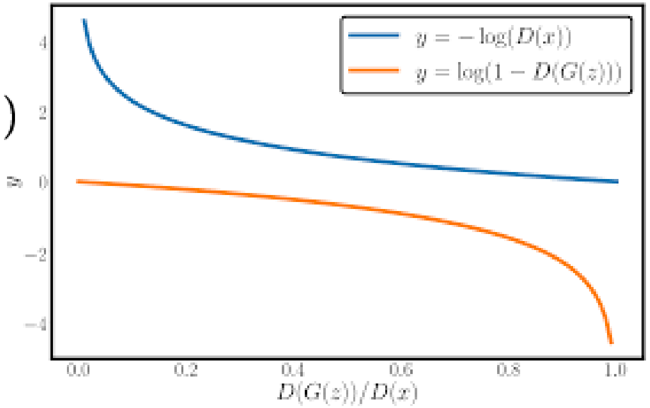
Generative Adversarial Networks: vanishing gradient

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

$$\nabla_{\Theta_G} V(G,D) = \nabla_{\Theta_G} E_{z\sim q(z)}[log(1 - D(G(z)))]$$

$$\nabla_a \log(1-\sigma(a)) = \frac{-\nabla_a \sigma(a)}{1-\sigma(a)} = \frac{-\sigma(a)(1-\sigma(a))}{1-\sigma(a)} = -\sigma(a) = -D(G(z))$$

- Gradient goes to 0 if D is confident, i.e. $D(G(z)) \rightarrow 0$
- Minimize $\boxed{-E_{z\sim p(z)}[\log(D(G(z)))]}$ for **generator**



Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G \max_D (E_{x \sim p_{data}}[\log D(x)] + E_{x \sim p_G}[\log(1 - D(x))])$$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G \max_D (E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_G}[\log(1 - D(x))])$$

$$= \min_G \max_D \int_X (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)))dx$$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G \max_D (E_{x \sim p_{data}}[\log D(x)] + E_{x \sim p_G}[\log(1 - D(x))])$$

$$= \min_G \max_D \int_X (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)))dx$$

$$= \min_G \int_X \max_D (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)))dx$$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$f(y) = a\log y + b\log(1-y)$$

$$= \min_{G}\max_{D}(E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_G}[\log(1 - D(x))])$$

$$= \min_{G}\max_{D}\int_X (p_{data}(x)\log D(x) + p_G(x)\log(1 - D(x)))dx$$

$$= \min_{G}\int_X \max_{D}(p_{data}(x)\log D(x) + p_G(x)\log(1 - D(x)))dx$$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G \max_D (E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_G}[\log(1-D(x))])$$

$$= \min_G \max_D \int_X (p_{data}(x)\log D(x) + p_G(x)\log(1-D(x)))dx$$

$$= \min_G \int_X \max_D (p_{data}(x)\log D(x) + p_G(x)\log(1-D(x)))dx$$

$$f(y) = a\log y + b\log(1-y)$$

$$f'(y) = \frac{a}{y} - \frac{b}{1-y}$$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G \max_D (E_{x \sim p_{data}}[\log D(x)] + E_{x \sim p_G}[\log(1 - D(x))])$$

$$= \min_G \max_D \int_X (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)))dx$$

$$= \min_G \int_X \max_D (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)))dx$$

$$f(y) = a \log y + b \log(1 - y)$$

$$f'(y) = \frac{a}{y} - \frac{b}{1-y}$$

$$f'(y) = 0 \Leftrightarrow y = \frac{a}{a+b}$$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G \max_D (E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_G}[\log(1-D(x))])$$

$$= \min_G \max_D \int_X (p_{data}(x) \log D(x) + p_G(x) \log(1-D(x)))dx$$

$$= \min_G \int_X \max_D (p_{data}(x) \log D(x) + p_G(x) \log(1-D(x)))dx$$

$$f(y) = a \log y + b \log(1-y)$$

$$f'(y) = \frac{a}{y} - \frac{b}{1-y}$$

$$f'(y) = 0 \Leftrightarrow y = \frac{a}{a+b}$$

Optimal Discriminator:

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}$$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} [\log(1 - D(G(z)))] \right)$$

$$= \min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{x \sim p_{G}} [\log(1 - D(x))] \right)$$

$$= \min_{G} \int_X (p_{data}(x) \log D_G^*(x) + p_G(x) \log(1 - D_G^*(x)))dx$$

Optimal Discriminator: $D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G \max_D (E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_G}[\log(1 - D(x))])$$

$$= \min_G \int_X (p_{data}(x) \log D_G^*(x) + p_G(x) \log(1 - D_G^*(x)))dx$$

$$= \min_G \int_X (p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x)+p_G(x)} + p_G(x) \log \frac{p_G(x)}{p_{data}(x)+p_G(x)})dx$$

Optimal Discriminator: $D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G \int_X (p_{data}(x)\log D^*_G(x) + p_G(x)\log(1 - D^*_G(x)))dx$$

$$= \min_G \int_X (p_{data}(x)\log \frac{p_{data}(x)}{p_{data}(x)+p_G(x)} + p_G(x)\log \frac{p_G(x)}{p_{data}(x)+p_G(x)})dx$$

$$= \min_G (E_{x\sim p_{data}}[\log \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \frac{p_G(x)}{p_{data}(x)+p_G(x)}])$$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G \int_X (p_{data}(x) \log D_G^*(x) + p_G(x) \log(1 - D_G^*(x)))dx$$

$$= \min_G \int_X (p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x)+p_G(x)} + p_G(x) \log \frac{p_G(x)}{p_{data}(x)+p_G(x)})dx$$

$$= \min_G (E_{x\sim p_{data}}[\log \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \frac{p_G(x)}{p_{data}(x)+p_G(x)}])$$

$$= \min_G (E_{x\sim p_{data}}[\log \frac{2\times p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \frac{2\times p_G(x)}{p_{data}(x)+p_G(x)}] - \log 4)$$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G (E_{x\sim p_{data}}[\log \frac{2\times p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \frac{2\times p_G(x)}{p_{data}(x)+p_G(x)}] - \log 4)$$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G (E_{x\sim p_{data}}[\log \frac{2\times p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \frac{2\times p_G(x)}{p_{data}(x)+p_G(x)}] - \log 4)$$

Kullback-Leibler Divergence: $KL(p,q) = E_{x\sim p}[log \frac{p(x)}{q(x)}]$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G (E_{x\sim p_{data}}[\log \frac{2\times p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \frac{2\times p_G(x)}{p_{data}(x)+p_G(x)}] - \log 4)$$

$$= \min_G (KL(p_{data}, \frac{p_{data}+p_G}{2}) + KL(p_G, \frac{p_{data}+p_G}{2}) - \log 4)$$

Kullback-Leibler Divergence: $KL(p,q) = E_{x\sim p}[log\frac{p(x)}{q(x)}]$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G (E_{x\sim p_{data}}[\log \frac{2\times p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \frac{2\times p_G(x)}{p_{data}(x)+p_G(x)}] - \log 4)$$

$$= \min_G (KL(p_{data}, \frac{p_{data}+p_G}{2}) + KL(p_G, \frac{p_{data}+p_G}{2}) - \log 4)$$

Jensen-Shannon Divergence: $JSD(p,q) = \frac{1}{2}KL(p, \frac{p+q}{2}) + \frac{1}{2}KL(q, \frac{p+q}{2})$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

$$\min_G \max_D (E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}[\log(1 - D(G(z)))])$$
$$= \min_G (E_{x\sim p_{data}}[\log \frac{2\times p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \frac{2\times p_G(x)}{p_{data}(x)+p_G(x)}] - \log 4)$$
$$= \min_G (KL(p_{data}, \frac{p_{data}+p_G}{2}) + KL(p_G, \frac{p_{data}+p_G}{2}) - \log 4)$$
$$= \min_G (2 \times JSD(p_{data}, p_G) - \log 4)$$

Jensen-Shannon Divergence: $JSD(p,q) = \frac{1}{2}KL(p, \frac{p+q}{2}) + \frac{1}{2}KL(q, \frac{p+q}{2})$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

$$\min_G \max_D (E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}[\log(1 - D(G(z)))])$$
$$= \min_G (E_{x\sim p_{data}}[\log \frac{2\times p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \frac{2\times p_G(x)}{p_{data}(x)+p_G(x)}] - \log 4)$$
$$= \min_G (KL(p_{data}, \frac{p_{data}+p_G}{2}) + KL(p_G, \frac{p_{data}+p_G}{2}) - \log 4)$$
$$= \min_G (2 \times JSD(p_{data}, p_G) - \log 4)$$

JSD is always nonnegative and zero when the two distributions are equal

=> the global minimum is $p_{data} = p_G$

Jensen-Shannon Divergence: $JSD(p,q) = \frac{1}{2}KL(p, \frac{p+q}{2}) + \frac{1}{2}KL(q, \frac{p+q}{2})$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

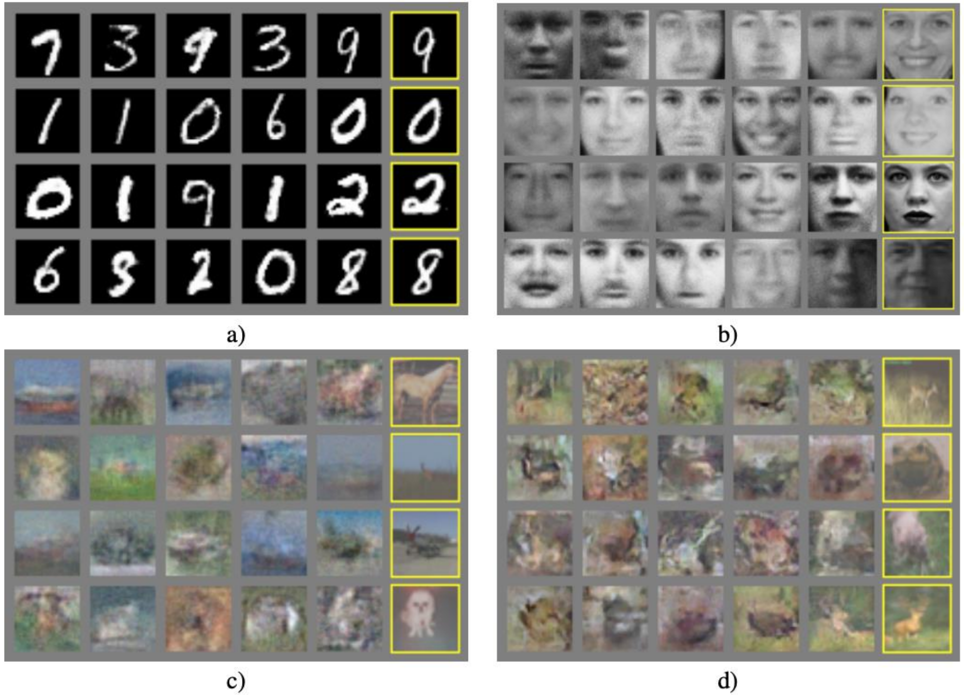
$$= \min_G (2 * JSD(p_{data}, p_G) - \log 4)$$

Summary: The global minimum of the minimax game happens when:

1. $D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}$ (Optimal discriminator for any G)
2. $p_G(x) = p_{data}(x)$ (Optimal generator for optimal D)

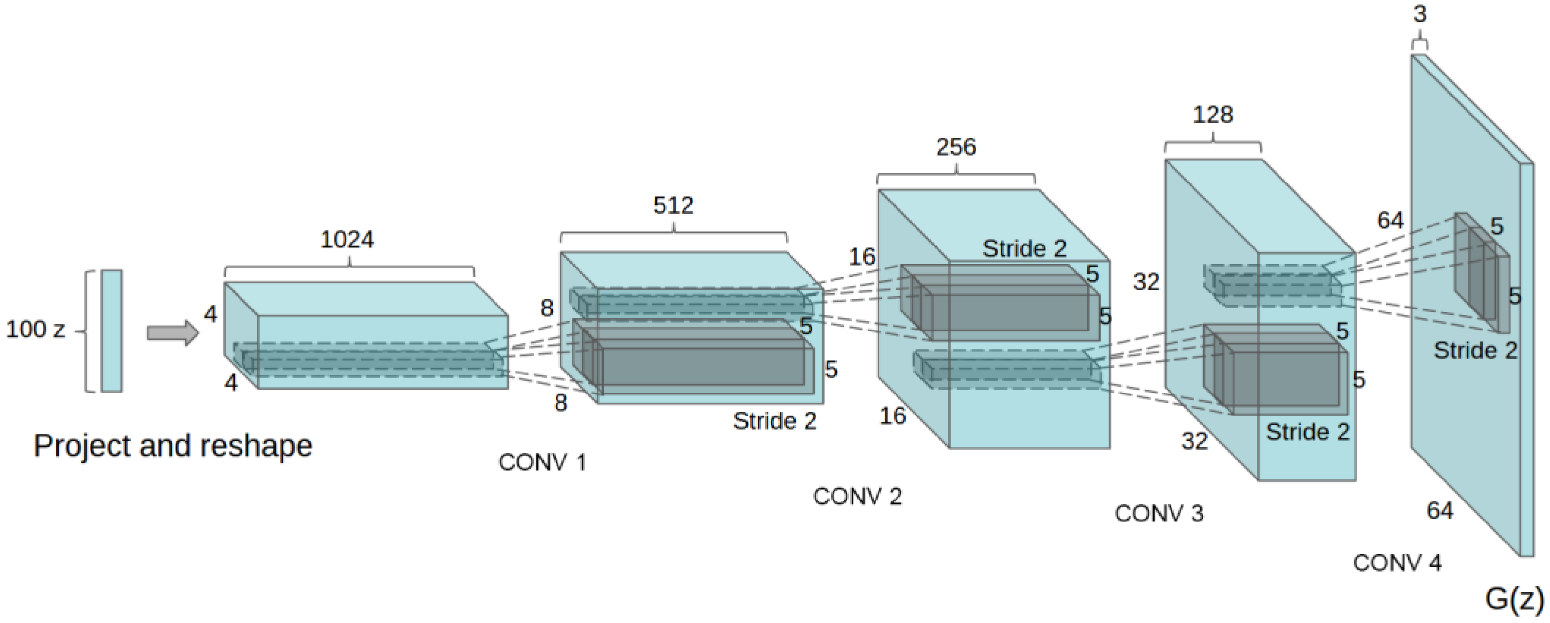
Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: results



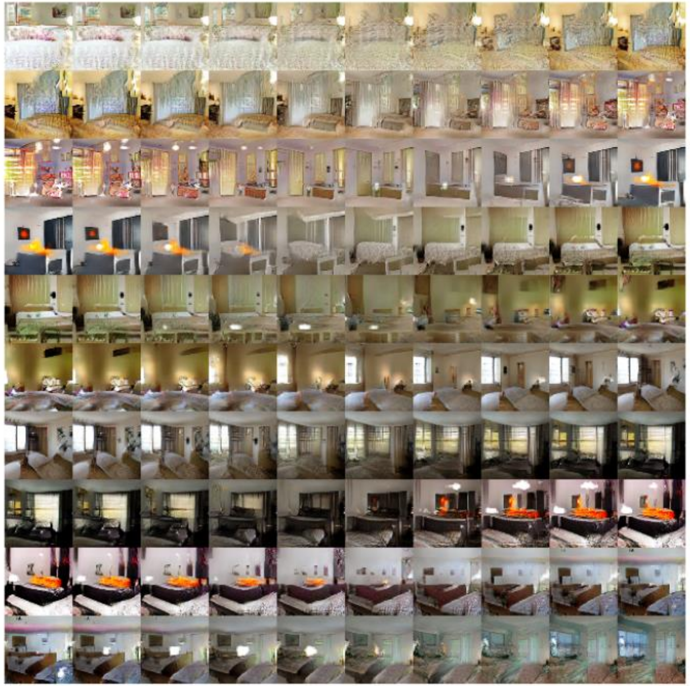
Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: DC-GAN



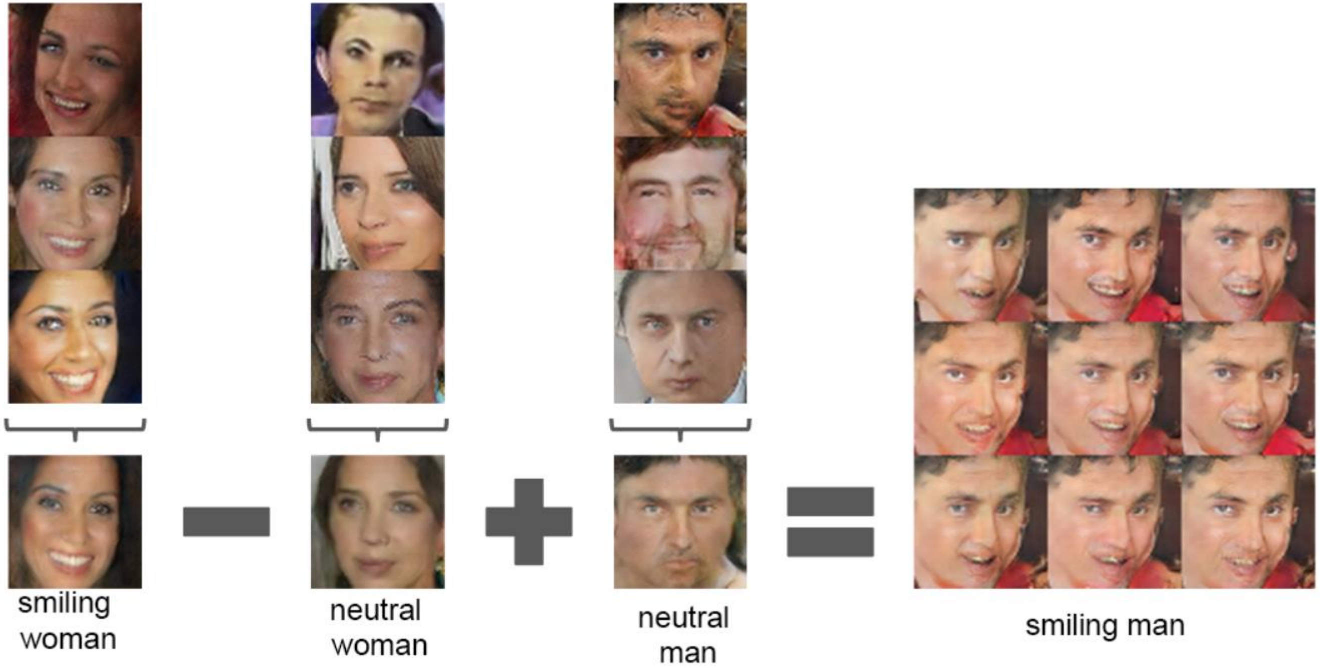
Alec Radford, Luke Metz, Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ICLR 2016

Generative Adversarial Networks: Interpolation



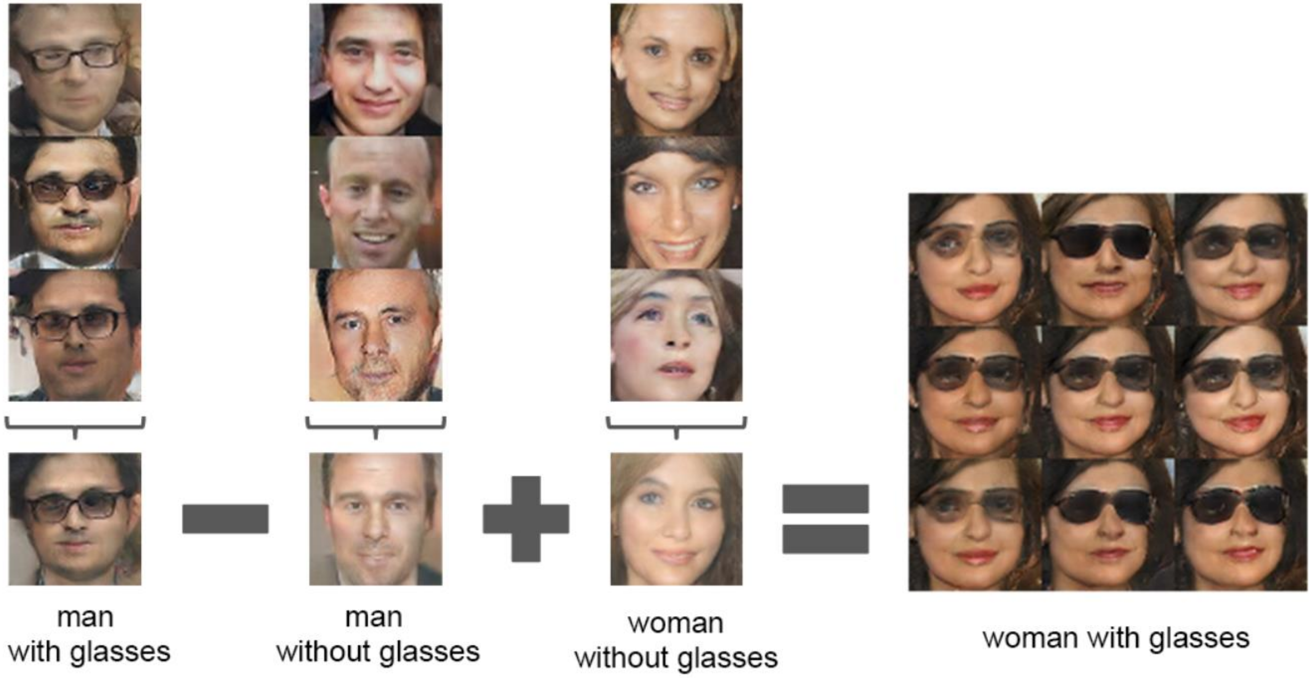
Alec Radford, Luke Metz, Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ICLR 2016

Generative Adversarial Networks: Vector Math



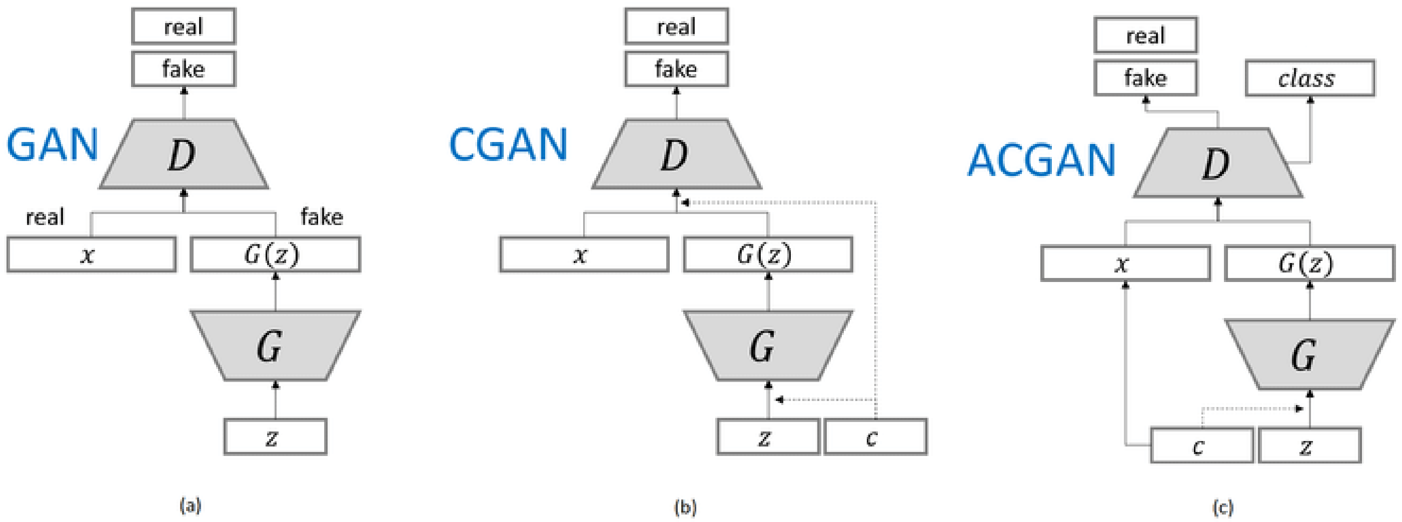
Alec Radford, Luke Metz, Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ICLR 2016

Generative Adversarial Networks: Vector Math



Alec Radford, Luke Metz, Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ICLR 2016

Conditional GANs



[b] Mehdi Mirza, Simon Osindero. Conditional Generative Adversarial Nets. 2014

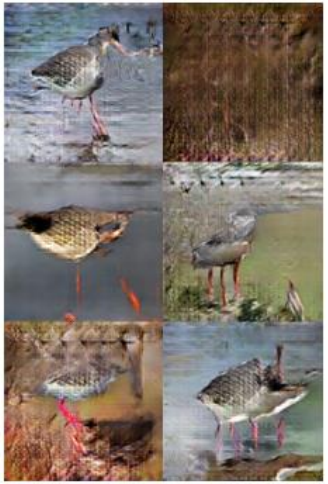
[c] Augustus Odena, Christopher Olah, Jonathon Shlens. Conditional Image Synthesis With Auxiliary Classifier GANs. ICML 2016

Conditional GANs


monarch butterfly


goldfinch


daisy


redshank


grey whale

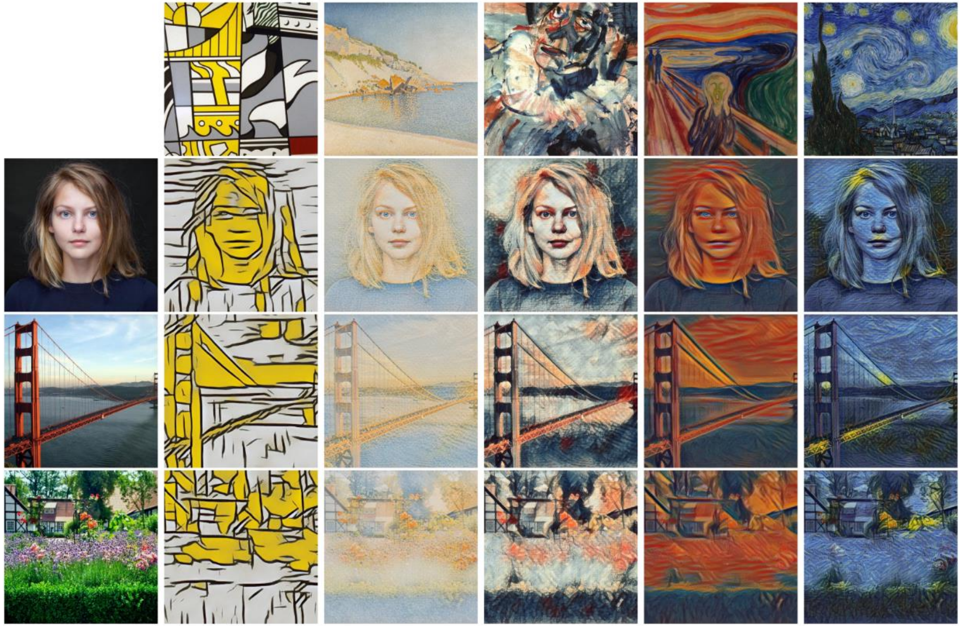
Augustus Odena, Christopher Olah, Jonathon Shlens. Conditional Image Synthesis With Auxiliary Classifier GANs. ICML 2016

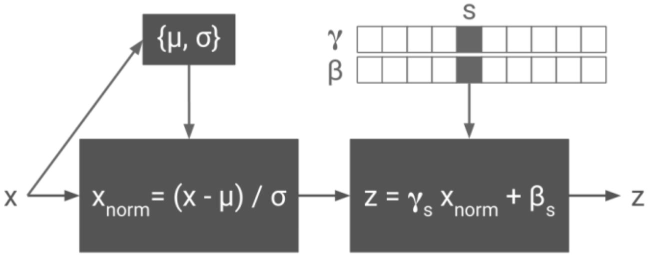
Conditional GANs: BigGAN

Figure 6: Samples generated by our BigGAN model at 512×512 resolution.

Andrew Brock, Jeff Donahue, Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. ICLR 2019 https://arxiv.org/abs/1809.11096

Conditional GANs: Conditional Batch Normalization

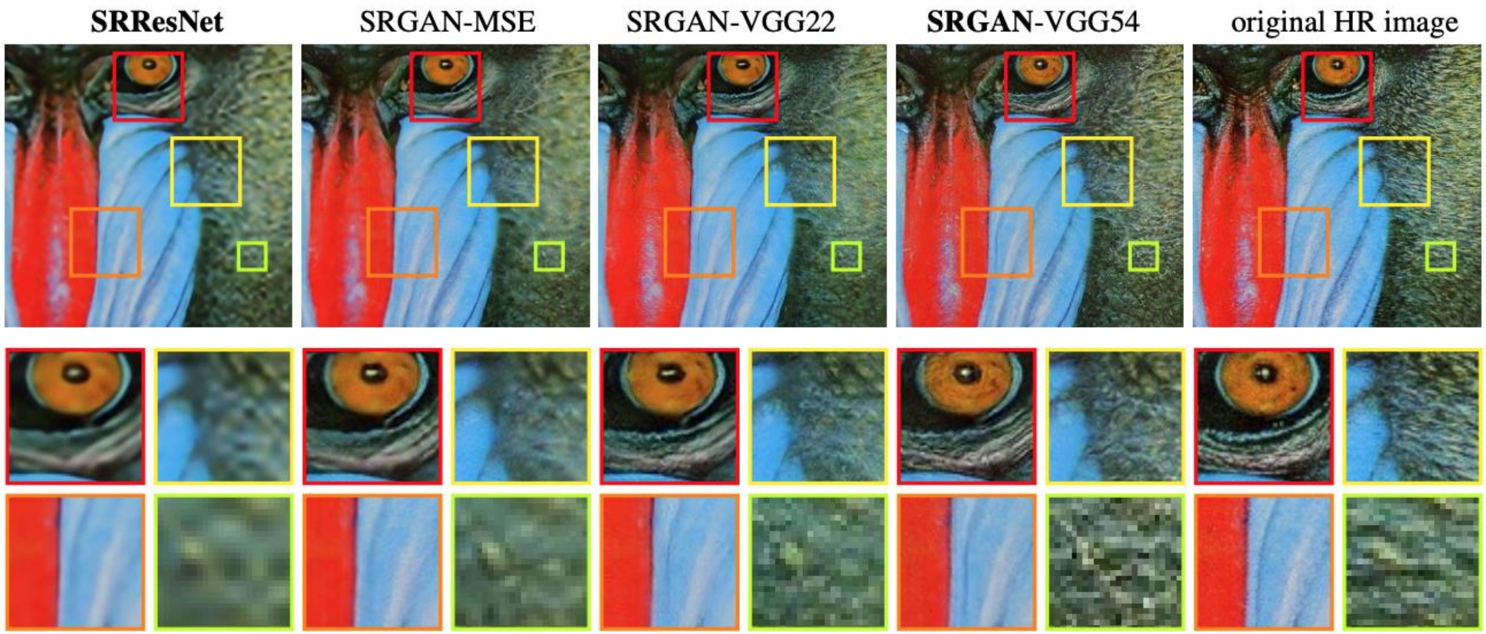




The input activation x is normalized across spatial dimensions and scaled and shifted using style-dependent parameter vectors γ_s, β_s where s indexes the style label.

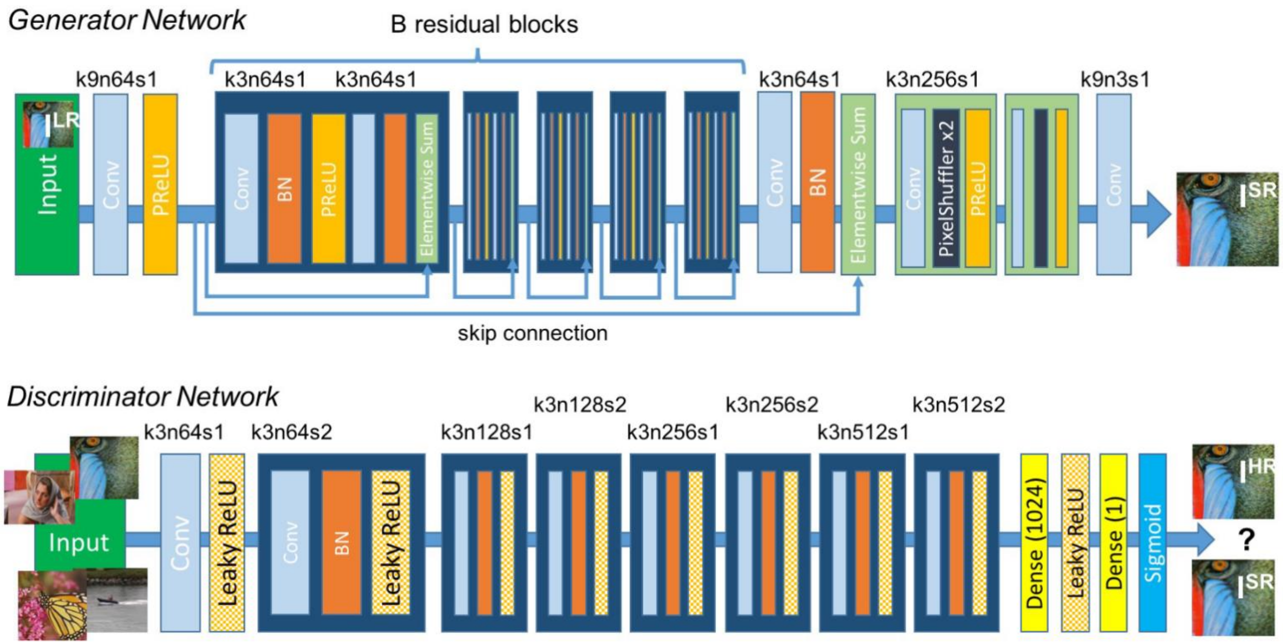
Vincent Dumoulin, Jonathon Shlens, Manjunath Kudlur, A Learned Representation For Artistic Style. ICLR 2017 https://arxiv.org/abs/1610.07629

Image Super-Resolution



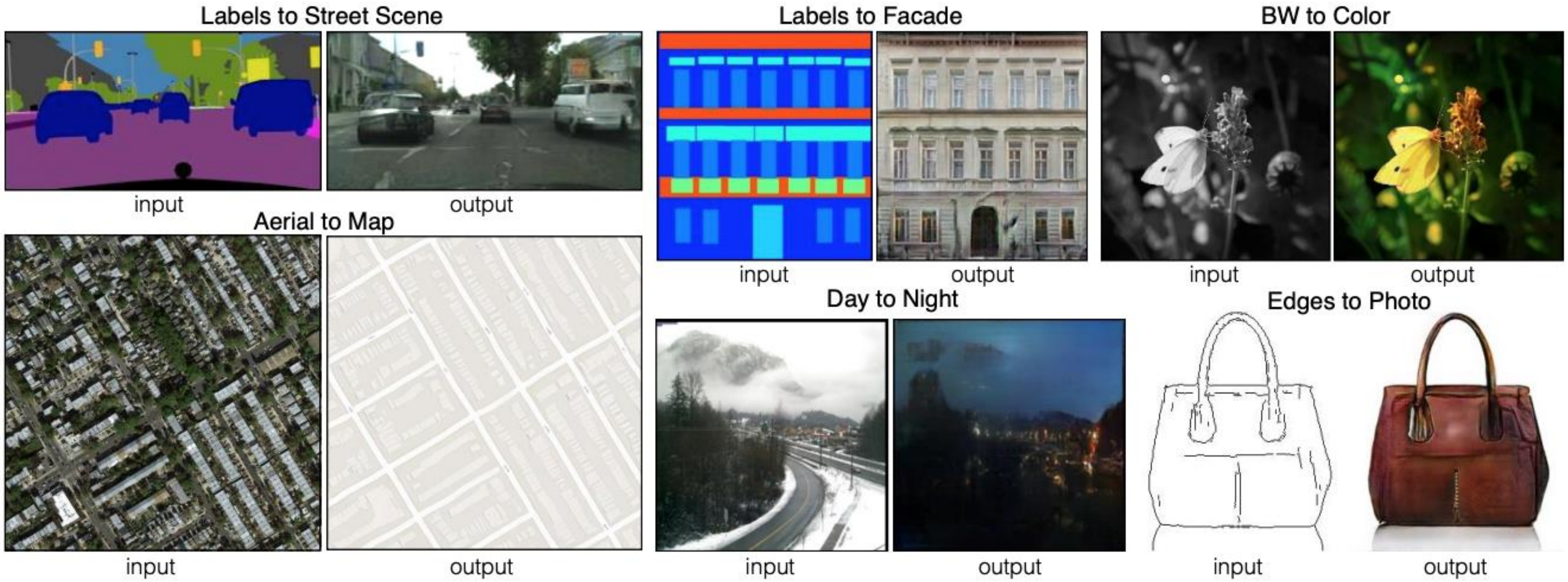
Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. CVPR 2017 https://arxiv.org/abs/1609.04802

Image Super-Resolution



Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, Wenzhe Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. CVPR 2017 https://arxiv.org/abs/1609.04802

Image-to-Image Translation: Pix2Pix



Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. CVPR 2017 https://arxiv.org/abs/1611.07004

Image-to-Image Translation: Pix2Pix

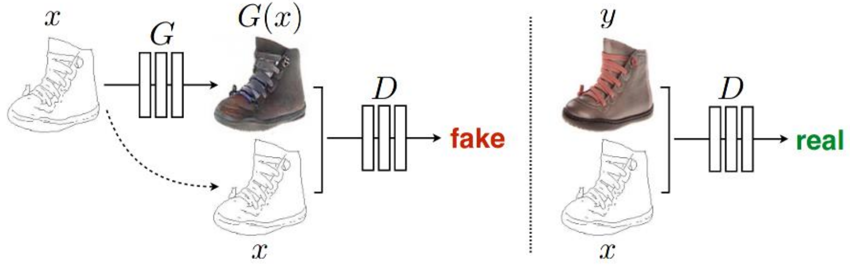
Objective:

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

where

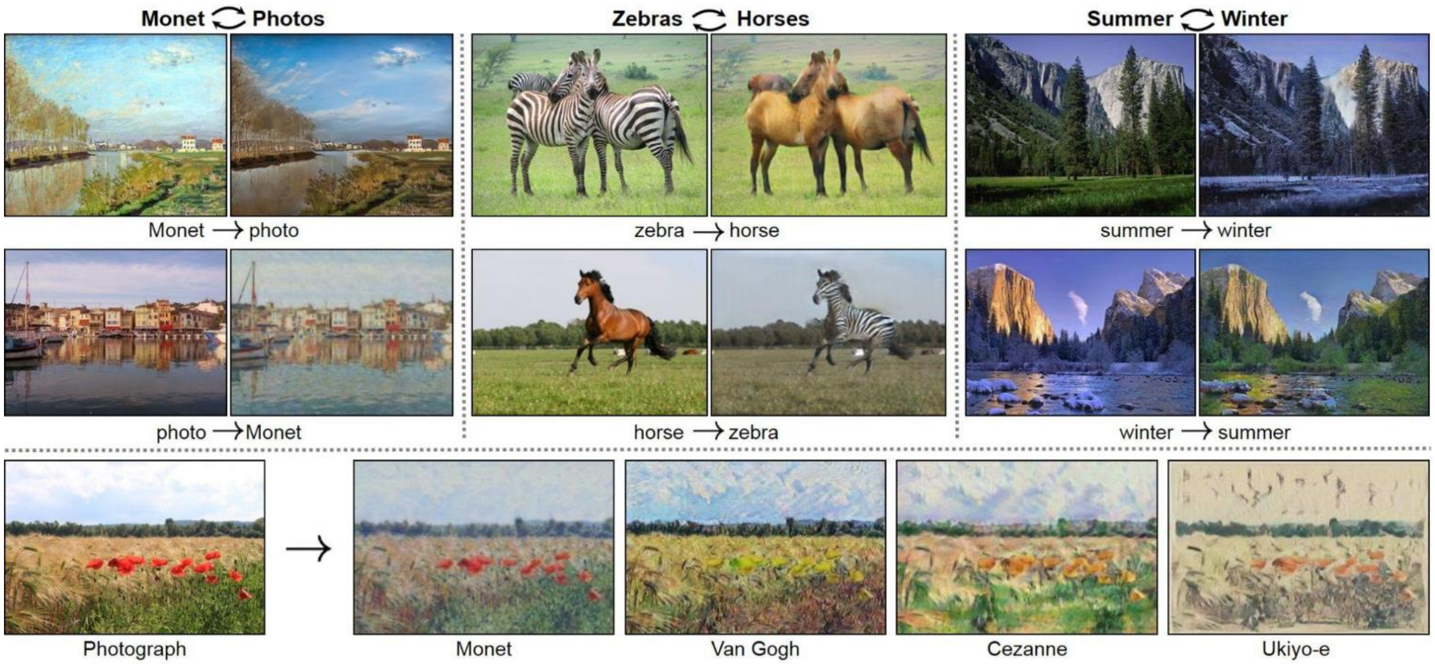
$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))]$$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]$$



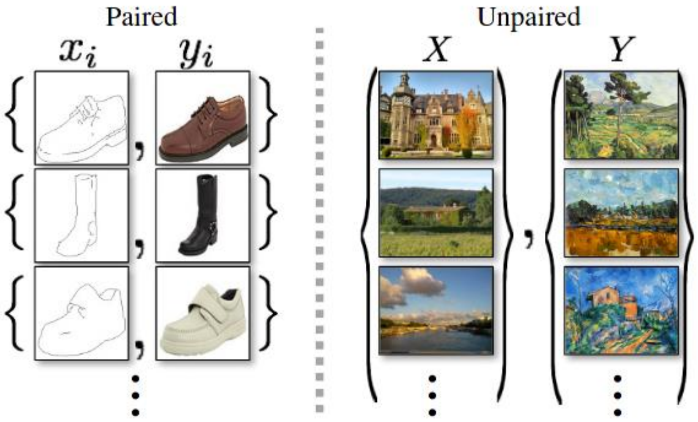
Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. CVPR 2017 https://arxiv.org/abs/1611.07004

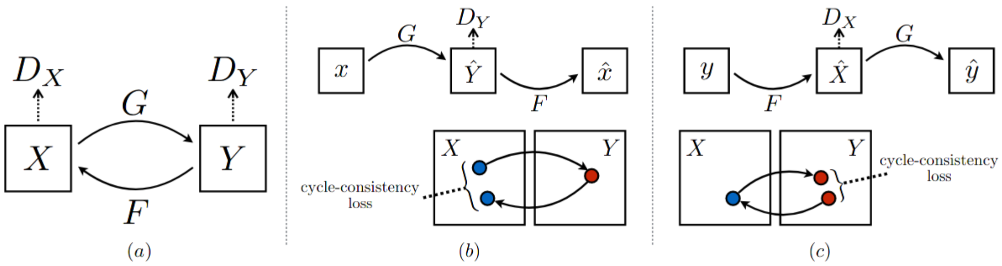
Unpaired Image-to-Image Translation: CycleGAN



Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. ICCV 2017
https://arxiv.org/abs/1703.10593

Unpaired Image-to-Image Translation: CycleGAN





Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. ICCV 2017
https://arxiv.org/abs/1703.10593

Unpaired Image-to-Image Translation: CycleGAN

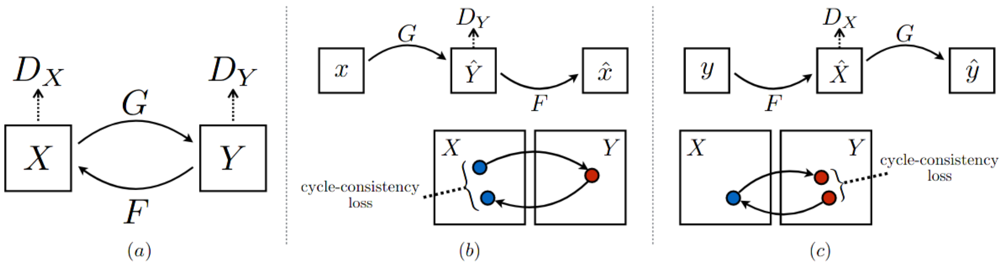
Objective:

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

where

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

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

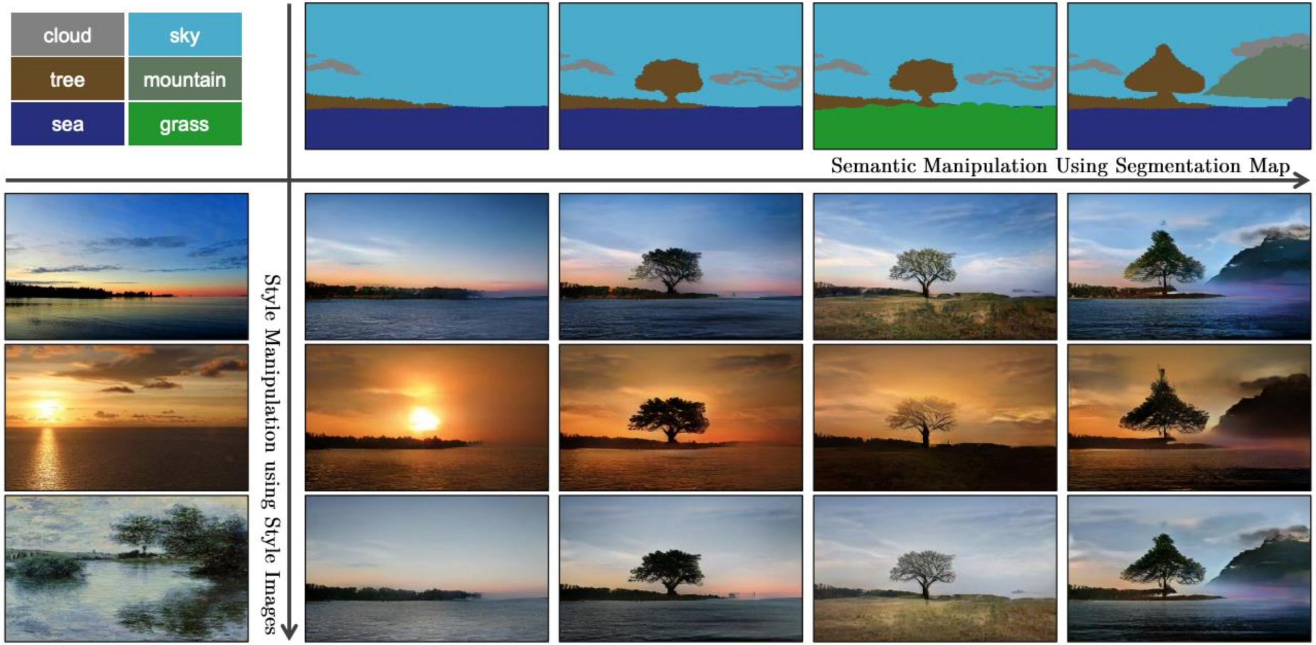


Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. ICCV 2017
https://arxiv.org/abs/1703.10593

Unpaired Image-to-Image Translation: CycleGAN

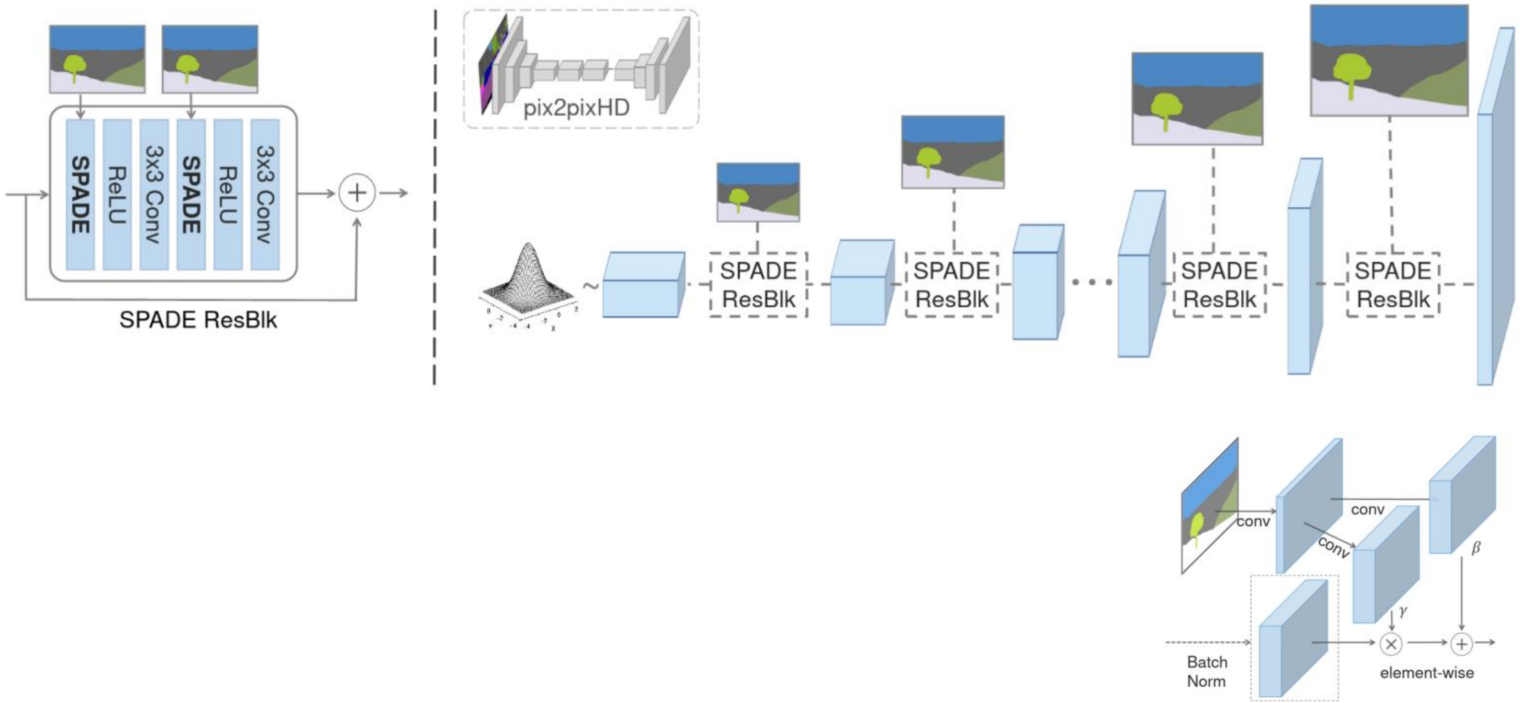
Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. ICCV 2017
https://arxiv.org/abs/1703.10593

Label Map to Image



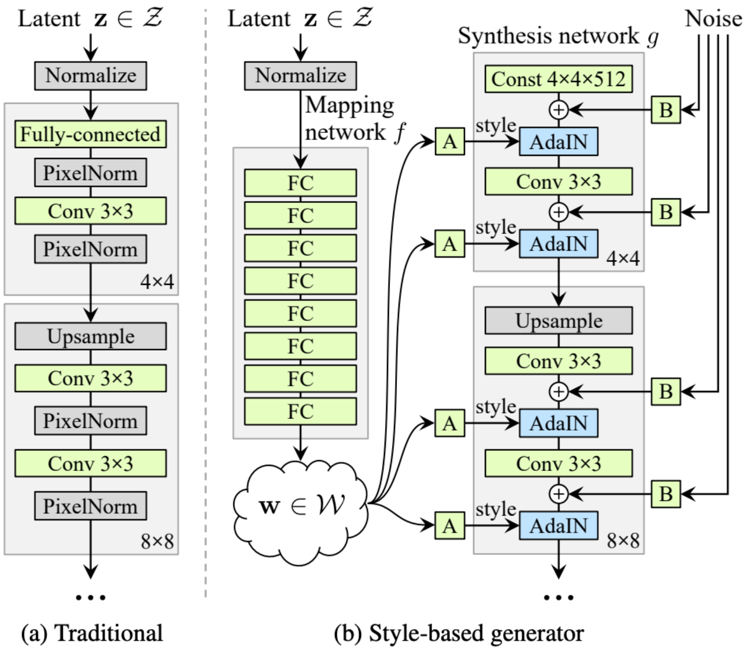
Taesung Park, Ming-Yu Liu, Ting-Chun Wang, Jun-Yan Zhu. Semantic Image Synthesis with Spatially-Adaptive Normalization. CVPR 2019 https://arxiv.org/abs/1903.07291

Label Map to Image



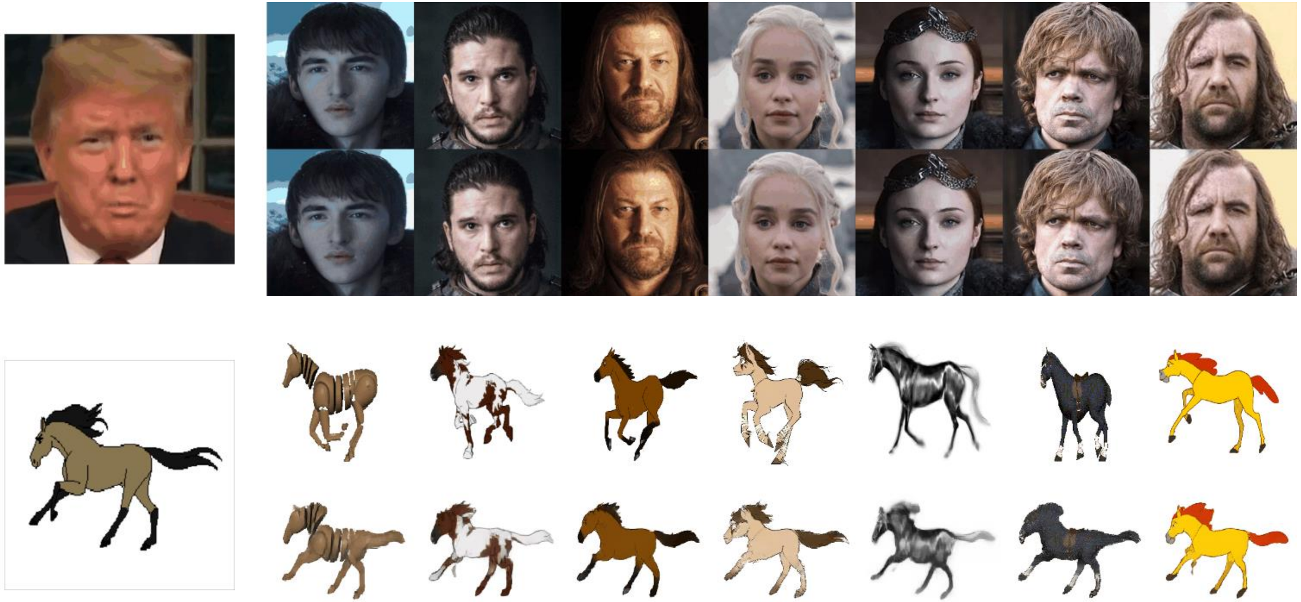
Taesung Park, Ming-Yu Liu, Ting-Chun Wang, Jun-Yan Zhu. Semantic Image Synthesis with Spatially-Adaptive Normalization. CVPR 2019 https://arxiv.org/abs/1903.07291

StyleGAN



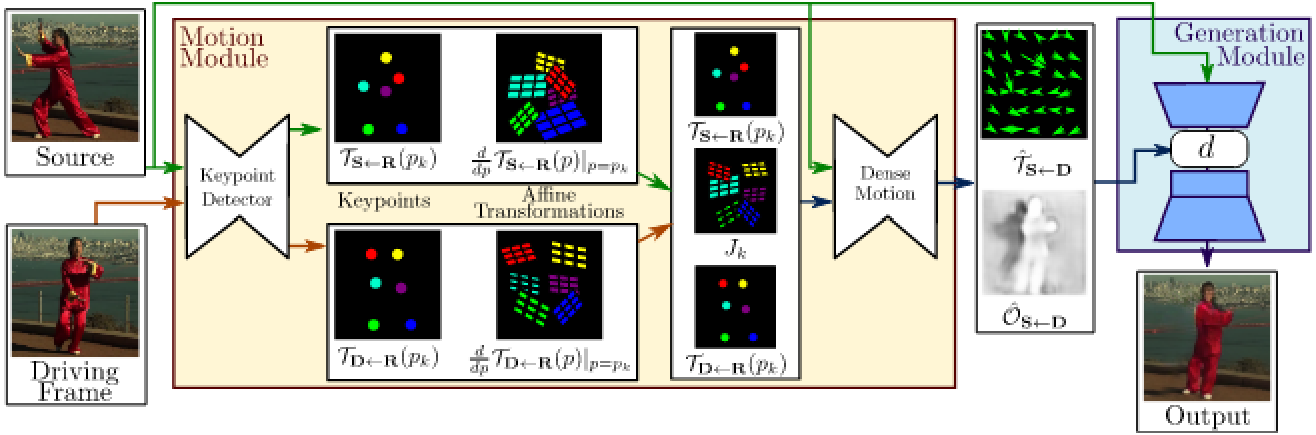
Tero Karras, Samuli Laine, Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. CVPR 2019

Video Generation



Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, Nicu Sebe. First Order Motion Model for Image Animation. NeurIPS 2019

Video Generation

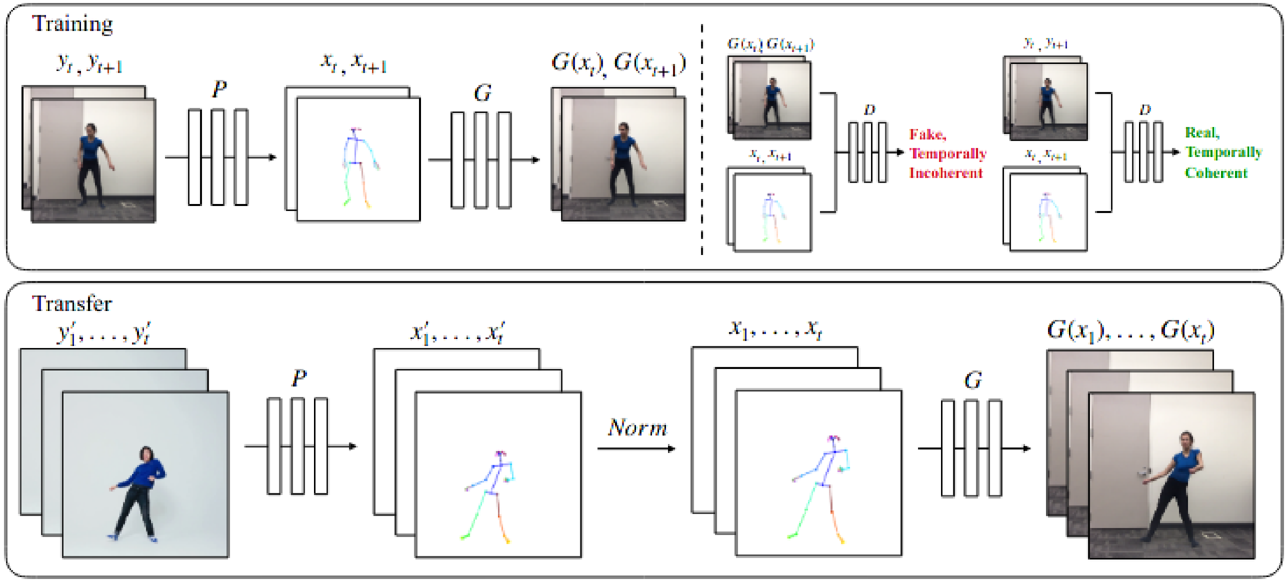


Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, Nicu Sebe. First Order Motion Model for Image Animation. NeurIPS 2019

Video Generation. Everybody Dance Now

Caroline Chan, Shiry Ginosar, Tinghui Zhou, Alexei A. Efros. Everybody Dance Now. ICCV 2019

Video Generation. Everybody Dance Now



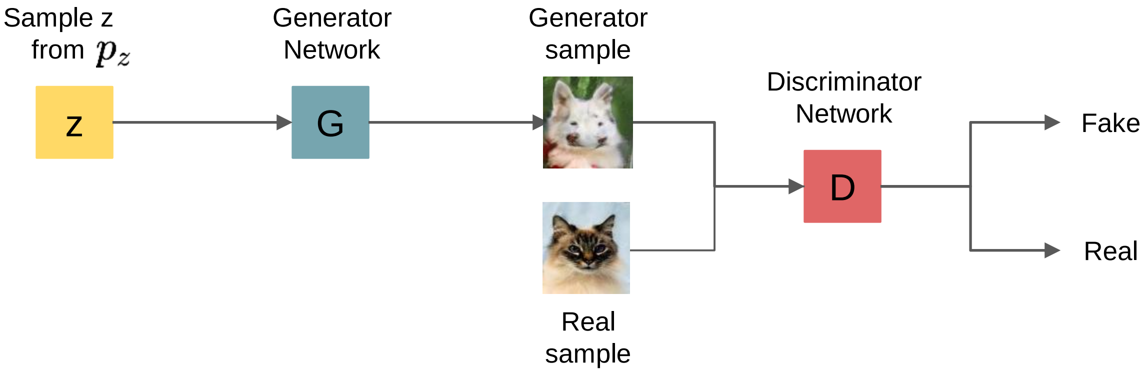
Caroline Chan, Shiry Ginosar, Tinghui Zhou, Alexei A. Efros. Everybody Dance Now. ICCV 2019

GAN Summary

Jointly train two networks:

Discriminator classifies data as real or fake

Generator generates data that fools the discriminator



Part 2

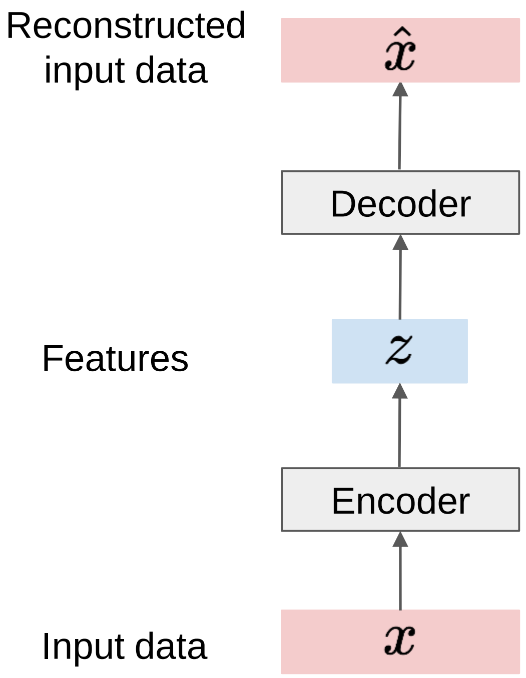
Outline

- Variational Autoencoders (VAE)
- Mode collapse
- Wasserstein GAN
- GAN evaluation

Autoencoders (non-variational)

Autoencoder learns latent features for data without any labels.

$$L = ||\hat{x} - x||_2^2$$

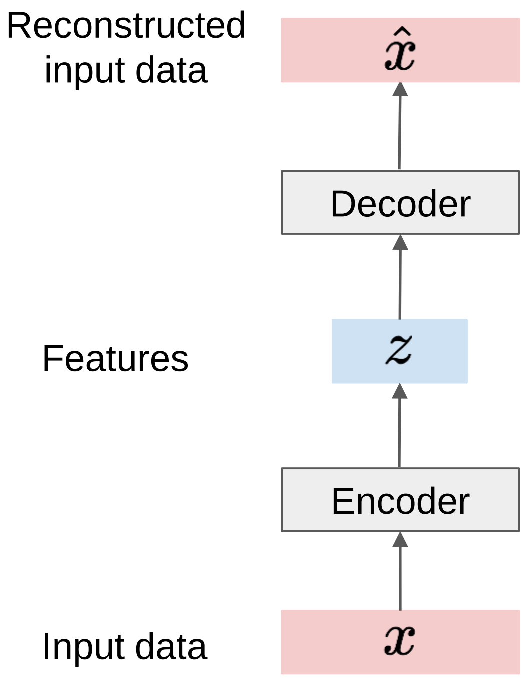


Autoencoders (non-variational)

Autoencoder learns latent features for data without any labels.

Features need to be low dimensional than the data.

$$L = ||\hat{x} - x||_2^2$$



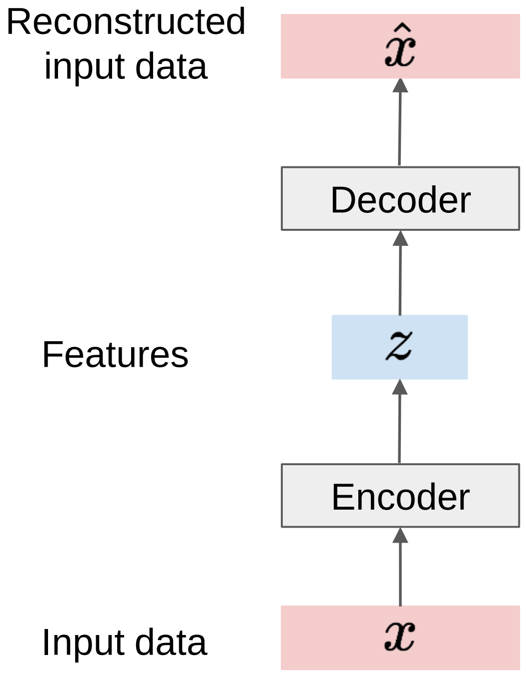
Autoencoders (non-variational)

Autoencoder learns latent features for data without any labels.

Features need to be low dimensional than the data.

Limitation: no way to produce any new content

$$L = ||\hat{x} - x||_2^2$$



Variational Autoencoders (VAE)

- VAE is an autoencoder whose training is regularised to avoid overfitting and ensure that the latent space has good properties that enable generative process.

Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014 https://arxiv.org/abs/1312.6114

Variational Autoencoders (VAE)

- VAE is an autoencoder whose training is regularised to avoid overfitting and ensure that the latent space has good properties that enable generative process.
- Instead of encoding an input as a single point, VAE encodes it as a distribution over the latent space.

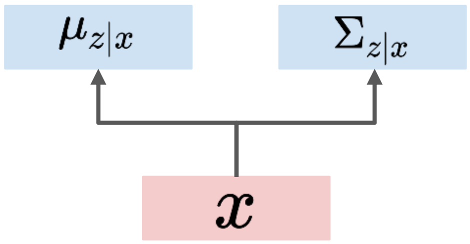
Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014 https://arxiv.org/abs/1312.6114

Variational Autoencoders (VAE)

Encoder network inputs data x and outputs distribution over latent codes z

Encoder Network

$$q_\phi(z|x) = N(\mu_{z|x}, \Sigma_{z|x})$$



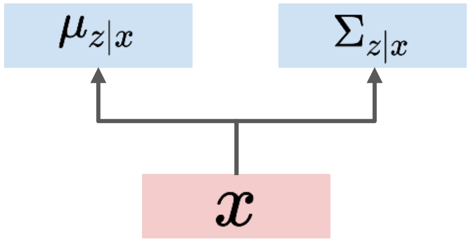
Variational Autoencoders (VAE)

Encoder network inputs data x and outputs distribution over latent codes z

Decoder network inputs latent code z and outputs distribution over data x

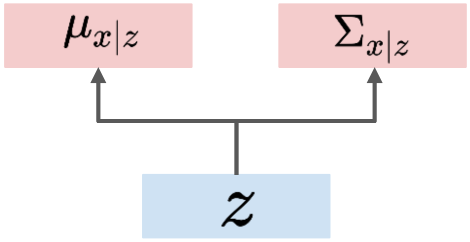
Encoder Network

$$q_\phi(z|x) = N(\mu_{z|x}, \Sigma_{z|x})$$



Decoder Network

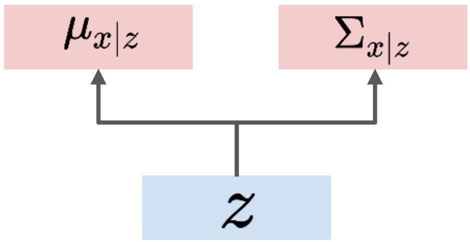
$$p_\theta(x|z) = N(\mu_{x|z}, \Sigma_{x|z})$$



Variational Autoencoders (VAE)

Decoder neural network represent p(x|z) where x is an image, z is latent factors to generate x.

$$p_\theta(x|z) = N(\mu_{x|z}, \Sigma_{x|z})$$



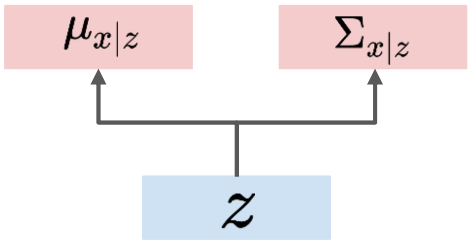
Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014 https://arxiv.org/abs/1312.6114

Variational Autoencoders (VAE)

Decoder neural network represent p(x|z) where x is an image, z is latent factors to generate x.

Assume prior p(z), e.g. Gaussian.

$$p_\theta(x|z) = N(\mu_{x|z}, \Sigma_{x|z})$$



Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014 https://arxiv.org/abs/1312.6114

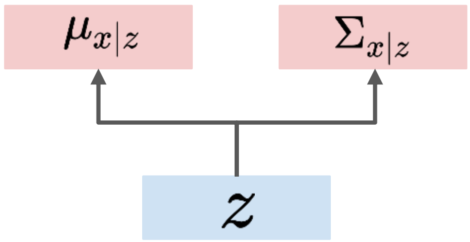
Variational Autoencoders (VAE)

Decoder neural network represent p(x|z) where x is an image, z is latent factors to generate x.

Assume prior p(z), e.g. Gaussian.

How to train this model?

$$p_\theta(x|z) = N(\mu_{x|z}, \Sigma_{x|z})$$



Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014 https://arxiv.org/abs/1312.6114

Variational Autoencoders (VAE)

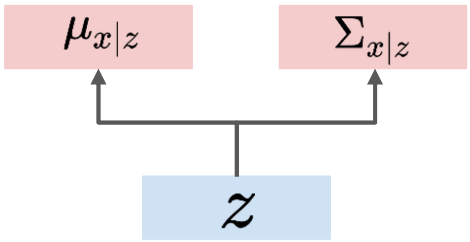
Decoder neural network represent p(x|z) where x is an image, z is latent factors to generate x.

Assume prior p(z), e.g. Gaussian.

How to train this model?

Maximize likelihood of data

$$p_\theta(x|z) = N(\mu_{x|z}, \Sigma_{x|z})$$



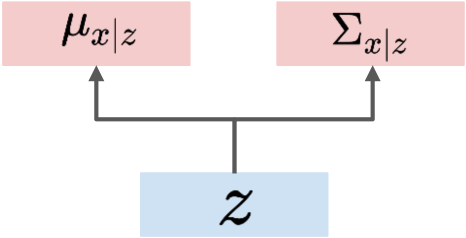
Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014 https://arxiv.org/abs/1312.6114

Variational Autoencoders (VAE)

Maximize likelihood of data

If we could observe z for each x, then we could train conditional generative model p(x|z) :

$$p_\theta(x) = \int p_\theta(x, z)dz = \int p_\theta(x|z)p_\theta(z)dz$$



Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014 https://arxiv.org/abs/1312.6114

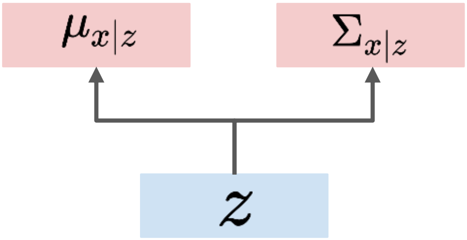
Variational Autoencoders (VAE)

Maximize likelihood of data

If we could observe z for each x, then we could train conditional generative model p(x|z) :

$$p_\theta(x) = \int p_\theta(x, z)dz = \int p_\theta(x|z)p_\theta(z)dz$$

Problem: impossible to integrate over all z

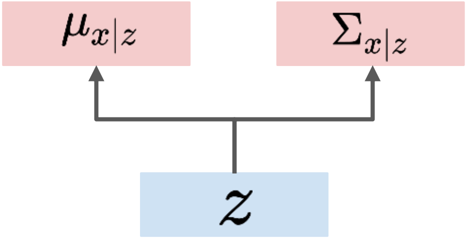


Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014 https://arxiv.org/abs/1312.6114

Variational Autoencoders (VAE)

Maximize likelihood of data

Bayes' Rule: $p_\theta(x) = \dfrac{p_\theta(x|z)p_\theta(z)}{p_\theta(z|x)}$



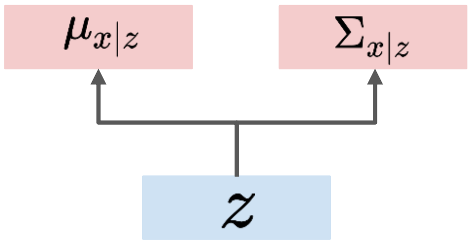
Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014 https://arxiv.org/abs/1312.6114

Variational Autoencoders (VAE)

Maximize likelihood of data

Bayes' Rule: $p_\theta(x) = \dfrac{p_\theta(x|z)p_\theta(z)}{p_\theta(z|x)}$

Problem: no way to compute $p_\theta(z|x)$



Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014 https://arxiv.org/abs/1312.6114

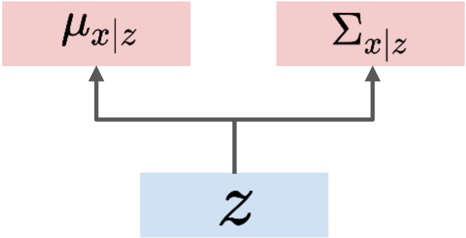
Variational Autoencoders (VAE)

Maximize likelihood of data

Bayes' Rule: $p_\theta(x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(z|x)}$

Problem: no way to compute $p_\theta(z|x)$

Solution: train another network (encoder) that learns $q_\phi(z|x) \approx p_\theta(z|x)$



Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014 https://arxiv.org/abs/1312.6114

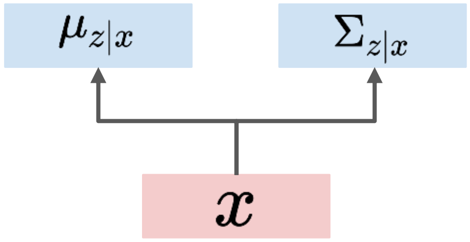
Variational Autoencoders (VAE)

Encoder network inputs data x and outputs distribution over latent codes z

Decoder network inputs latent code z and outputs distribution over data x

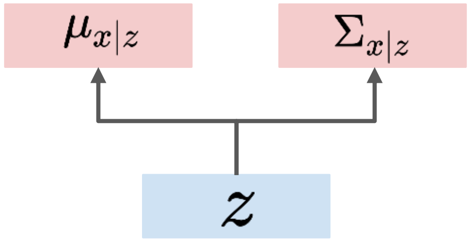
Encoder Network

$$q_\phi(z|x) = N(\mu_{z|x}, \Sigma_{z|x})$$



Decoder Network

$$p_\theta(x|z) = N(\mu_{x|z}, \Sigma_{x|z})$$

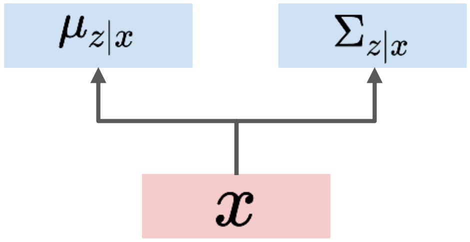


Variational Autoencoders (VAE)

Jointly train **encoder** q and **decoder** p

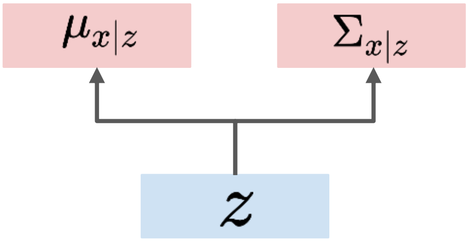
Encoder Network

$$q_\phi(z|x) = N(\mu_{z|x}, \Sigma_{z|x})$$



Decoder Network

$$p_\theta(x|z) = N(\mu_{x|z}, \Sigma_{x|z})$$



Variational Autoencoders (VAE)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)}$$

Variational Autoencoders (VAE)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

Variational Autoencoders (VAE)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}$$

Variational Autoencoders (VAE)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z[\log \frac{q_\phi(z|x)}{p(z)}] + E_z[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}]$$

Variational Autoencoders (VAE)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z[\log \frac{q_\phi(z|x)}{p(z)}] + E_z[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}]$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - KL(q_\phi(z|x), p(z)) + KL(q_\phi(z|x), p_\theta(z|x))$$

Kullback-Leibler Divergence: $KL(p, q) = E_{x\sim p}[log\frac{p(x)}{q(x)}]$

Variational Autoencoders (VAE)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z[\log \frac{q_\phi(z|x)}{p(z)}] + E_z[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}]$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - KL(q_\phi(z|x), p(z)) + KL(q_\phi(z|x), p_\theta(z|x))$$

$KL \geq 0$ => dropping the last term gives a lower bound on the data likelihood

Kullback-Leibler Divergence: $KL(p,q) = E_{x\sim p}[log\frac{p(x)}{q(x)}]$

Variational Autoencoders (VAE)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z[\log \frac{q_\phi(z|x)}{p(z)}] + E_z[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}]$$

$$= E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - KL(q_\phi(z|x), p(z)) + KL(q_\phi(z|x), p_\theta(z|x))$$

$$\log p_\theta(x) \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - KL(q_\phi(z|x), p(z))$$

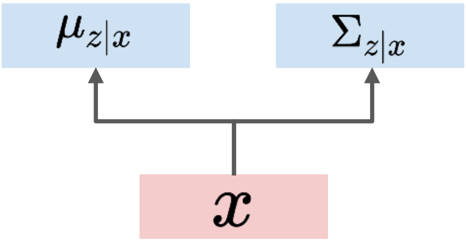
Variational Autoencoders (VAE)

Jointly train **encoder** q and **decoder** p to maximize the **variational lower bound** on the data likelihood

$$\log p_\theta(x) \geq E_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)] - KL(q_\phi(z|x), p(z))$$

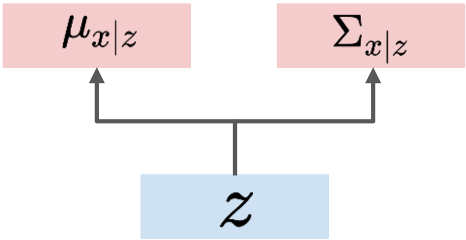
Encoder Network

$$q_\phi(z|x) = N(\mu_{z|x}, \Sigma_{z|x})$$



Decoder Network

$$p_\theta(x|z) = N(\mu_{x|z}, \Sigma_{x|z})$$



Variational Autoencoders (VAE)

Closed form solution when q_ϕ is diagonal Gaussian and p is unit Gaussian:

$$\log p_\theta(x) \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - \boxed{KL(q_\phi(z|x), p(z))}$$

$$-KL(q_\phi(z|x), p(z)) = \int_Z q_\phi(z|x) \log \frac{p(z)}{q_\phi(z|x)} dz = \int_Z N(z, \mu_{z|x}, \Sigma_{z|x}) \log \frac{N(z,0,I)}{N(z,\mu_{z|x},\Sigma_{z|x})} dz$$

$$= \sum_{j=1}^{J} (1 + \log((\sum_{z|x})_j^2) - (\mu_{z|x})_j^2 - (\Sigma_{z|x})_j^2)$$

Variational Autoencoders (VAE)

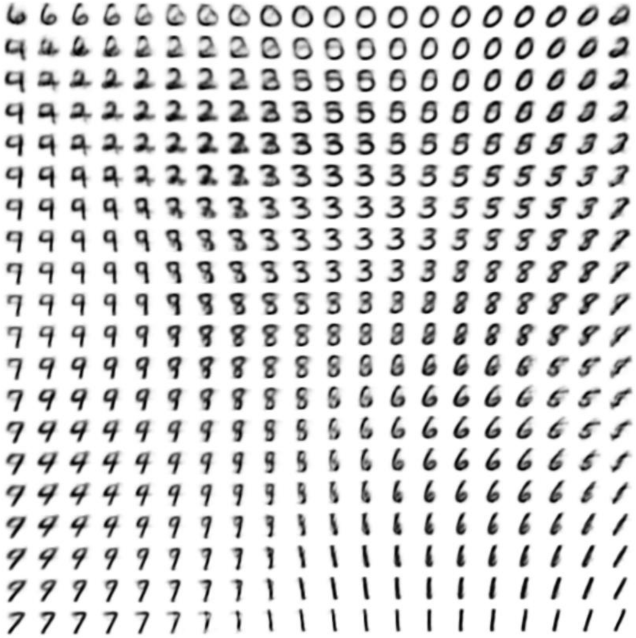
Closed form solution when q_ϕ is diagonal Gaussian and p is unit Gaussian:

$$\log p_\theta(x) \geq \boxed{E_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)]} - KL(q_\phi(z|x), p(z))$$

$E_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)]$ is data reconstruction term

Variational Autoencoders (VAE)

Learned data manifold for generative models with two-dimensional latent space:



Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes, ICLR 2014 https://arxiv.org/abs/1312.6114

Variational Autoencoders. Summary

Pros:

- Allows inference of $q(z|x)$, can be useful feature representation for other tasks

Cons:

- Generated images are blurrier than lower quality compared to state-of-the-art (GAN)

Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes, ICLR 2014 https://arxiv.org/abs/1312.6114

Generative Models Summary

- **Variational Autoencoders (VAEs)** introduces a latent z and maximize a lower bound:

  $$\log p_\theta(x) \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - KL(q_\phi(z|x), p(z))$$

  Latent z allows for interpolation and editing applications

- **Generative Adversarial Networks (GANs)** do not model p(x) but allow us to draw samples from p(x). Difficult to evaluate but best qualitative results today

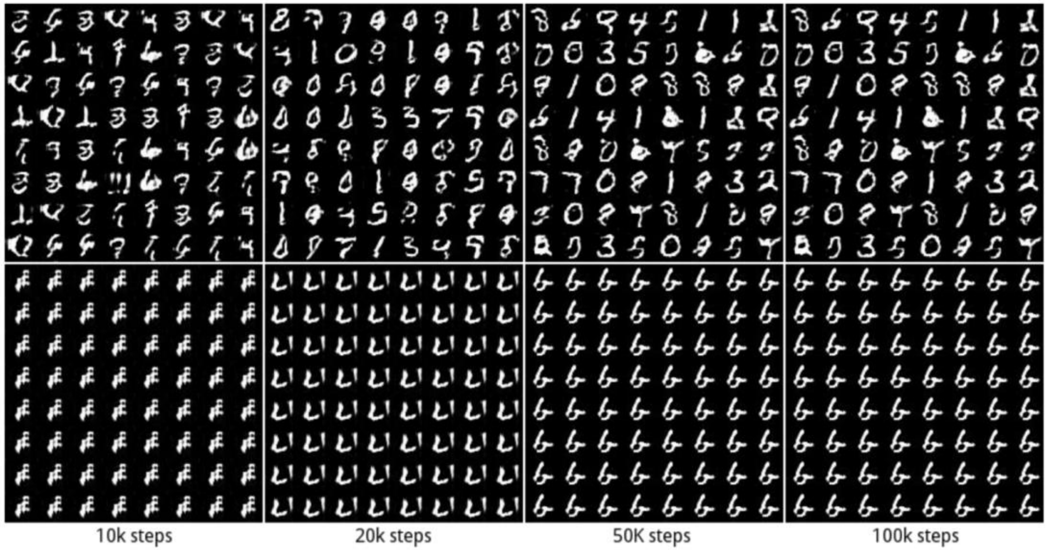
Problems of GANs

- Mode collapse:
  - G collapses providing limited sample variety
- Non-convergence:
  - model parameters oscillate, destabilize and never converge
- Diminished gradient:
  - D is so successful that the G gradient vanishes and learns nothing

Mode collapse

Real-life data is multimodal (10 in MNIST)

Mode collapse: when few modes generated



Partial mode collapse

The generator produces realistic and diverse samples, but much less diverse than the real-world data distribution.

Solutions to mode collapse

- Wasserstein loss [1]
  - Trains the discriminator to optimality without worrying about vanishing gradients.
  - If the discriminator doesn't get stuck in local minima, it learns to reject the outputs that the generator stabilizes on.
- Unrolling [2]
  - Uses a generator loss function that incorporates not only the current discriminator's classifications, but also the outputs of future discriminator versions
  - The generator can't over-optimize for a single discriminator.

[1] Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

[2] Luke Metz, Ben Poole, David Pfau, Jascha Sohl-Dickstein. Unrolled Generative Adversarial Networks. https://arxiv.org/abs/1611.02163

Wasserstein GAN. Criticizing is easy

- GAN can optimize the discriminator easier than the generator.

Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

Wasserstein GAN. Criticizing is easy

- GAN can optimize the discriminator easier than the generator.
- An optimal discriminator produces good information for the generator to improve. But if the generator is not doing a good job yet, the gradient for the generator diminishes and the generator learns nothing

Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

Wasserstein GAN. Criticizing is easy

- GAN can optimize the discriminator easier than the generator.
- An optimal discriminator produces good information for the generator to improve. But if the generator is not doing a good job yet, the gradient for the generator diminishes and the generator learns nothing

Original GAN generator's gradient: $-\nabla_{\theta_g} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \rightarrow \boldsymbol{0}$

Alternative: $\nabla_{\theta_g} \log D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)$

Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

Wasserstein GAN. Criticizing is easy

- GAN can optimize the discriminator easier than the generator.
- An optimal discriminator produces good information for the generator to improve. But if the generator is not doing a good job yet, the gradient for the generator diminishes and the generator learns nothing

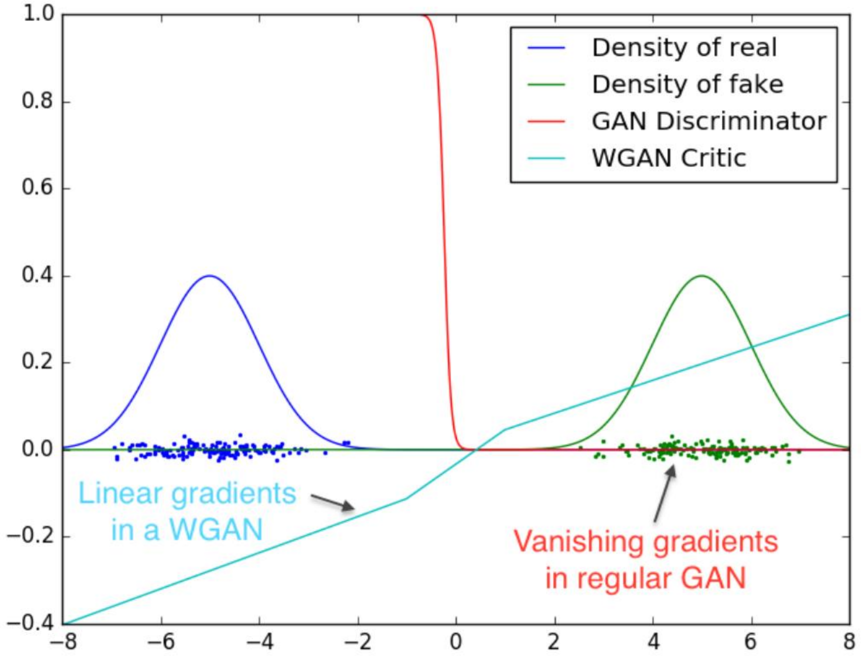
Original GAN generator's gradient: $-\nabla_{\theta_g} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \rightarrow \boldsymbol{0}$

Alternative: $\nabla_{\theta_g} \log D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)$

Problem: large variance of gradients
that make the model unstable

Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

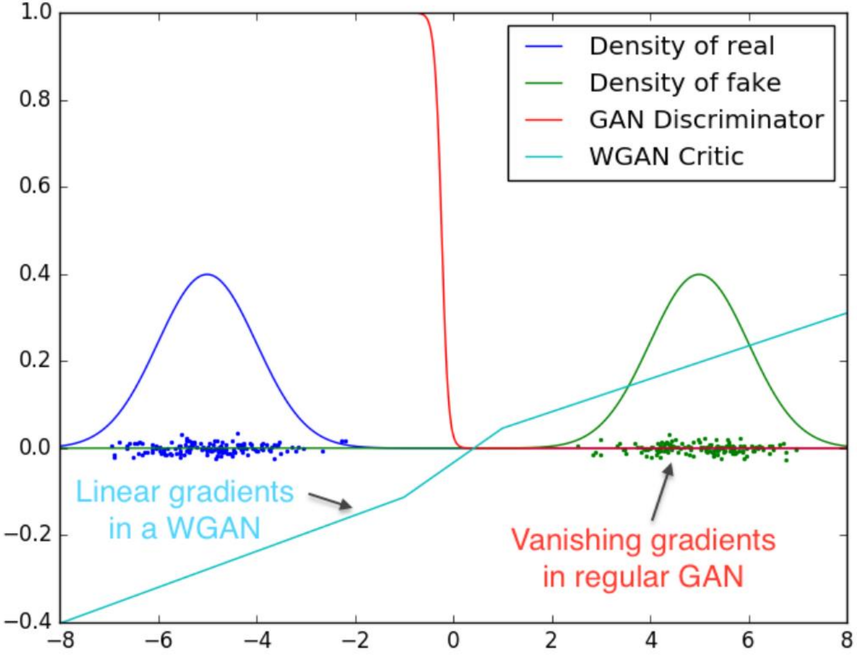
Wasserstein GAN



Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

Wasserstein GAN

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$



Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

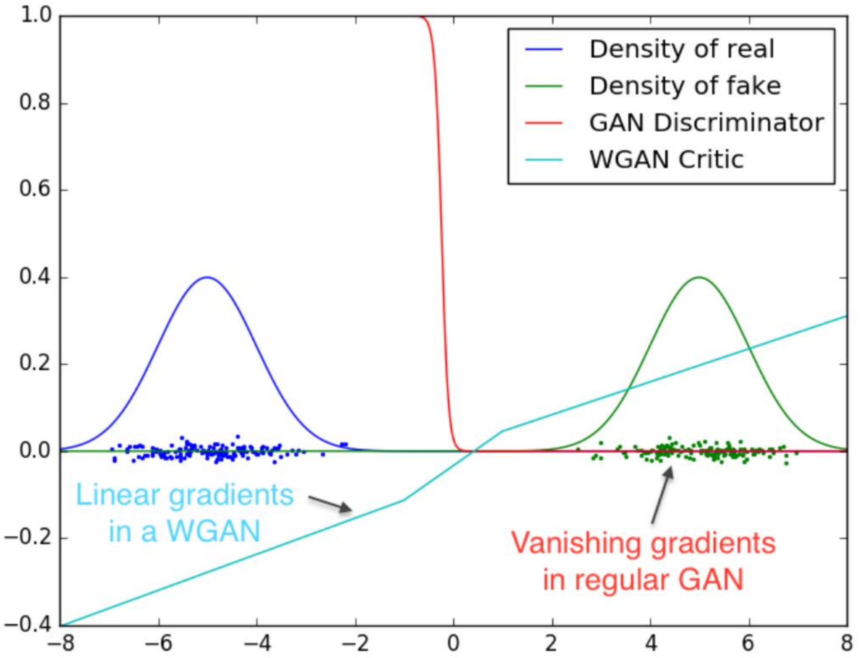
Wasserstein GAN

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

where

- *sup* is the least upper bound
- ***f*** is a 1-Lipschitz function following constraint:

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2|.$$



Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

Wasserstein GAN

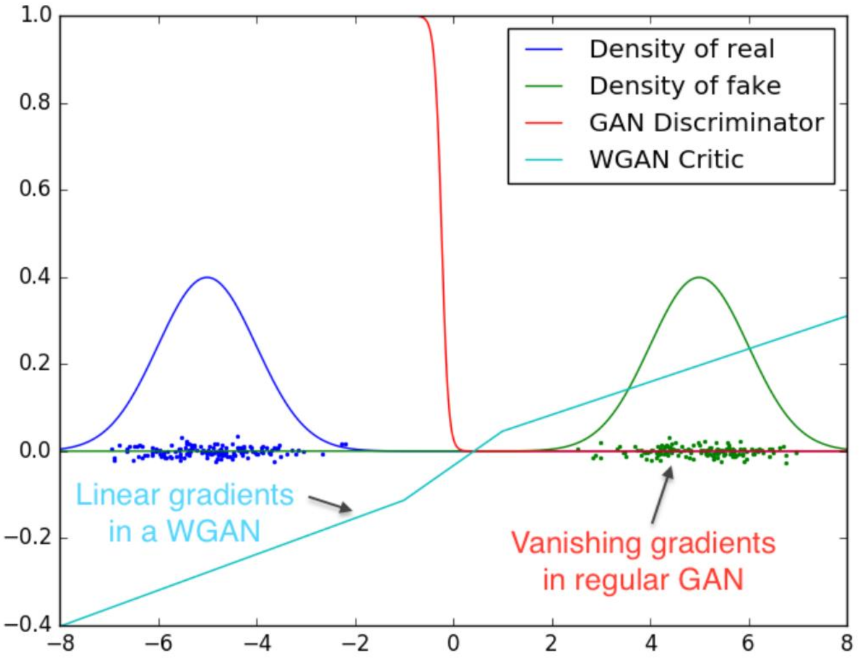
$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

where

- *sup* is the least upper bound
- ***f*** is a 1-Lipschitz function following constraint:

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2|.$$

We can build a deep network to calculate the Wasserstein distance.



Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

Wasserstein GAN

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_\theta}[f(x)]$$

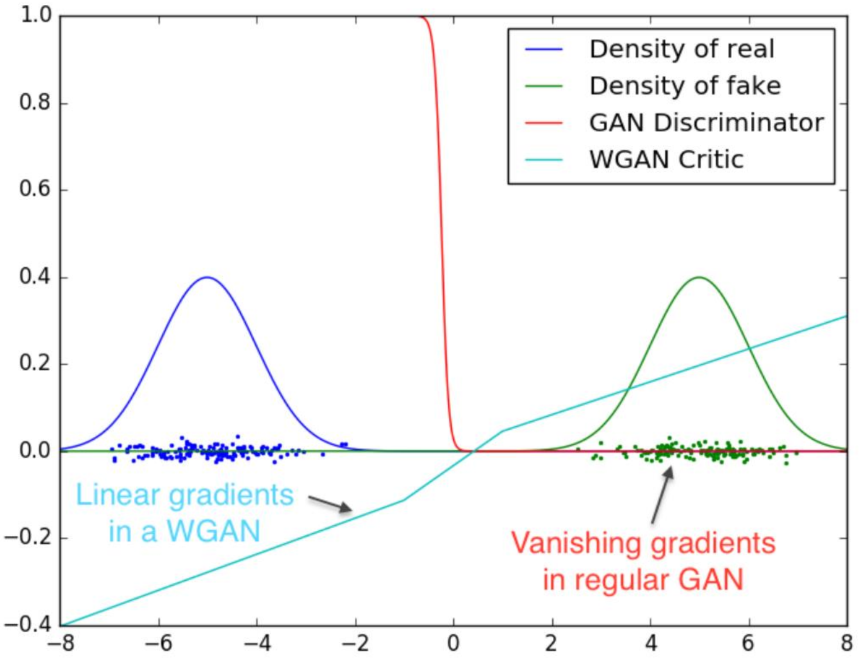
where

- *sup* is the least upper bound
- ***f*** is a 1-Lipschitz function following constraint:

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2|.$$

We can build a deep network to calculate the Wasserstein distance.

This network is very similar to the discriminator, just without the sigmoid function and outputs a scalar score rather than a probability.



Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

Wasserstein GAN

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \le 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

while θ has not converged **do**

    for $t = 0, ..., n_{\text{critic}}$ **do**

        Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.

        Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.

        $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$

        $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$

        $w \leftarrow \text{clip}(w, -c, c)$

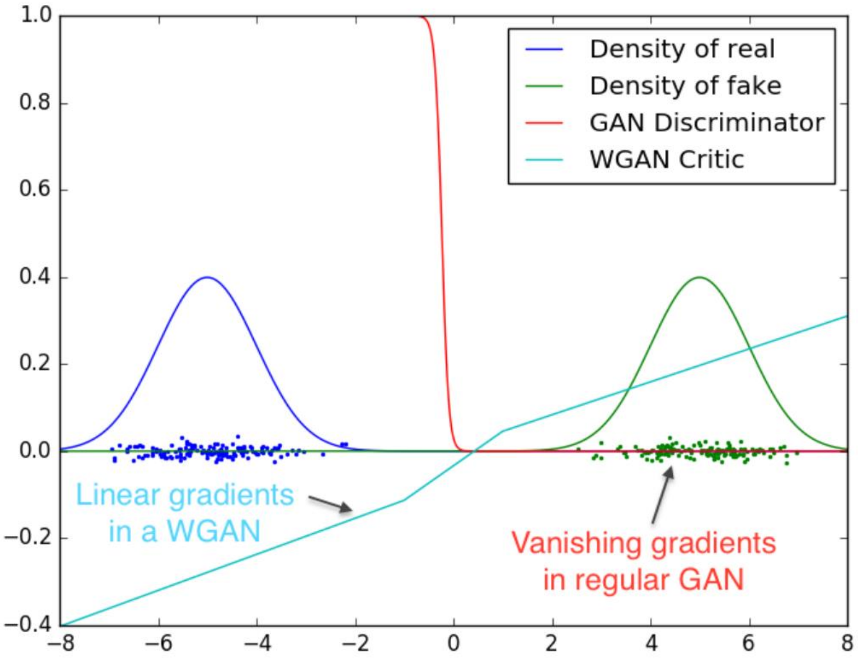
    end for

    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.

    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$

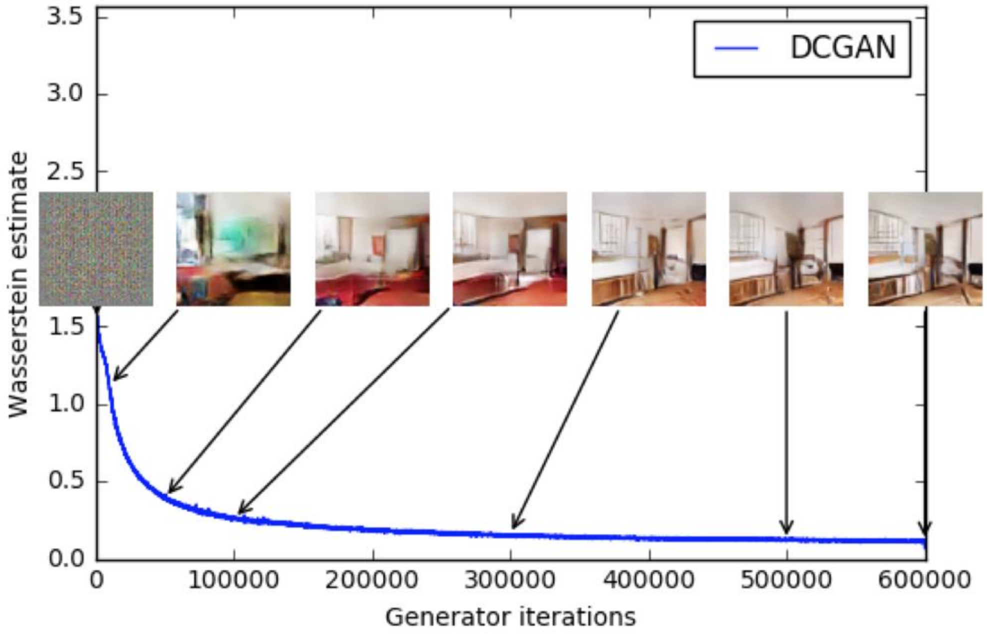
    $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$

end while



Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

Wasserstein GAN



Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

Wasserstein GAN

- Wasserstein criterion allows us to train **D** until optimality. When the criterion reaches the optimal value, it simply provides a loss to the generator that we can train as any other neural network.

Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

Wasserstein GAN

- Wasserstein criterion allows us to train **D** until optimality. When the criterion reaches the optimal value, it simply provides a loss to the generator that we can train as any other neural network.
- We no longer need to balance **G** and **D** capacity properly.

Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

Wasserstein GAN

- Wasserstein criterion allows us to train **D** until optimality. When the criterion reaches the optimal value, it simply provides a loss to the generator that we can train as any other neural network.
- We no longer need to balance **G** and **D** capacity properly.
- Wasserstein loss leads to a higher quality of the gradients to train **G**.

Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

Wasserstein GAN

- Wasserstein criterion allows us to train **D** until optimality. When the criterion reaches the optimal value, it simply provides a loss to the generator that we can train as any other neural network.
- We no longer need to balance **G** and **D** capacity properly.
- Wasserstein loss leads to a higher quality of the gradients to train **G**.
- WGANs are **more robust** than common GANs to the architectural choices for the generator and hyperparameter tuning

Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

GANs evaluation

The objective function for the generator and the discriminator usually measures how well they are doing relative to the opponent.

It is not a good metric in measuring the image quality or its diversity.

GANs evaluation

- Inception Score (IS) [1]
- Frechet Inception Distance (FID) [2]
- Human-based ratings and preference judgments

[1] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen. Improved Techniques for Training GANs. NeurIPS 2016 https://arxiv.org/abs/1606.03498

[2] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. NeurIPS 2017 https://arxiv.org/abs/1706.08500

Inception Score (IS)

IS uses two criteria in measuring the performance of GAN:

- The **quality** of the generated images
- their **diversity**

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen. Improved Techniques for Training GANs. NeurIPS 2016 https://arxiv.org/abs/1606.03498

Inception Score (IS)

- **Quality**: use an Inception network to predict conditional probability p(y|x) — where y is the label and x is the generated data
- **Diversity**: calculate marginal probability:

$$p(y) = \int_z p(y|x = G(z))dz$$

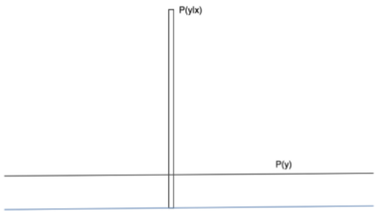
Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen. Improved Techniques for Training GANs. NeurIPS 2016 https://arxiv.org/abs/1606.03498

Inception Score (IS)

- **Quality**: use an Inception network to predict conditional probability p(y|x) — where y is the label and x is the generated data
- **Diversity**: calculate marginal probability: $p(y) = \int_z p(y|x = G(z))dz$

We want

- the conditional probability **p(y|x)** to be highly predictable (**low entropy**) i.e. given an image, we should know the object type easily
- the data distribution **p(y)** should be uniform (**high entropy**)



Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen. Improved Techniques for Training GANs. NeurIPS 2016 https://arxiv.org/abs/1606.03498

Inception Score (IS)

Compute their KL-divergence to combine these two criteria:

$$IS(G) = \exp(E_{x \sim p_g} KL(p(y|x) || p(y)))$$

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen. Improved Techniques for Training GANs. NeurIPS 2016 https://arxiv.org/abs/1606.03498

Inception Score (IS)

Limitations:

- IS is limited by what the Inception classifier can detect, which is linked to the training data (ILSVRC)
- IS can misrepresent the performance if it only generates one image per class. p(y) will still be uniform even though the diversity is low

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen. Improved Techniques for Training GANs. NeurIPS 2016 https://arxiv.org/abs/1606.03498

Frechet Inception Distance (FID)

- Use the **Inception network** to extract features from an intermediate layer

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. NeurIPS 2017 https://arxiv.org/abs/1706.08500

Frechet Inception Distance (FID)

- Use the Inception network to extract features from an intermediate layer
- Model data distribution for these features using a multivariate Gaussian distribution with mean μ and covariance Σ

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. NeurIPS 2017 https://arxiv.org/abs/1706.08500

Frechet Inception Distance (FID)

- Use the Inception network to extract features from an intermediate layer
- Model data distribution for these features using a multivariate Gaussian distribution with mean μ and covariance Σ
- The FID between the real images x and generated images g:

$$FID(x,g) = ||\mu_x - \mu_g||_2^2 + Tr(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{\frac{1}{2}})$$

where *Tr* sums up all the diagonal elements

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. NeurIPS 2017 https://arxiv.org/abs/1706.08500

Frechet Inception Distance (FID)

- **Lower** FID values mean **better** image quality and diversity
- FID is sensitive to mode collapse, the distance increases when modes are missed
- FID is more robust to noise than IS. If the model only generates one image per class, the distance will be high

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. NeurIPS 2017 https://arxiv.org/abs/1706.08500

Q&A