Generative Adversarial Networks (GANs)

Past, Present, and Future

Zhifei Zhang

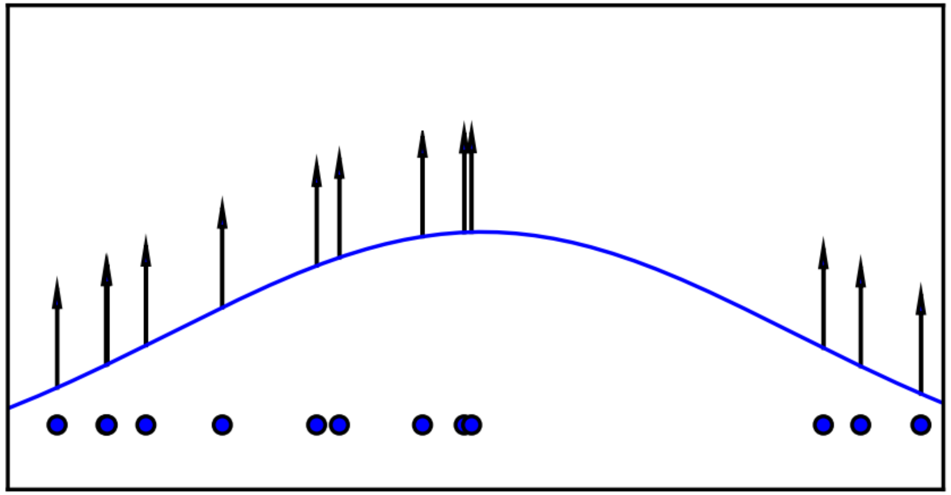
Outline

1. Basic idea of GANs
2. Time line of development
3. Drawbacks & solutions
4. Future directions

Basic idea of GANs

In probability and statistics, a **generative model** is a model for **randomly** generating observable data values, typically given some **hidden parameters**.

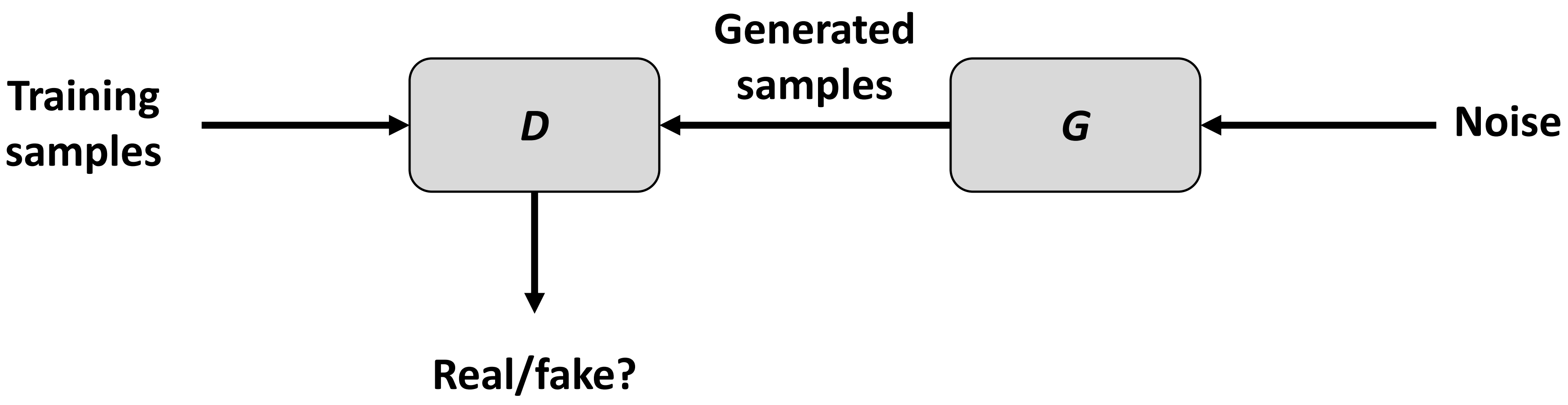
Maximum Likelihood



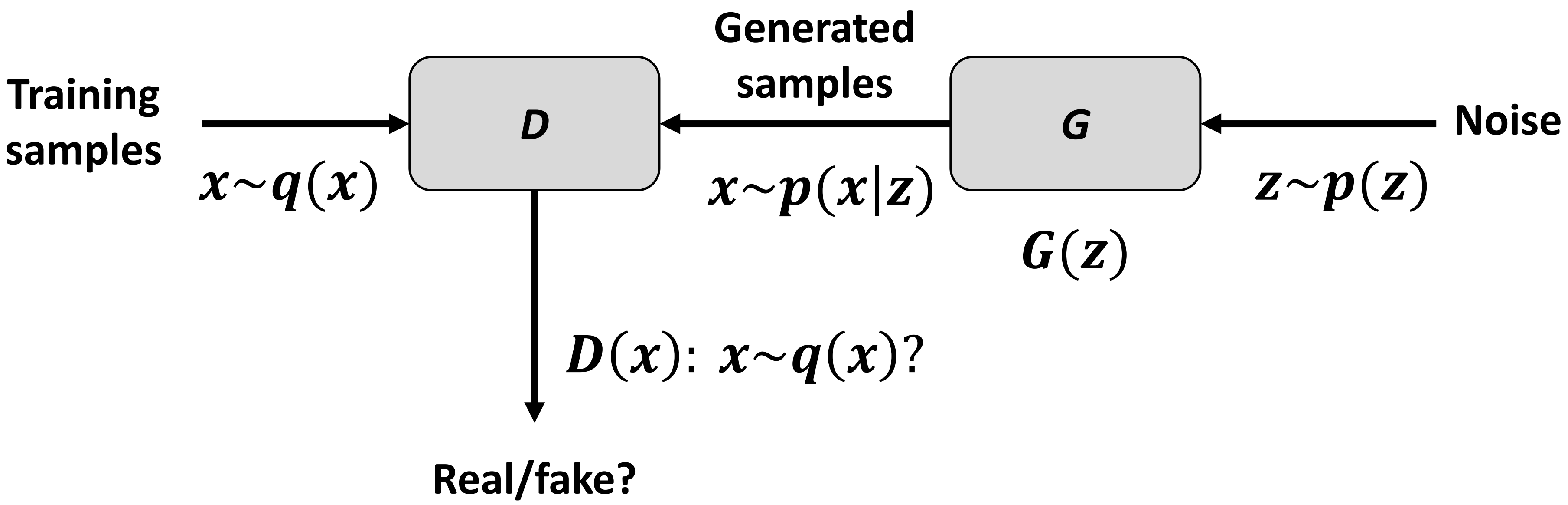
$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{x \sim p_{\text{data}}} \log p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{\theta})$$

Basic idea of GANs

- Two neural networks against each other:
  - A generator network *G*
    - Mimic training samples to fool the discriminator
  - A discriminator network *D*
    - Discriminate training samples and generated samples



Basic idea of GANs



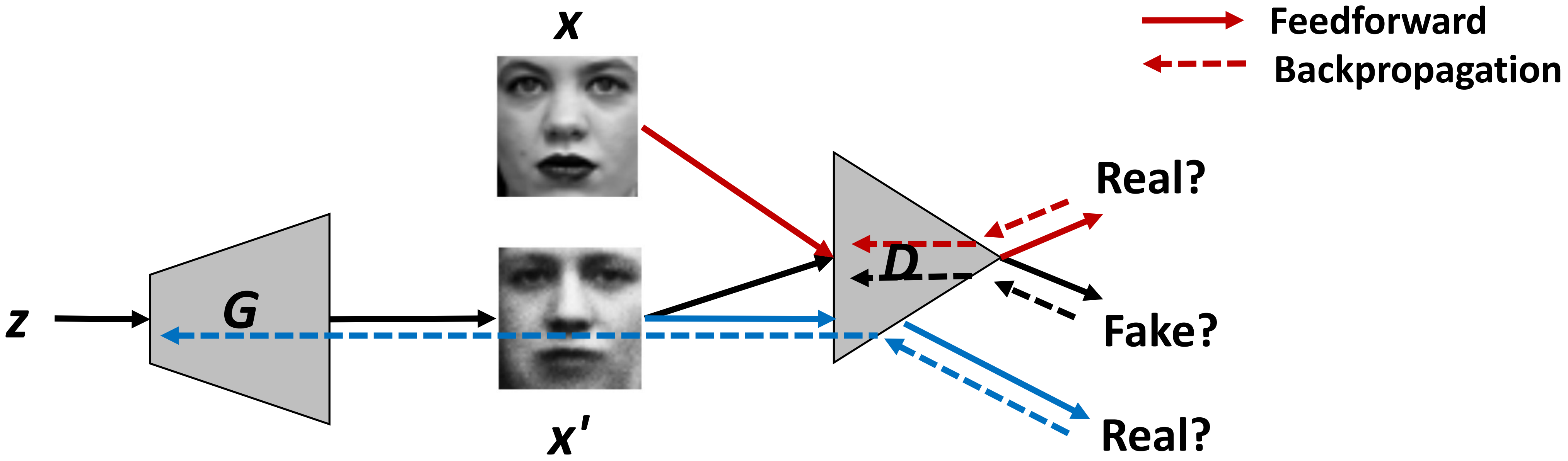
For *D*: $\max_{D} \; \mathbb{E}_{x \sim q(x)}[\log(D(x))] + \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z)))]$

For *G*: $\min_{G} \; \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z)))]$

Basic idea of GANs

The objective function of GANs:

$$\min_{G} \max_{D} \ \mathbb{E}_{x\sim q(x)}[\log(D(x))] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))]$$



Time line

2014

GAN [NIPS]

Born

2015

Laplacian Pyramid [NIPS]

Fermenting

2016

DCGAN [ICLR]

InfoGAN [NIPS]

RNN+GAN [ICML]

VAE+GAN [ICML]

CoGAN [NIPS]

AAE [ICLR]

CatGAN [ICLR]

Super-Resolution [ECCV] [CVPR]

Latent-Manipulation [ECCV] [CVPR]

Domain transformation [ICML] [CVPR]

Booming of Improvements and Applications:

- Higher resolution
- Flexible manipulation

2017

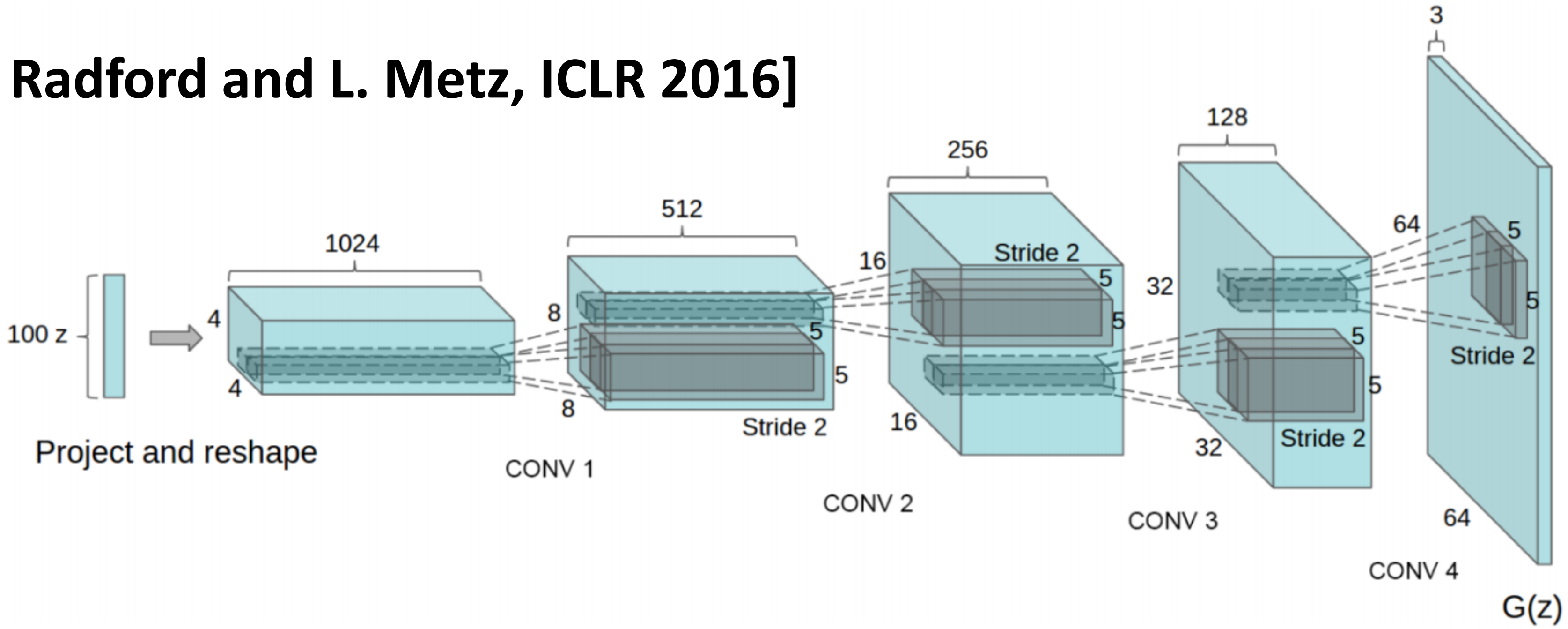
Instability [ICLR]

Mode Missing [ICLR]

Theory: Drawbacks & Solutions

Time line

DCGAN [A. Radford and L. Metz, ICLR 2016]

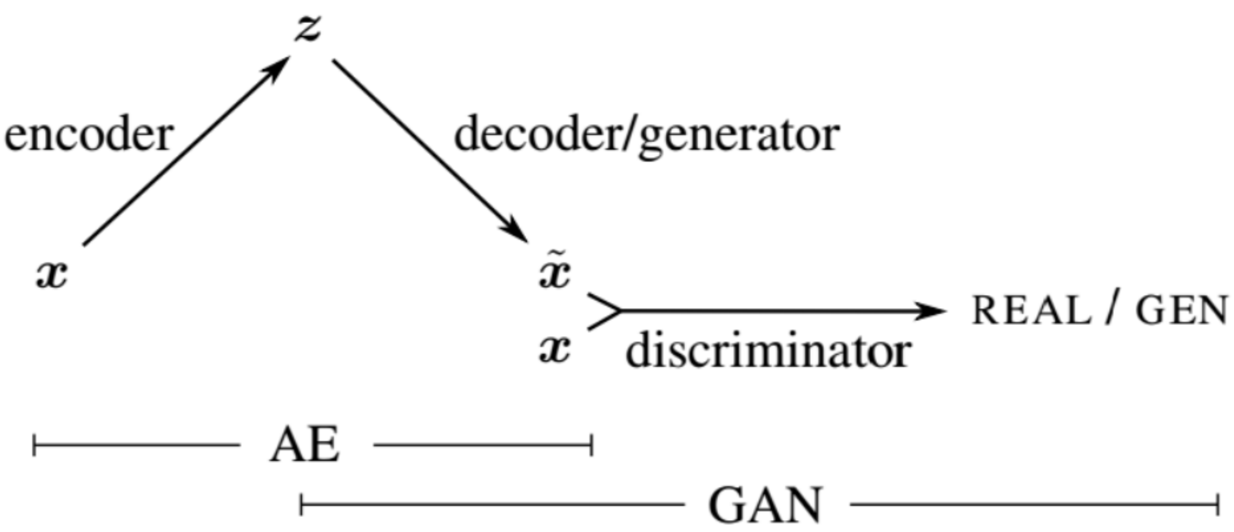


Architecture guidelines for stable Deep Convolutional GANs

- 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.

Time line

VAE+GAN [A. B. L. Larsen et al., ICML 2016]



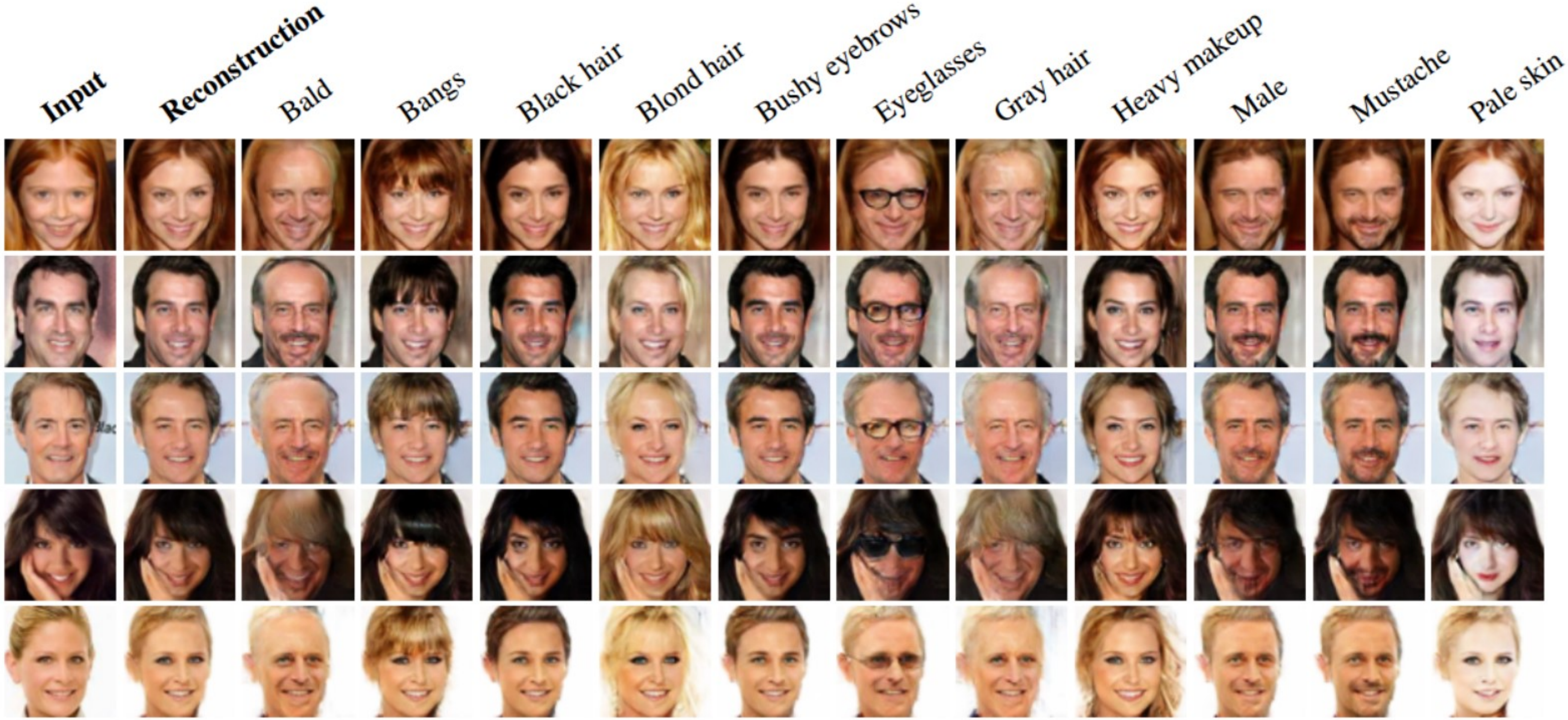
- **GAN encourages higher resolution compared to VAE**
- **Controllable generation and manipulation on *z***

Time line

VAE+GAN [A. B. L. Larsen et al., ICML 2016]

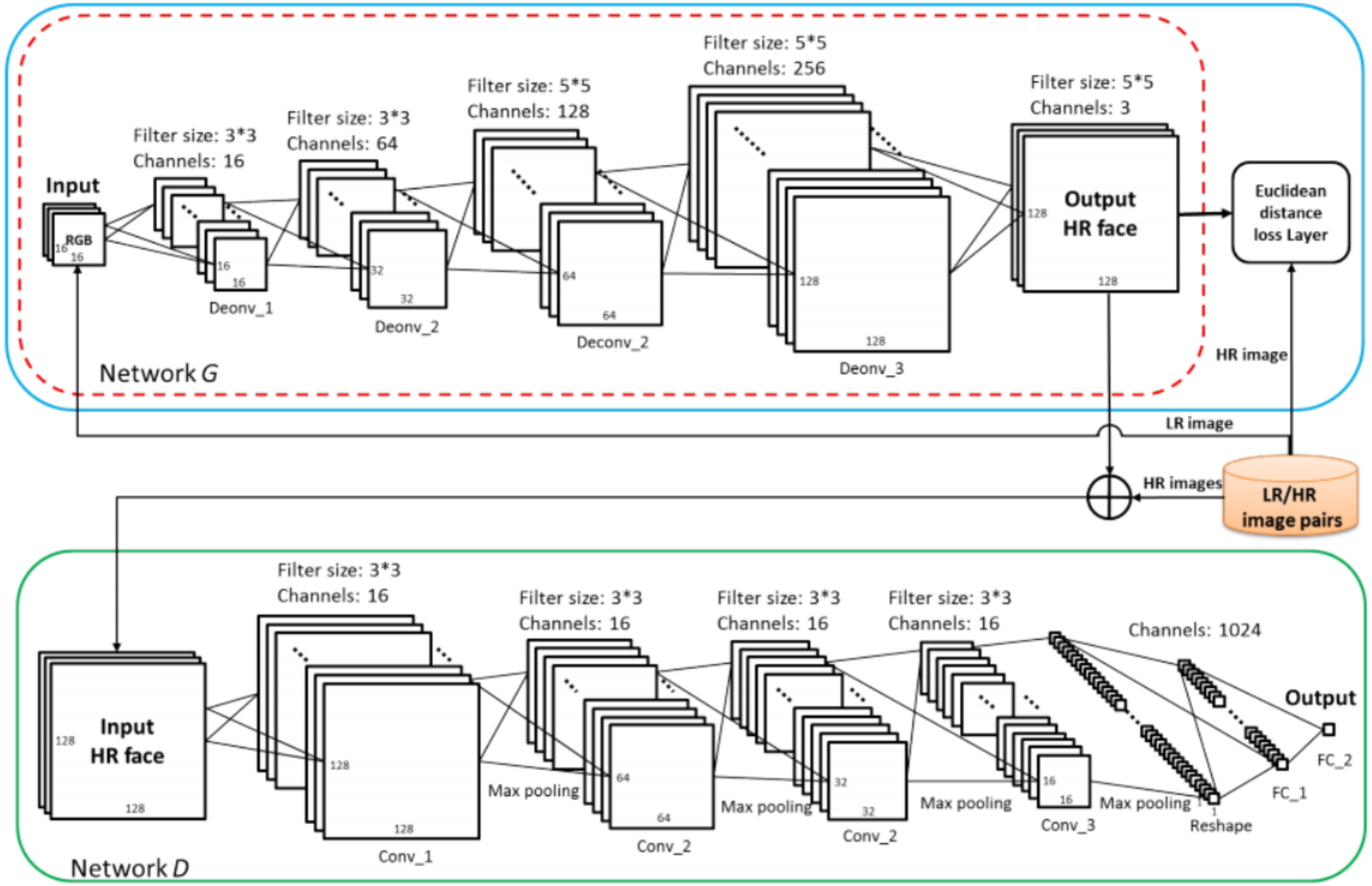
Time line

VAE+GAN [A. B. L. Larsen et al., ICML 2016]



Time line

Super-Resolution [X. Yu and F. Porikli, ECCV 2016]



- 8× scaling factor

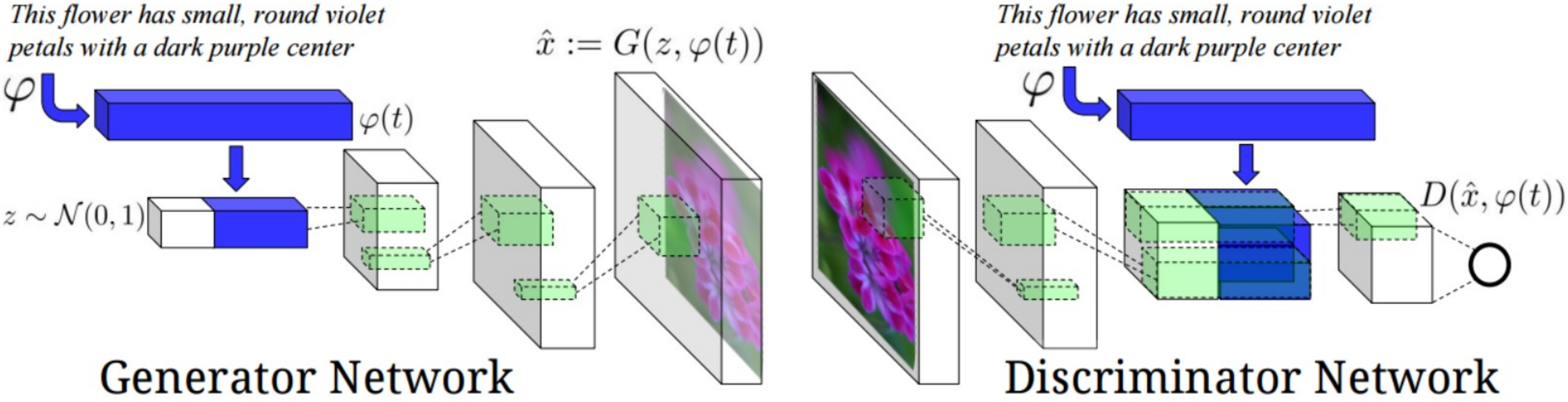
Time line

Super-Resolution [X. Yu and F. Porikli, ECCV 2016]

(a) LR (b) HR (c) bicubic (d) [5] (e) [7] (f) [10] (g) [16] (h) [8] (i) Ours

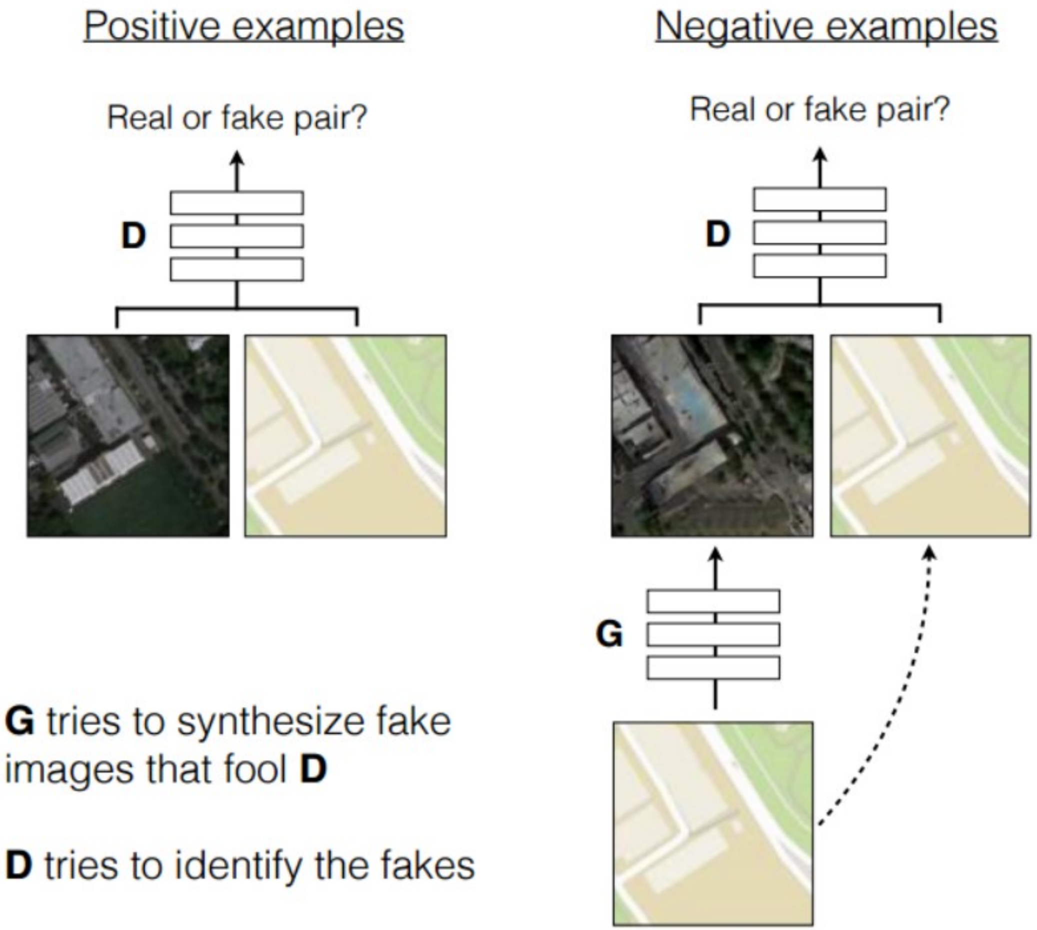
Time line

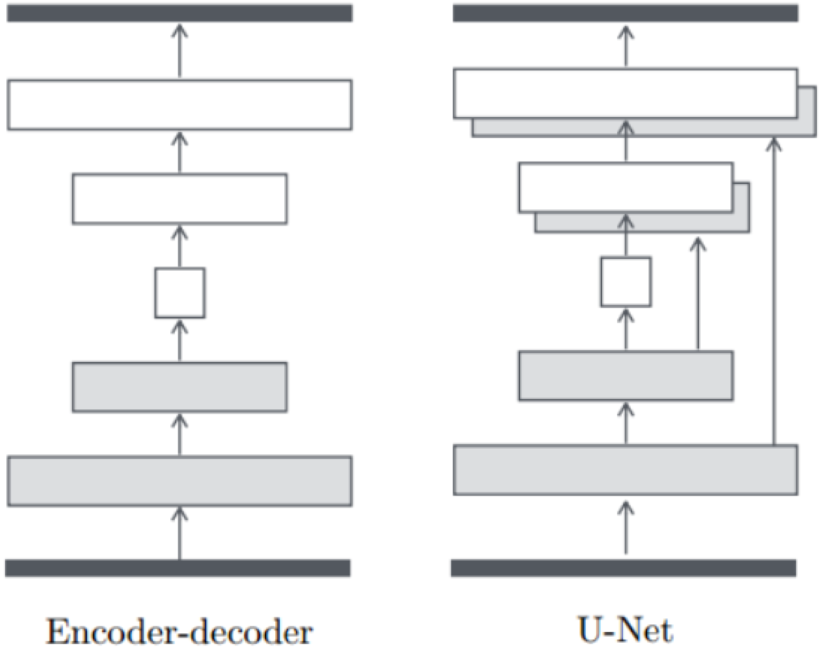
Domain Transformation1 [S. Reed et al., ICML 2016]



Time line

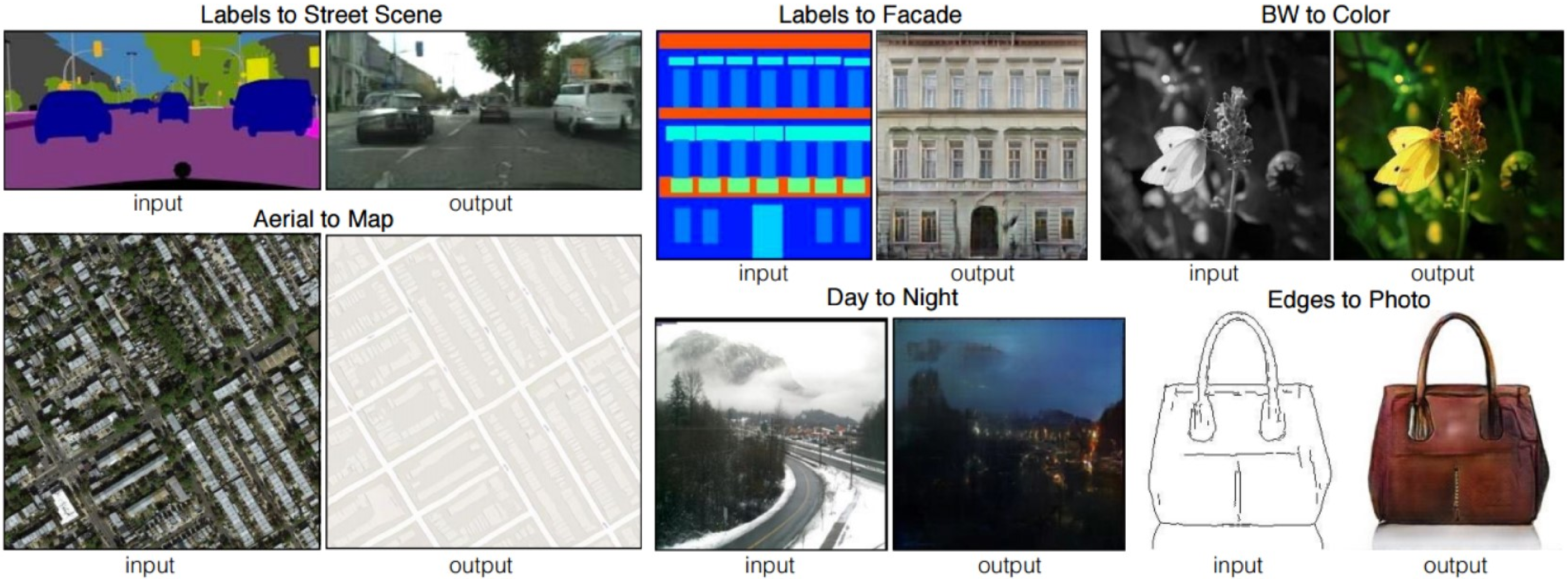
Domain Transformation2 [P. Isola et al., CVPR 2017]



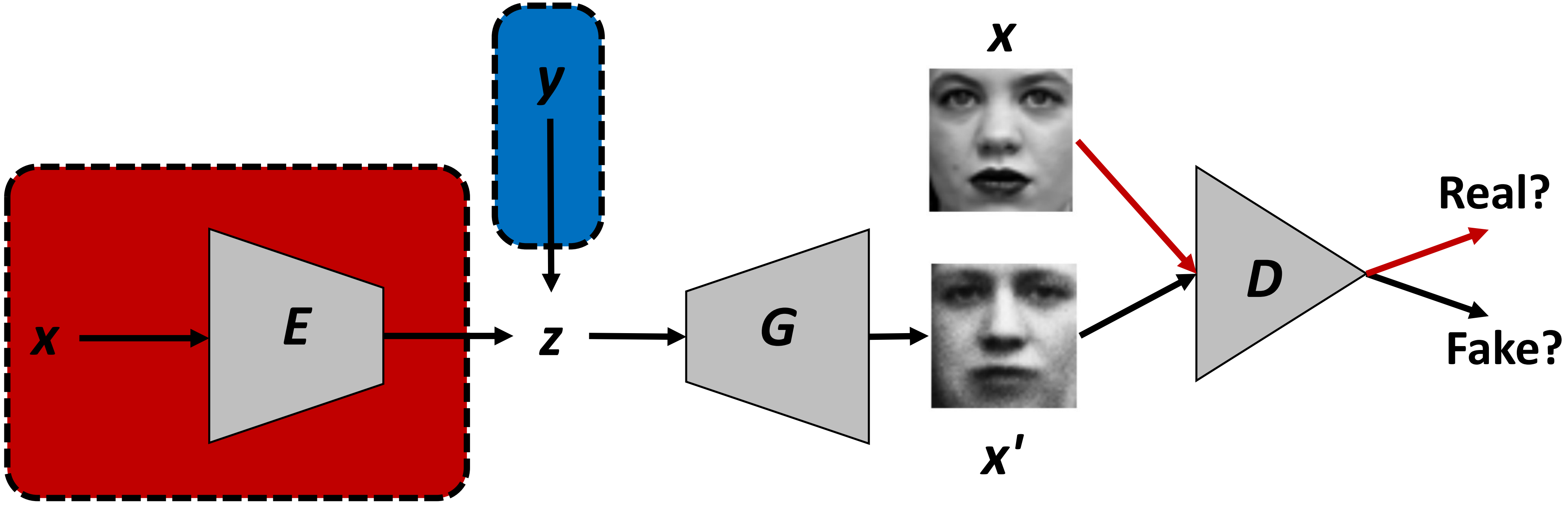


Time line

Domain Transformation2 [P. Isola et al., CVPR 2017]



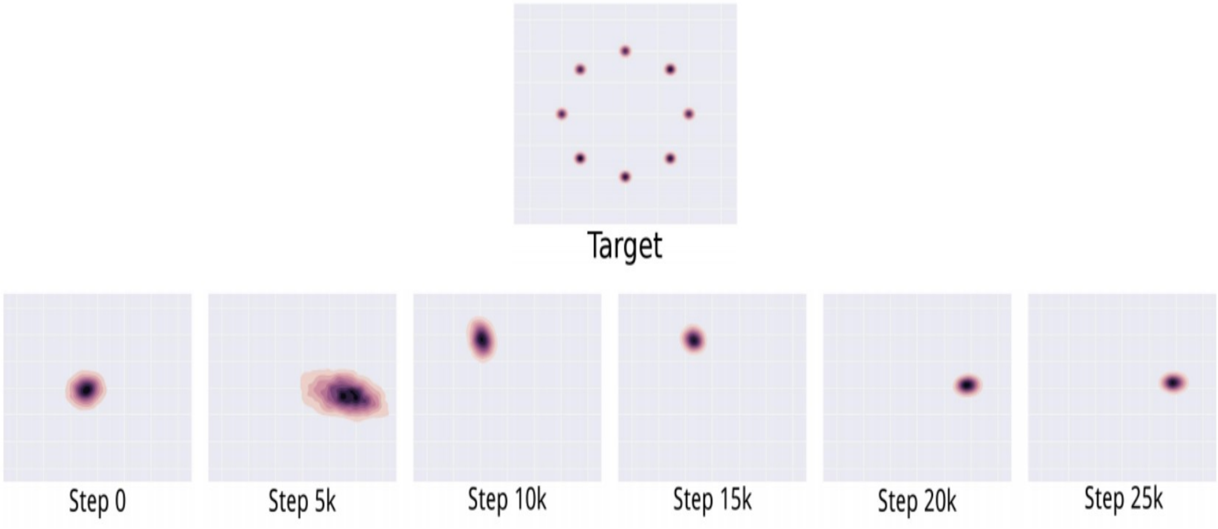
Summary

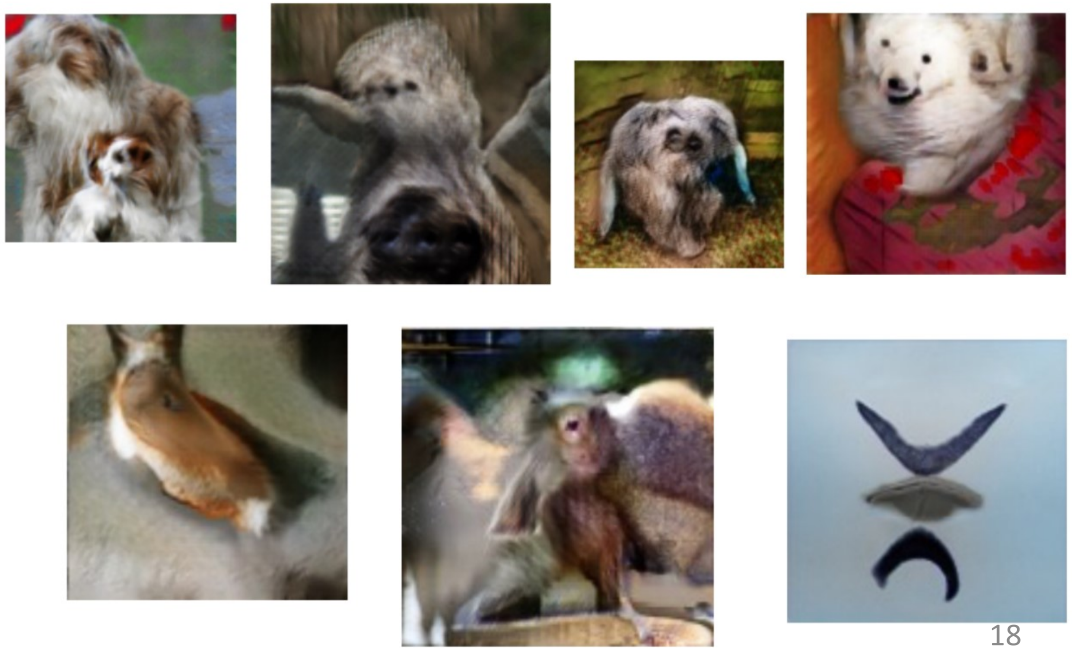


- Seldom use original GAN
- Concatenate an encoder to *G*
- Concatenate extra feature to *z*

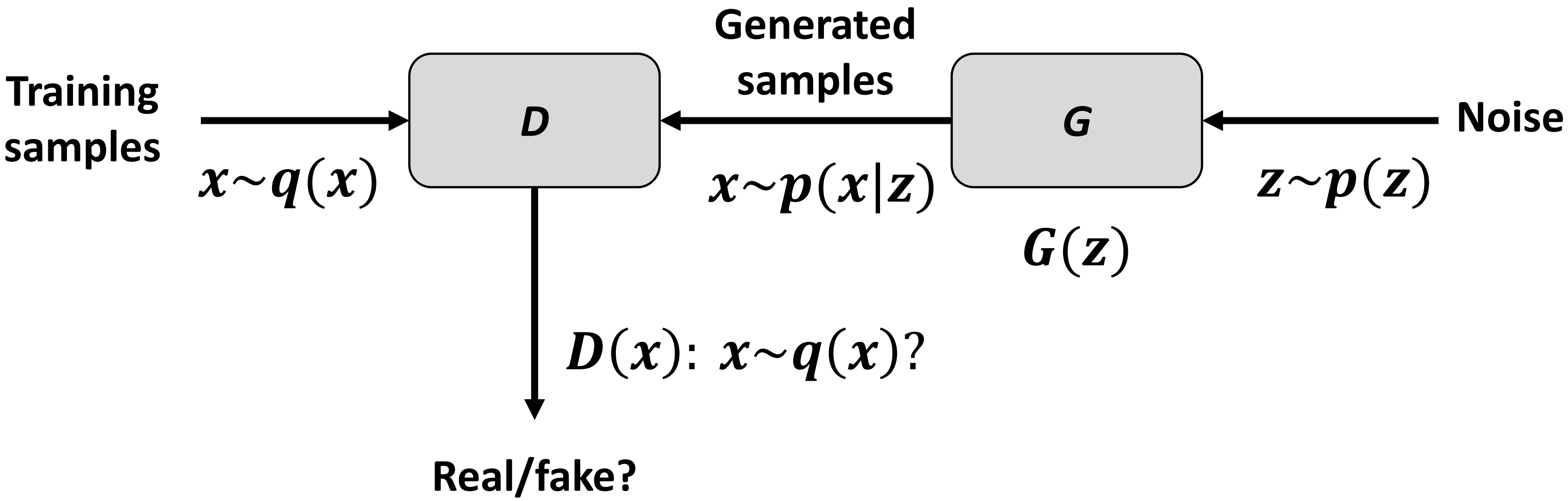
Drawbacks

- Hard to train, e.g., mode missing.
- Generate something but not real actually.
- Hard to learn to generate discrete data, e.g., text.





Drawbacks (mode missing)



$$\min_G \max_D \; \mathbb{E}_{x\sim q(x)}[\log(D(x))] + \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))]$$

$$\min_G \max_D \; \int_x q(x)\log(D(x))\,dx + \int_z p(z)\log(1 - D(G(z)))\,dz$$

Drawbacks (mode missing)

Fix *G*,

$$\min_G \max_D \int_x q(x)\log(D(x))\,dx + \int_z p(z)\log(1 - D(G(z)))\,dz$$

$$= \max_D \int_x q(x)\log\big(D(x)\big) + p_g(x)\log\big(1 - D(x)\big)\,dx$$

$$\boldsymbol{D^*(x) = \frac{q(x)}{q(x) + p_g(x)}}$$

Fix *D,**

$$= \min_G \int_x q(x)\log\left(\frac{q(x)}{q(x)+p_g(x)}\right) + p_g(x)\log\left(1 - \frac{q(x)}{q(x)+p_g(x)}\right)dx$$

$$= \min_G \int_x q(x)\log\left(\frac{q(x)}{q(x)+p_g(x)}\right) + p_g(x)\log\left(\frac{p_g(x)}{q(x)+p_g(x)}\right)dx$$

$$= \min_G D_{KL}(q||q+p_g) + D_{KL}(p_g||q+p_g)$$

$$= \min_G D_{KL}(q||\frac{q+p_g}{2}) + D_{KL}(p_g||\frac{q+p_g}{2}) - 2\log 2$$

$$= \min_G 2D_{JS}(q||p_g) - 2\log 2$$

$q(x)<p_g(x)$: Unrealistic sample

$q(x)>p_g(x)$: Mode missing

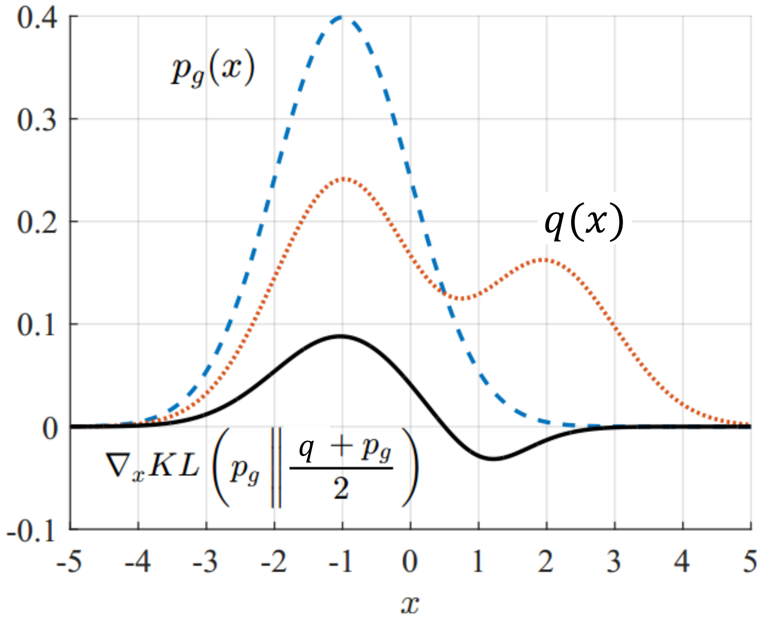
Jensen-Shannon divergence is a symmetric metric.

Drawbacks (mode missing)

However, we cannot ensure D^* in practice.

$$\min_G \quad \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))]$$

$$= \min_G \quad D_{KL}(p_g || \frac{q + p_g}{2}) - \log 2$$



- **Punish more on generating unrealistic samples**
- **Punish less on mode missing**

If relax the assumption of D^*,

$$\min_G \quad \mathbb{E}_{z\sim p(z)}[\log(1 - D(G(z)))]$$

only punish on generating unrealistic samples.

Drawbacks (unrealistic generation)

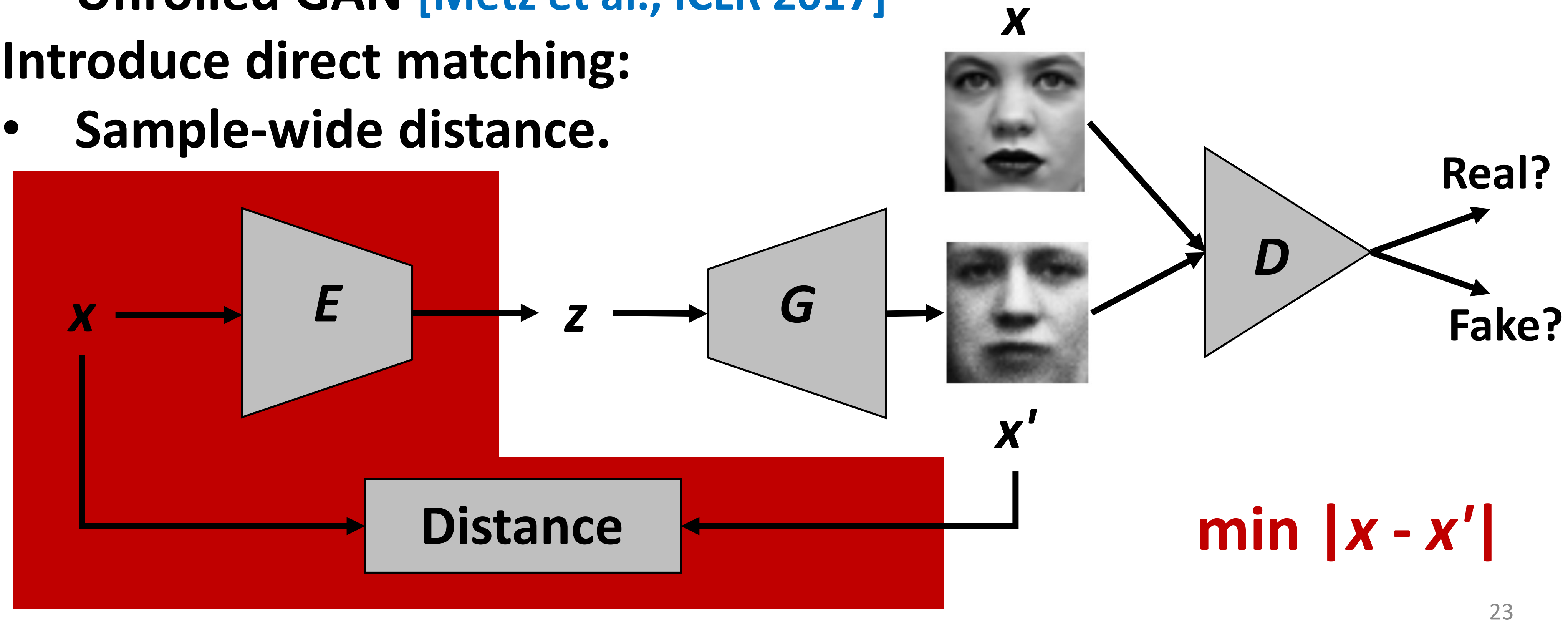
In GANs, the generated distribution is matched to the distribution specified by *D*, rather than to the real distribution.

Ideally, $p_g(x) = q(x)$, direct matching.

In practice, $D(x|p_g) = D(x|q)$, indirect matching.

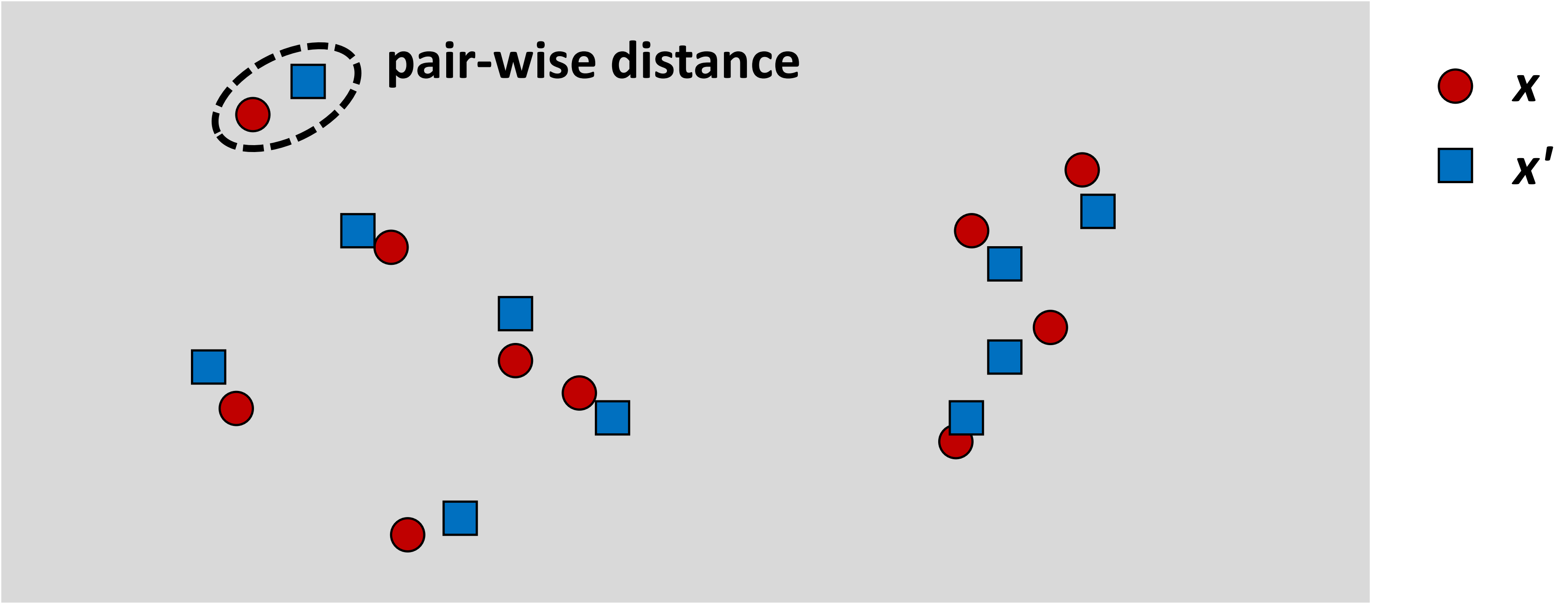
Drawbacks (Solutions)

- Achieve D^*:
  - Update D multiple times for each update of G.
  - Unrolled GAN [Metz et al., ICLR 2017]
- Introduce direct matching:
  - Sample-wide distance.



Drawbacks (Solutions)

$$|x - x'| = |x - F(x)|$$

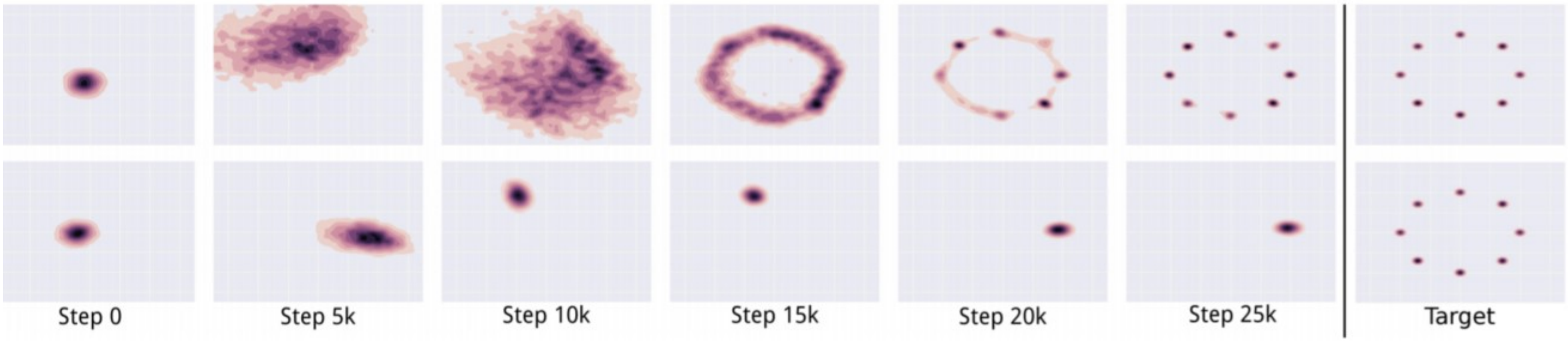


Mont Carlo → $q(x) = p_g(x)$

Drawbacks (Solutions)

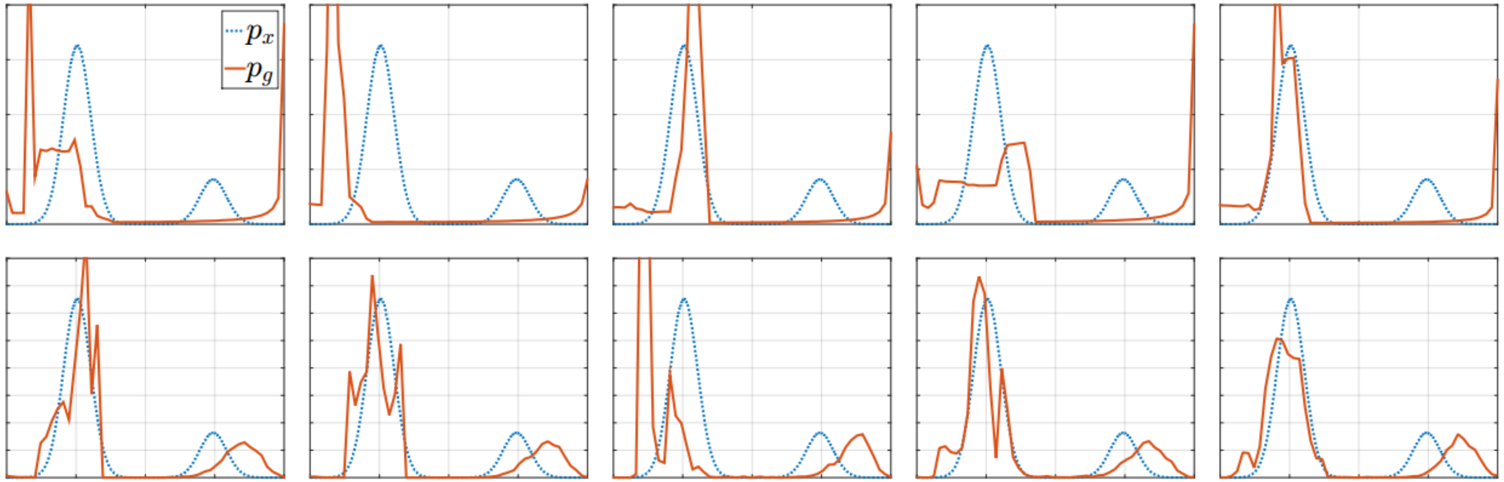
Unrolled GAN

GAN



GAN

GAN+E



Future Directions

- After explosion of applications, theoretical understanding becomes more attractive.
- Better objective functions to ensure convergence.
- There is not a good way to quantify how good the generated samples are.
- Generating discrete samples.
- One-shot imitation learning.
- Connection to reinforcement learning.


thank you