Generative Models II: Generative Adversarial Network (GAN)

M. Soleymani
Sharif University of Technology
Spring 2024

Most slides are based on Fei Fei Li and colleagues lectures, cs231n

Generative Models

PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i | x_1, ..., x_{i-1})$$

VAEs define intractable density function with latent **z**:

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

Cannot optimize directly, derive and optimize lower bound on likelihood instead

What if we give up on explicitly modeling density, and just want ability to sample?

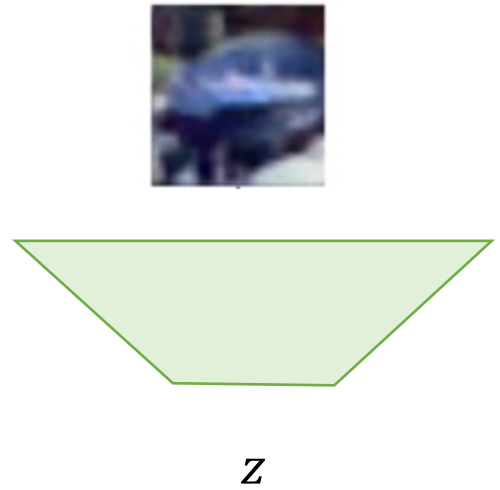
GANs: don't work with any explicit density function!
Instead, take game-theoretic approach: learn to generate from training distribution through 2-player game

Generative Adversarial Networks

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

- Solution:
  - Sample from a simple distribution, e.g. random noise.
  - Then, learn transformation to training distribution.

- Q: What can we use to represent this complex transformation?
  - A neural network!

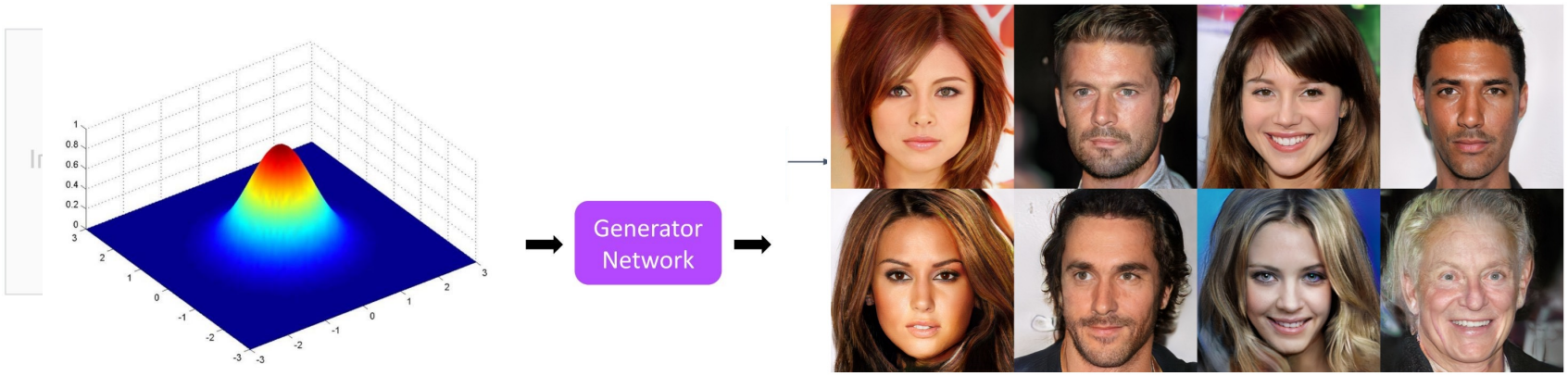


Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014.

Generative Modeling

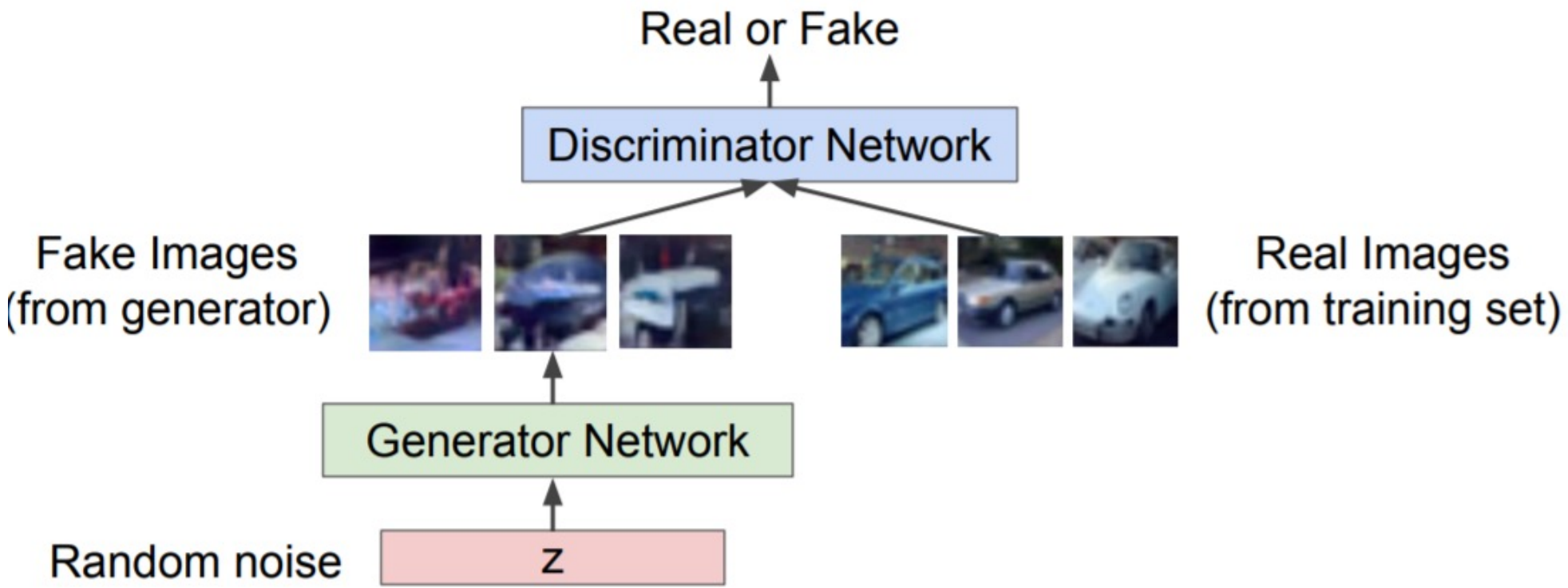
Idea: Let's learn a complex function (aka a neural network) to transform a simple distribution sample into a complex one!

- VAEs
- GANs

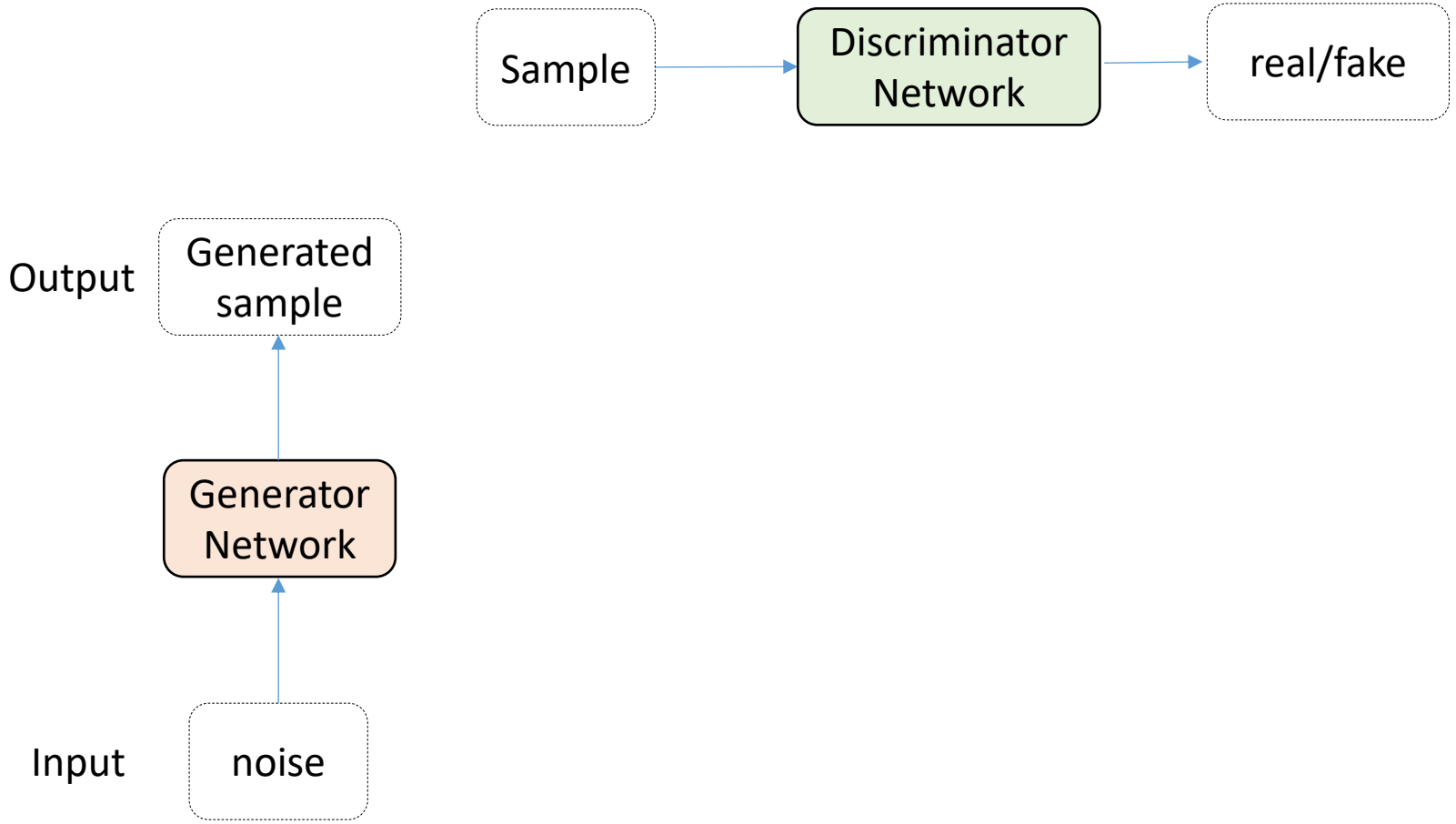


Training GANs: Two-player game

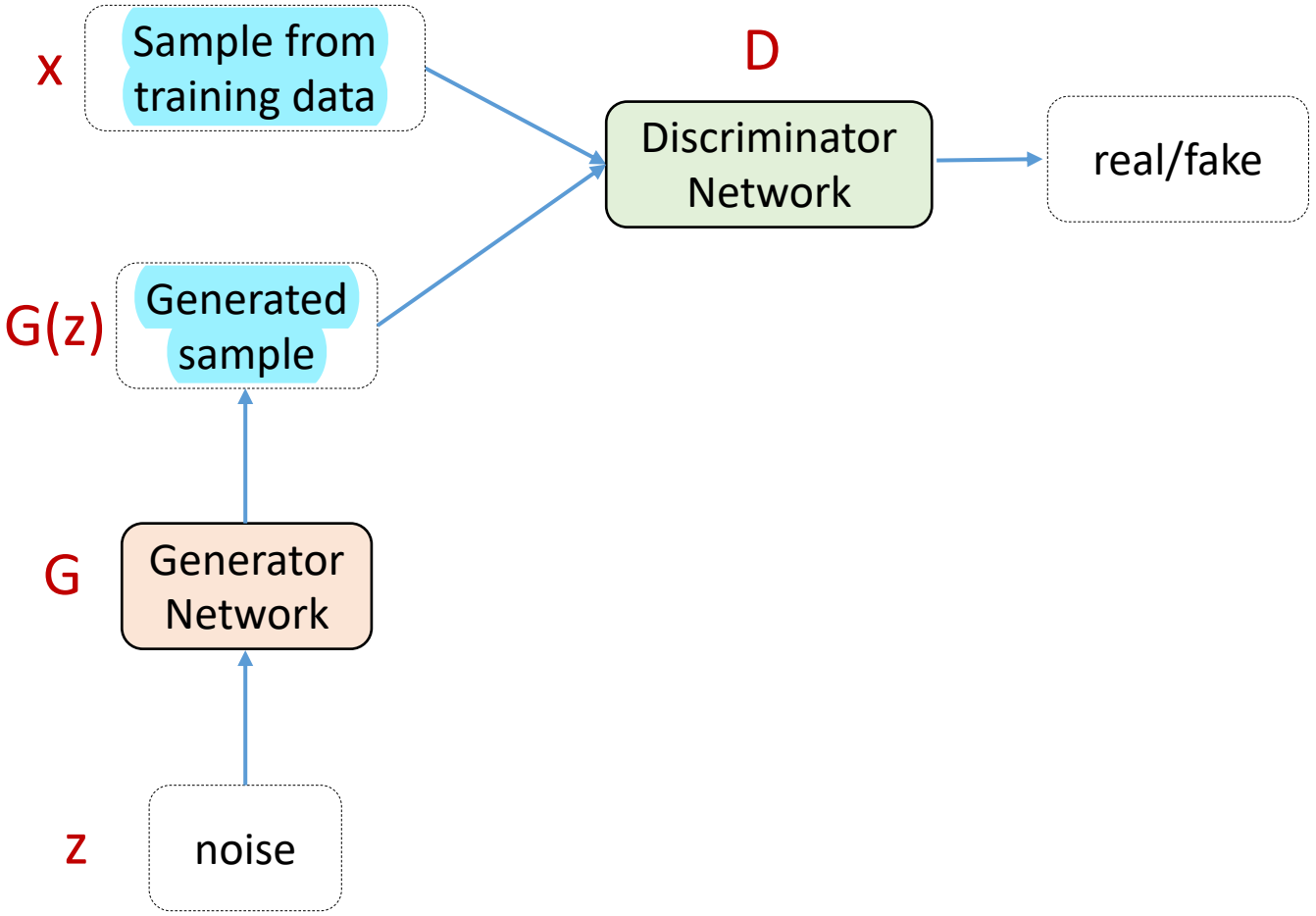
- Generator network: try to fool the discriminator by turning noise into real-looking data
- Discriminator network: try to distinguish real data from fakes created by the generator



GAN architecture



GAN architecture



Training GANs

- Generator network: intend to generate real-looking samples
- Discriminator network: intend to distinguish between real and fake samples

Discriminator output for real samples

Discriminator output for generated samples

$$J^{(D)} = \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

$$J^{(G)} = -\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log\left(1 - D\left(G(\boldsymbol{z})\right)\right)\right]$$

$$\theta_D^* = \max_{\theta_D} \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D_{\theta_D}(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log\left(1 - D_{\theta_D}\left(G_{\theta_G}(\boldsymbol{z})\right)\right)\right]$$

$$\theta_G^* = \min_{\theta_G} \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log\left(1 - D_{\theta_D}\left(G_{\theta_G}(\boldsymbol{z})\right)\right)\right]$$

Ian Goodfellow et al., Generative Adversarial Nets, NIPS 2014.

Training GANs: Two-player game

- Generator network: intend to generate real-looking samples
  - try to fool the discriminator by generate real-looking samples
- Discriminator network: intend to distinguish between real and fake samples
  - to evaluate the generated samples by generator network

Discriminator output for real samples

Discriminator output for generated samples

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{\boldsymbol{x}\sim p_{data}} \left[\log D_{\theta_D}(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})} \left[\log\left(1 - D_{\theta_D}\left(G_{\theta_G}(\boldsymbol{z})\right)\right)\right]$$

Discriminator (θ_D) wants to maximize objective such that D(x) is close to 1 (real) and D(G(z)) is close to 0 (fake)

Generator (θ_G) wants to minimize objective such that D(G(z)) is close to 1 (discriminator is fooled into thinking generated G(z) is real)

Ian Goodfellow et al., Generative Adversarial Nets, NIPS 2014.

Training GANs: Two-player game

- Train jointly in a minimax game:

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{\boldsymbol{x}\sim p_{data}}\left[\log D_{\theta_D}(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log\left(1 - D_{\theta_D}\left(G_{\theta_G}(\boldsymbol{z})\right)\right)\right]$$

- Objectives of the generators and discriminators:

$$J^{(D)} = \mathbb{E}_{\boldsymbol{x}\sim p_{data}}\left[\log D(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}\left[\log\left(1 - D(G(\boldsymbol{z}))\right)\right]$$

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

The training procedure for G is to maximize the probability of D making a mistake.

Ian Goodfellow et al., Generative Adversarial Nets, NIPS 2014.

GAN: Adversarial

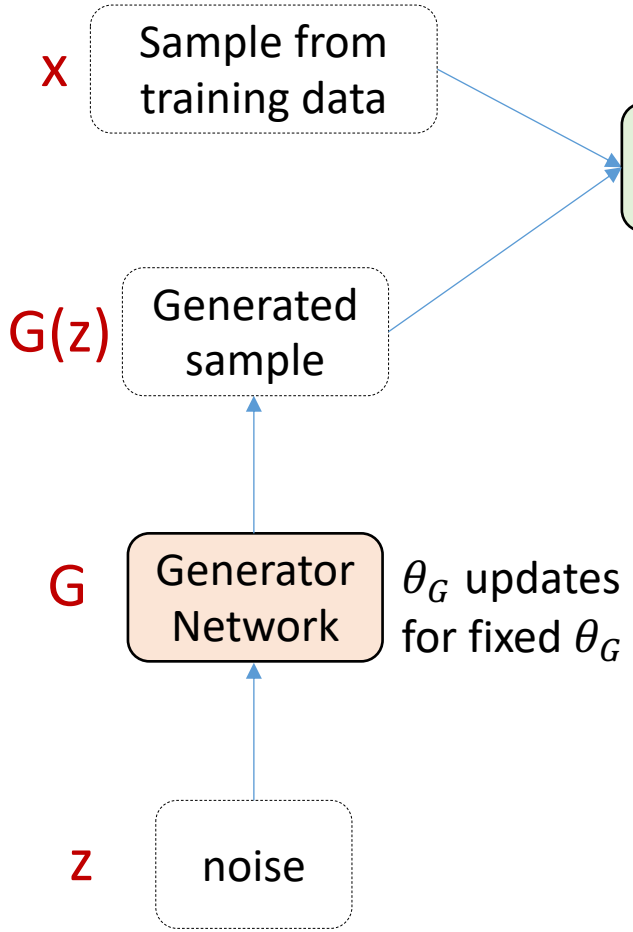
- Two networks:
  - Generator G: creates (fake) samples that the discriminator cannot distinguish
    - trained to capture the data distribution
  - Discriminator D: distinguish fake and real samples
    - estimate the probability that a sample came from the training data rather than G.

- Equilibrium is a saddle point of the discriminator loss

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

Ian J. Goodfellow et al. Generative Adversarial Networks, 2014.

GAN Training



Each player's cost depends on the parameters of other player. However, each player can only optimize its own parameters.

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

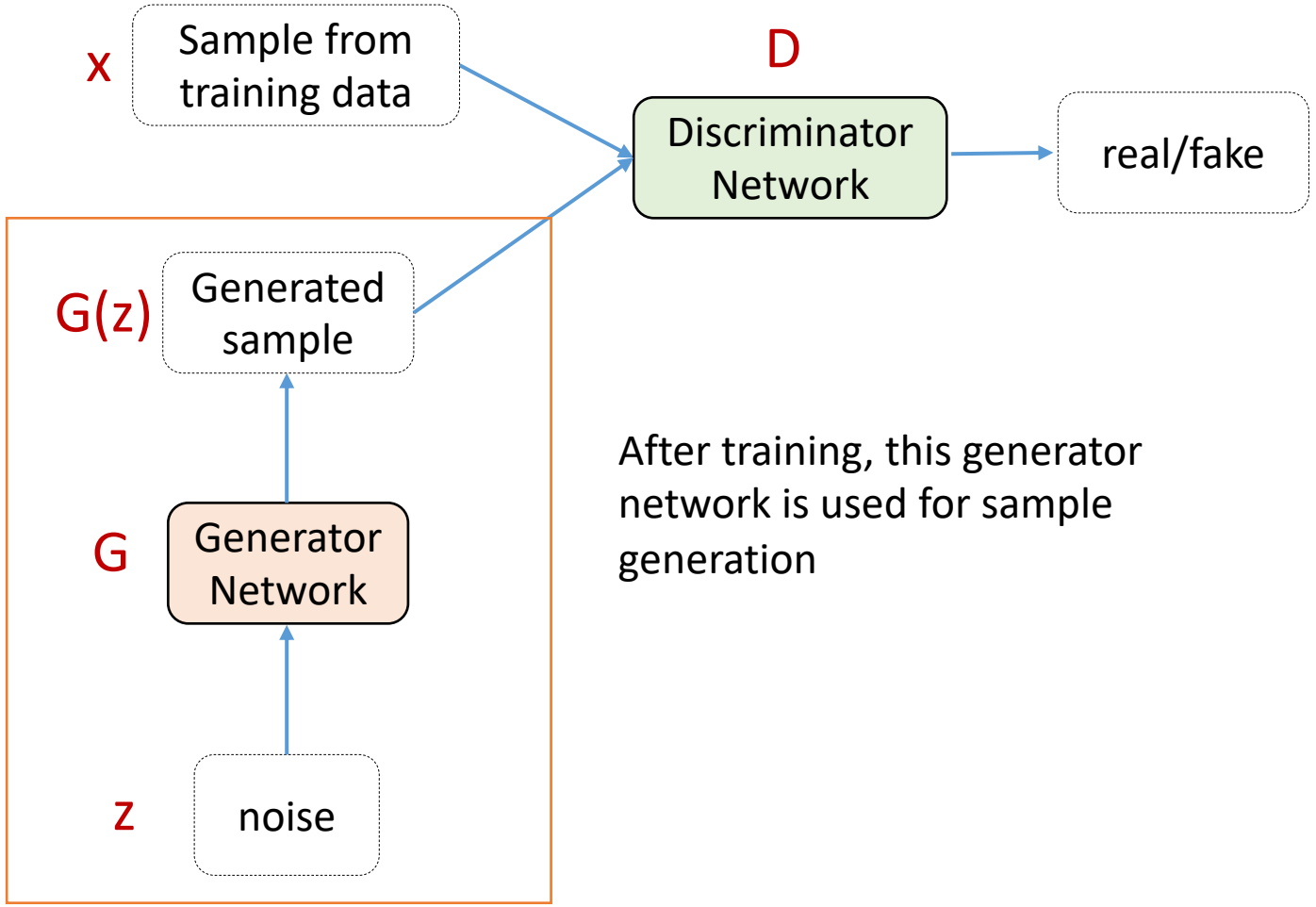
1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

GAN architecture



After training, this generator network is used for sample generation

Training GAN

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:

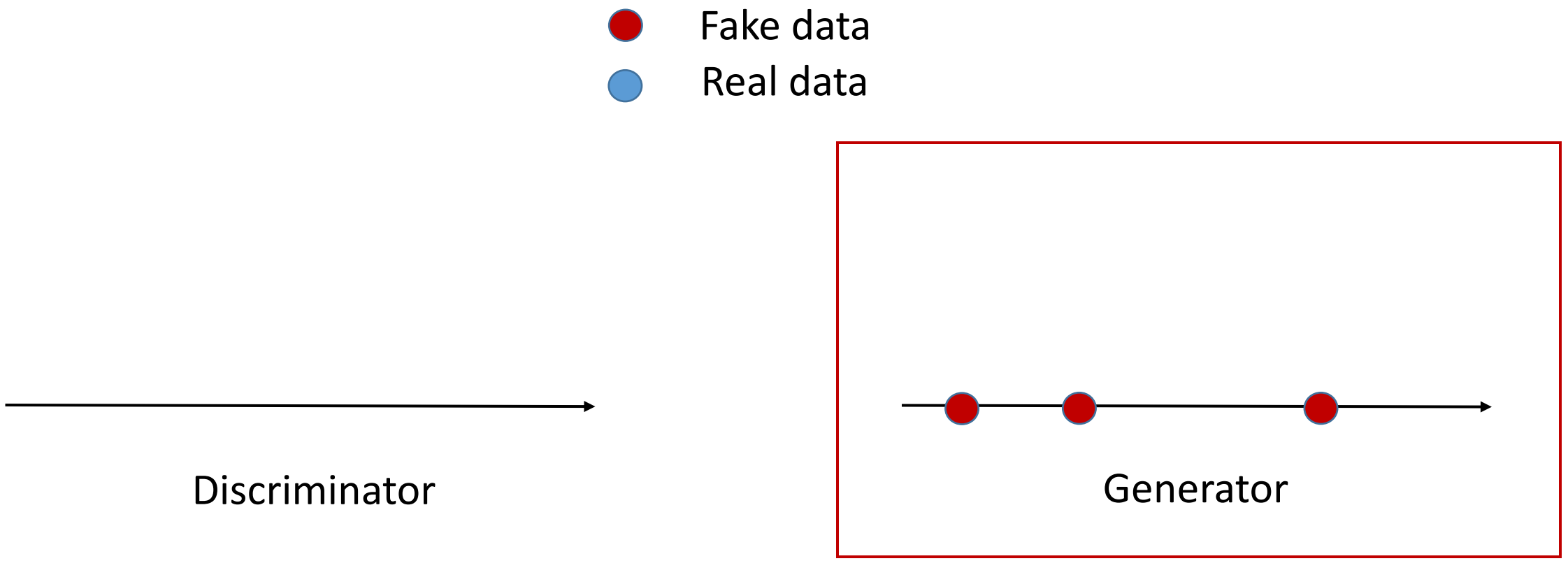
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

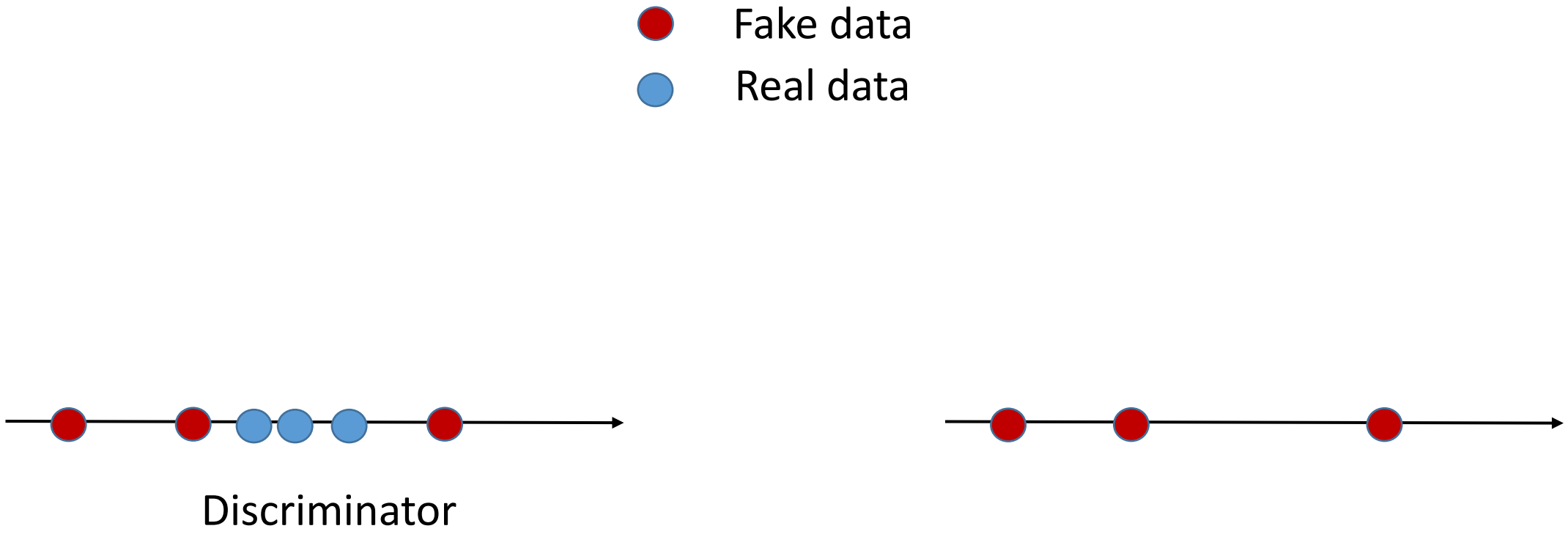
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014.

GAN: Hypothetical Example



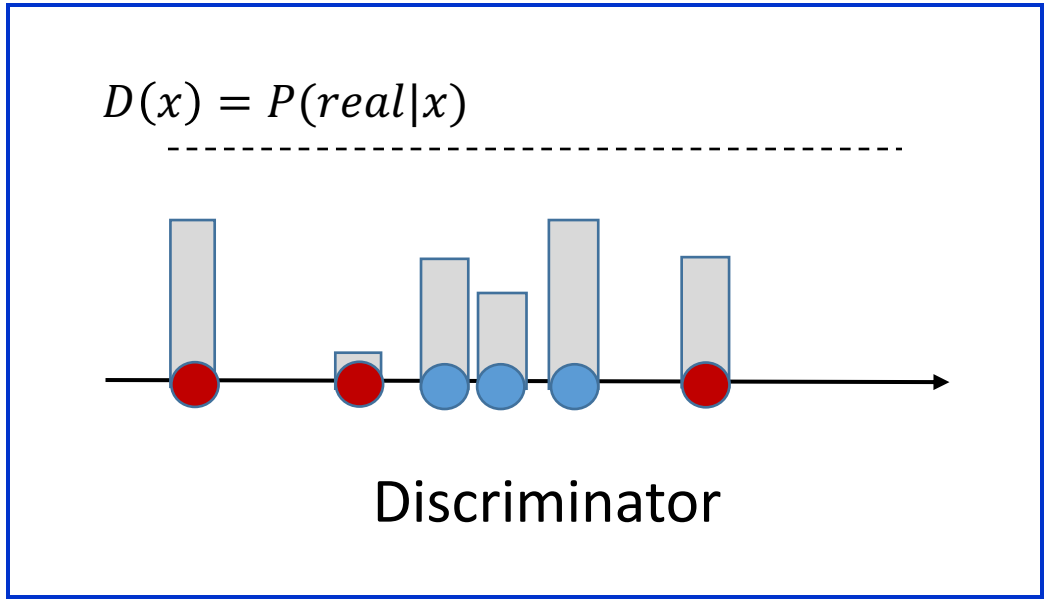
GAN: Hypothetical Example



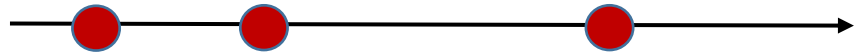
GAN: Hypothetical Example

Fake data

Real data



Initialization

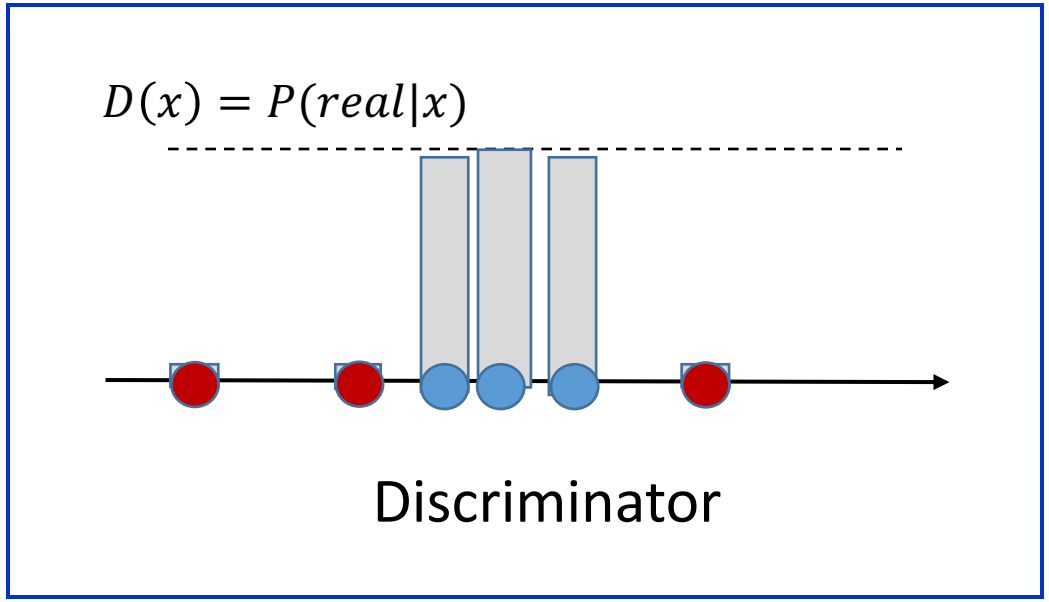


Generator

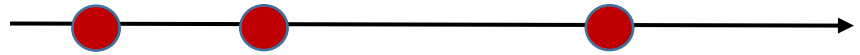
GAN: Hypothetical Example

Fake data

Real data



After training

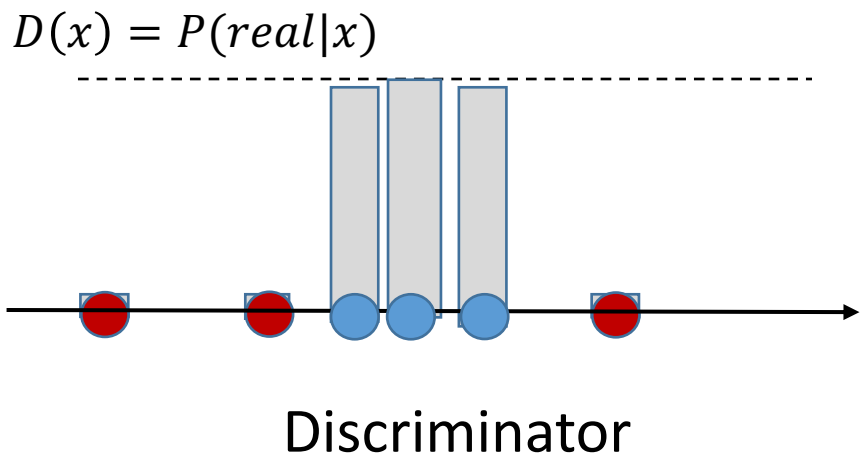


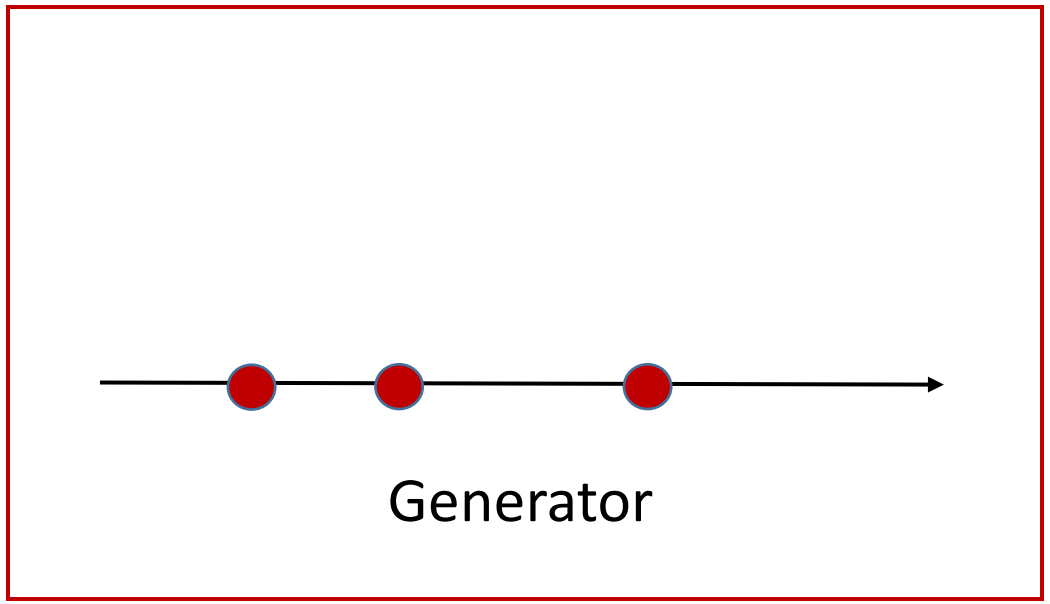
Generator

GAN: Hypothetical Example

Fake data

Real data

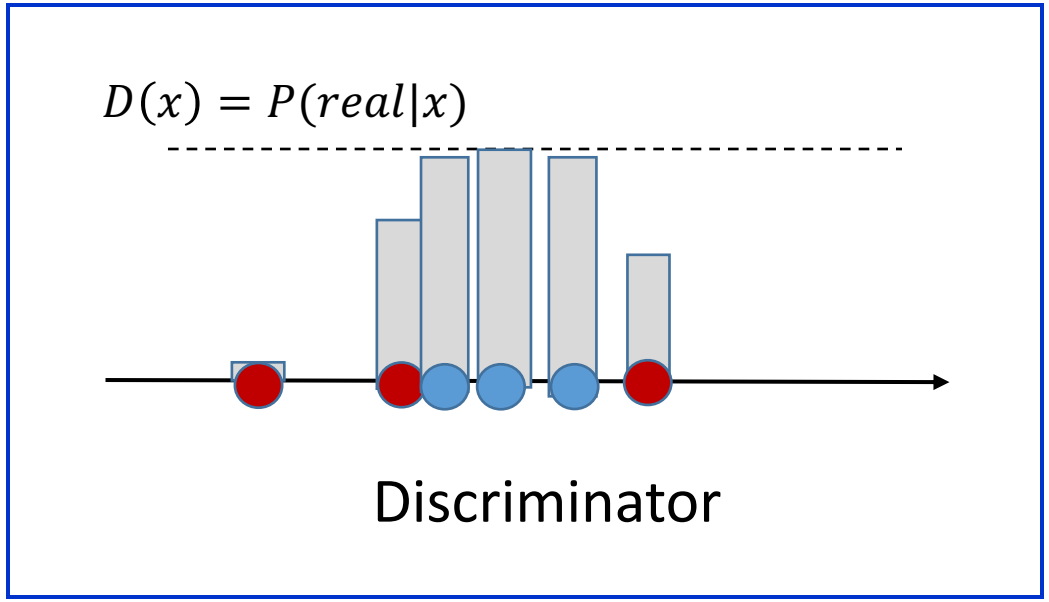




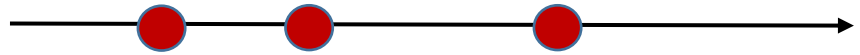
GAN: Hypothetical Example

Fake data

Real data



previous

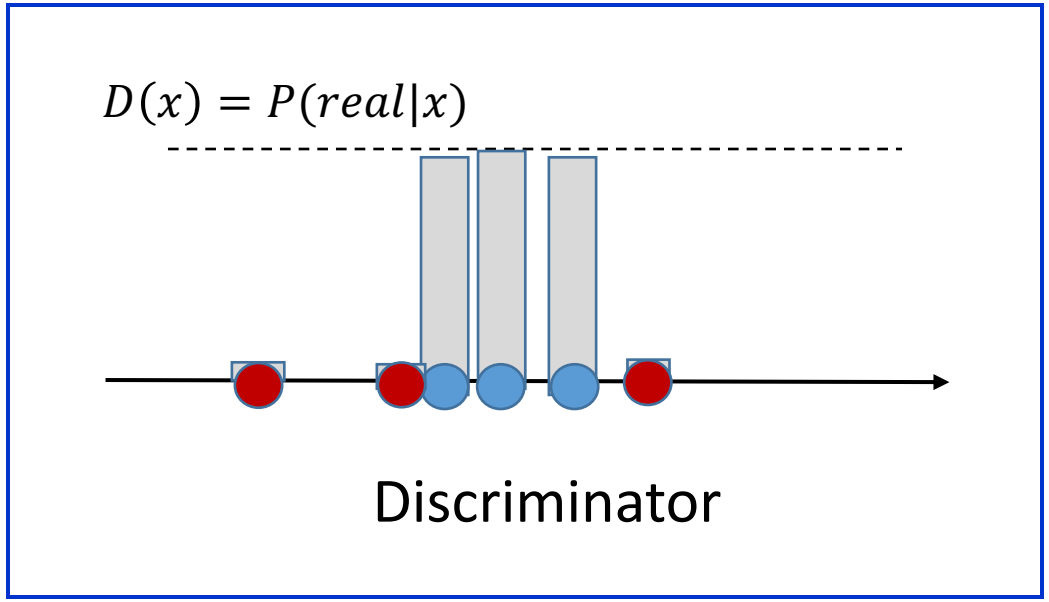


Generator

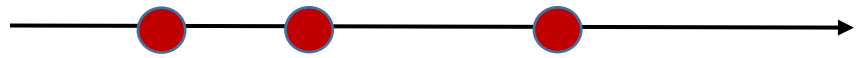
GAN: Hypothetical Example

Fake data

Real data



new



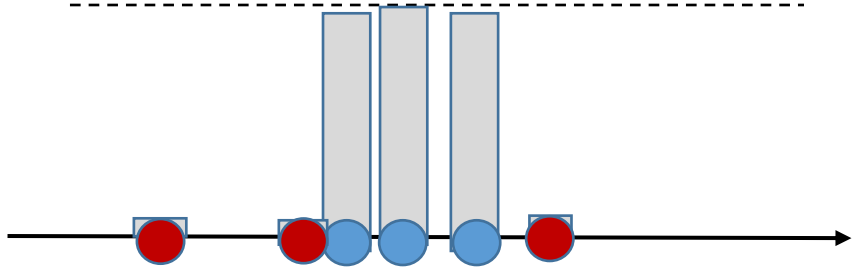
Generator

GAN: Hypothetical Example

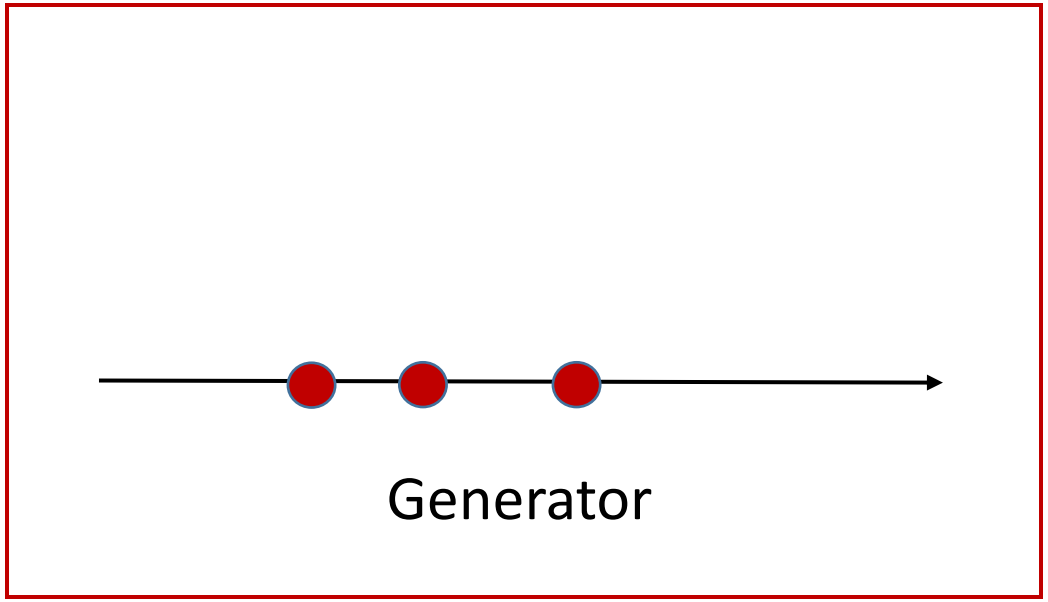
Fake data

Real data

$D(x) = P(real|x)$



Discriminator

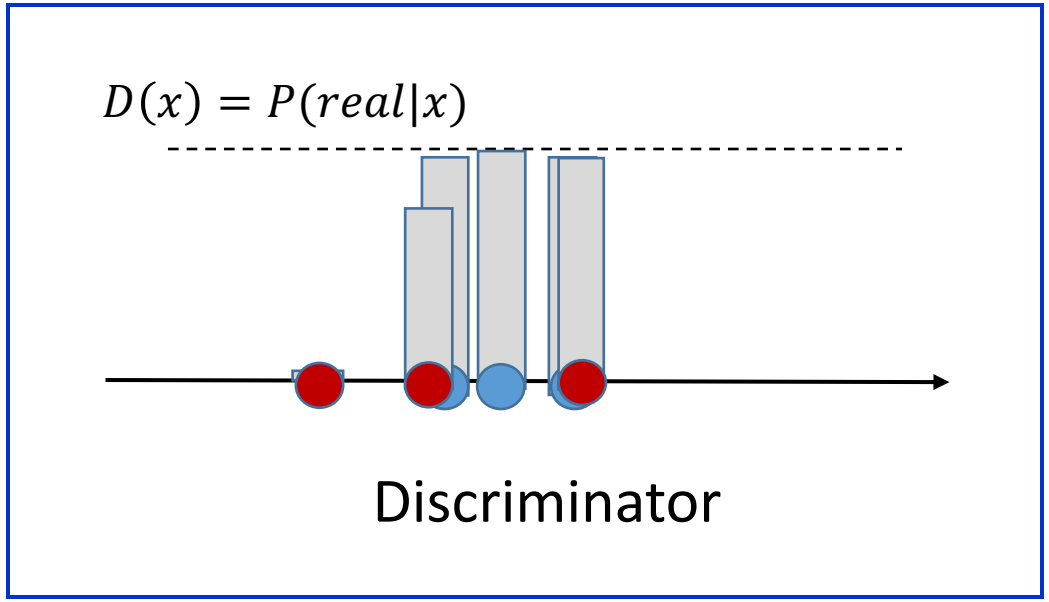


Generator

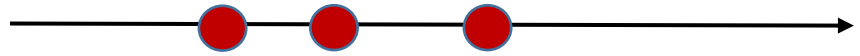
GAN: Hypothetical Example

Fake data

Real data



previous

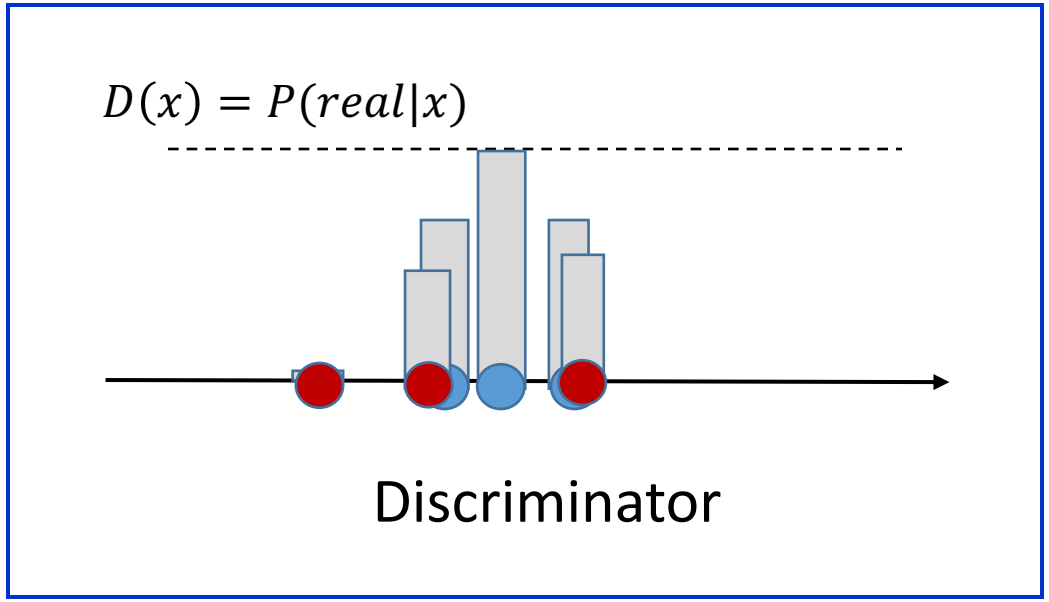


Generator

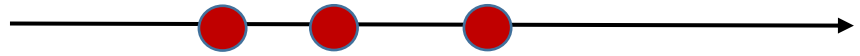
GAN: Hypothetical Example

Fake data

Real data



new



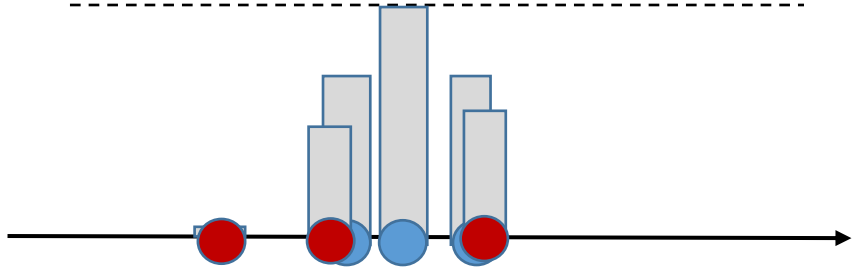
Generator

GAN: Hypothetical Example

Fake data

Real data

$D(x) = P(real|x)$



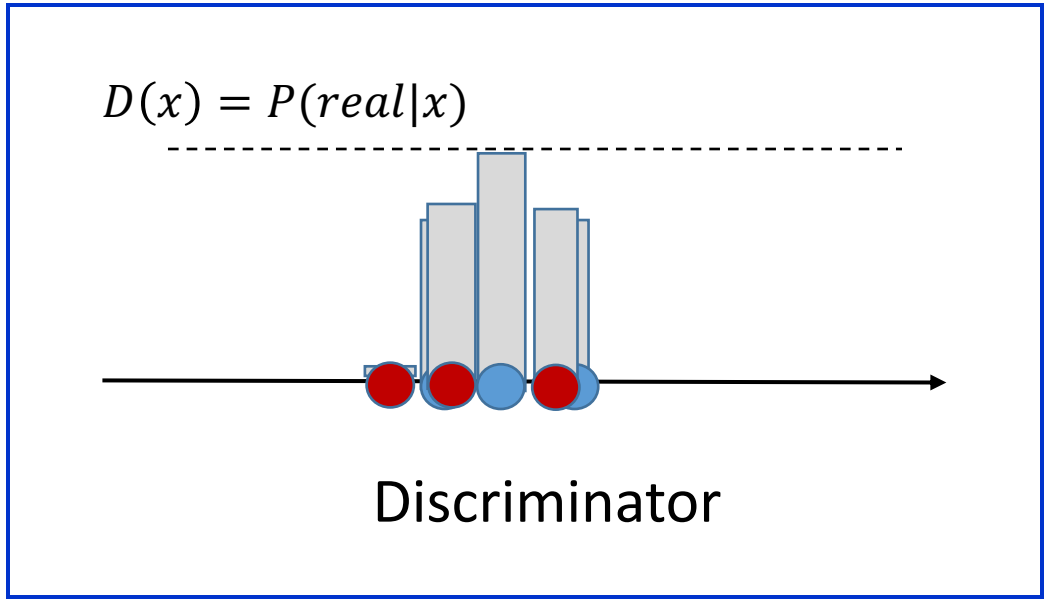
Discriminator

Generator

GAN: Hypothetical Example

Fake data

Real data



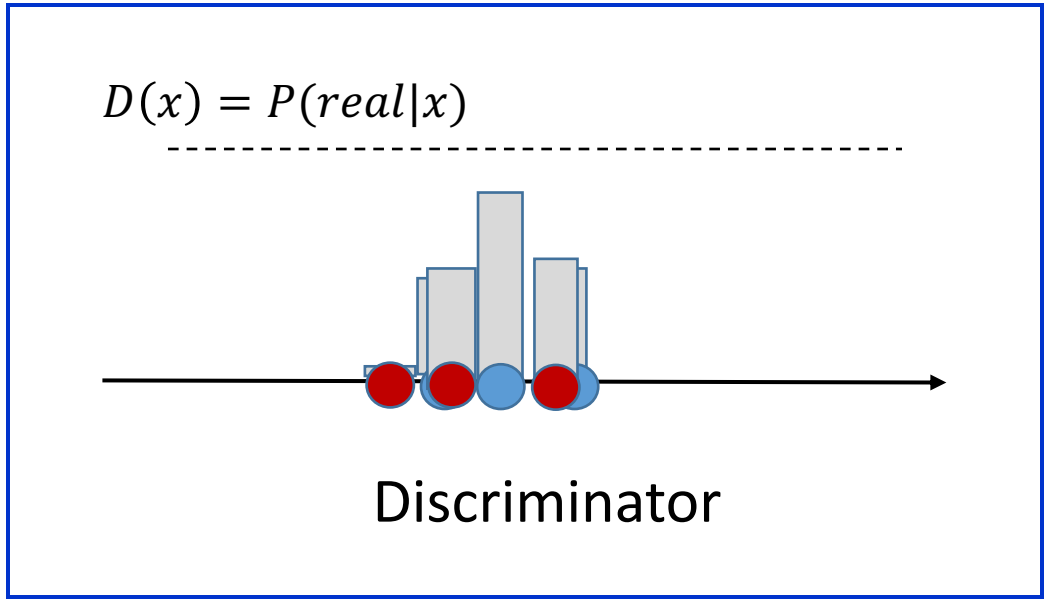
previous

Generator

GAN: Hypothetical Example

Fake data

Real data



Generator

new

Optimal Discriminator and Generator

- For a fix generator g, the optimal discriminator is:

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

- If we insert the above discriminator in the objective:
  - when both models have sufficient capacity, Nash equilibrium of the game corresponds to:

$$p_g \approx p_{data}$$

G(z) being drawn from the same distribution as the training data

D(x) will be 1/2 for all x.

Ian Goodfellow et al., Generative Adversarial Nets, NIPS 2014.

Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

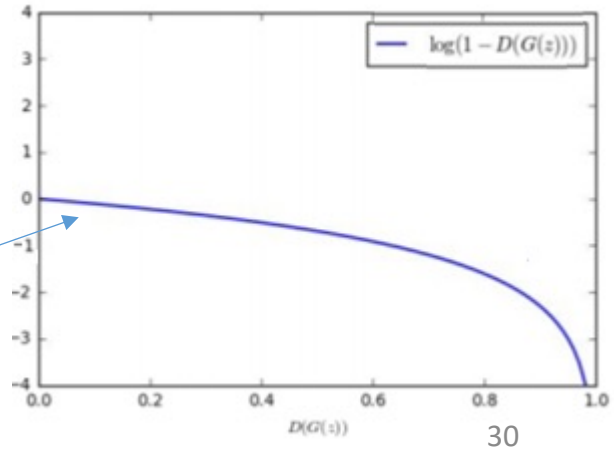
1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Gradient is relatively flat for likely fake samples while we intend to improve generator from them.



Ian Goodfellow et al., Generative Adversarial Nets, NIPS 2014.

Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

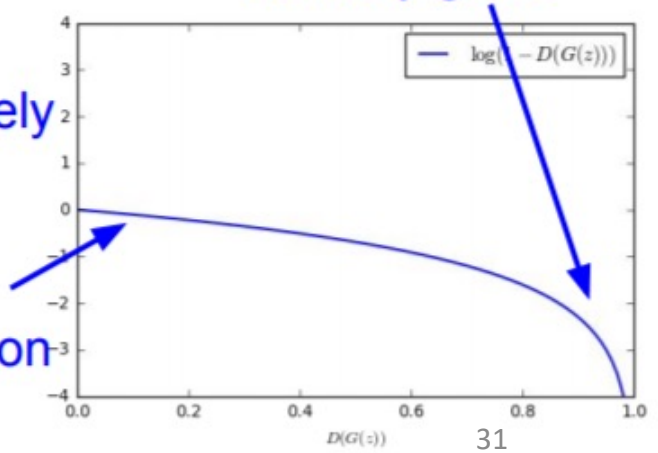
2. **Gradient descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!

Gradient signal dominated by region where sample is already good



Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014.

Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

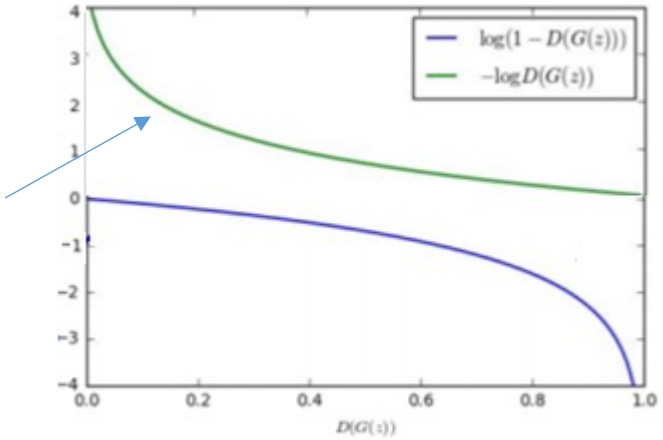
1. **Gradient ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient ascent** on generator

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Higher gradient for likely fake samples as we intend.



In generator, instead of minimizing likelihood of discriminator being correct, maximize likelihood of discriminator being wrong.

Ian Goodfellow et al., Generative Adversarial Nets, NIPS 2014.

Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

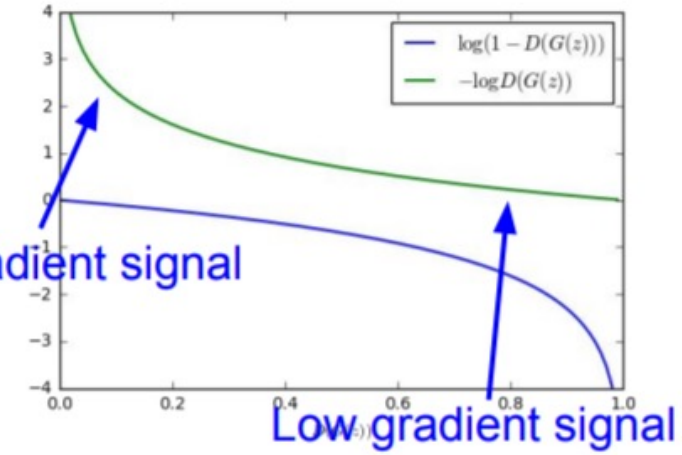
$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Instead: Gradient ascent** on generator, different objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.
Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.

Aside: Jointly training two networks is challenging, can be unstable. Choosing objectives with better loss landscapes helps training, is an active area of research.



Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014.

Putting it together: GAN training algorithm

Some find k=1 more stable, others use k > 1, no best rule.

Recent work (e.g. Wasserstein GAN) alleviates this problem, better stability!

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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

  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 ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

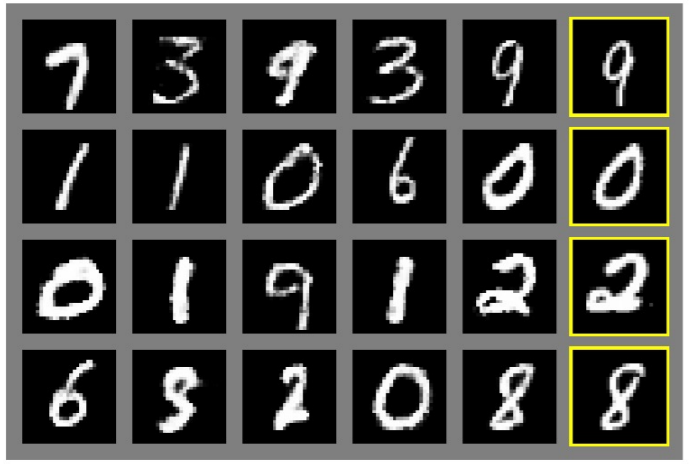
end for

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014.

Generate samples by GAN

Nearest training sample to the generated sample in the second rightmost column

MNIST



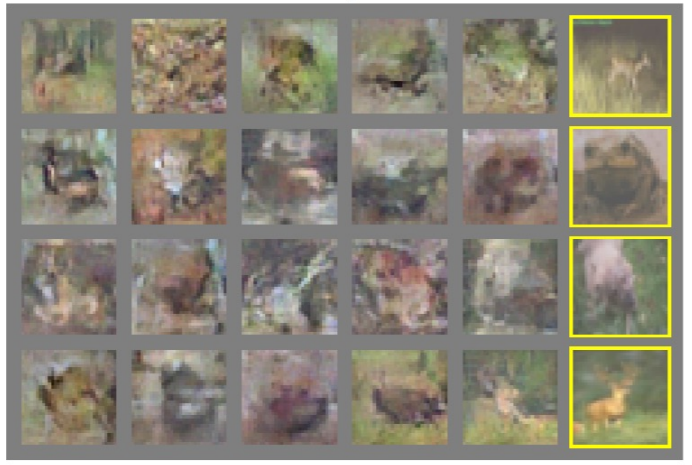
a)

b)

TFD

CIFAR-10

c)



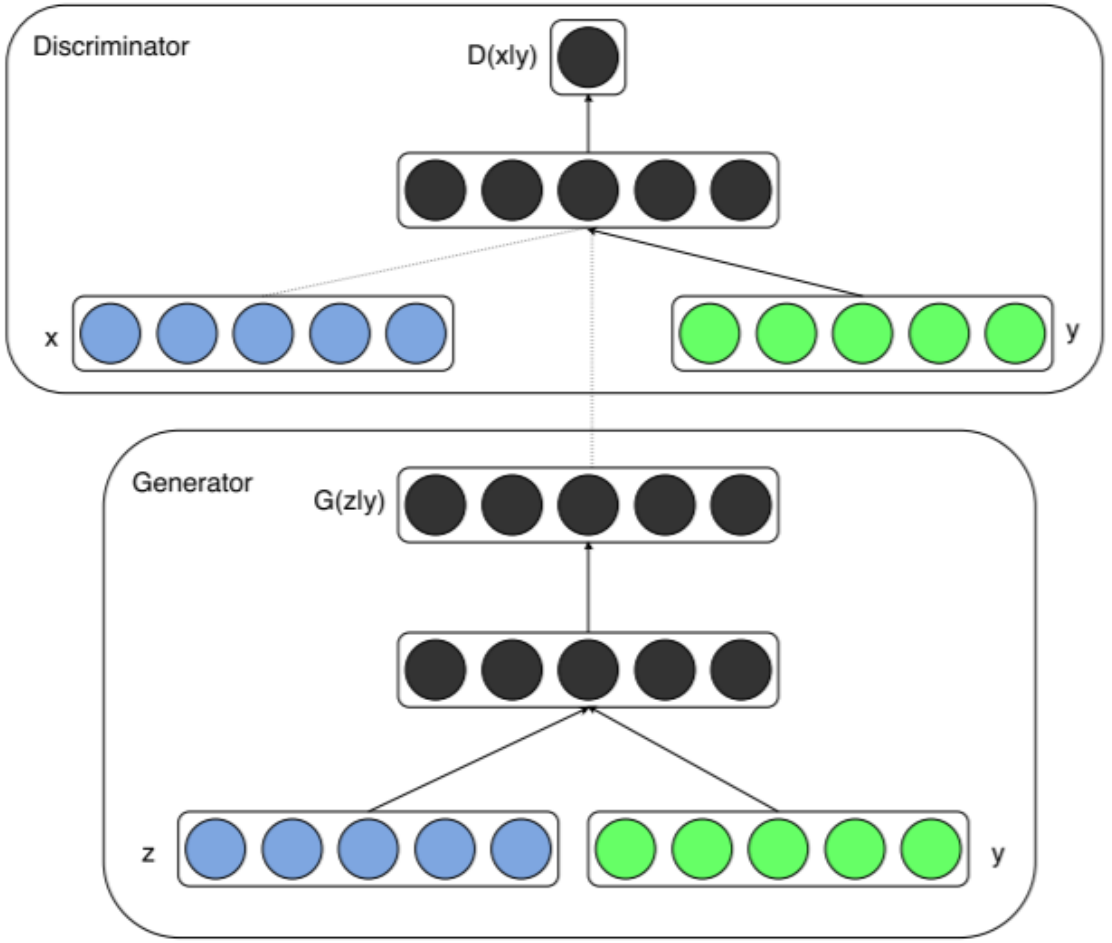
d)

CIFAR-10

Ian Goodfellow et al., Generative Adversarial Nets, NIPS 2014.

Conditional GAN

Learning a conditional model p(x|y) often gives much better samples



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

M. Mirza, S. Osindero, Conditional Generative Adversarial Nets, 2014.

Conditional GAN: MNIST

Figure 2: Generated MNIST digits, each row conditioned on one label

M. Mirza, S. Osindero, Conditional Generative Adversarial Nets, 2014.

Deep Convolutional Generative Adversarial Networks (DCGAN)

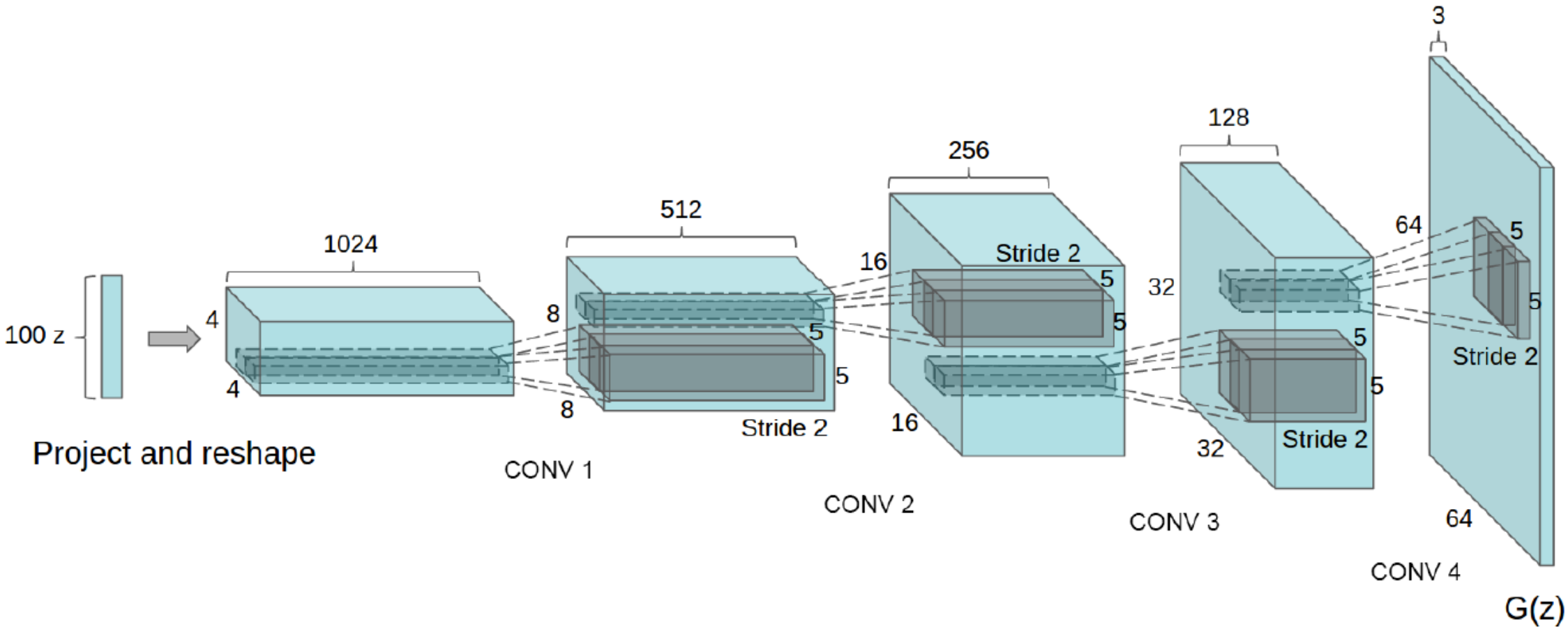
- Generator: an upsampling network with fractionally-strided convolutions
- Discriminator is a convolutional network
- Conditioning on a class label improves the generated samples

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.

Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016.

DCGAN: Convolutional Architecture



Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016.

DCGAN

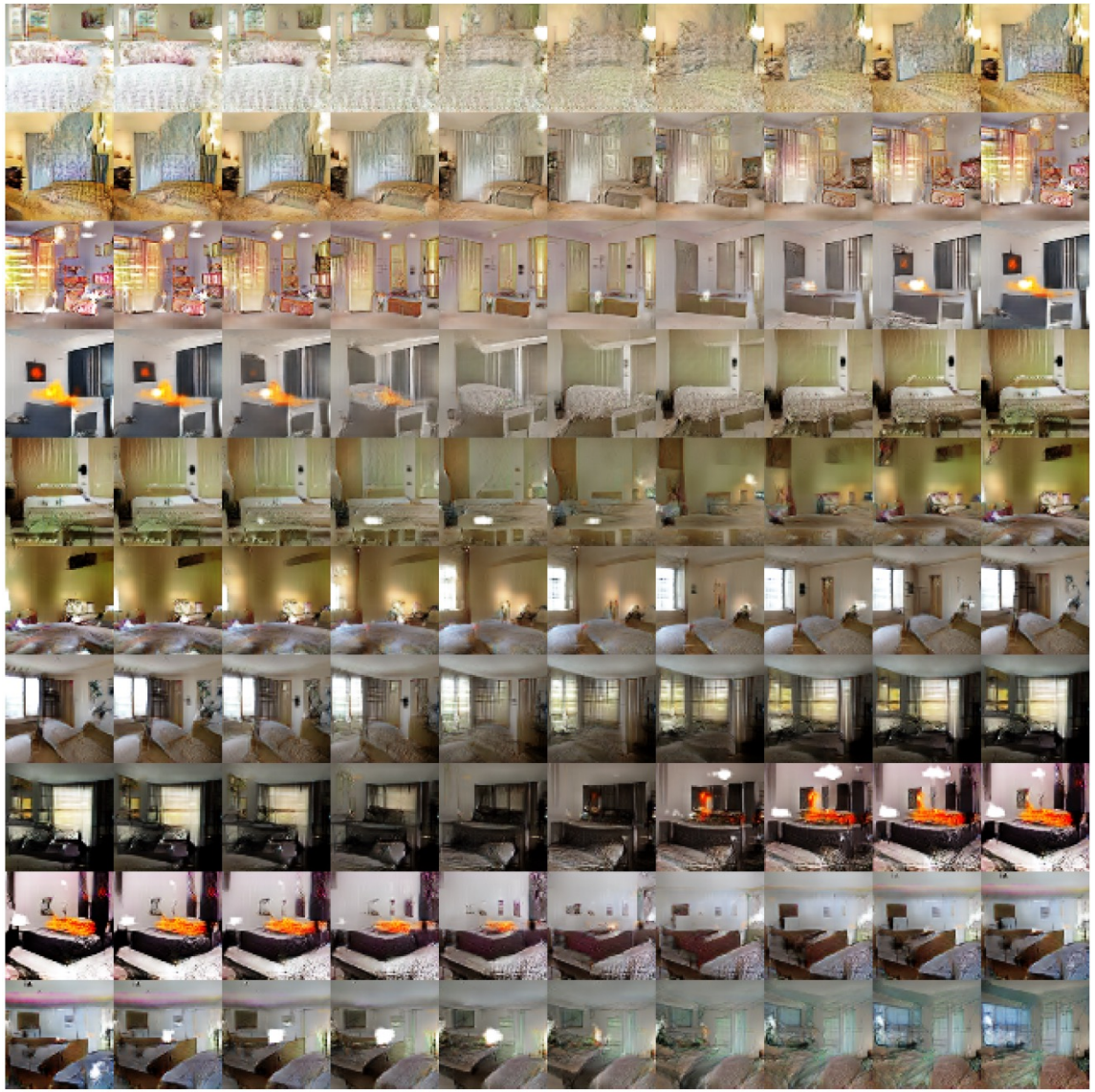
Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016.

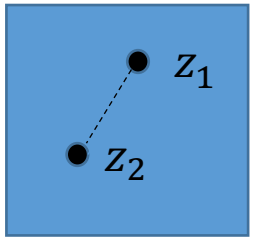
Disentangling the representation

- GANs learn a representation z of the image x.
  - this representation can capture useful high-level abstract semantic properties of data
  - However, it is difficult to make use of it.

DCGAN: Latent space

Interpolation between a series of 9 random points in z





Z space

Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016.

Vector arithmetic for visual concepts

Samples generated by model

smiling woman

neutral woman

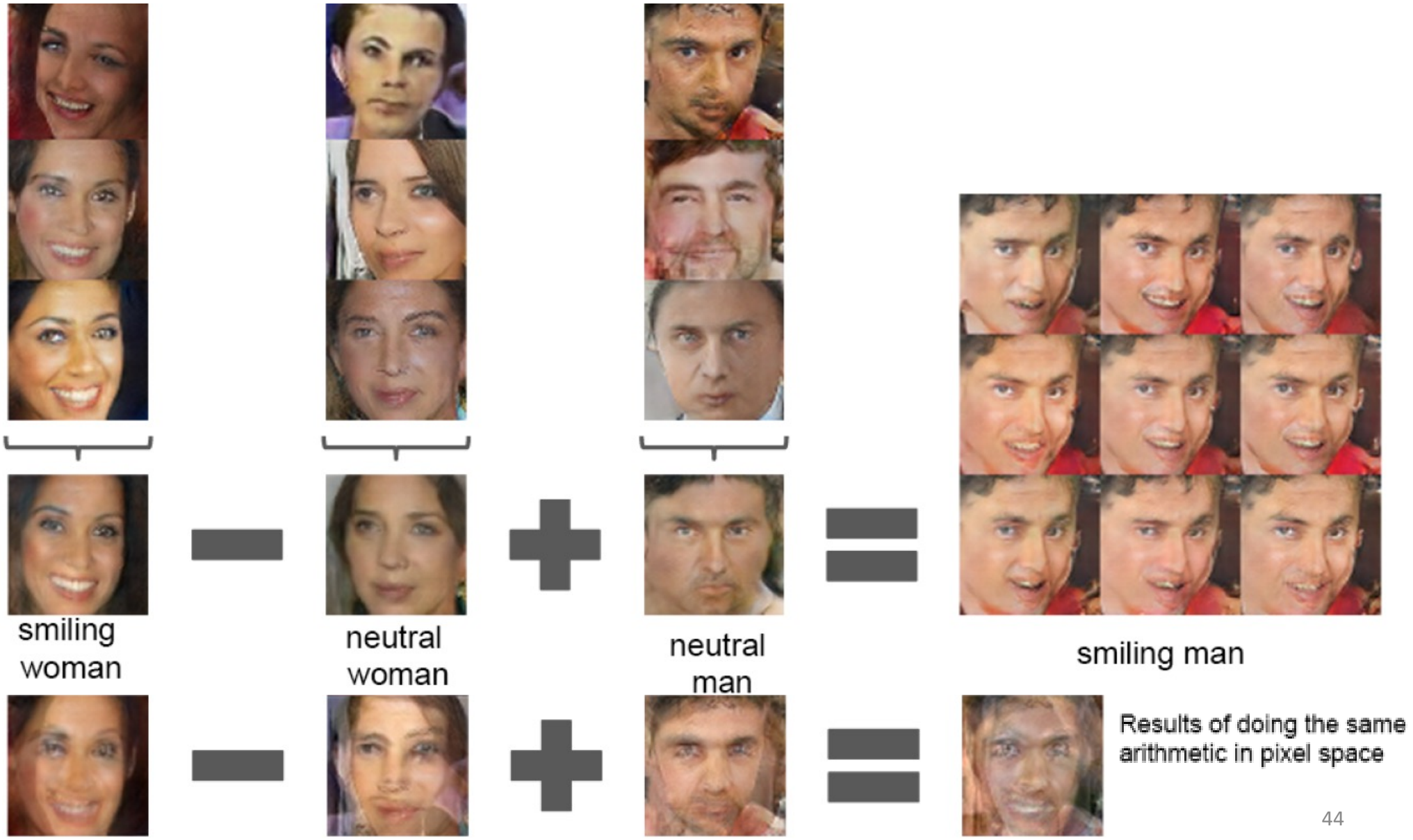
neutral man

Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016.

Vector arithmetic for visual concepts

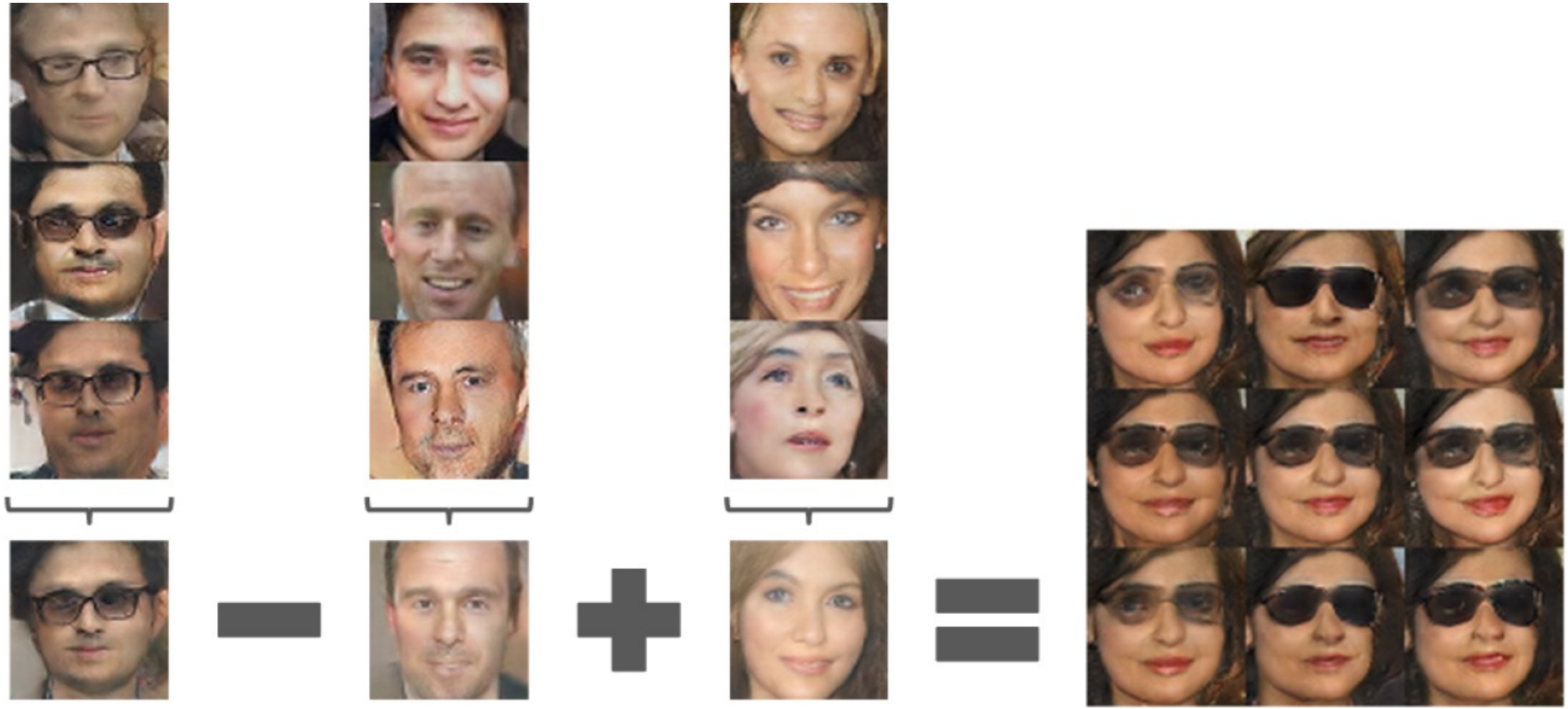
Samples generated by model

Sample corresponding to average Z of the above images



Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016.

Vector arithmetic for visual concepts



Radford et al., Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR 2016.

GANs: Applications

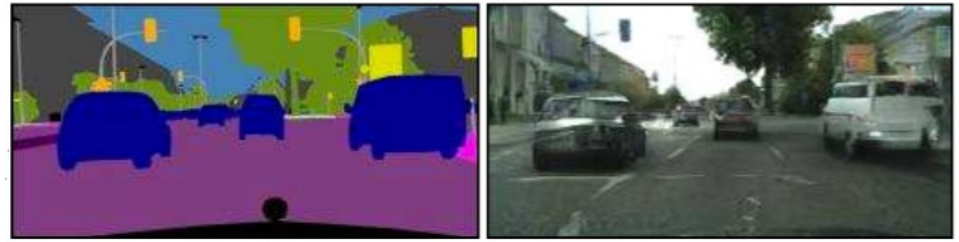
GAN Applications

- Conditional image generation tasks
  - Text-to-Image Generation
  - Image to image tasks
    - style transfer, super-resolution, in-painting, and etc

- Other generative models
  - Speech synthesis: Text to Speech
  - Text generation ?

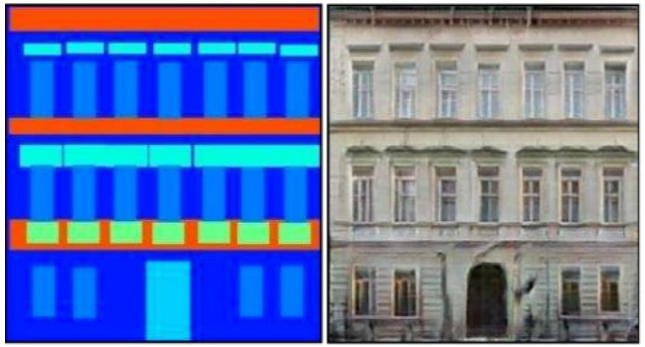
Image-to-Image Translation (pix2pix)

Labels to Street Scene



input output

Labels to Facade



input output

BW to Color

input output

Aerial to Map

input output

Day to Night

input output

Edges to Photo

input output

Phillip Isola et al., Image-to-Image Translation with Conditional Adversarial Networks, CVPR 2017.

Image-to-Image Translation (pix2pix)

- Supervised image-to-image translation tasks

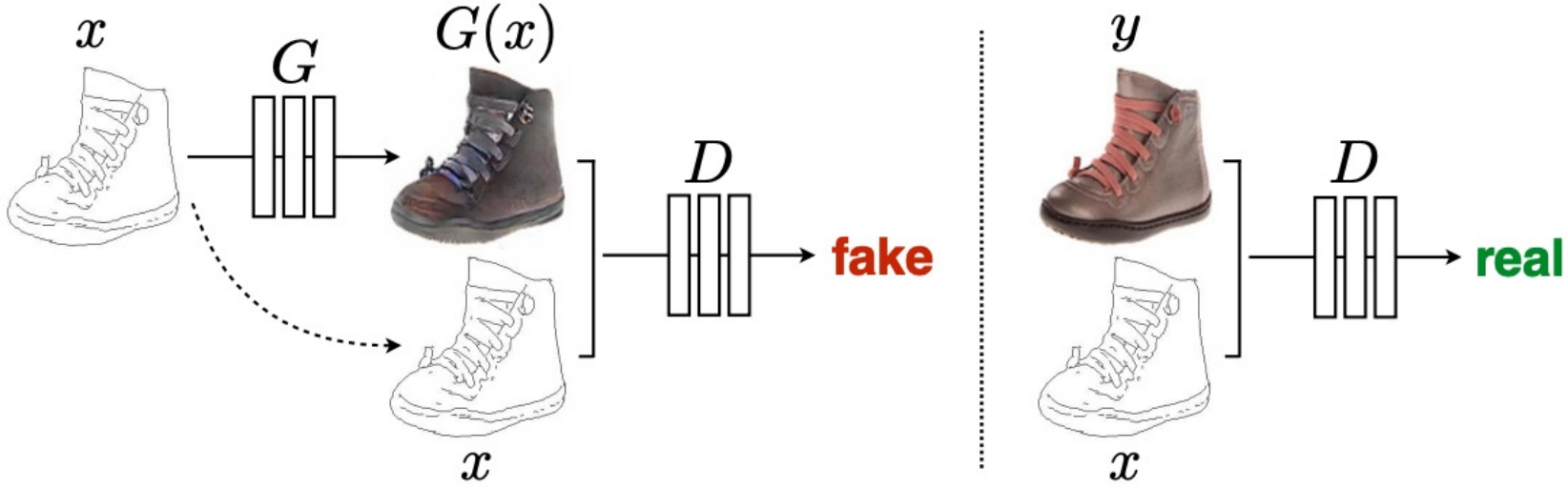
- Euclidean distance (used in the traditional methods) causes blurring
  - minimized by averaging all plausible outputs

- GAN-based networks not only learn the mapping from input to output, but also learn a loss function to train this mapping.
  - Objective: “make the output indistinguishable from reality”
    - this automatically learn a loss function appropriate for satisfying this goal

Image-to-Image Translation (pix2pix)

$$L_{cGAN}(G, D) = E_{x,y}[\log D(x, y)] + E_{x,z}[\log(1 - D(x, G(x, z))]$$

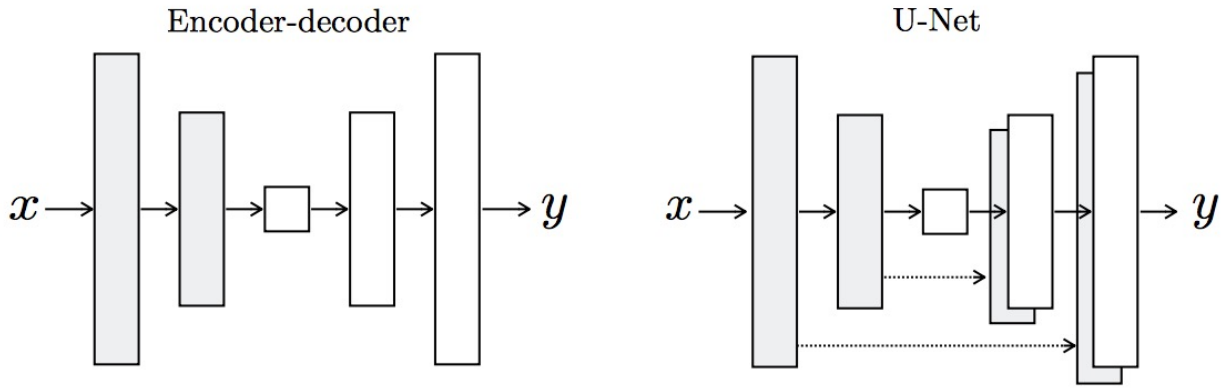
$$G^* = \arg\min_G \max_D L_{cGAN}(G, D) + \lambda E_{x,y,z}[\|y - G(x, z)\|_1]$$



Phillip Isola et al., Image-to-Image Translation with Conditional Adversarial Networks, CVPR 2017.

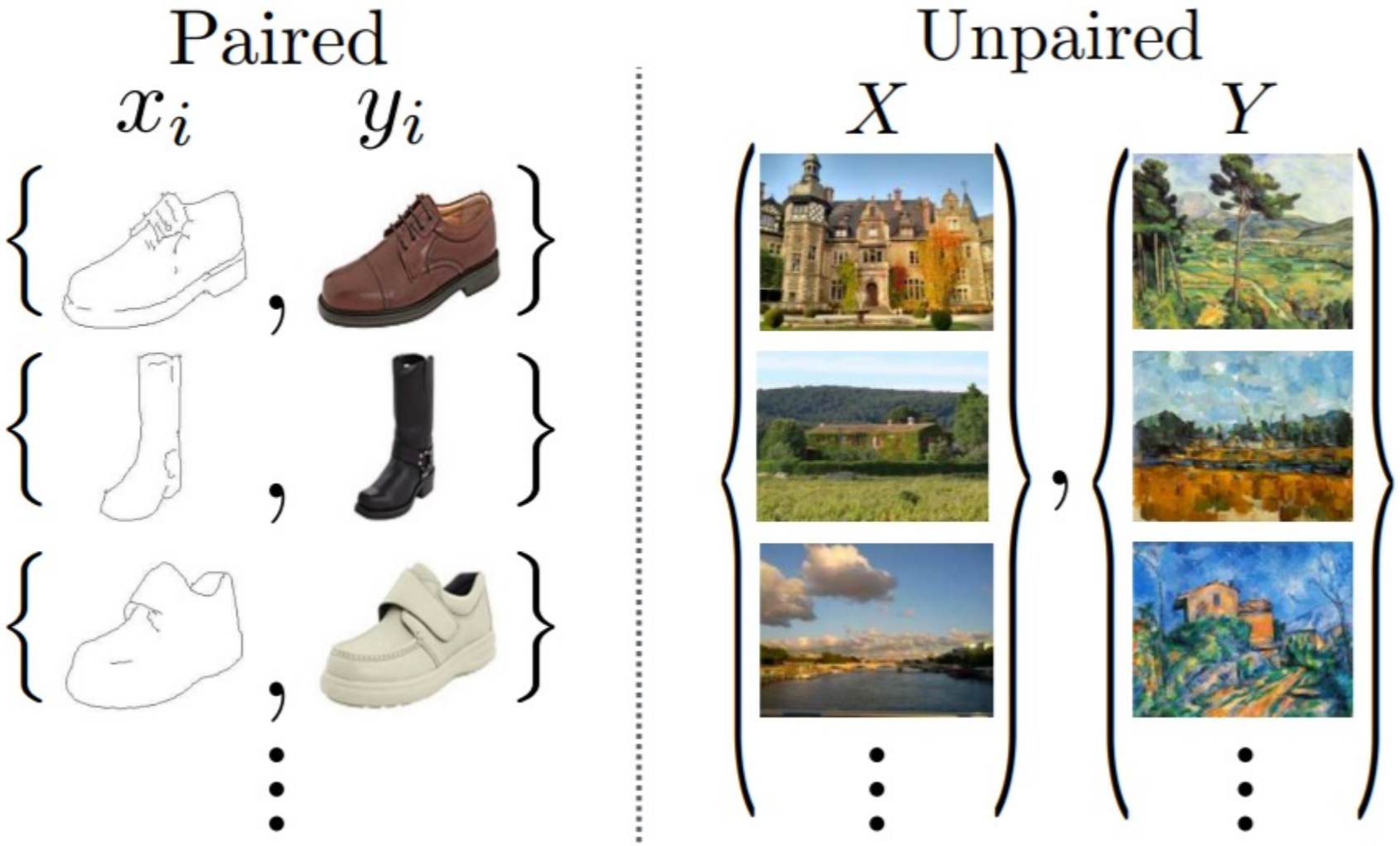
Image-to-Image Translation (pix2pix)

- Provides noise only in the form of dropout, applied on several layers of our generator at both training and test time.
- Generator has a "U-Net"-based architecture
- Discriminator is a convolutional "PatchGAN" classifier, which only penalizes structure at the scale of image patches.
  - classify if each N × N patch in an image is real or fake and this discriminator is run convolutionally across the image and all responses are averaged.



Phillip Isola et al., Image-to-Image Translation with Conditional Adversarial Networks, CVPR 2017.

Unpaired data



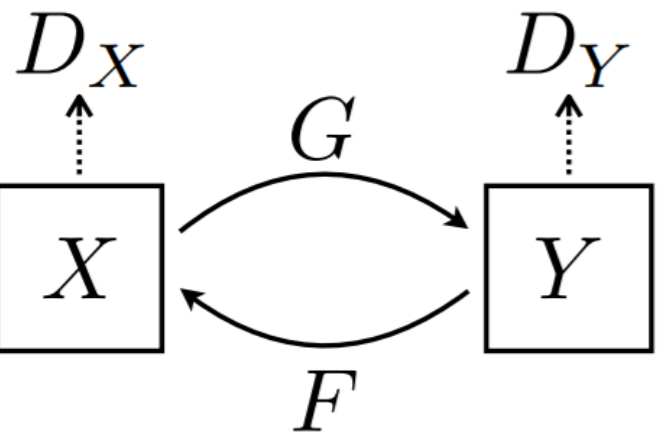
Jun-Yan Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017.

cycleGAN

- For many tasks, paired training data will not be available.
- learn a mapping $G: X \rightarrow Y$ in the absence of paired examples
  - such that the distribution of images from $G(X)$ is indistinguishable from the distribution Y using an adversarial loss.
- learn to synthesize pairs of corresponding images without correspondence supervision
- couple it with an inverse mapping $F: Y \rightarrow X$ and introduce a cycle consistency loss to push $F(G(X)) \approx X$ (and vice versa).

Jun-Yan Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017.

cycleGAN



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

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

encourages G to translate X into outputs indistinguishable from domain Y
while DY aims to distinguish between translated samples G(x) and real samples y

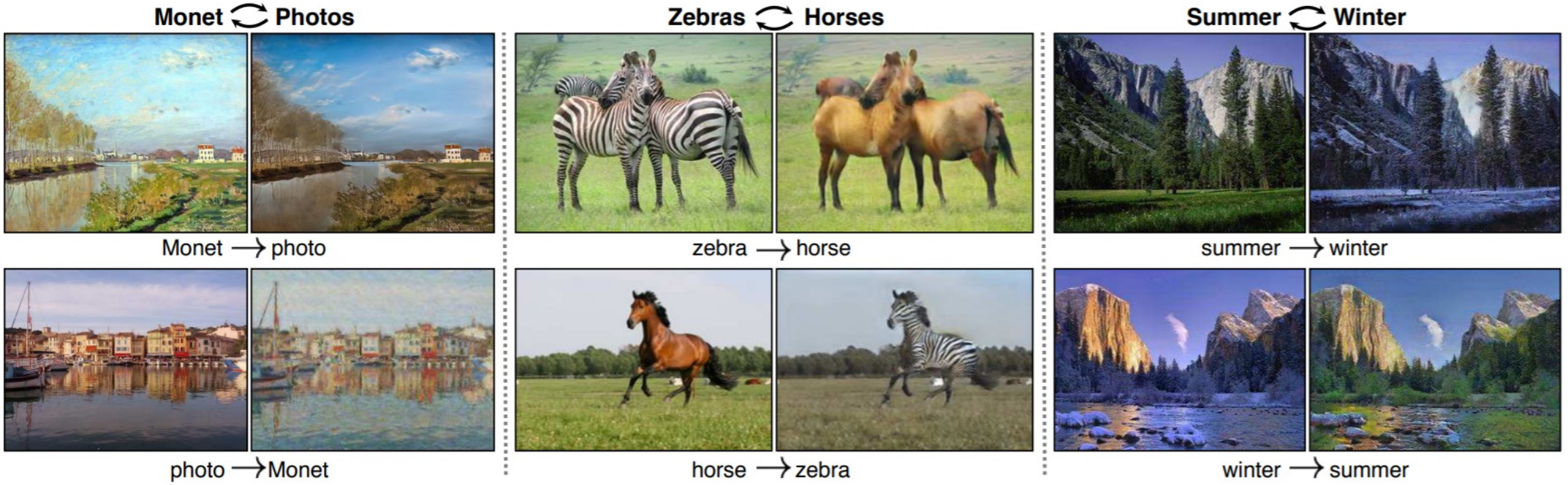
$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

$F(G(x)) \approx x$

$G(F(y)) \approx y$

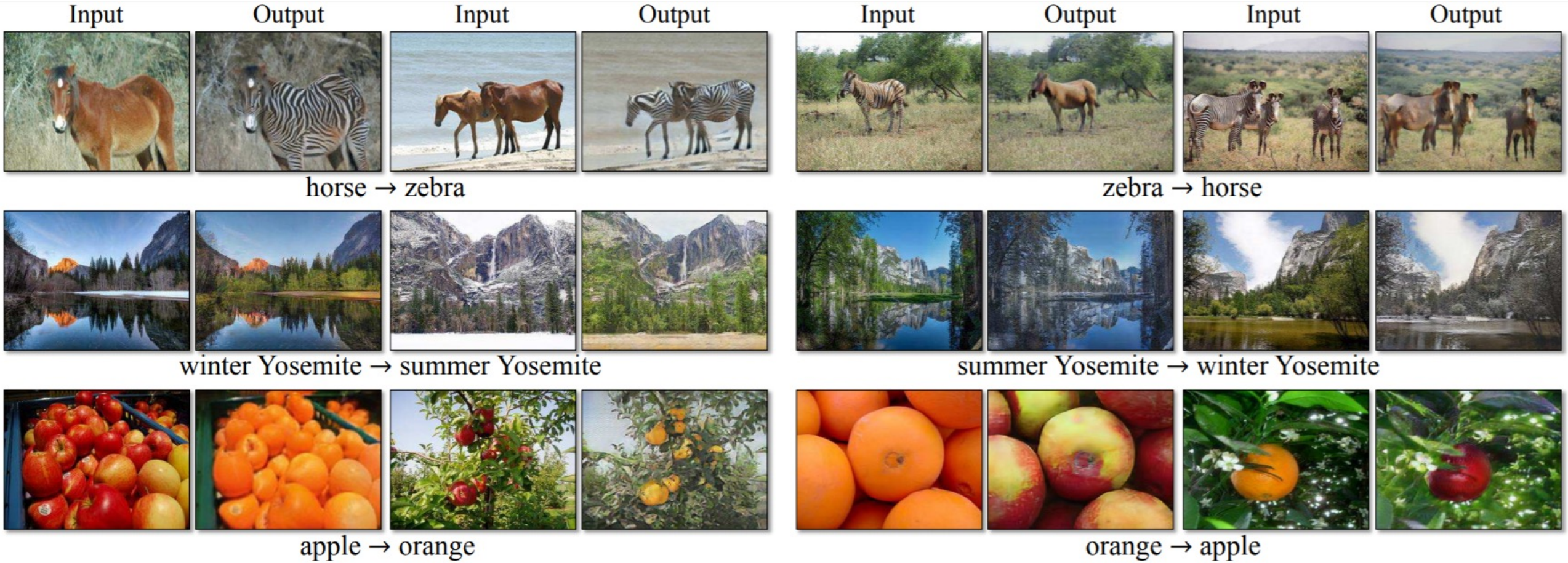
Jun-Yan Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017.

CycleGAN



Jun-Yan Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017.

CycleGAN



Jun-Yan Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017.

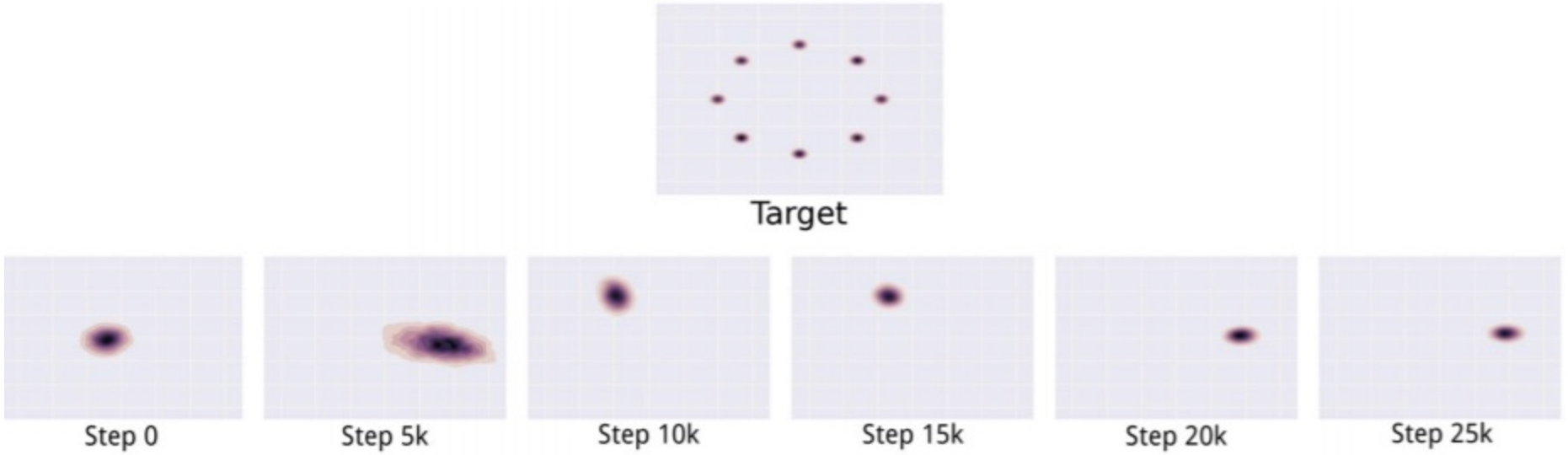
GANs: Advances

GAN Challenges

See also: https://github.com/soumith/ganhacks for tips and tricks for trainings GANs

- Difficulty of training
  - training GANs requires finding Nash equilibria in high dimensional, continuous, non-convex games
  - GANs easily suffer from mode collapse
    - Thus, applications of GANs are often limited to problems where it is acceptable for the model to produce a small number of distinct outputs
- Thus, the quality and variation of the generated samples is a challenge.
- Moreover, there is no clearly justified way to quantitatively score samples.

Mode Collapse



Metz et al, Unrolled Generative Adversarial Networks, ICLR 2017.

Better Training

- Wasserstein GAN
- f-GAN
- LSGAN
- EBGAN
- BEGAN
- …

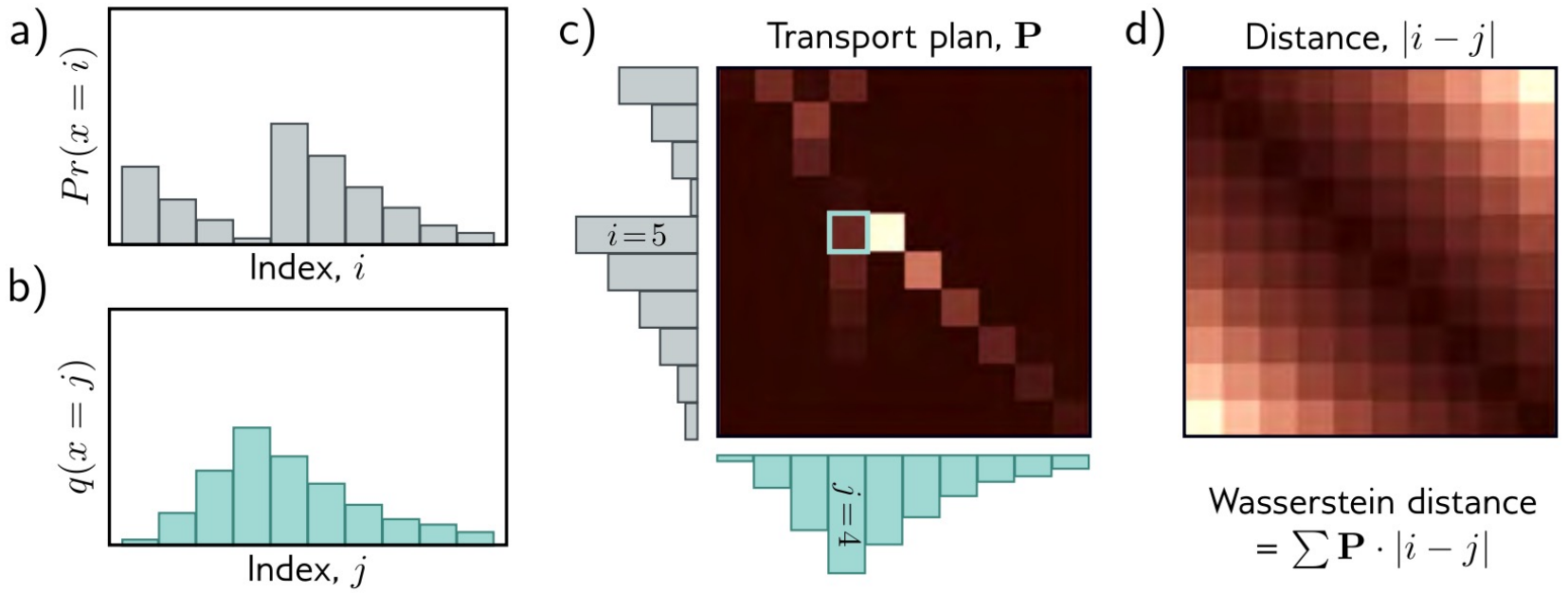
Vanishing Gradients

- If the probability distributions are completely disjoint, KL distance is infinite, and any small change to the generator will not decrease the loss.
  - $KL(q||p)$ is possibly infinite when there are points such that $p(x) = 0$ and $q(x) > 0$

- Wasserstein distance is well-defined even when the distributions are disjoint and decreases smoothly as they become closer to one another.

Wasserstein Distance

- The quantity of work required to transport the probability mass from one distribution to create the other.
  - "work": the mass multiplied by the distance moved.



$$D_w\Big[Pr(x)||q(x)\Big] = \min_{\mathbf{P}} \left[\sum_{i,j} P_{ij} \cdot |i-j|\right],$$

subject to the constraints that:

$$\begin{aligned} \textstyle\sum_j P_{ij} &= Pr(x=i) && \text{initial distribution of } Pr(x) \\ \textstyle\sum_i P_{ij} &= q(x=j) && \text{initial distribution of } q(x) \\ P_{ij} &\geq 0 && \text{non-negative masses.} \end{aligned}$$

Wasserstein GAN

- The Earth Mover (EM) or Wasserstein-1 distance then is the "cost" of the optimal transport plan:

$$W(p,q) = \inf_{\gamma \in \Pi(p,q)} \mathbb{E}_{(x,y)\sim\gamma} [\|x - y\|]$$

γ(x, y) indicates how much "mass" must be transported from x to y

- The equivalent of the dual form

$$W(p,q) = \max_{w} [\mathbb{E}_{x\sim p}[f_w(x)] - \mathbb{E}_{x\sim q}[f_w(x)]]$$
$$s.t. \|f_w\|_L \leq 1$$

the absolute gradient of f_w w.r.t. x is less than one

- W-GAN minimized $W(p_{data}, p_g)$ in the dual form

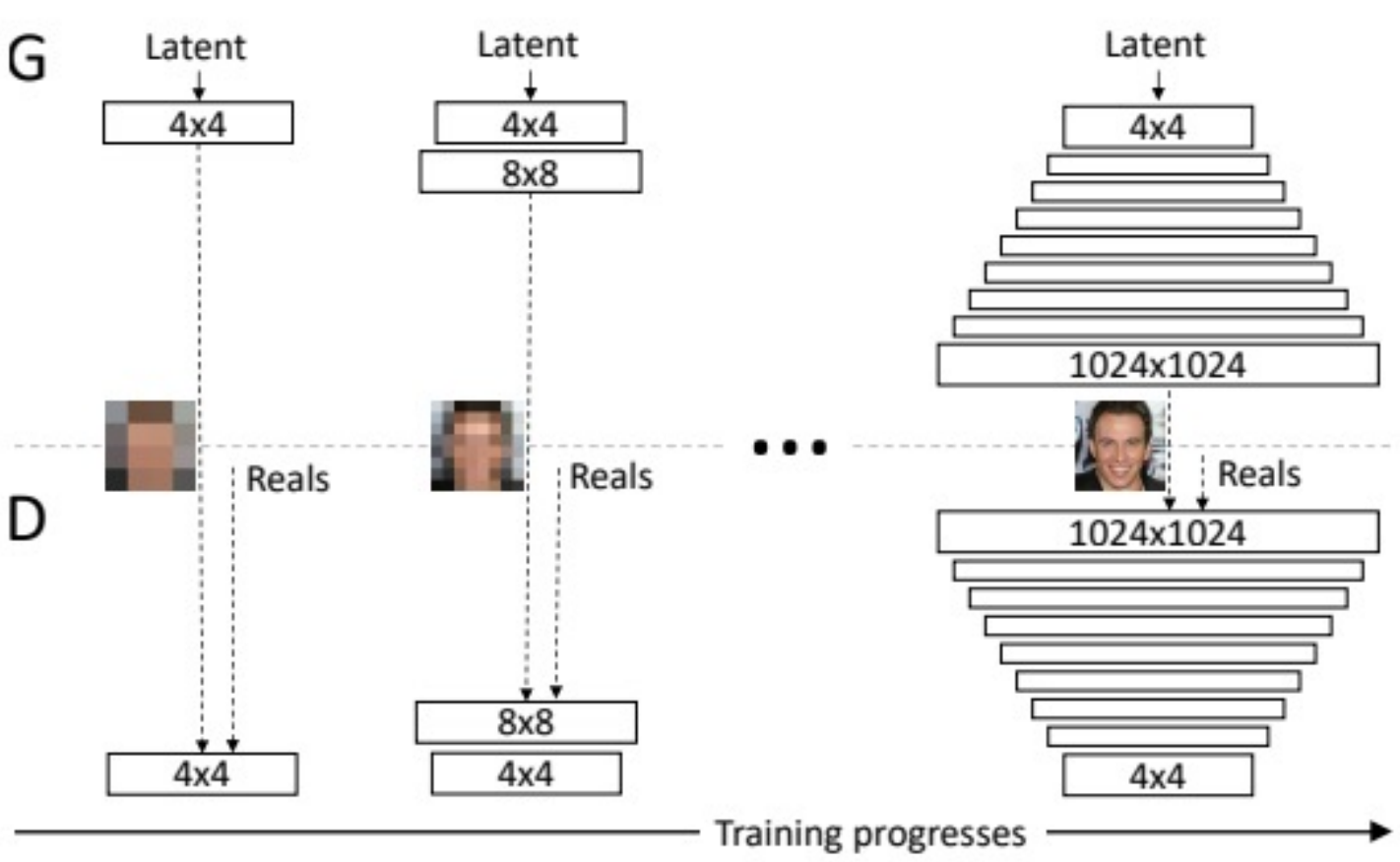
Arjovsky et al., Wasserstein GAN, 2017

GAN Zoo (2017)

https://github.com/hindupuravinash/the-gan-zoo

- 3D-ED-GAN - Shape Inpainting using 3D Generative Adversarial Network and Recurrent Convolutional Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling (github)
- 3D-IWGAN - Improved Adversarial Systems for 3D Object Generation and Reconstruction (github)
- 3D-RecGAN - 3D Object Reconstruction from a Single Depth View with Adversarial Learning (github)
- ABC-GAN - ABC-GAN: Adaptive Blur and Control for improved training stability of Generative Adversarial Networks (github)
- ABC-GAN - GANs for LIFE: Generative Adversarial Networks for Likelihood Free Inference
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- ACGAN - Coverless Information Hiding Based on Generative adversarial networks
- ACtuAL - ACtuAL: Actor-Critic Under Adversarial Learning
- AdaGAN - AdaGAN: Boosting Generative Models
- AdvGAN - Generating adversarial examples with adversarial networks
- AE-GAN - AE-GAN: adversarial eliminating with GAN
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AF-DCGAN - AF-DCGAN: Amplitude Feature Deep Convolutional GAN for Fingerprint Construction in Indoor Localizatic System
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference (github)
- AlignGAN - AlignGAN: Learning to Align Cross-Domain Images with Conditional Generative Adversarial Networks
- AM-GAN - Activation Maximization Generative Adversarial Nets
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- APE-GAN - APE-GAN: Adversarial Perturbation Elimination with GAN
- ARAE - Adversarially Regularized Autoencoders for Generating Discrete Structures (github)
- ARDA - Adversarial Representation Learning for Domain Adaptation
- ARIGAN - ARIGAN: Synthetic Arabidopsis Plants using Generative Adversarial Network
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- ATA-GAN - Attention-Aware Generative Adversarial Networks (ATA-GANs)
- Attention-GAN - Attention-GAN for Object Transfiguration in Wild Images
- AttGAN - Arbitrary Facial Attribute Editing: Only Change What You Want
- UGAN - Enhancing Underwater Imagery using Generative Adversarial Networks
- Unim2im - Unsupervised Image-to-Image Translation with Generative Adversarial Networks (github)
- UNIT - Unsupervised Image-to-image Translation Networks (github)
- Unrolled GAN - Unrolled Generative Adversarial Networks (github)
- UV-GAN - UV-GAN: Adversarial Facial UV Map Completion for Pose-invariant Face Recognition
- VAE-GAN - Autoencoding beyond pixels using a learned similarity metric
- VariGAN - Multi-View Image Generation from a Single-View
- VAW-GAN - Voice Conversion from Unaligned Corpora using Variational Autoencoding Wasserstein Generative Adversarial Networks
- VEEGAN - VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning (github)
- VGAN - Generating Videos with Scene Dynamics (github)
- VGAN - Generative Adversarial Networks as Variational Training of Energy Based Models (github)
- VGAN - Text Generation Based on Generative Adversarial Nets with Latent Variable
- ViGAN - Image Generation and Editing with Variational Info Generative Adversarial Networks
- VIGAN - VIGAN: Missing View Imputation with Generative Adversarial Networks
- VoiceGAN - Voice Impersonation using Generative Adversarial Networks
- VOS-GAN - VOS-GAN: Adversarial Learning of Visual-Temporal Dynamics for Unsupervised Dense Prediction in Videos
- VRAL - Variance Regularizing Adversarial Learning
- WaterGAN - WaterGAN: Unsupervised Generative Network to Enable Real-time Color Correction of Monocular Underwater Images
- WaveGAN - Synthesizing Audio with Generative Adversarial Networks
- weGAN - Generative Adversarial Nets for Multiple Text Corpora
- WGAN - Wasserstein GAN (github)
- WGAN-GP - Improved Training of Wasserstein GANs (github)
- WS-GAN - Weakly Supervised Generative Adversarial Networks for 3D Reconstruction
- XGAN - XGAN: Unsupervised Image-to-Image Translation for many-to-many Mappings
- ZipNet-GAN - ZipNet-GAN: Inferring Fine-grained Mobile Traffic Patterns via a Generative Adversarial Neural Network
- α-GAN - Variational Approaches for Auto-Encoding Generative Adversarial Networks (github)
- β-GAN - Annealed Generative Adversarial Networks
- Δ-GAN - Triangle Generative Adversarial Networks

Progressive growing of GANs



Karras et al., ICLR 2018

BigGAN

Brocks et al., Large Scale GAN Training for High Fidelity Natural Image Synthesis, ICLR 2019

Evaluation

- Inception Score (IS)
  - based on the output of pretrained InceptionV3 on generated samples
  - It composed of an entropy term and a class coverage term
    - Entropy term shows to what extent the classification model confidently predicts a single label for each image
    - predictions of the classification model on generated images are evenly distributed over all labels

- Frechet Inception Distance (FID)
  - Finds the Wasserstein distance of the distribution of generated images and the distribution of real images (in the representation space like deepest layer of Inception v3)

GANs

- Don’t work with an explicit density function
- Take game-theoretic approach: learn to generate from training distribution through 2-player game

- Pros:
  - Beautiful, state-of-the-art samples!
- Cons:
  - Trickier / more unstable to train
  - Can’t solve inference queries such as p(x), p(z|x)
- Active areas of research:
  - Better loss functions, more stable training (Wasserstein GAN, LSGAN, many others)
  - Conditional GANs, GANs for all kinds of applications
  - The quality of generated samples is still a challenge
  - Controlling the diversity of the generated samples is difficult

Generative Models: Summary