Diffusion Models and Scientific Generative AI

—— Lecture 3: Foudations of Generative Models (2)

Professor Siyu Zhu
AI3 institute, Fudan University

Contents

- **Last Class Review**
- Generative Model
  - Autoregressive Model (PixelRNN/PixelCNN)
  - Variational Autoencoder (VAEs)
  - Generative Adversarial Networks (GAN)

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a function to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden structure of the data

Examples: Clustering, dimensionality reduction, density estimation, etc.

Self-Supervised Learning

Data: (x, pseudo generated y)
No manual labels!

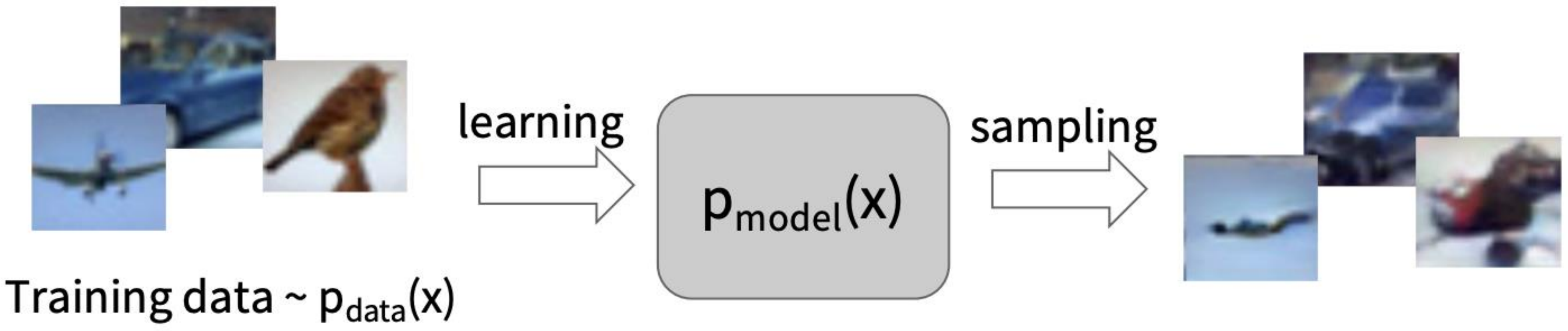
Goal: Learn to generate good features (reduce the data to useful/generic features)

Example: Classification in downstream applications where we have limited data

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Generative Model

- **Given training data, generate new samples from same distribution.**

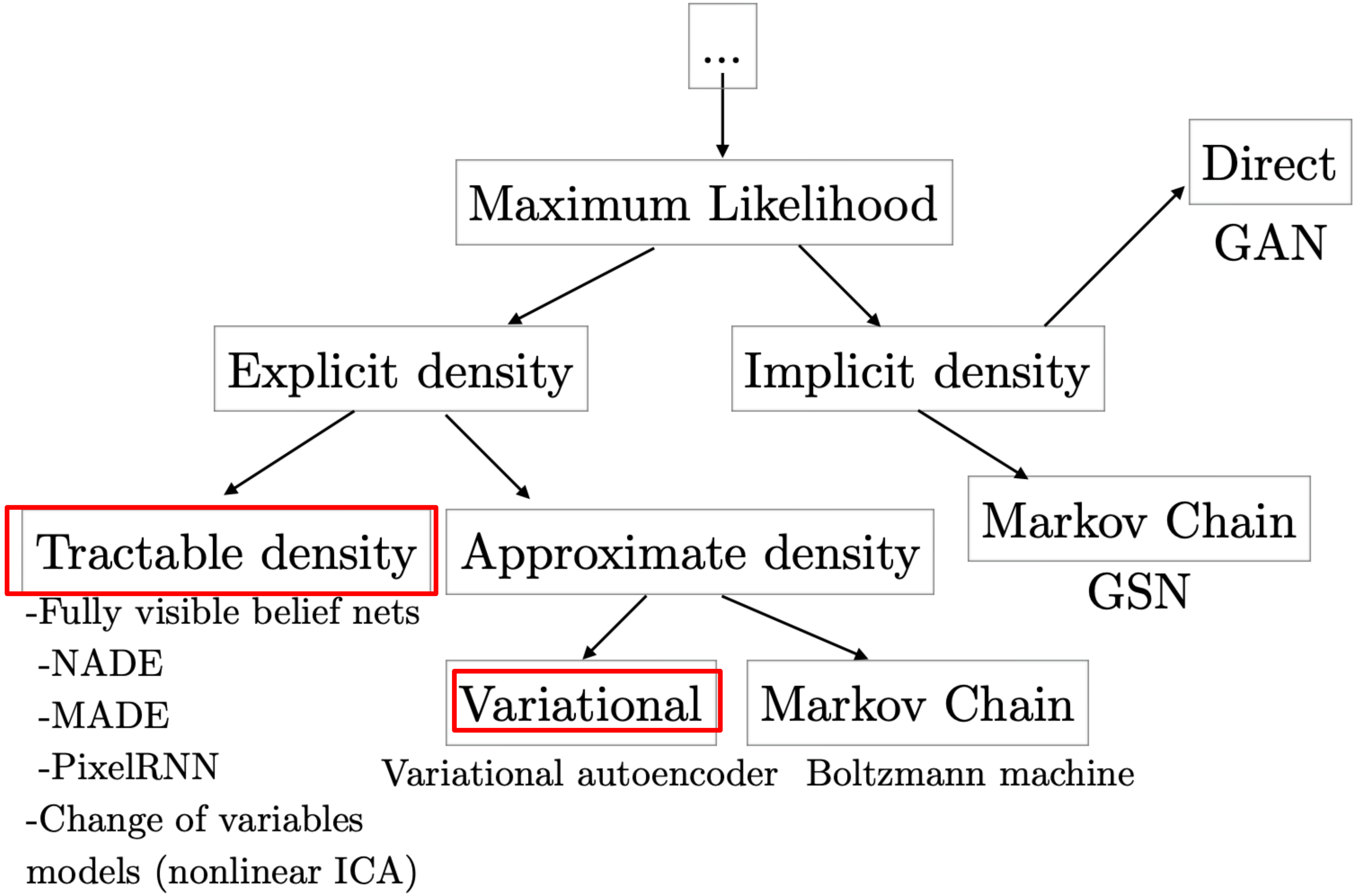


Formulate as density estimation problems:

- Explicit density estimation: explicitly define and solve for $p_{model}(x)$
- Implicit density estimation: learn model that can sample from $p_{model}(x)$ without explicitly defining it.

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

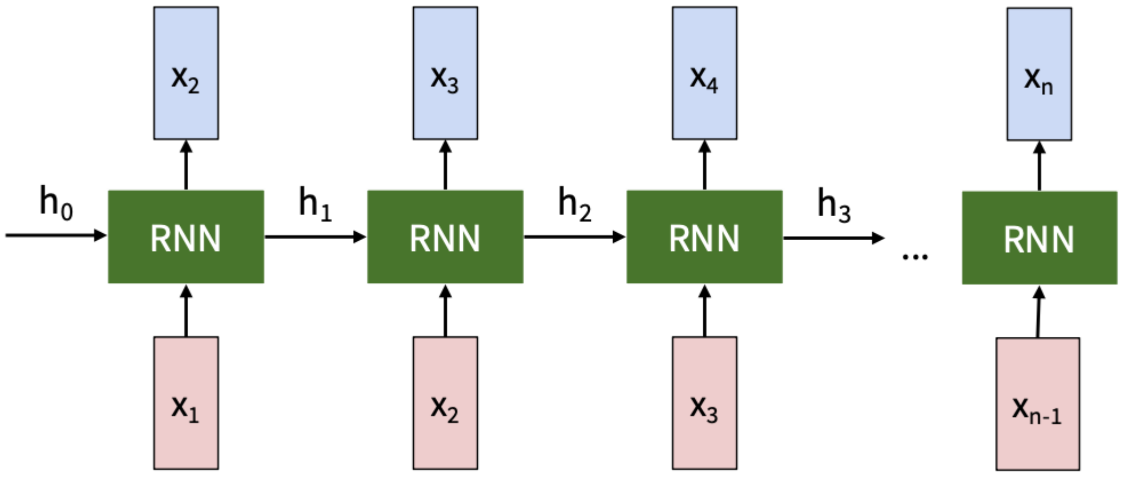
Taxonomy of Generative Models



slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

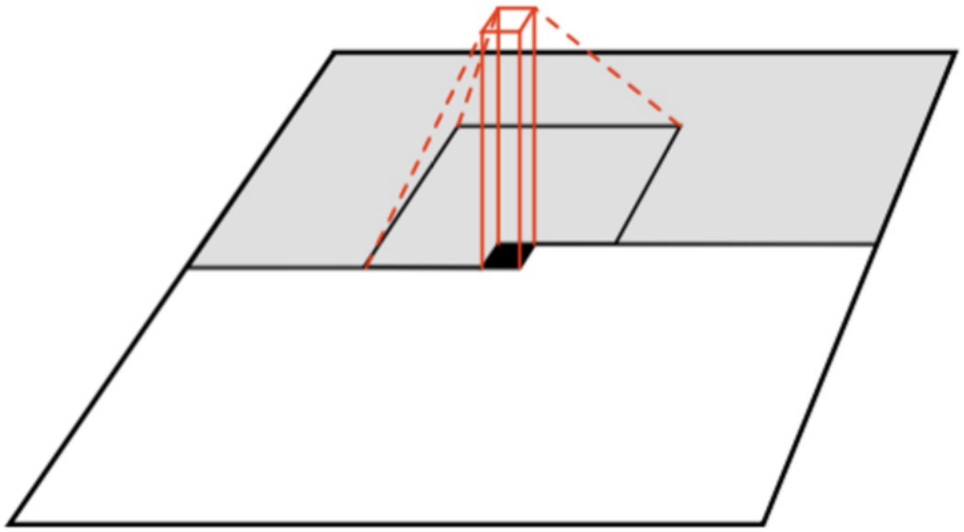
Autoregressive Models

- **PixelRNN/ PixelCNN [van der Oord et al. 2016]**



$$p(x_i|x_1, ..., x_{i-1})$$

PixelRNN

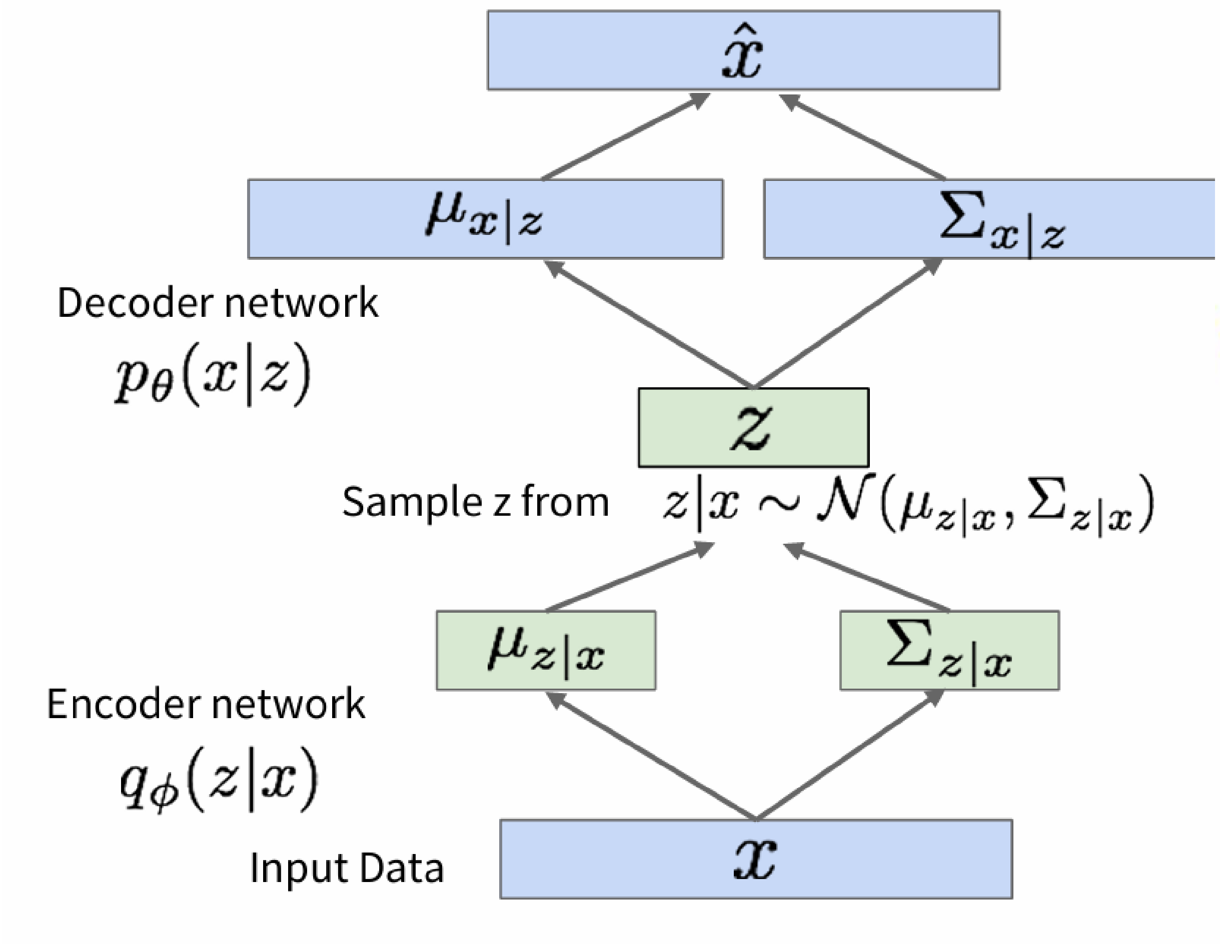


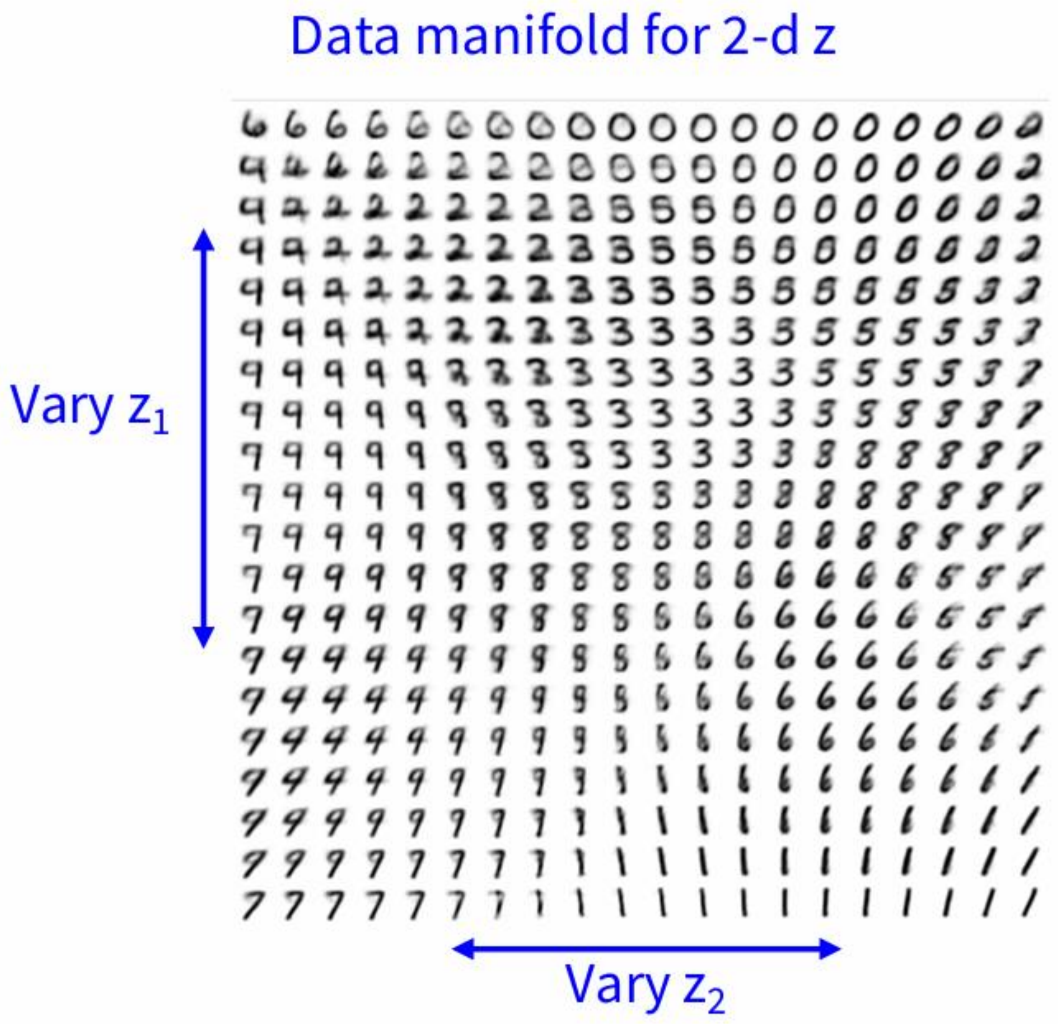
PixelCNN

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Variational Autoencoder (VAE)

- **VAE [Kingma et al. 2014]**

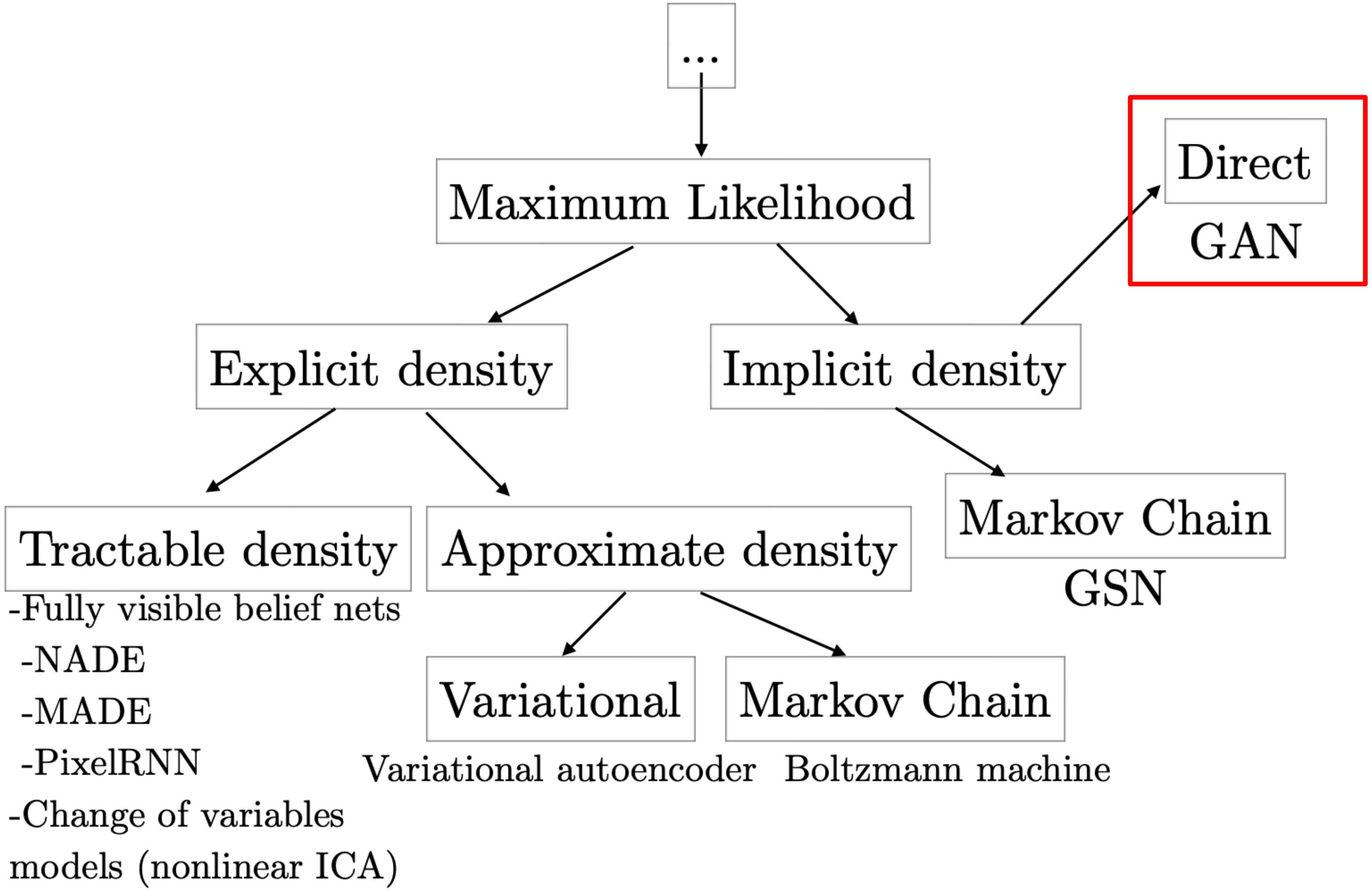




Contents

- Last Class Review
- Generative Model
  - Autoregressive Model (PixelRNN/PixelCNN)
  - Variational Autoencoder (VAEs)
  - Generative Adversarial Networks (GAN)

Taxonomy of Generative Models



slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Generative Adversarial Networks

PixelRNN/CNNs 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

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Generative Adversarial Networks

PixelRNN/CNNs define tractable density function, optimize likelihood of training data:

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

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?

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Generative Adversarial Networks

PixelRNN/CNNs 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: not modeling any explicit density function!

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Generative Adversarial Networks

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

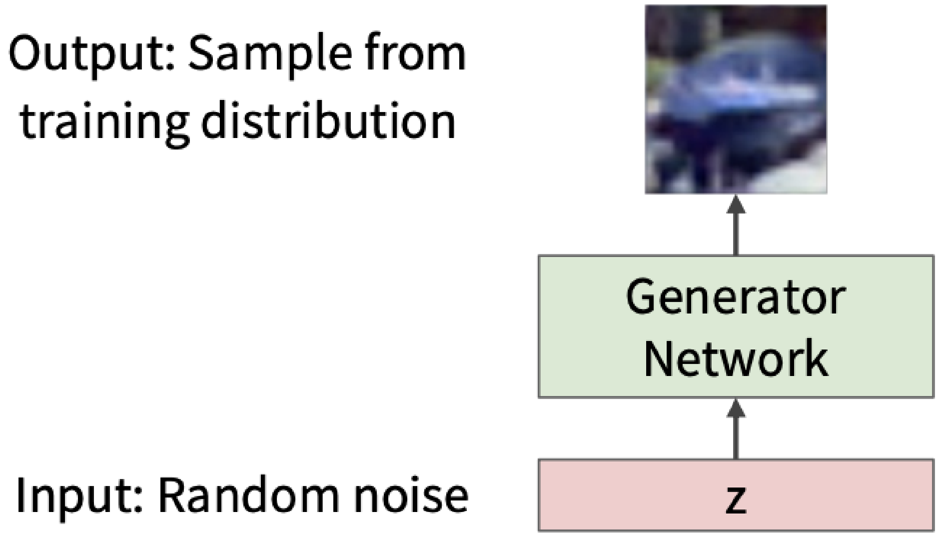
Solution: Sample from a simple distribution we can easily sample from, e.g. random noise. Learn transformation to training distribution.

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

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 we can easily sample from, e.g. random noise. Learn transformation to training distribution.



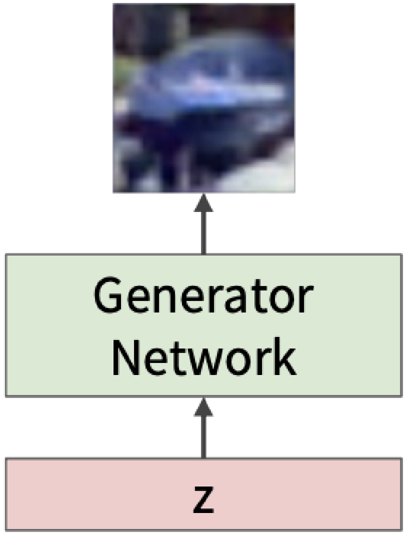
Generative Adversarial Networks

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

Solution: Sample from a simple distribution we can easily sample from, e.g. random noise. Learn transformation to training distribution.

But we don’t know which sample z maps to which training image -> can’t learn by reconstructing training images

Output: Sample from training distribution



Input: Random noise

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

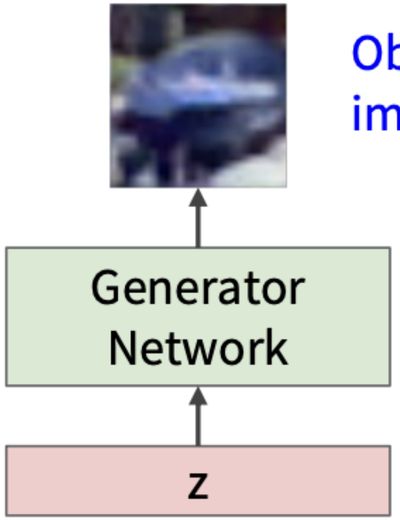
Generative Adversarial Networks

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

Solution: Sample from a simple distribution we can easily sample from, e.g. random noise. Learn transformation to training distribution.

But we don't know which sample z maps to which training image -> can't learn by reconstructing training images

Output: Sample from training distribution



Objective: generated images should look "real"

Input: Random noise

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Generative Adversarial Networks

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

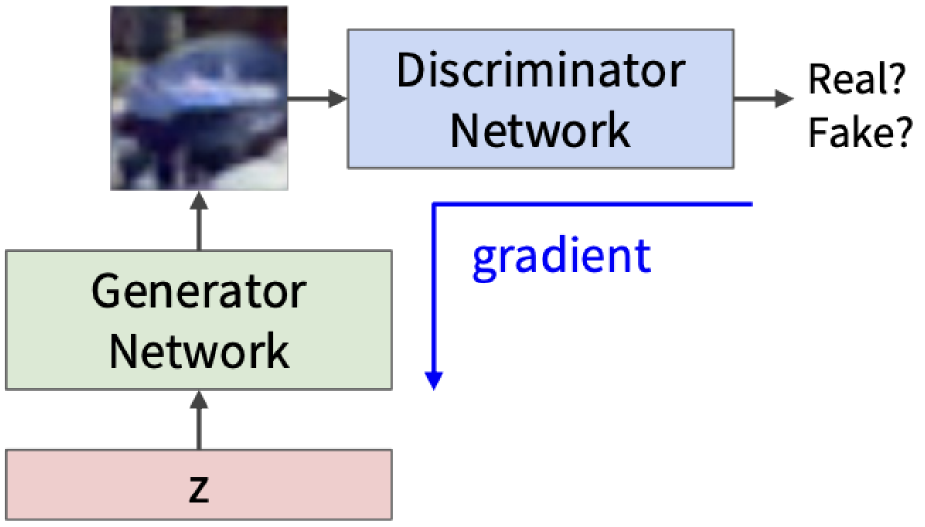
Solution: Sample from a simple distribution we can easily sample from, e.g. random noise. Learn transformation to training distribution.

But we don’t know which sample z maps to which training image -> can’t learn by reconstructing training images

Solution: Use a discriminator network to tell whether the generate image is within data distribution (“real”) or not

Output: Sample from training distribution

Input: Random noise



slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

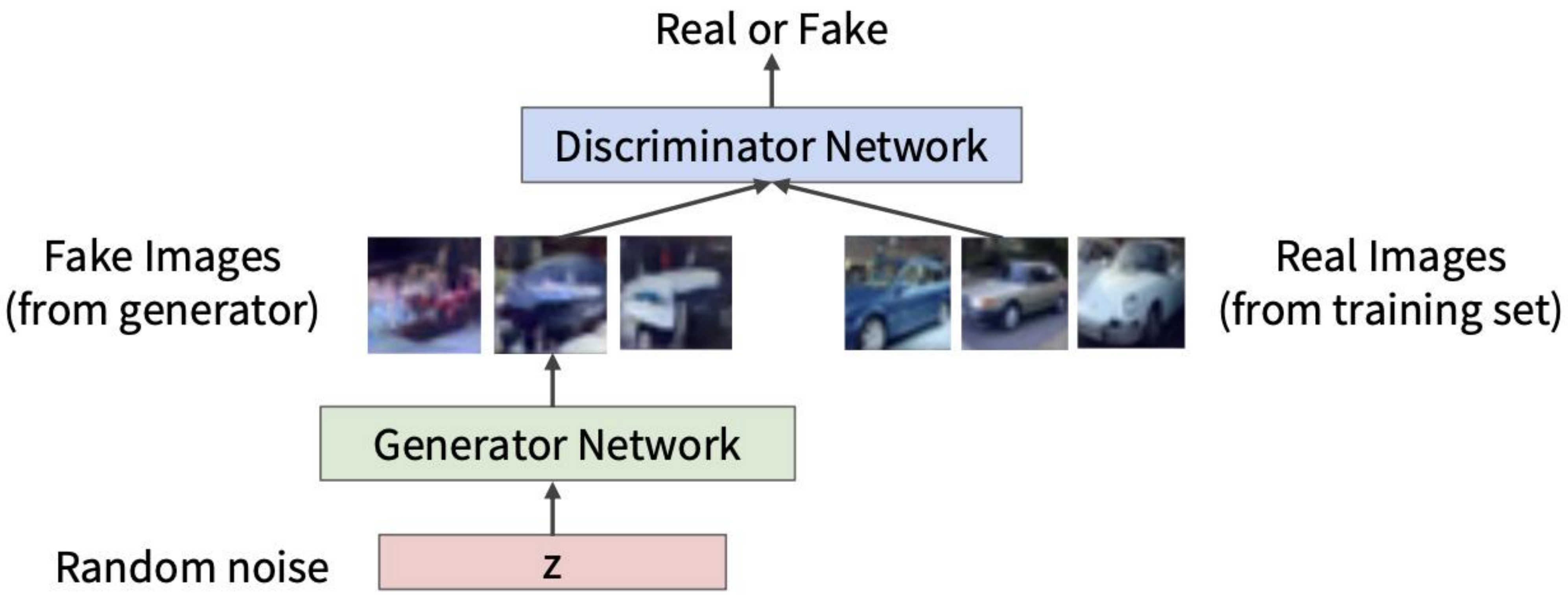
Training GANs: Two-player game

- **Optimization Goal**

Discriminator network: try to distinguish between real and fake images
Generator network: try to fool the discriminator by generating real-looking images

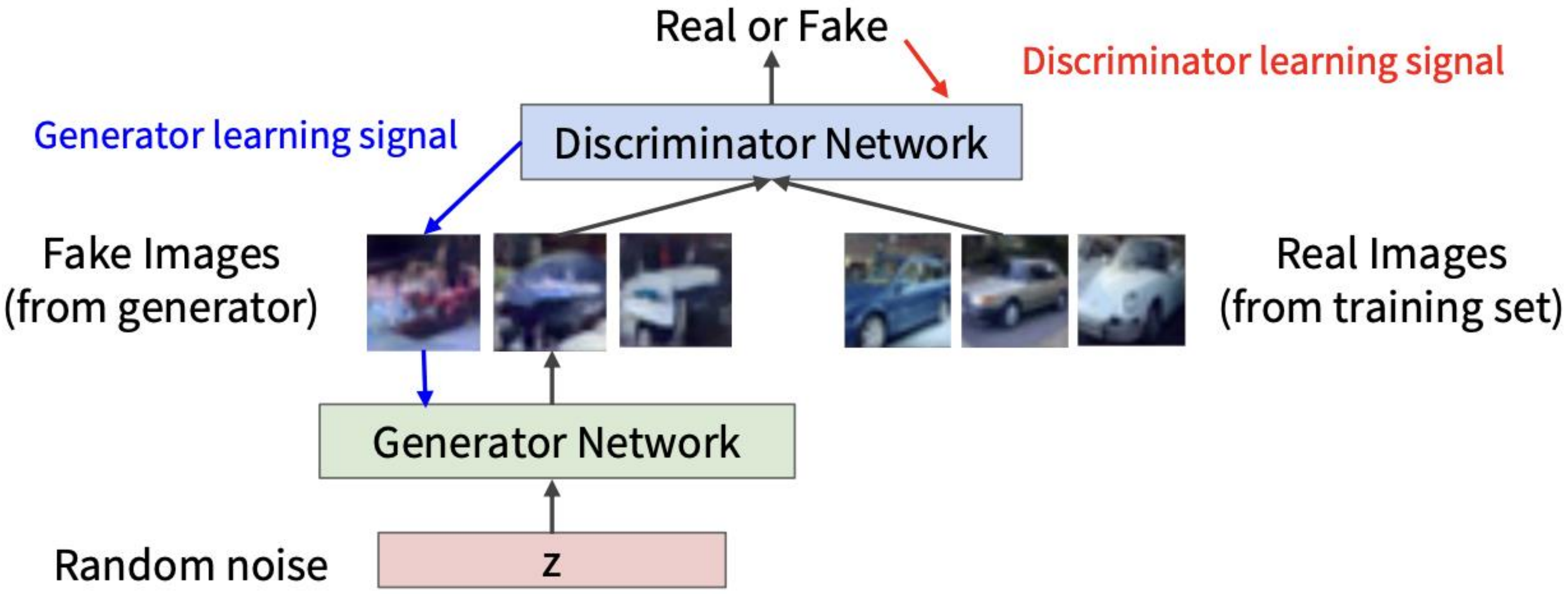
Training GANs: Two-player game

Discriminator network: try to distinguish between real and fake images
Generator network: try to fool the discriminator by generating real-looking images



Training GANs: Two-player game

Discriminator network: try to distinguish between real and fake images
Generator network: try to fool the discriminator by generating real-looking images



slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Training GANs: Two-player game

Discriminator network: try to distinguish between real and fake images
Generator network: try to fool the discriminator by generating real-looking images

Train jointly in minimax game

Minimax objective function:

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

Generator objective

Discriminator objective

Training GANs: Two-player game

Discriminator network: try to distinguish between real and fake images
Generator network: try to fool the discriminator by generating real-looking images

Train jointly in minimax game

Discriminator outputs likelihood in (0,1) of real image

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

- 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)

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

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

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

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

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:

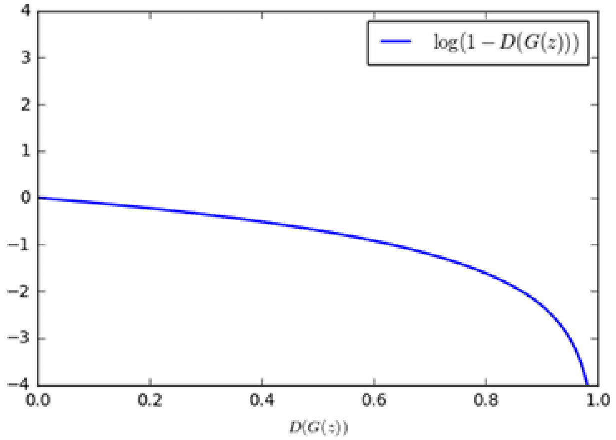
1. Gradient ascent on discriminator

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

2. Gradient descent on generator

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

When sample is likely fake, want to learn from it to improve generator (move to the right on X axis).



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

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

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

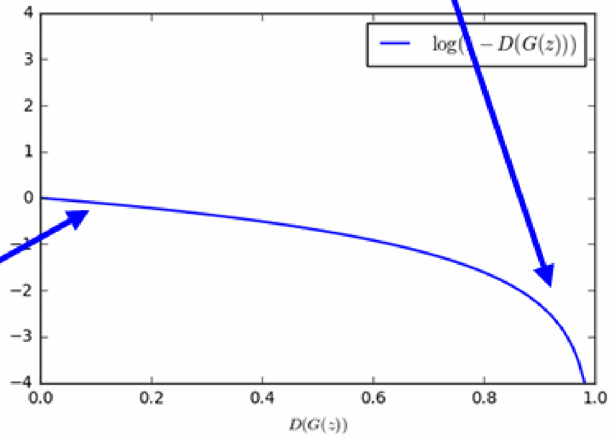
$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

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

When sample is likely fake, want to learn from it to improve generator (move to the right on X axis).

But gradient in this region is relatively flat!

Gradient signal dominated by region where sample is already good



slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

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:

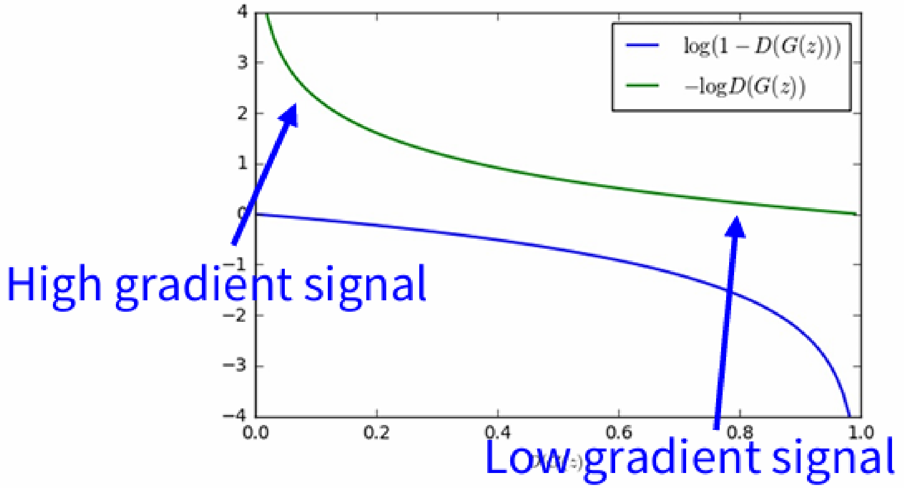
1. Gradient ascent on discriminator

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

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.



slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Training GANs: Two-player game

Putting it together: GAN training algorithm

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

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Training GANs: Two-player game

Putting it together: GAN training algorithm

for number of training iterations **do**

  for $\boxed{k \text{ steps}}$ **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

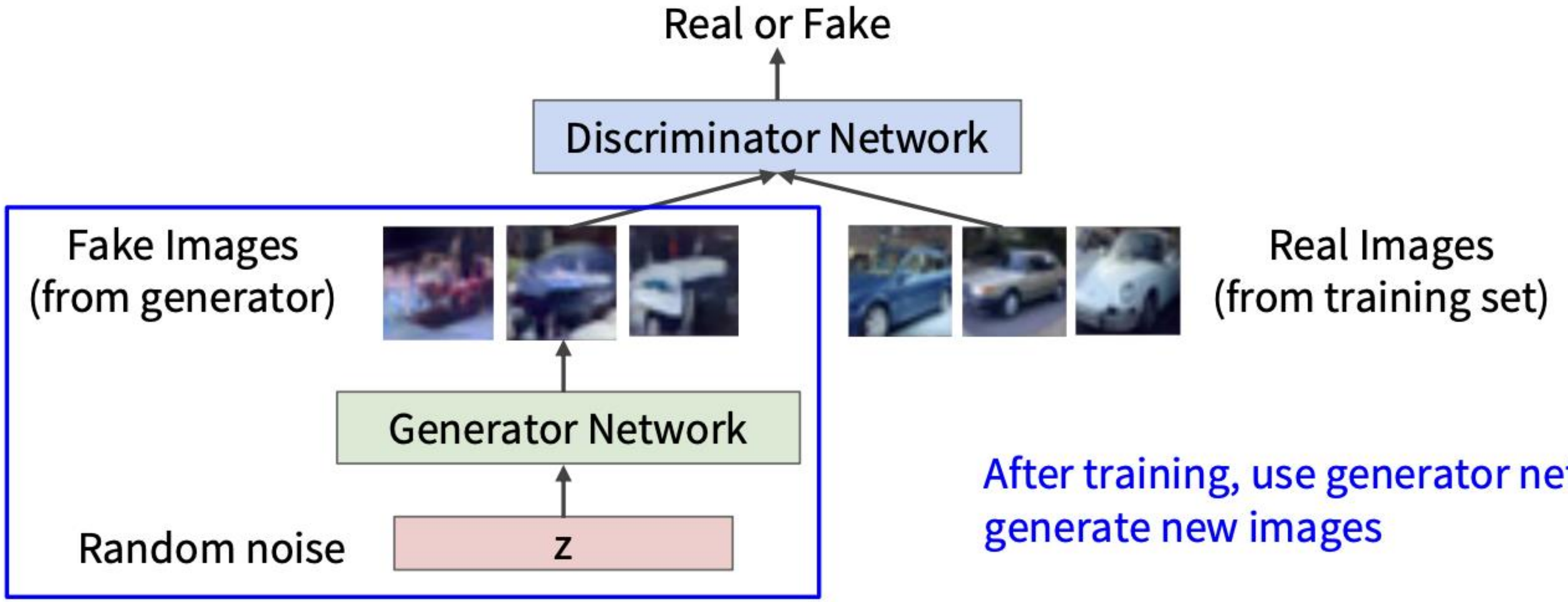
Some find k=1 more stable, others use k > 1, no best rule.

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

Arjovsky et al. "Wasserstein gan." arXiv preprint arXiv:1701.07875 (2017)
Berthelot, et al. "Began: Boundary equilibrium generative adversarial networks." arXiv preprint arXiv:1703.10717 (2017)

Training GANs: Two-player game

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

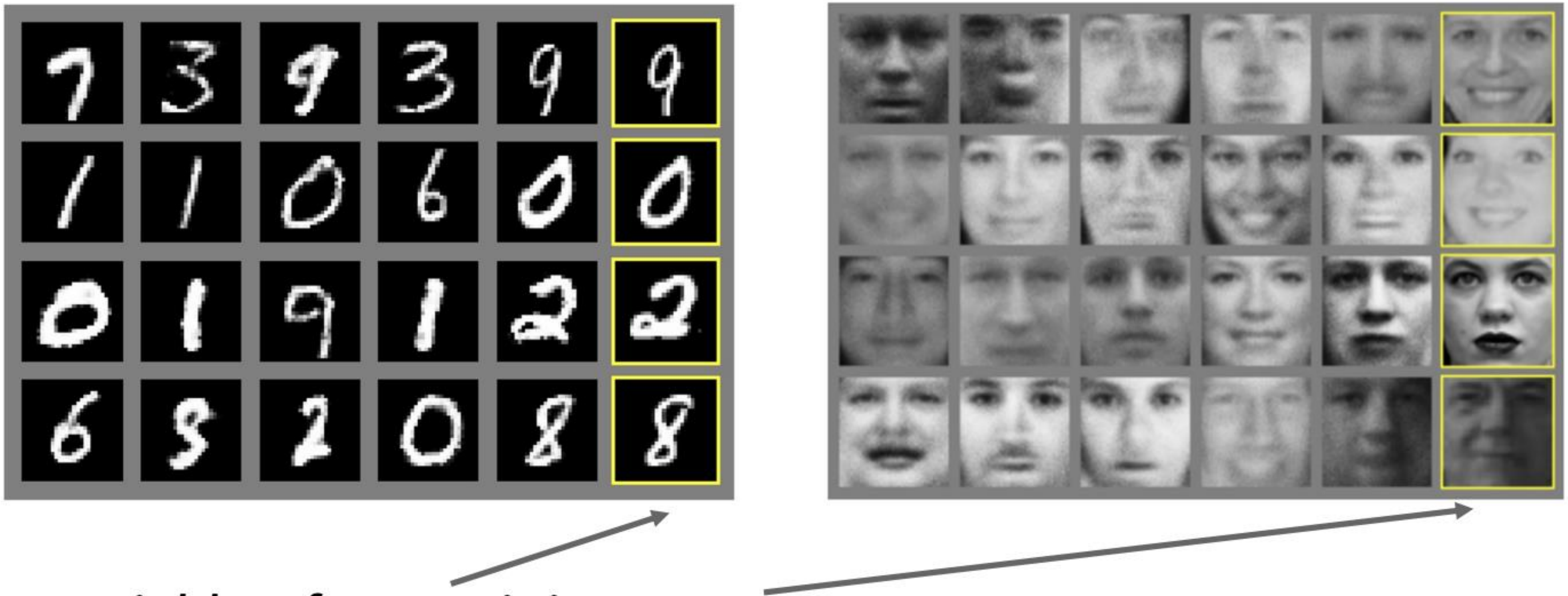


After training, use generator network to generate new images

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Generative Adversarial Nets

Generated samples

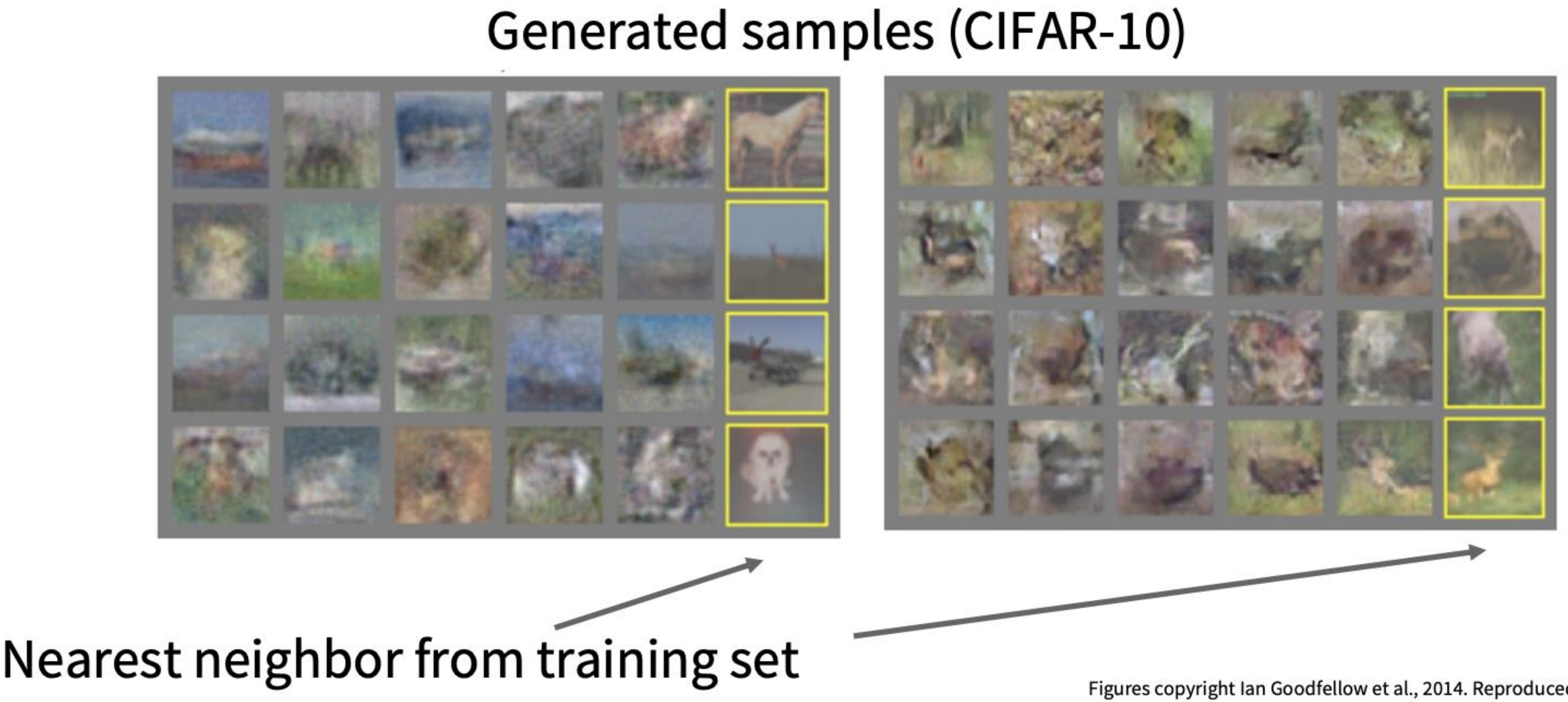


Nearest neighbor from training set

Figures copyright Ian Goodfellow et al., 2014. Reproduced with permission.

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Generative Adversarial Nets



Generative Adversarial Nets: Convolutional Architectures

Generator is an upsampling network with fractionally-strided convolutions
Discriminator is a convolutional network

Architecture guidelines for stable Deep Convolutional GANs

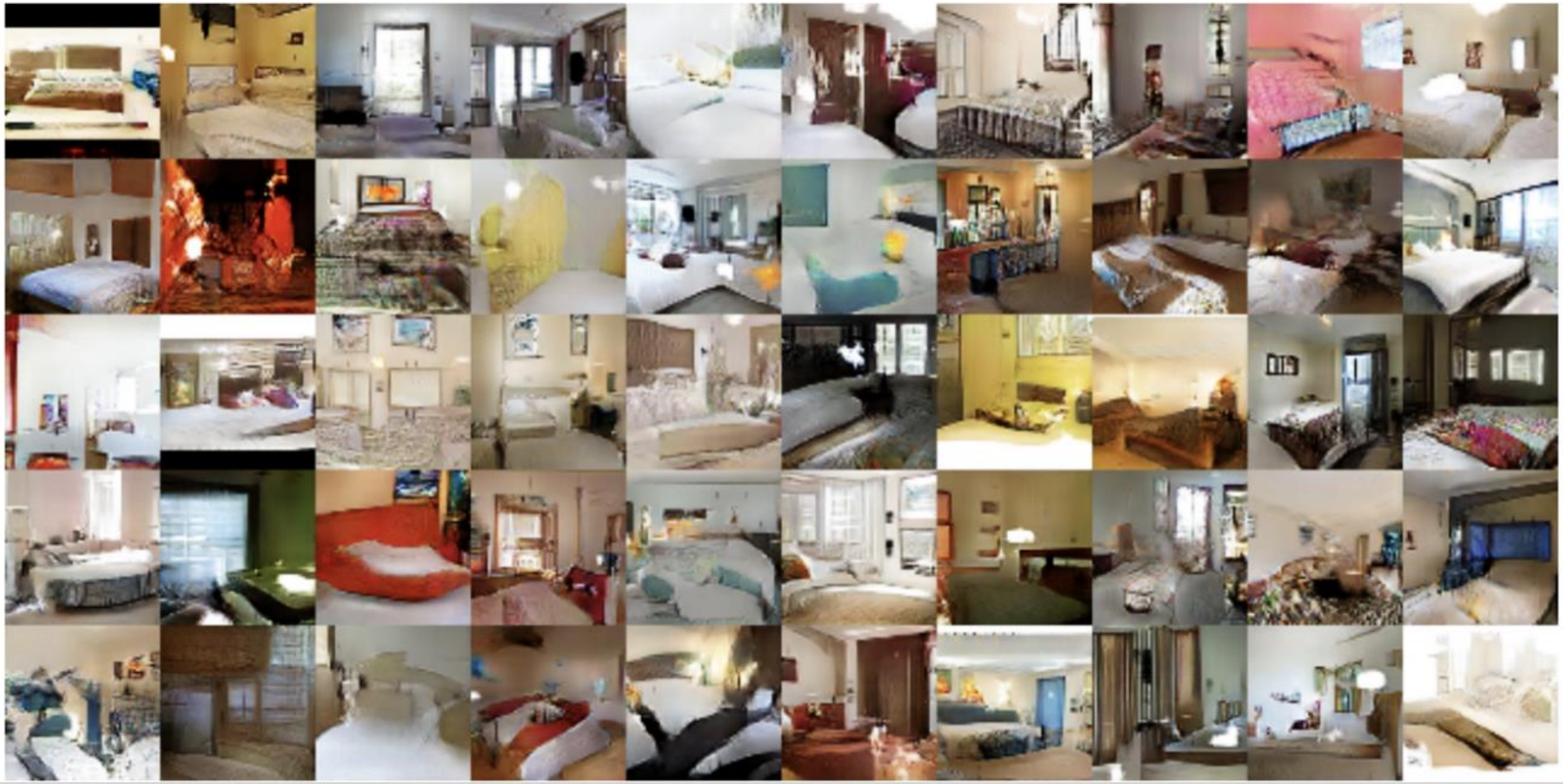
- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except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

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Generative Adversarial Nets: Convolutional Architectures

Samples from the model look much better!

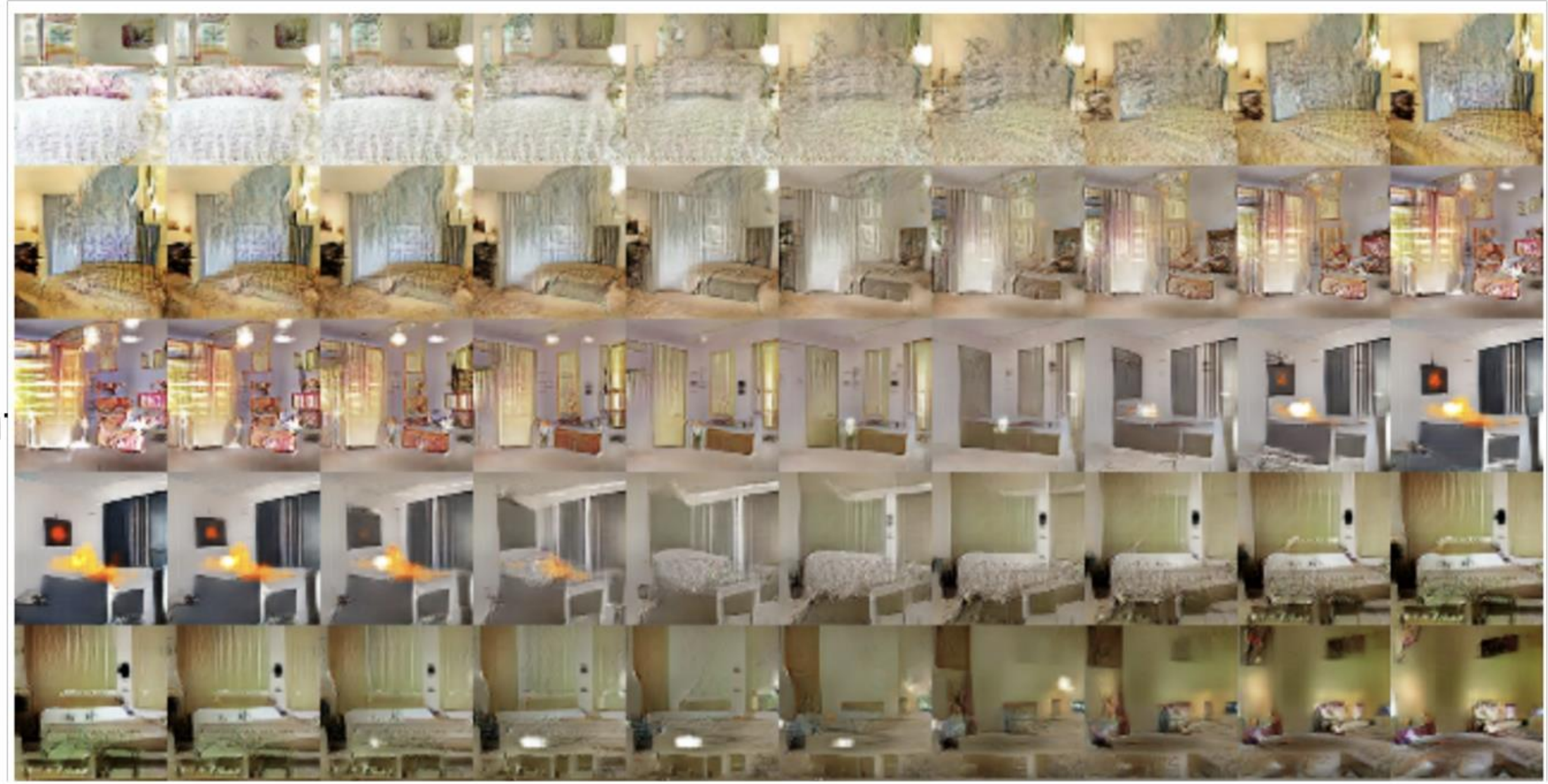


Radford et al, ICLR 2016

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Generative Adversarial Nets: Convolutional Architectures

Interpolating between random points in latent space

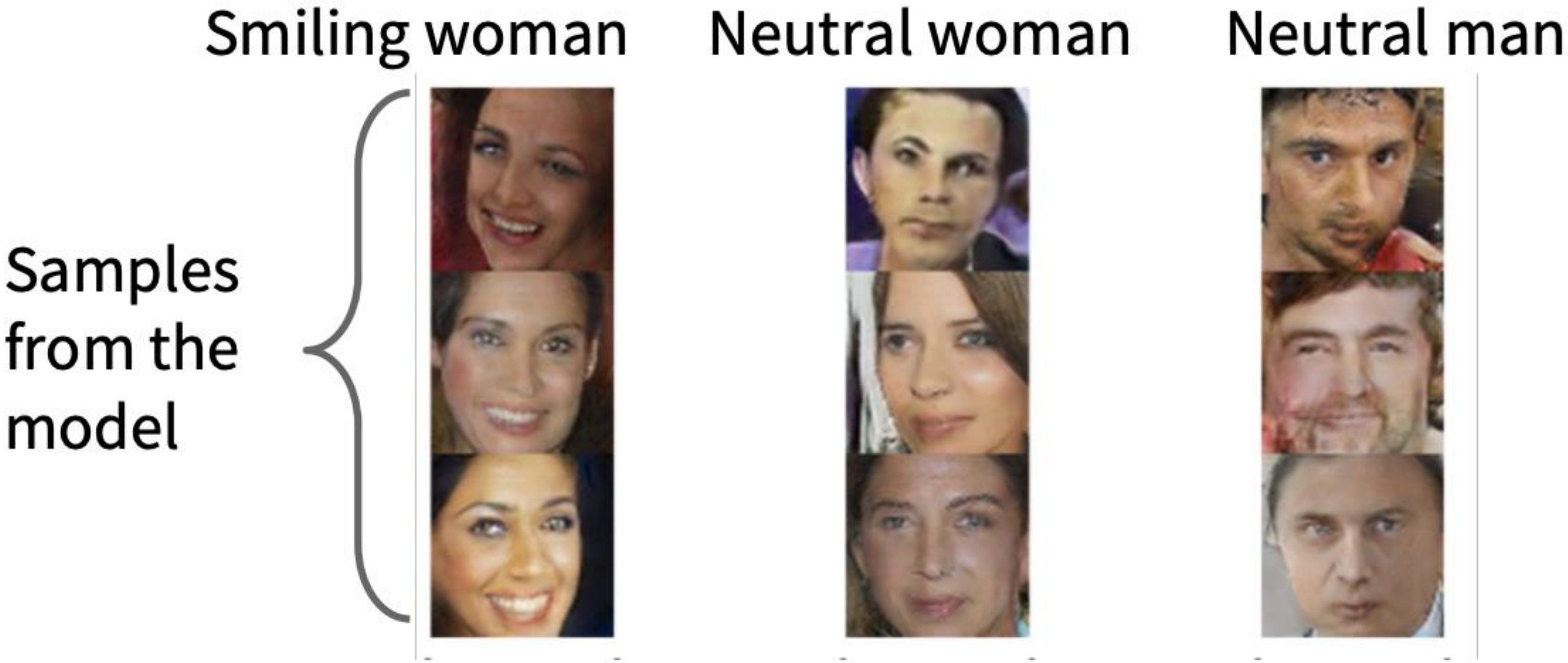


Radford et al, ICLR 2016

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Generative Adversarial Nets: Interpretable Vector Math

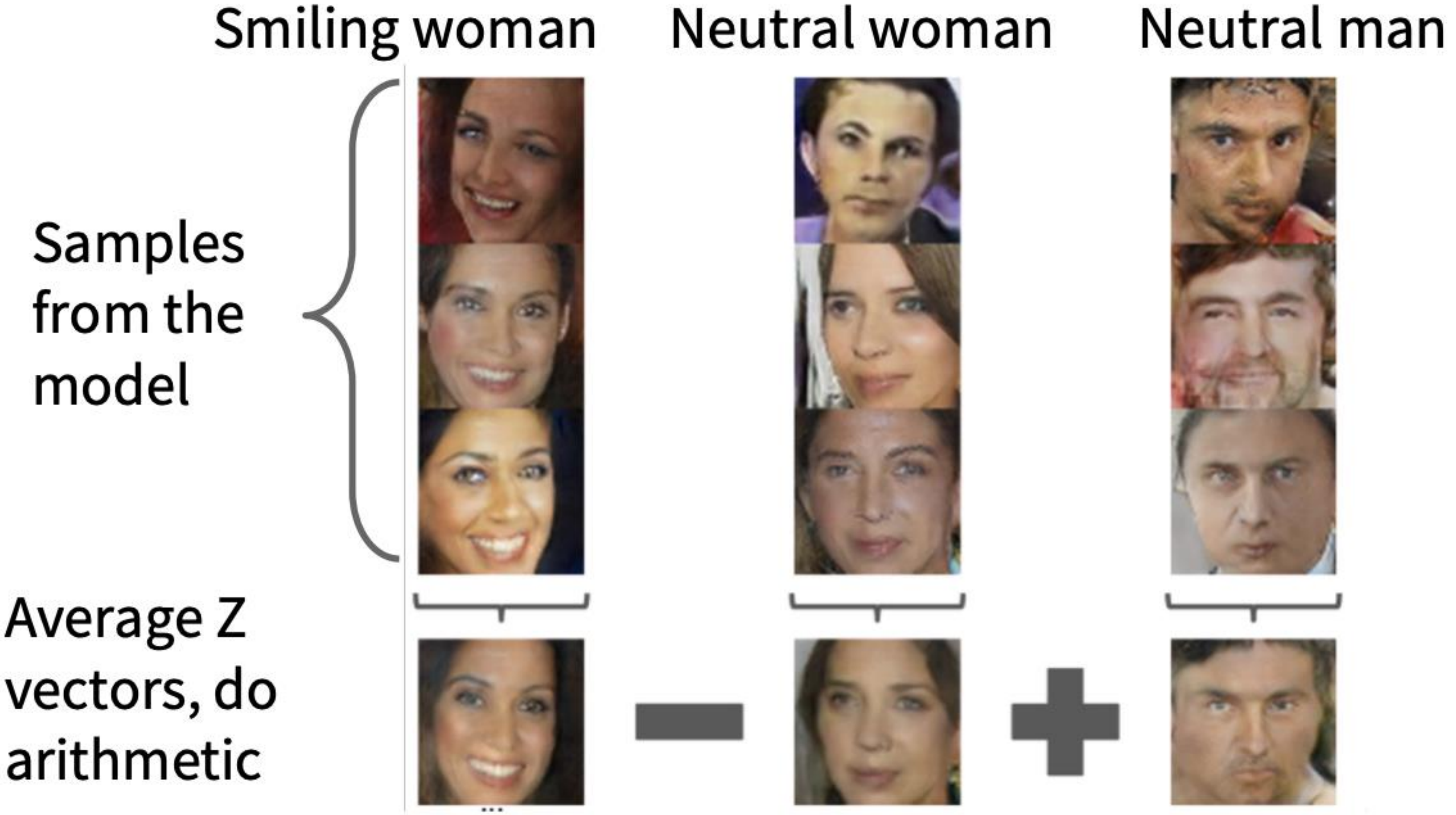
Radford et al, ICLR 2016



slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Generative Adversarial Nets: Interpretable Vector Math

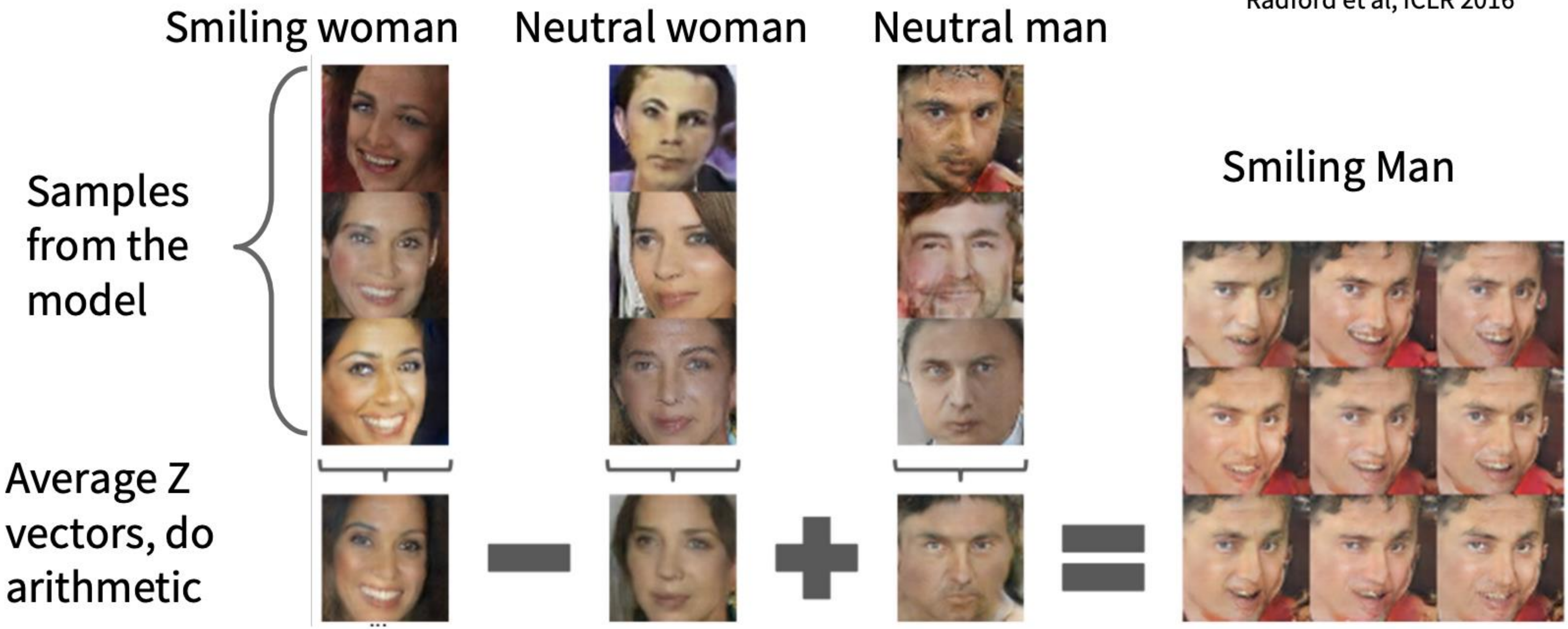
Radford et al, ICLR 2016



slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Generative Adversarial Nets: Interpretable Vector Math

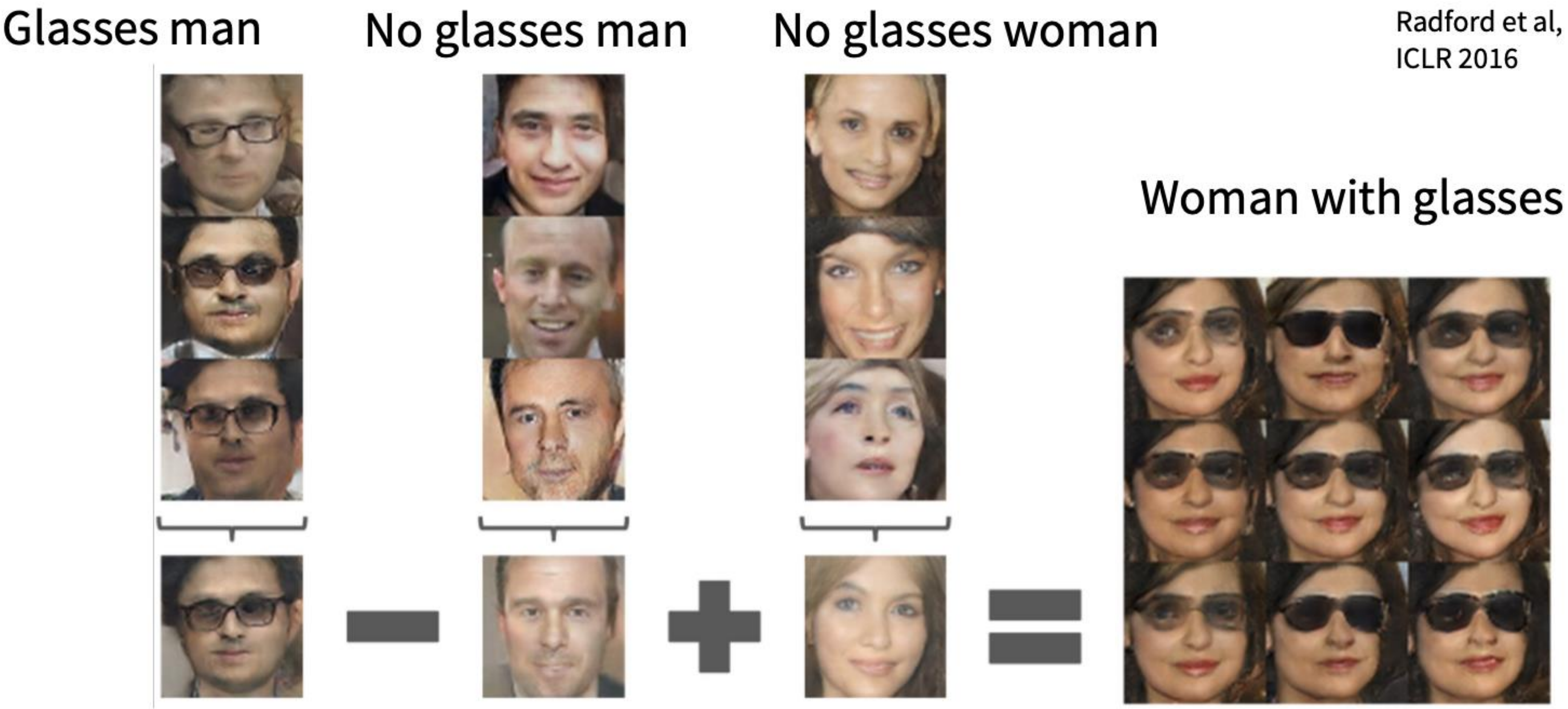
Radford et al, ICLR 2016



slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

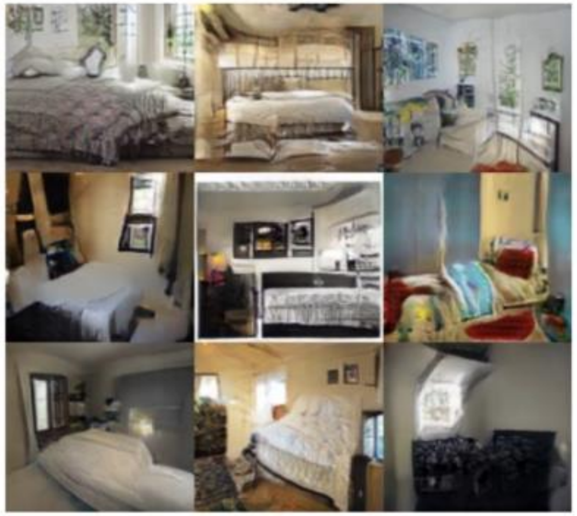
Generative Adversarial Nets: Interpretable Vector Math

Radford et al, ICLR 2016



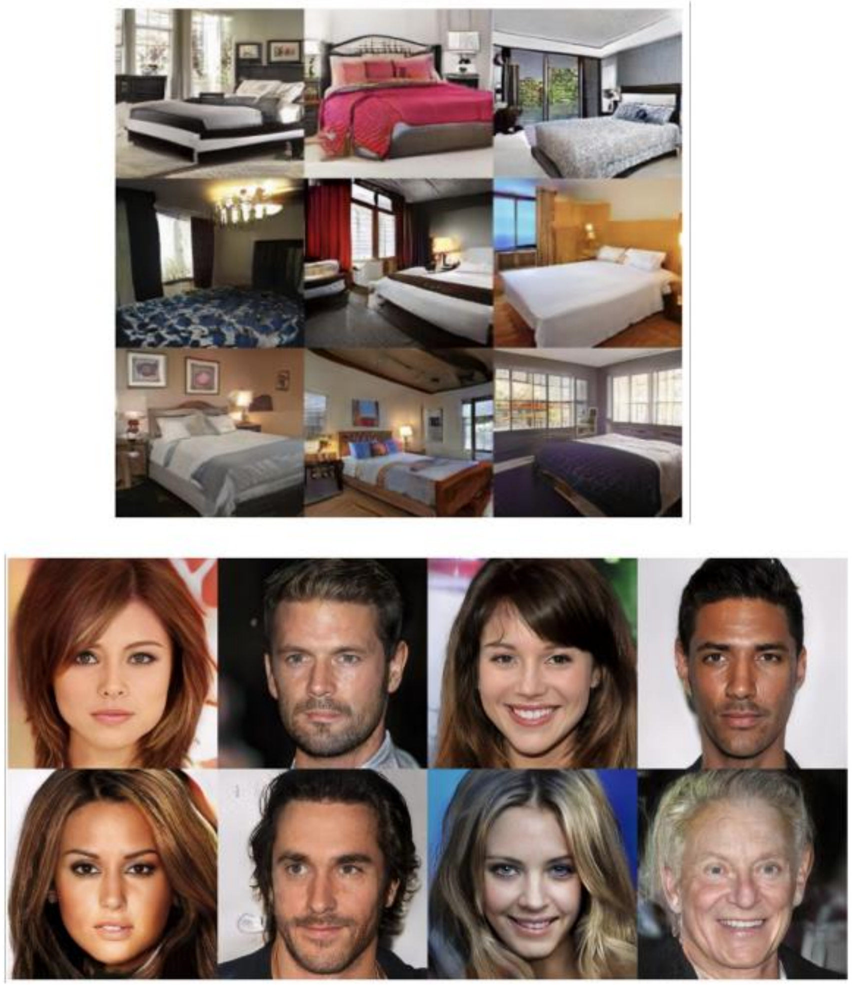
Generative Adversarial Nets: Interpretable Vector Math

Better training and generation



LSGAN, Zhu 2017.

Wasserstein GAN, Arjovsky 2017.
Improved Wasserstein GAN, Gulrajani 2017.

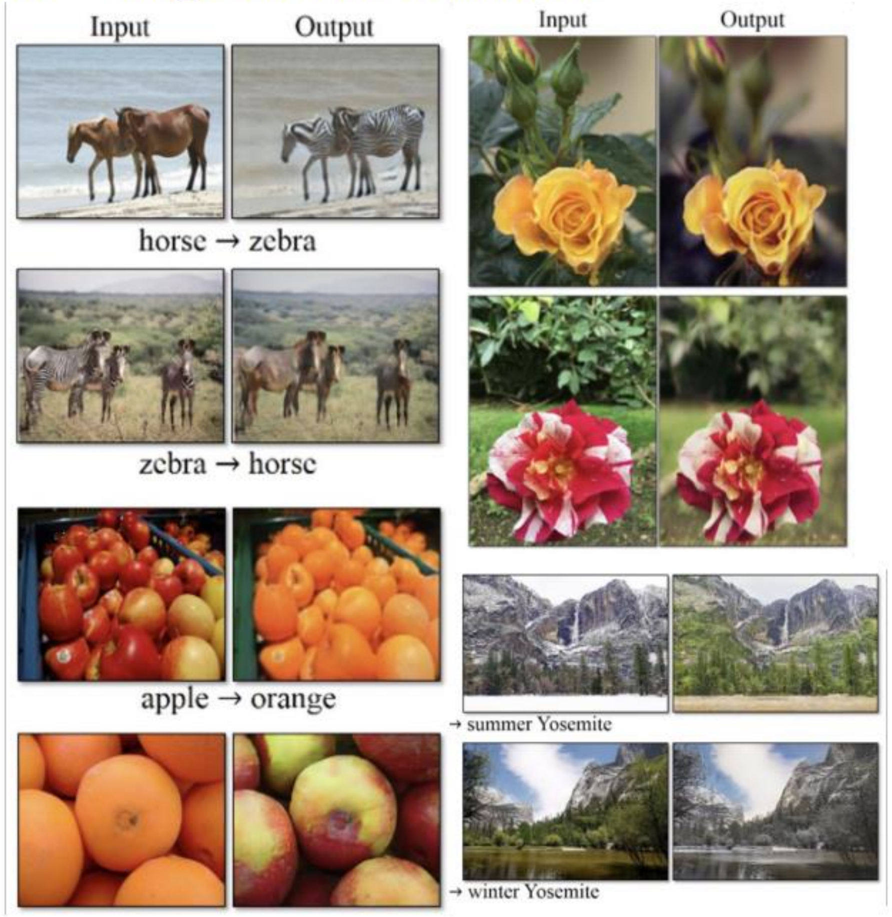


Progressive GAN, Karras 2018.

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

2017: Explosion of GANs

Source->Target domain transfer



CycleGAN. Zhu et al. 2017.

Text -> Image Synthesis

Reed et al. 2017.

Many GAN applications

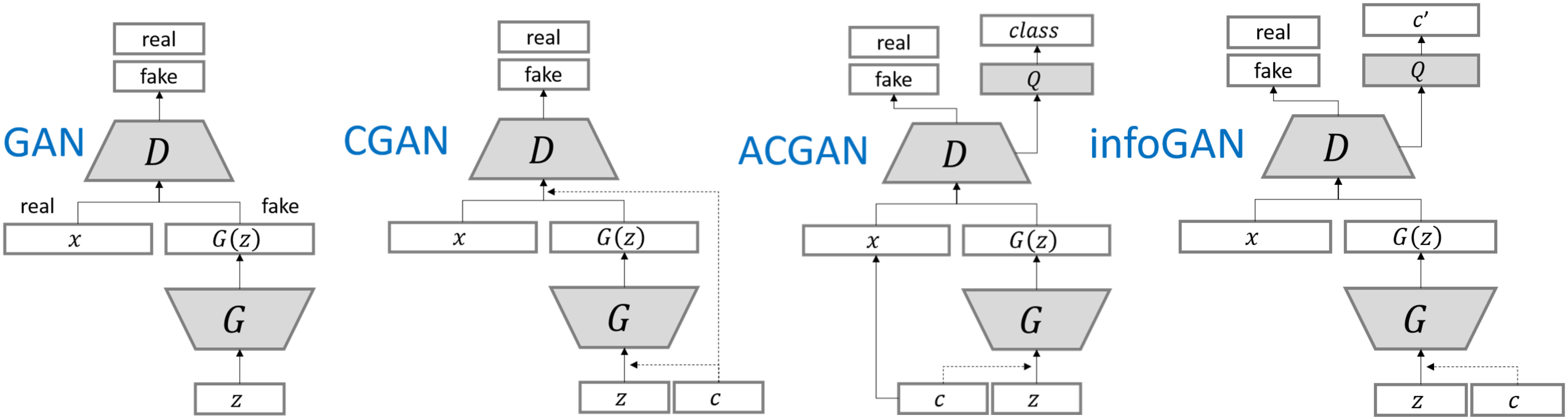
Pix2pix. Isola 2017. Many examples at https://phillipi.github.io/pix2pix/

2019: BigGAN

Brock et al., 2019

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Generative Adversarial Nets



GAN Goodfellow et al. NIPS 2014	Conditional GAN Mehdi et al. ArXiv. 1411.1784 2014	AC-GAN Odena et al. ICML 2017	InfoGAN Chen et al. NIPS 2016

VQ-GAN

Problems:

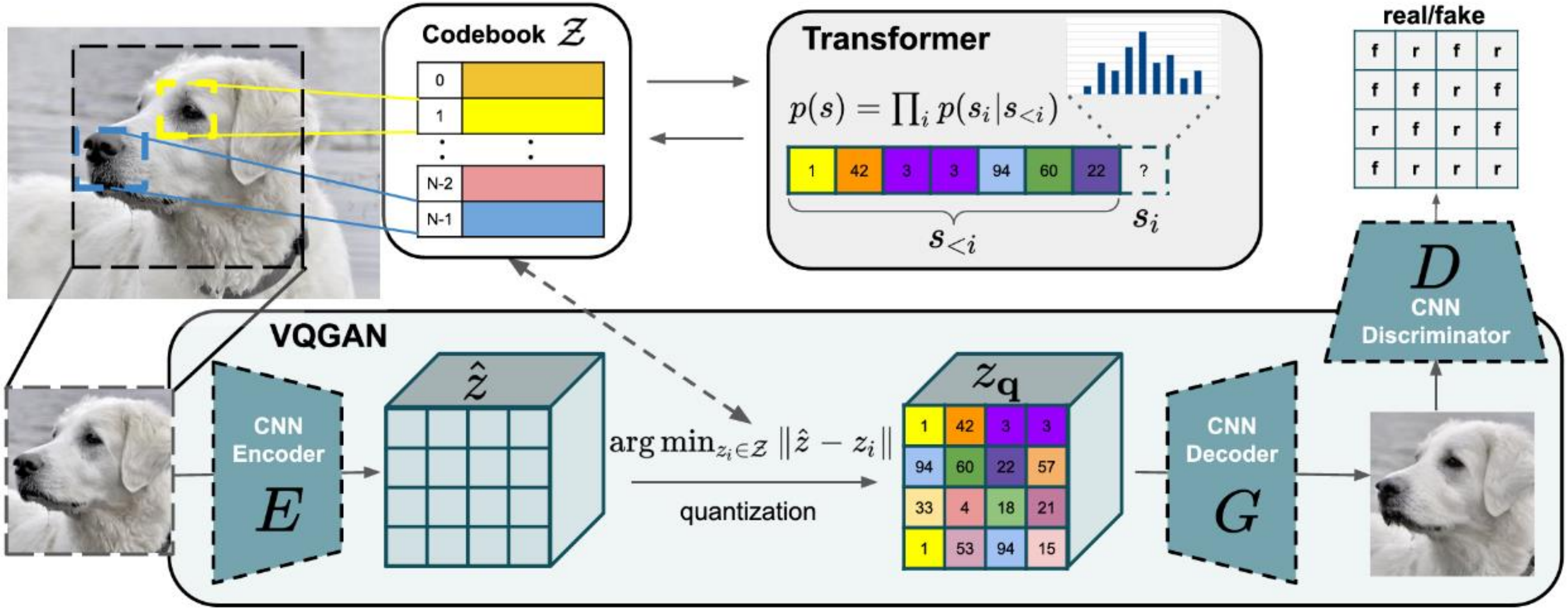
- Insufficient fine details in reconstructions & low resolution
- Restricted long-term dependencies

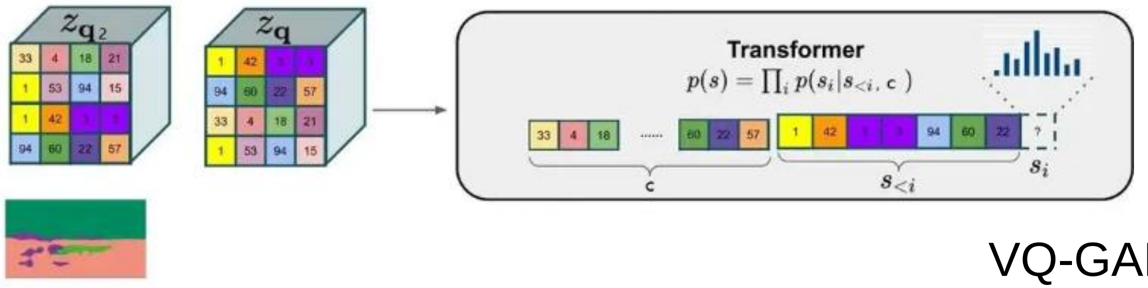
Solution:

- Perceptual loss & generative adversarial nets
- Transformer in discrete latent space

Generations from VQ-VAE

VQ-GAN

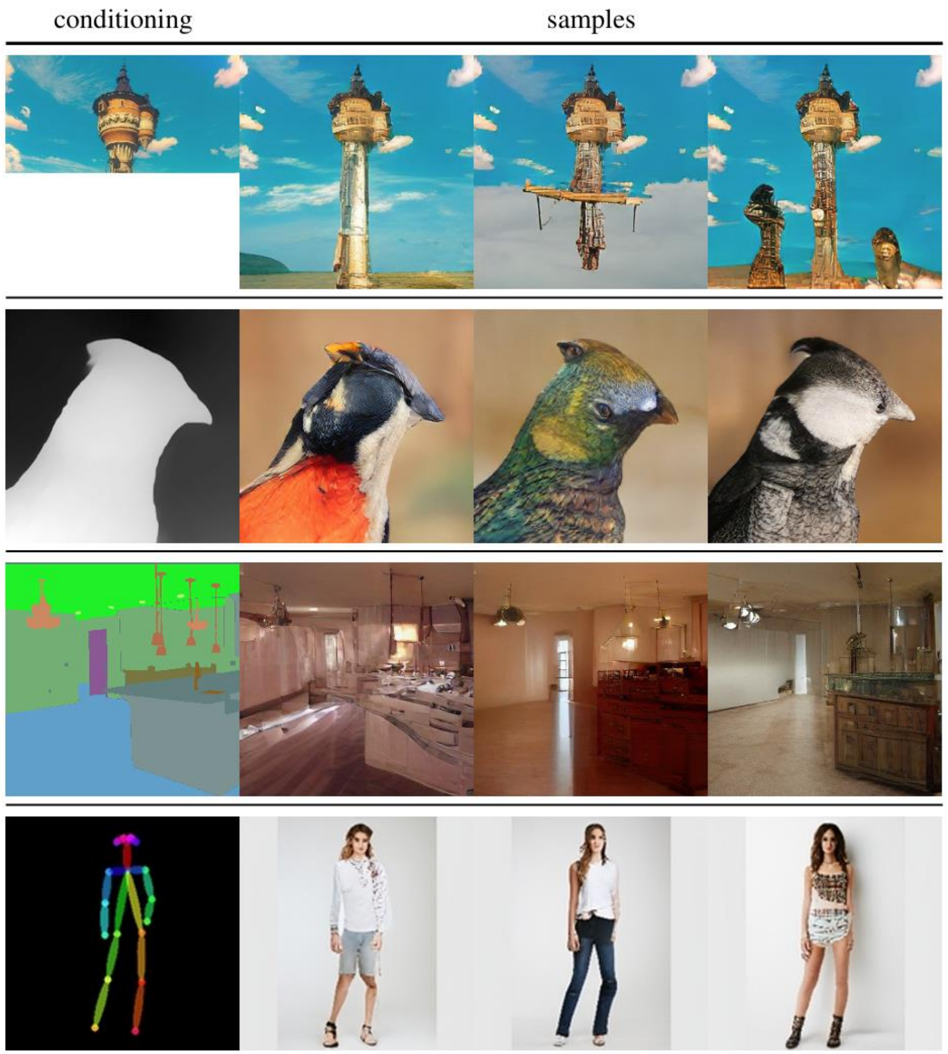




VQ-GAN
Esser et al. ICCV 2021

- Generative adversarial nets
- Perceptual loss
- Conditional inference

VQ-GAN



Generations with conditioning inputs

Generations with different resolution

Summary: 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 samples!

Cons:

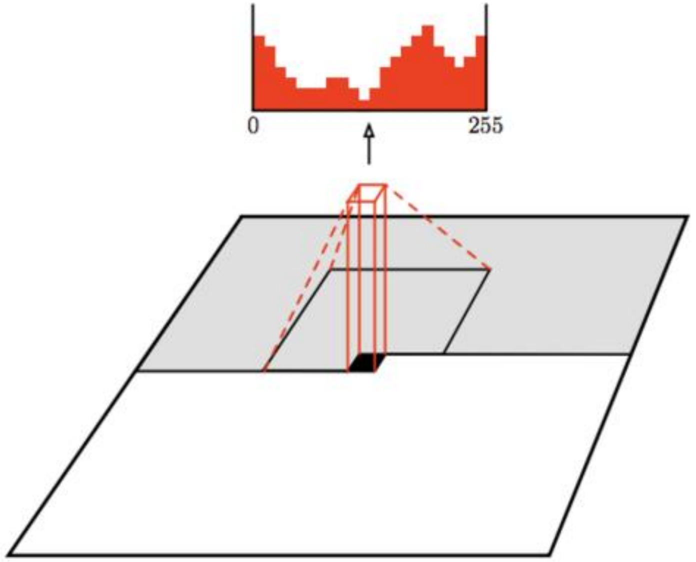
- Trickier / more unstable to train
- Can't solve inference queries such as p(x), p(z|x)

Active areas of research:

- Better loss functions, more stable training (Wasserstein GAN, LSGAN, many others)
- Conditional GANs, GANs for all kinds of applications

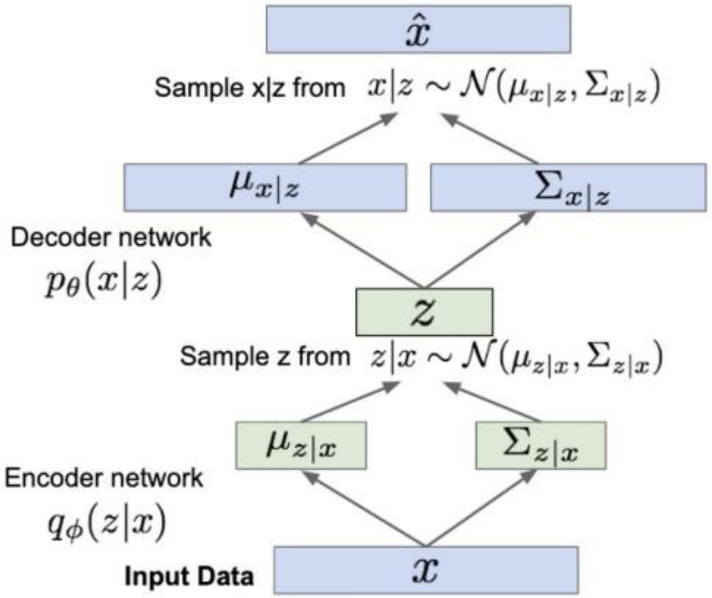
Summary

Autoregressive models:
PixelRNN, PixelCNN



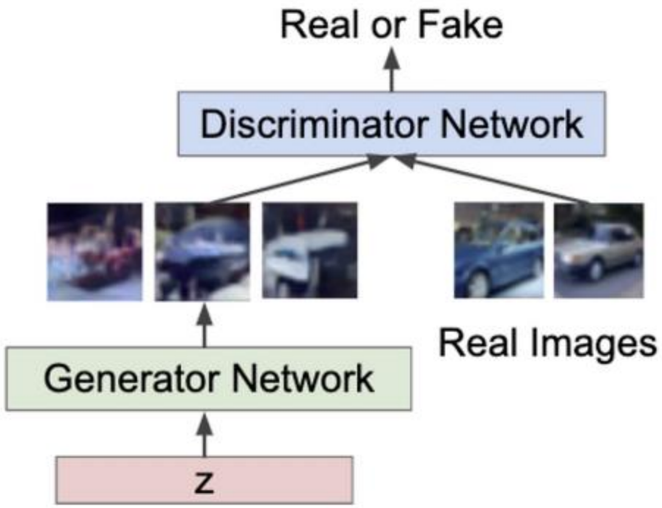
Van der Oord et al, “Conditional image generation with pixelCNN decoders”, NIPS 2016

Variational Autoencoders



Kingma and Welling, “Auto-encoding variational bayes”, ICLR 2013

Generative Adversarial Networks (GANs)



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

slide from https://cs231n.stanford.edu/slides/2024/lecture_13.pdf

Thanks!