Lecture 20:
Generative Models, Part 2

Midterm Regrades

The deadline for requesting midterm regrades is
Tuesday November 17, 11:59pm

If you've submitted a regrade request and haven't heard back,
make a new post about it on Piazza

Assignment 5: Object Detection

Single-stage detector
Two-stage detector

~~Due on Monday November 16, 11:59pm~~
Due on Wednesday November 18, 11:59pm

Assignment 6: Generative Models

Variational Auto-Encoders
Generative Adversarial Networks

Planning to release on **Wednesday November 18**
Will be due **three weeks** after release (due to Thanksgiving), currently **Wednesday December 9**

Wednesday: Guest Lecture – Andrej Karpathy

Senior Director of AI at Tesla
Founding member of OpenAI
PhD at Stanford
Designer of CS231N

Talk Title:
Applying ConvNets in Practice

Last Time: 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!

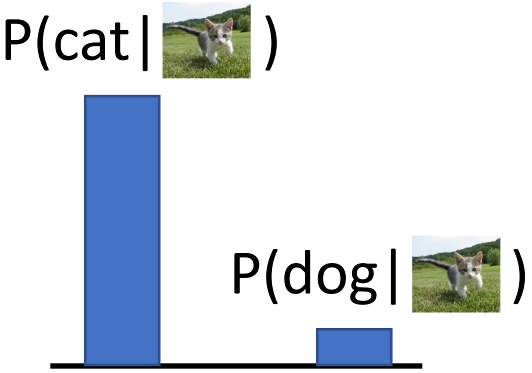
Goal: Learn some underlying hidden *structure* of the data

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

Last Time: Discriminative vs Generative Models

Discriminative Model:
Learn a probability distribution p(y|x)

Data: x



Generative Model:
Learn a probability distribution p(x)

Density Function
p(x) assigns a positive number to each possible x; higher numbers mean x is more likely

Density functions are **normalized**:

$$\int_X p(x)dx = 1$$

Different values of x **compete** for density

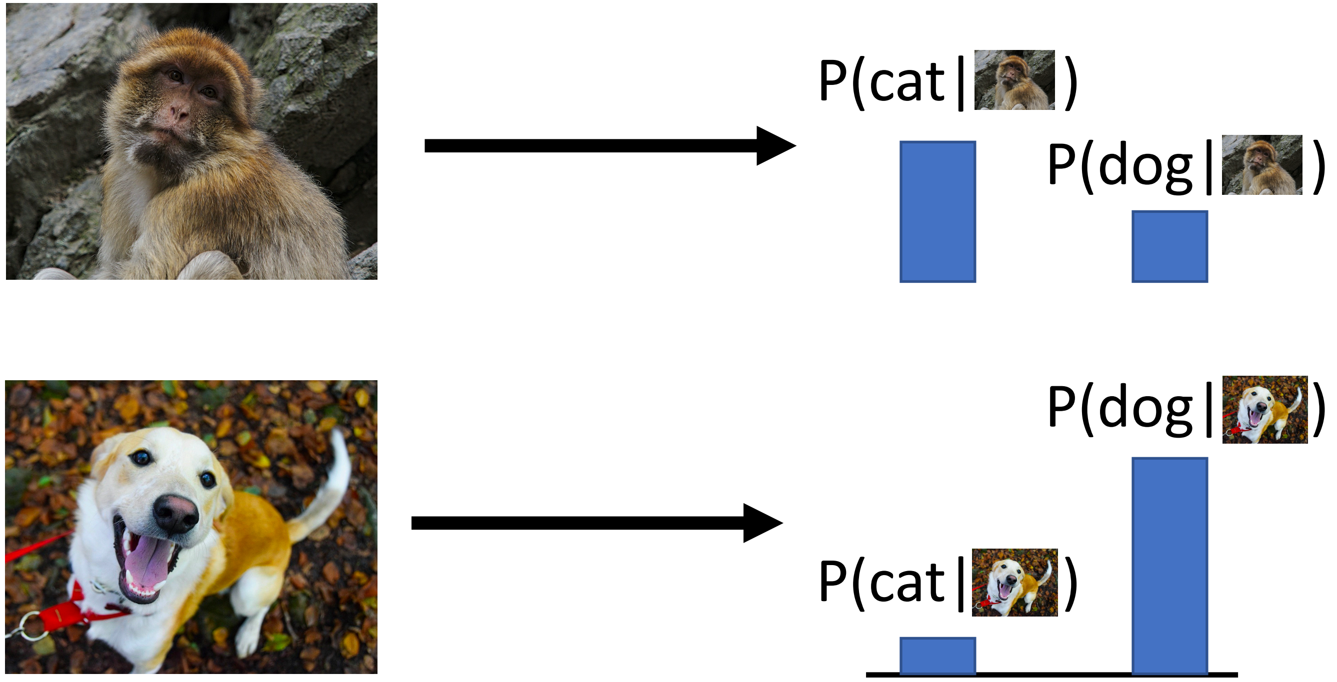
Conditional Generative Model: Learn p(x|y)

Last Time: Discriminative vs Generative Models

Discriminative Model: Learn a probability distribution p(y|x)

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)



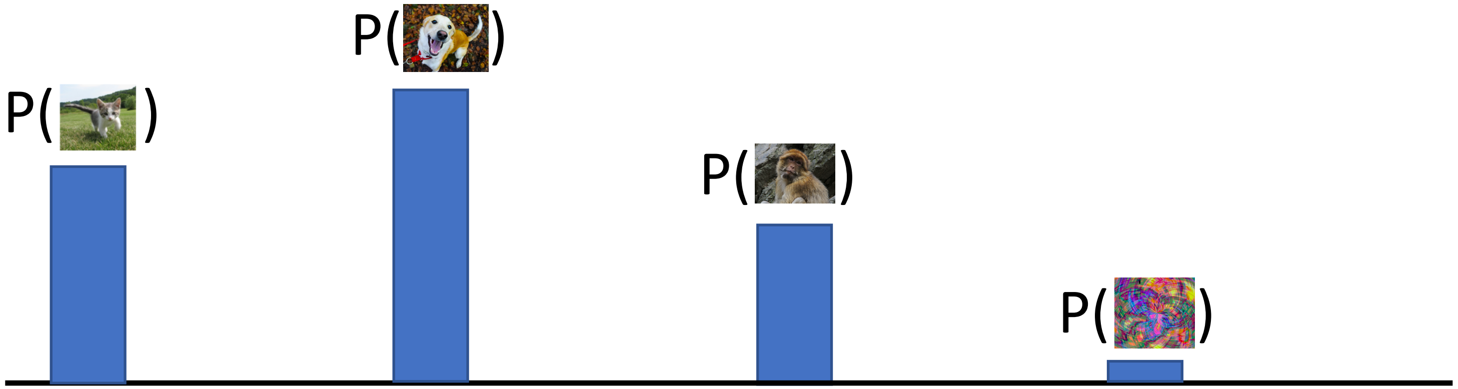
Discriminative model: No way for the model to handle unreasonable inputs; it must give label distributions for all images

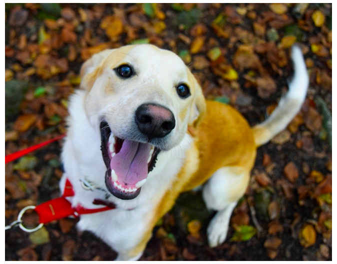
Last Time: Discriminative vs Generative Models

Discriminative Model: Learn a probability distribution p(y|x)

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)



...

Generative model: All possible images compete with each other for probability mass

Requires deep image understanding! Is a dog more likely to sit or stand? How about 3-legged dog vs 3-armed monkey?

Last Time: Discriminative vs Generative Models

Discriminative Model: Learn a probability distribution p(y|x)

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)

Recall **Bayes' Rule:**

$$P(x \mid y) = \frac{P(y \mid x)}{P(y)} P(x)$$

- $P(x \mid y)$: **Conditional Generative Model**
- $P(y \mid x)$: **Discriminative Model**
- $P(y)$: **Prior over labels**
- $P(x)$: **(Unconditional) Generative Model**

We can build a conditional generative model from other components!

Last Time: Taxonomy of Generative Models

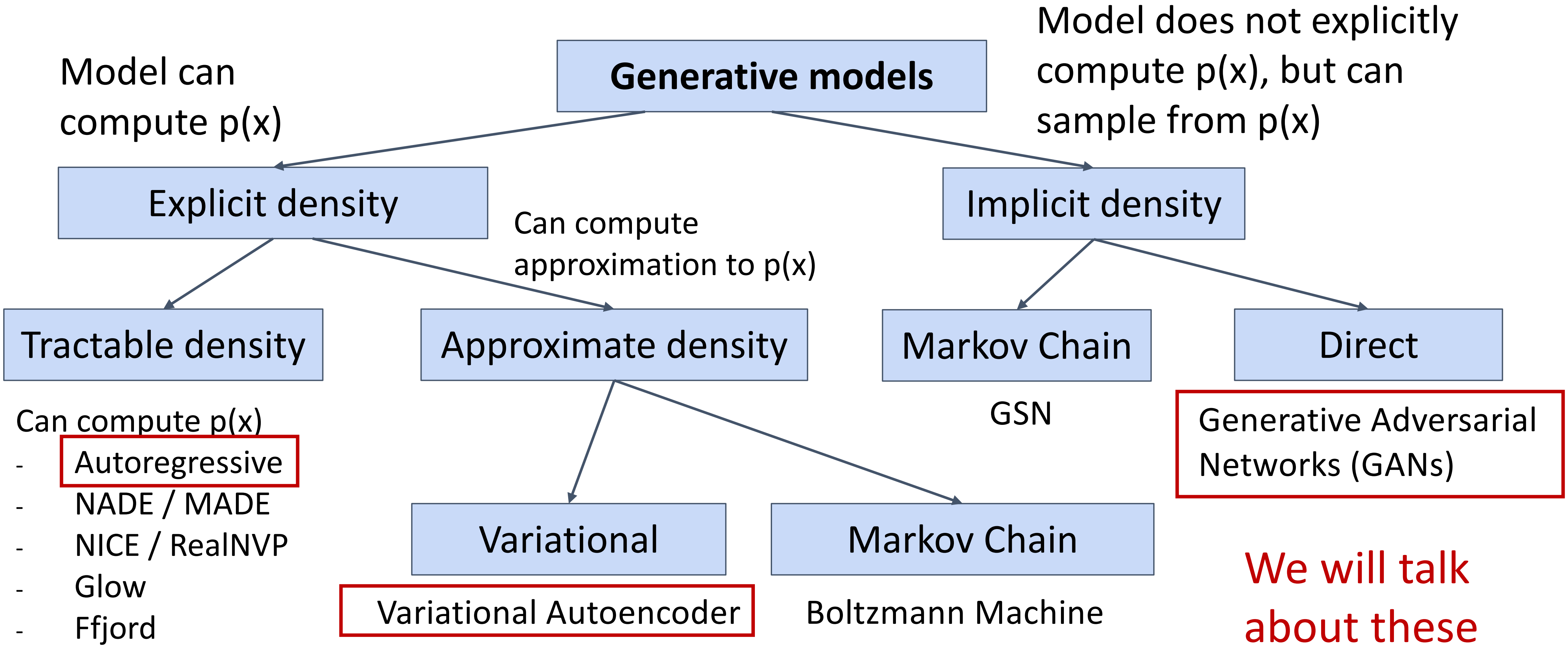


Figure adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

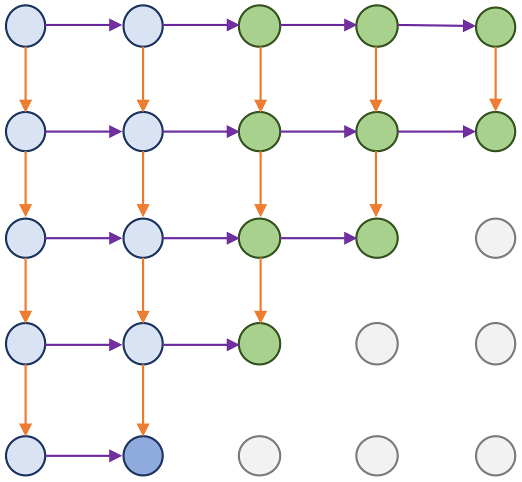
Last Time: Autoregressive Models

Explicit Density Function

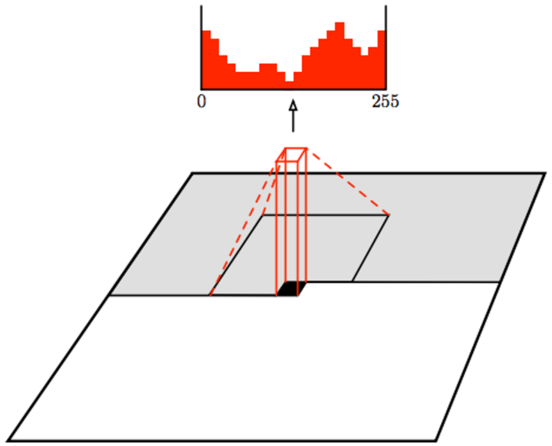
$$p(x) = p(x_1, x_2, x_3, \ldots, x_T)$$
$$= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \ldots$$
$$= \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$

Train by maximizing log-likelihood of training data

PixelRNN



PixelCNN



Van den Oord et al, "Pixel Recurrent Neural Networks", ICML 2016

Van den Oord et al, "Conditional Image Generation with PixelCNN Decoders", NeurIPS 2016

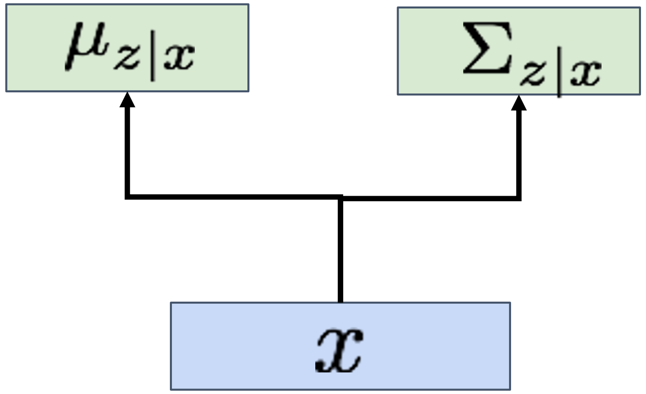
Last Time: Variational Autoencoders

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

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

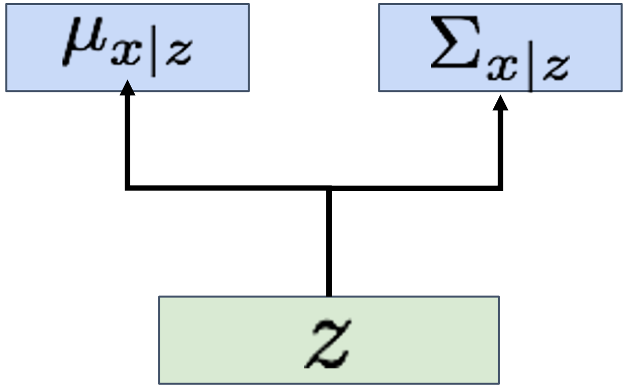
Encoder Network

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



Decoder Network

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

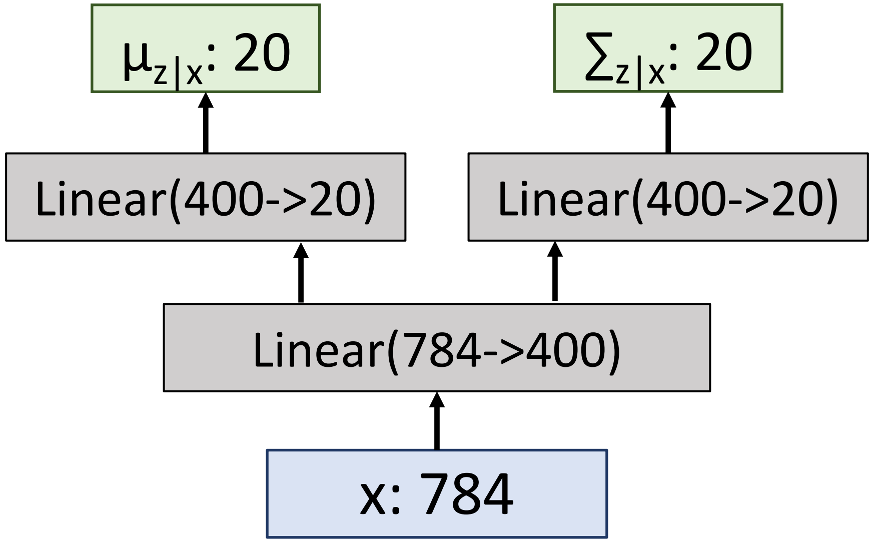


Example: Fully-Connected VAE

x: 28x28 image, flattened to 784-dim vector
z: 20-dim vector

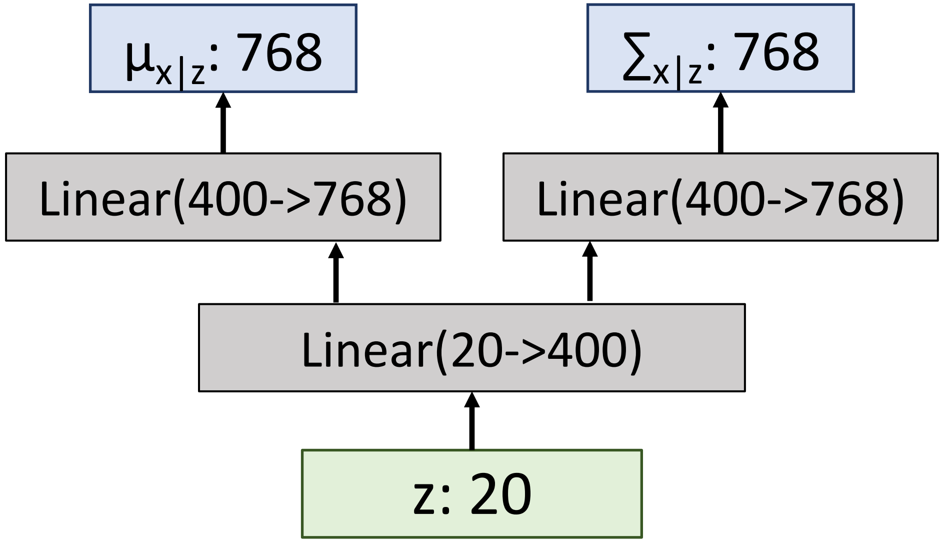
Encoder Network

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



Decoder Network

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



Variational Autoencoders

Train by maximizing the
variational lower bound

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

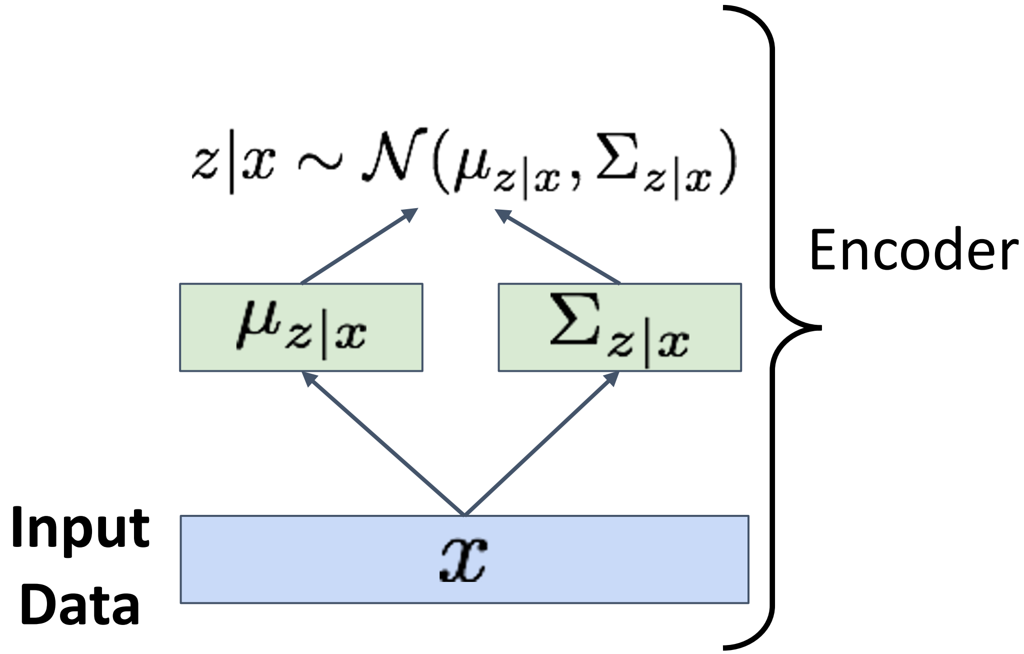
Input Data x

Variational Autoencoders

Train by maximizing the
variational lower bound

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

1. Run input data through **encoder** to get a distribution over latent codes

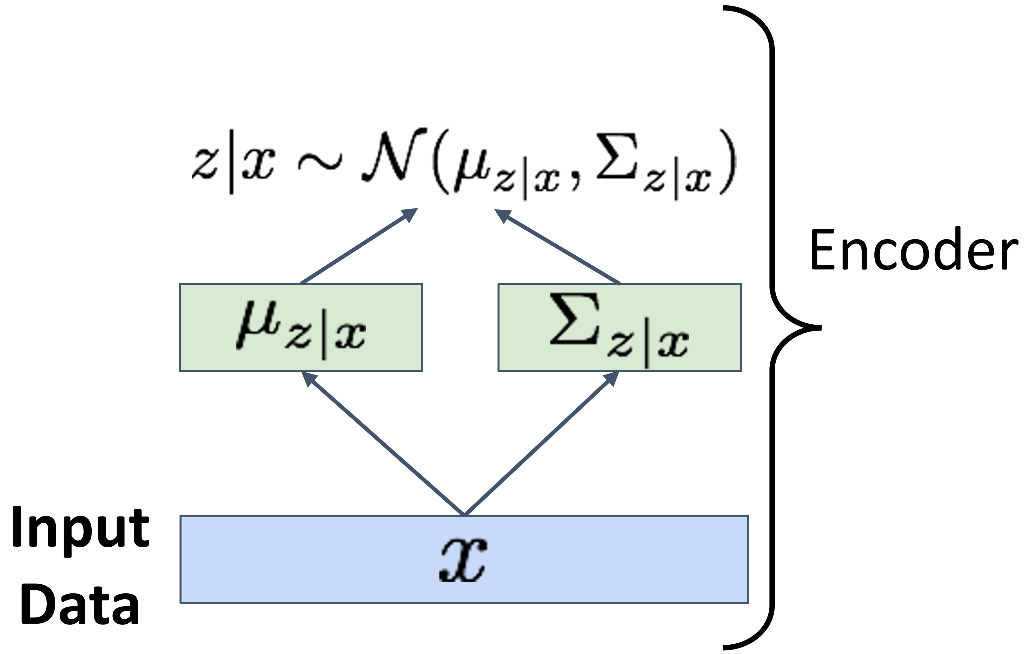


Variational Autoencoders

Train by maximizing the
variational lower bound

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

1. Run input data through **encoder** to get a distribution over latent codes
2. **Encoder output should match the prior p(z)!**



Variational Autoencoders

Train by maximizing the **variational lower bound**

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

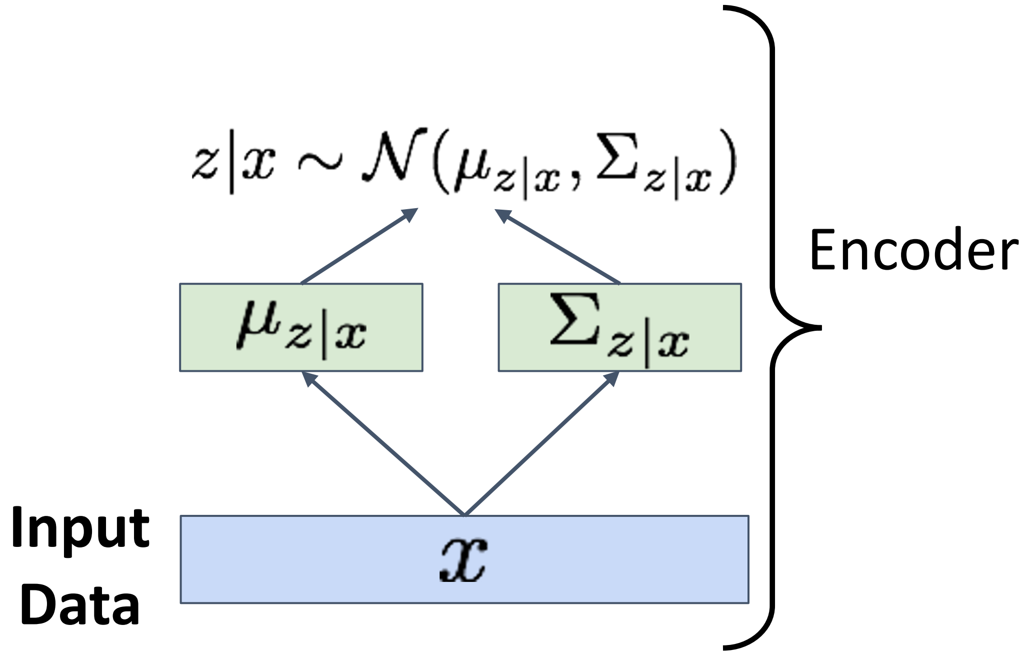
Closed form solution when q_ϕ is diagonal Gaussian and p is unit Gaussian! (Assume z has dimension J)

1. Run input data through **encoder** to get a distribution over latent codes
2. **Encoder output should match the prior p(z)!**

$$-D_{KL}\left(q_\phi(z|x), p(z)\right) = \int_Z q_\phi(z|x) \log \frac{p(z)}{q_\phi(z|x)} dz$$

$$= \int_Z N\left(z; \mu_{z|x}, \Sigma_{z|x}\right) \log \frac{N(z; 0, I)}{N\left(z; \mu_{z|x}, \Sigma_{z|x}\right)} dz$$

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

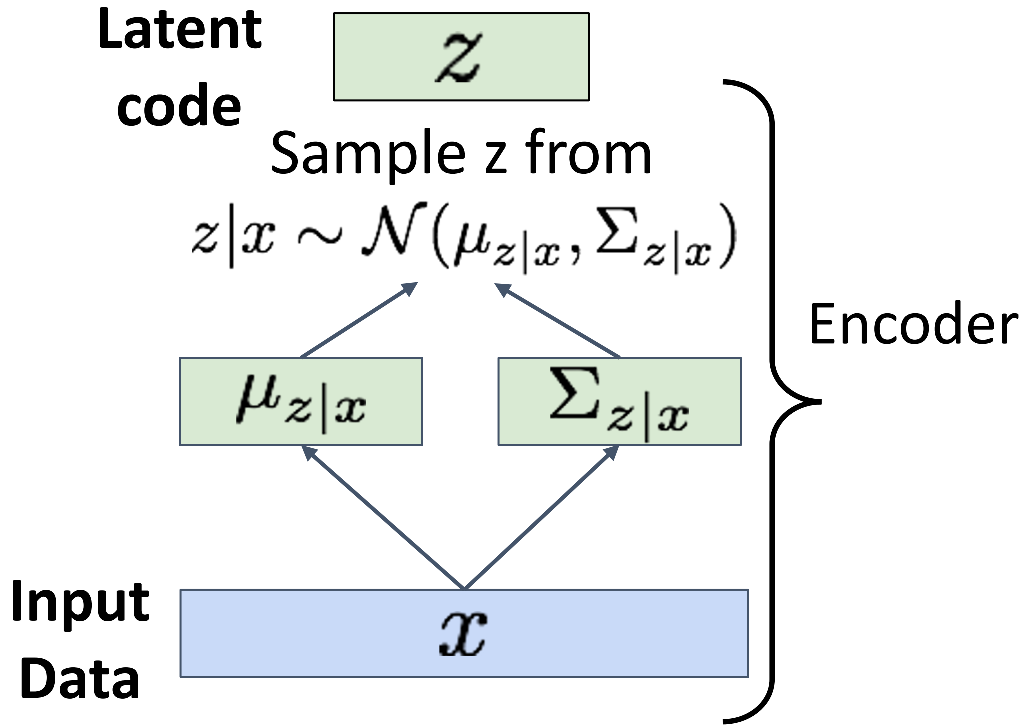


Variational Autoencoders

Train by maximizing the
variational lower bound

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

1. Run input data through **encoder** to get a distribution over latent codes
2. **Encoder output should match the prior p(z)!**
3. Sample code z from encoder output

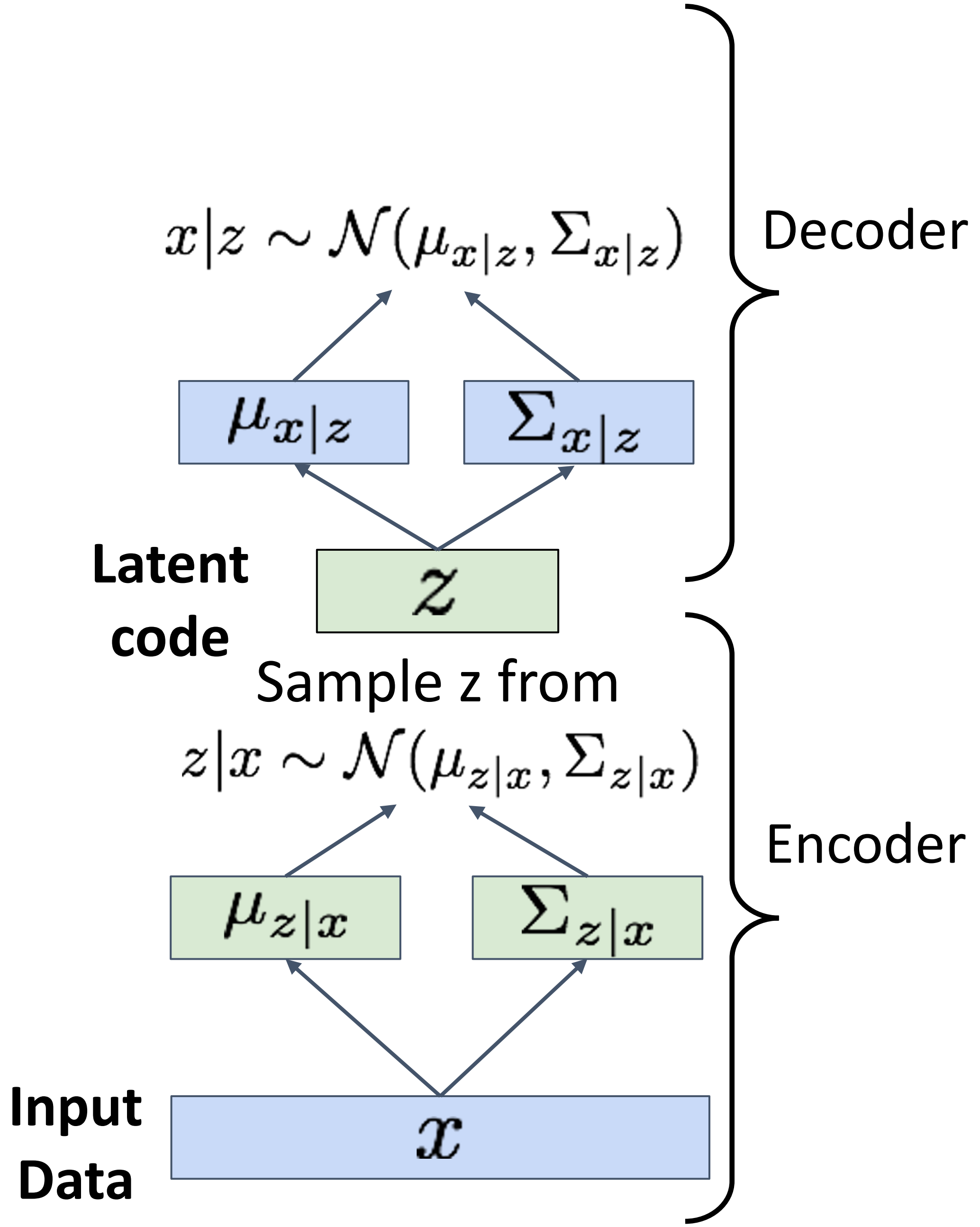


Variational Autoencoders

Train by maximizing the **variational lower bound**

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

1. Run input data through **encoder** to get a distribution over latent codes
2. **Encoder output should match the prior p(z)!**
3. Sample code z from encoder output
4. Run sampled code through **decoder** to get a distribution over data samples

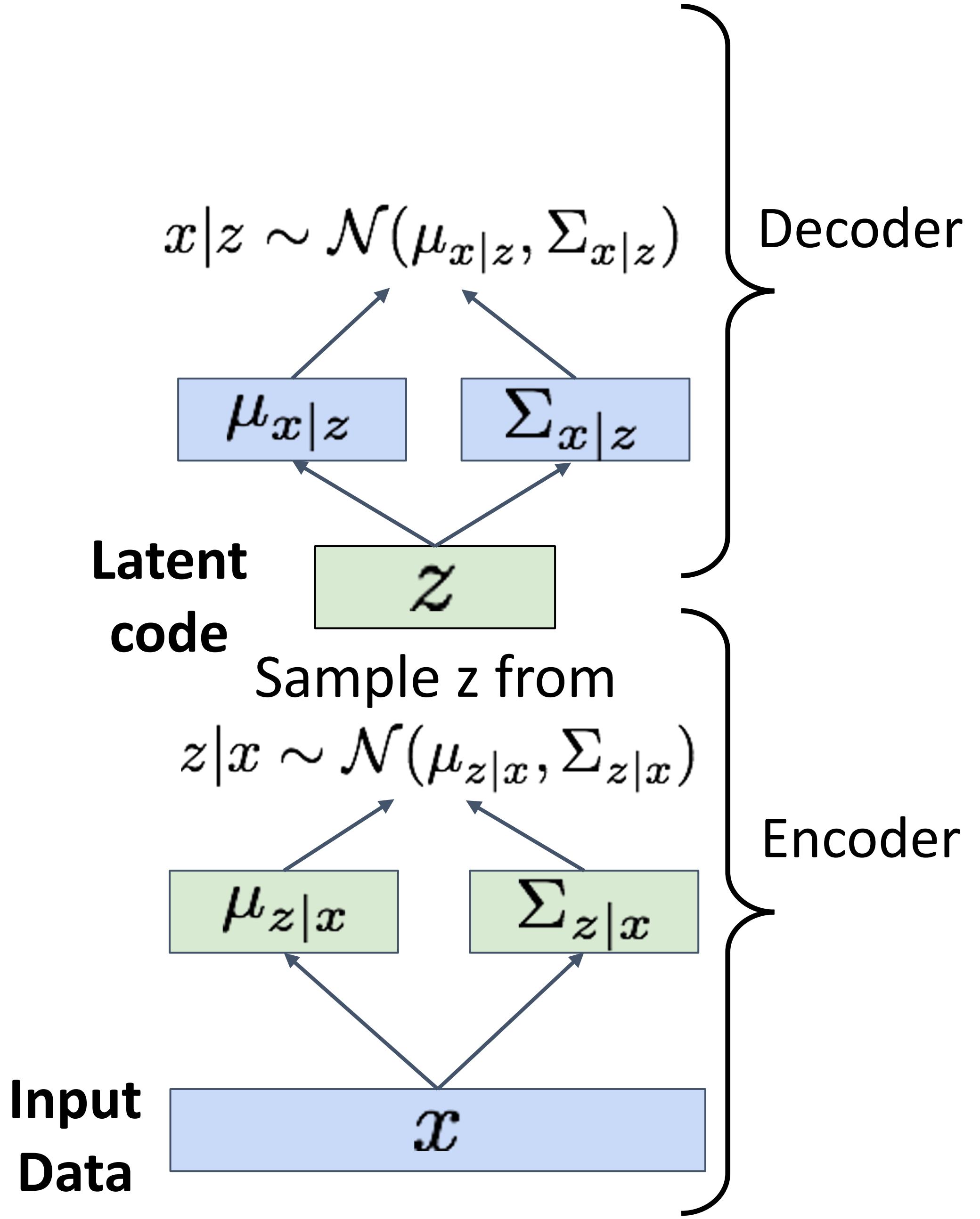


Variational Autoencoders

Train by maximizing the **variational lower bound**

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

1. Run input data through **encoder** to get a distribution over latent codes
2. **Encoder output should match the prior p(z)!**
3. Sample code z from encoder output
4. Run sampled code through **decoder** to get a distribution over data samples
5. **Original input data should be likely under the distribution output from (4)!**

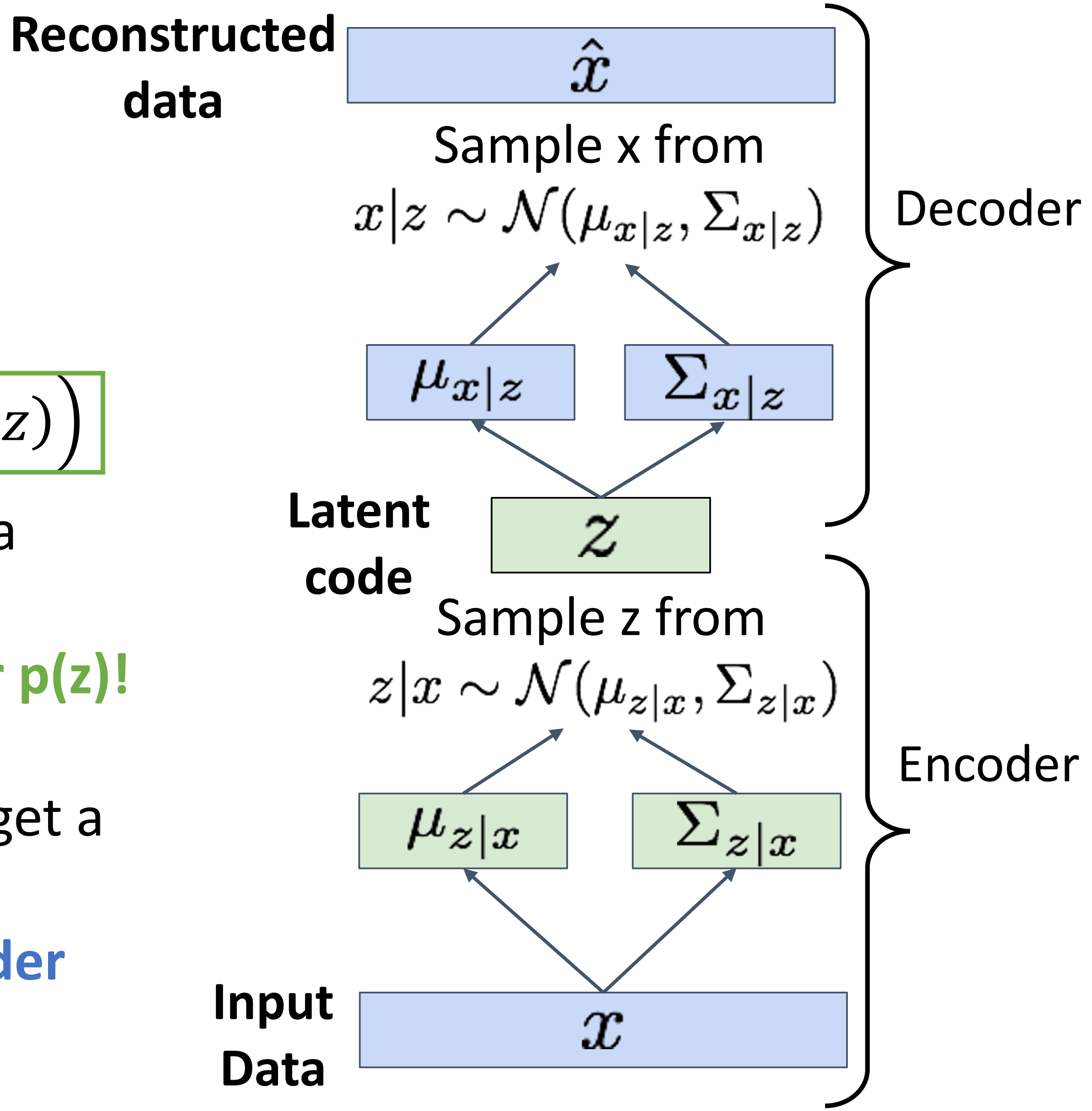


Variational Autoencoders

Train by maximizing the **variational lower bound**

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

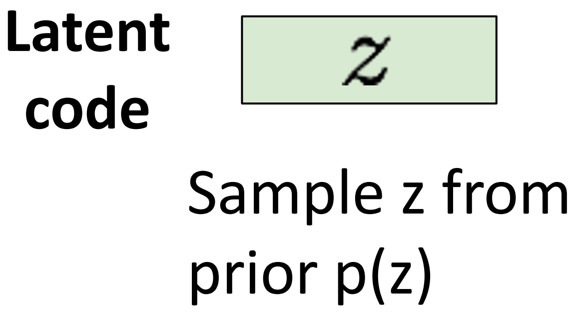
1. Run input data through **encoder** to get a distribution over latent codes
2. **Encoder output should match the prior p(z)!**
3. Sample code z from encoder output
4. Run sampled code through **decoder** to get a distribution over data samples
5. **Original input data should be likely under the distribution output from (4)!**
6. Can sample a reconstruction from (4)



Variational Autoencoders: Generating Data

After training we can generate new data!

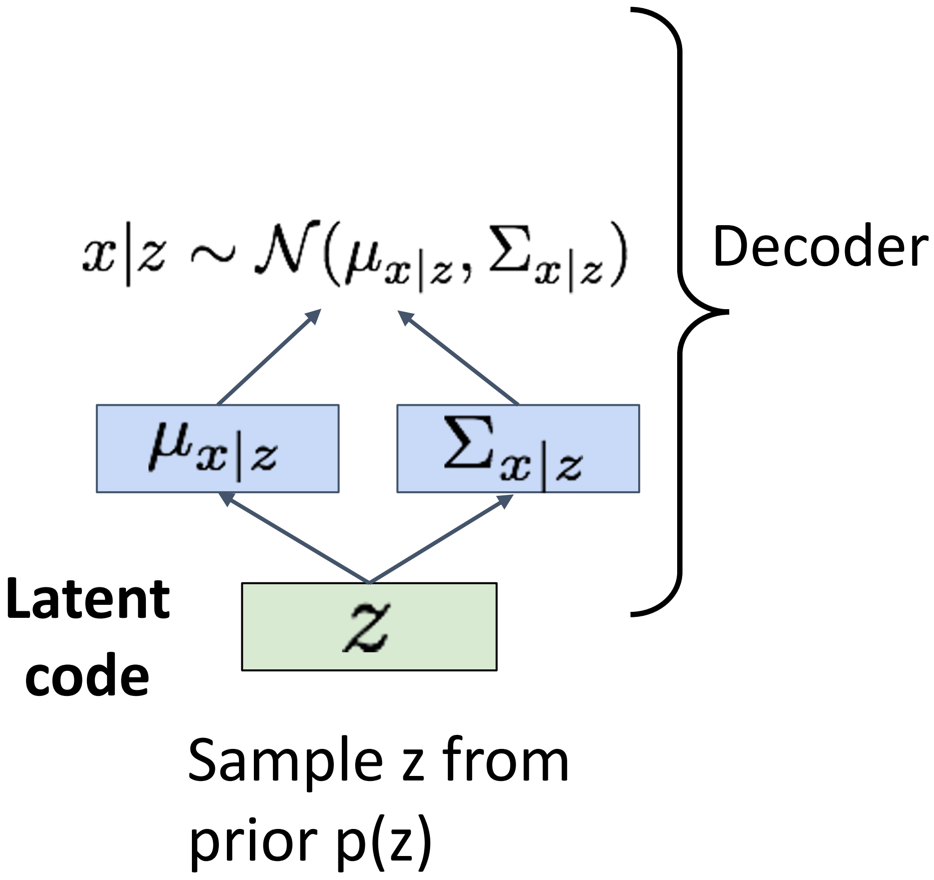
1. Sample z from prior p(z)



Variational Autoencoders: Generating Data

After training we can generate new data!

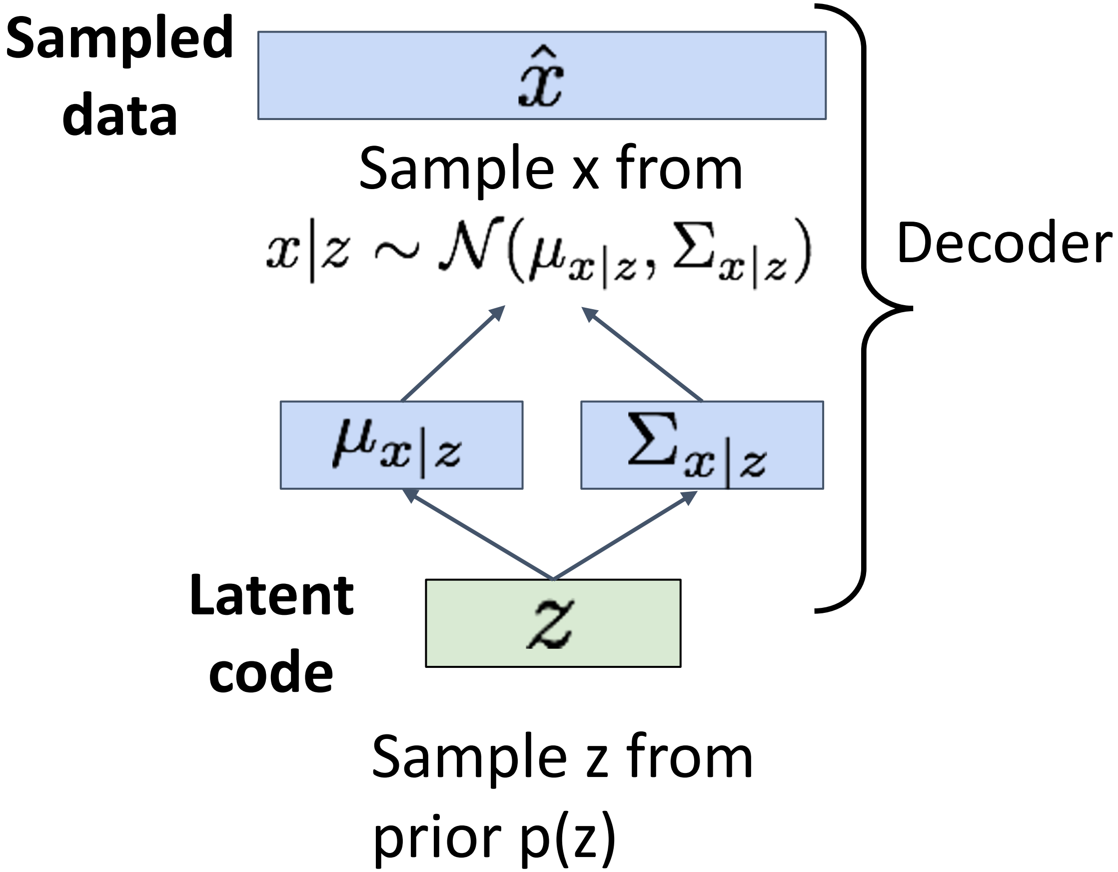
1. Sample z from prior p(z)
2. Run sampled z through decoder to get distribution over data x



Variational Autoencoders: Generating Data

After training we can generate new data!

1. Sample z from prior p(z)
2. Run sampled z through decoder to get distribution over data x
3. Sample from distribution in (2) to generate data



Variational Autoencoders: Generating Data

32x32 CIFAR-10

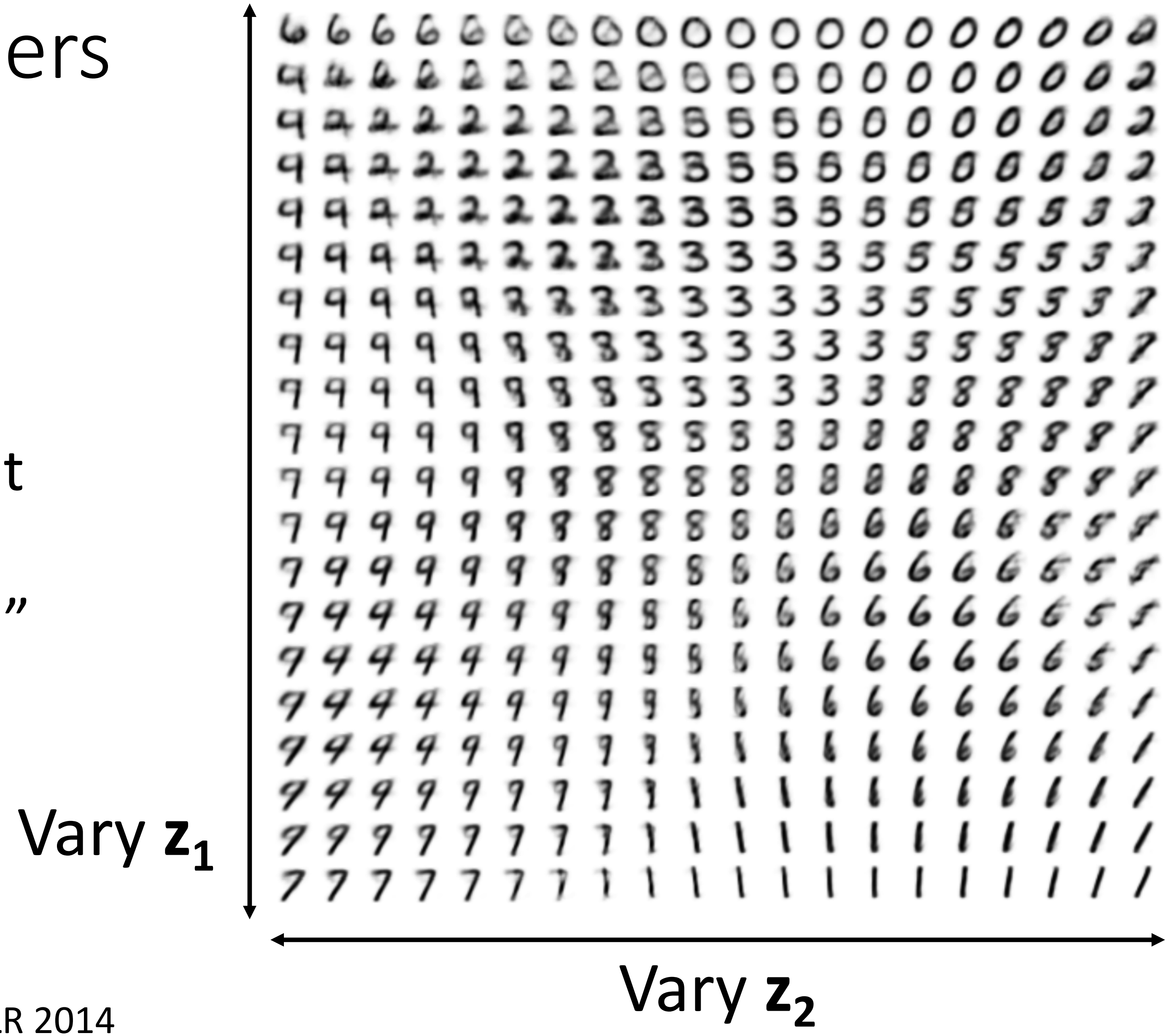
Labeled Faces in the Wild

Figures from (L) Dirk Kingma et al. 2016; (R) Anders Larsen et al. 2017.

Variational Autoencoders

The diagonal prior on p(z) causes dimensions of z to be independent

"Disentangling factors of variation"

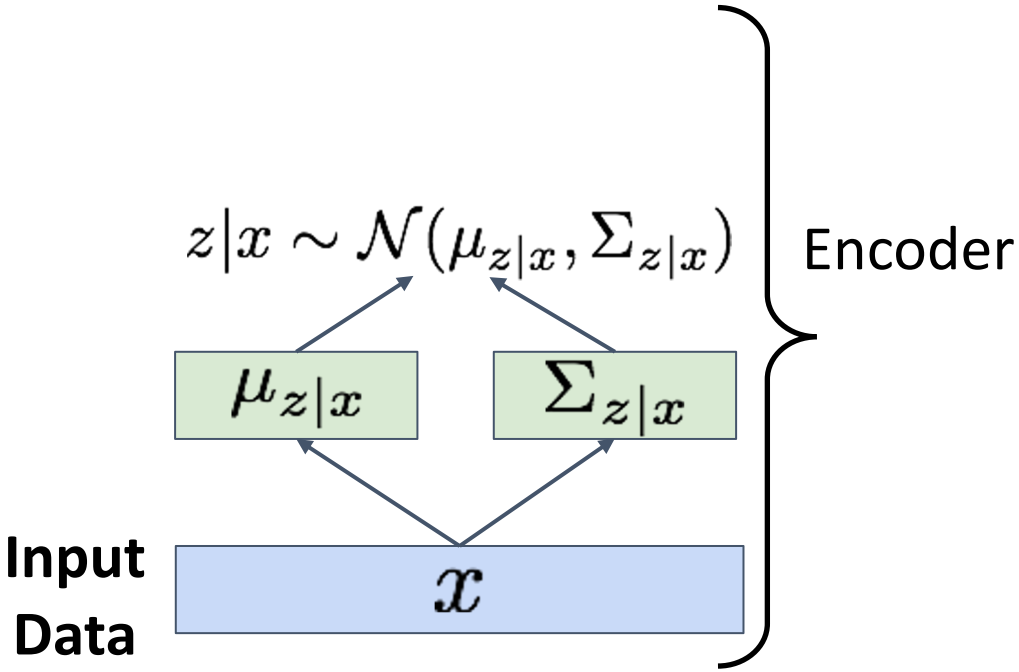


Kingma and Welling, Auto-Encoding Variational Beyes, ICLR 2014

Variational Autoencoders

After training we can **edit images**

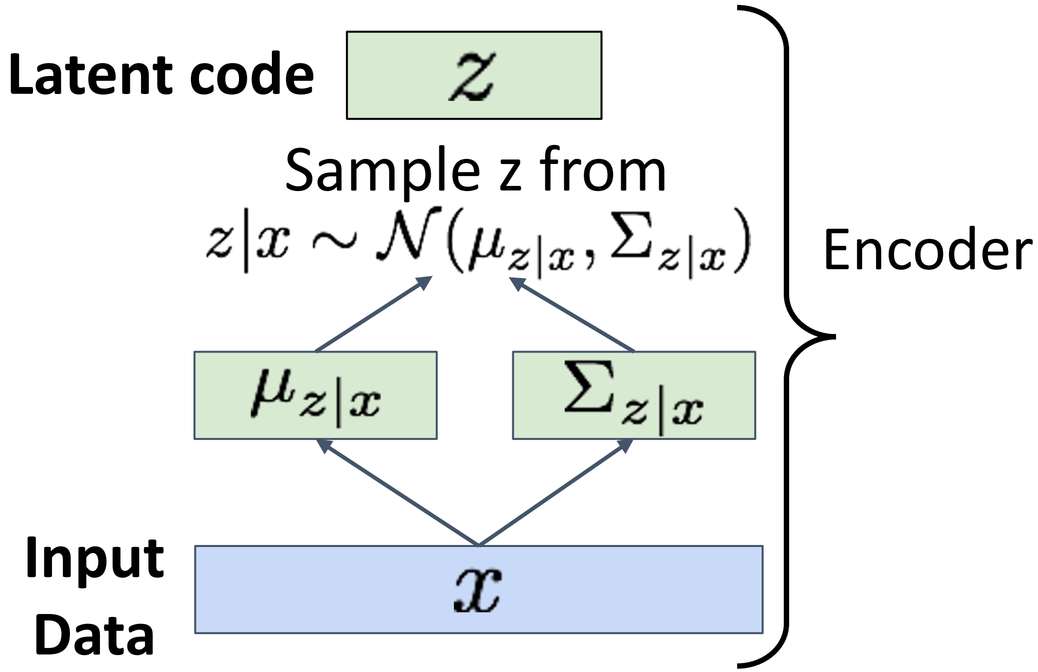
1. Run input data through **encoder** to get a distribution over latent codes



Variational Autoencoders

After training we can **edit images**

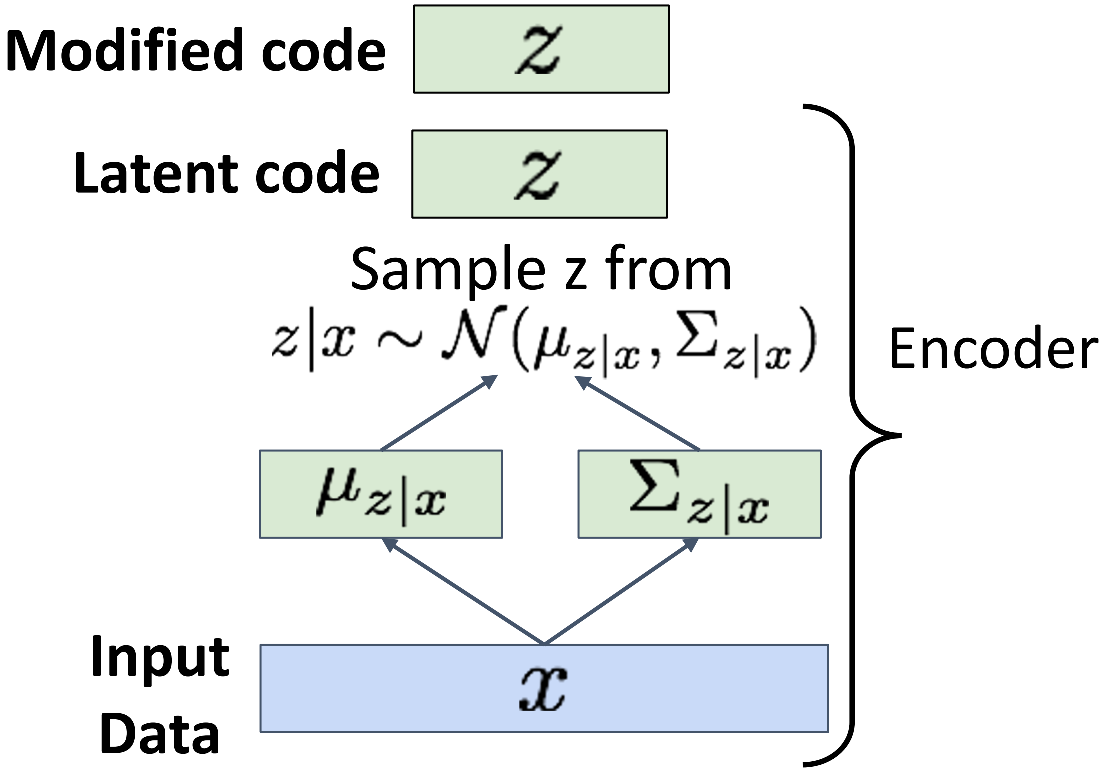
1. Run input data through **encoder** to get a distribution over latent codes
2. Sample code z from encoder output



Variational Autoencoders

After training we can **edit images**

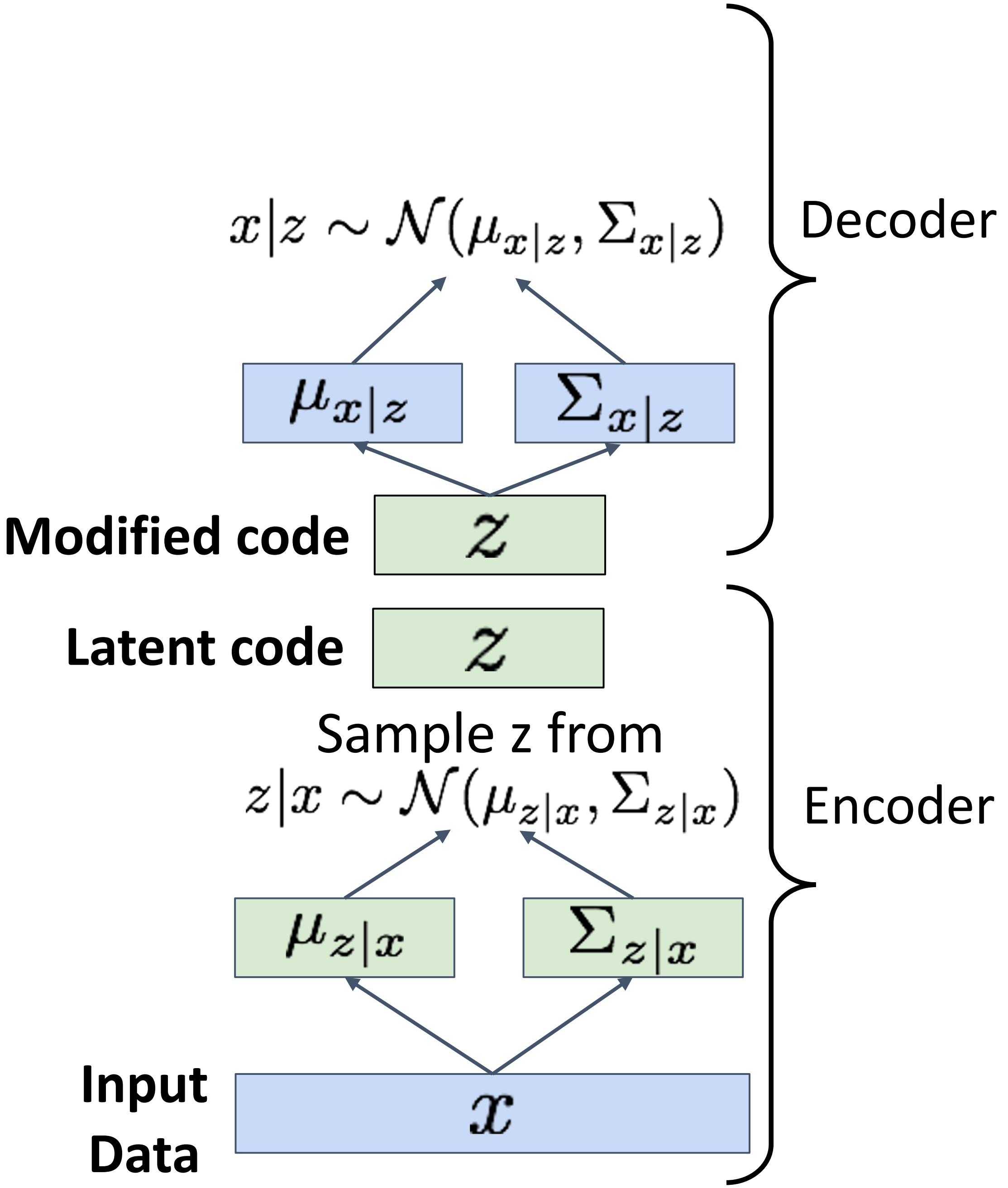
1. Run input data through **encoder** to get a distribution over latent codes
2. Sample code z from encoder output
3. Modify some dimensions of sampled code



Variational Autoencoders

After training we can **edit images**

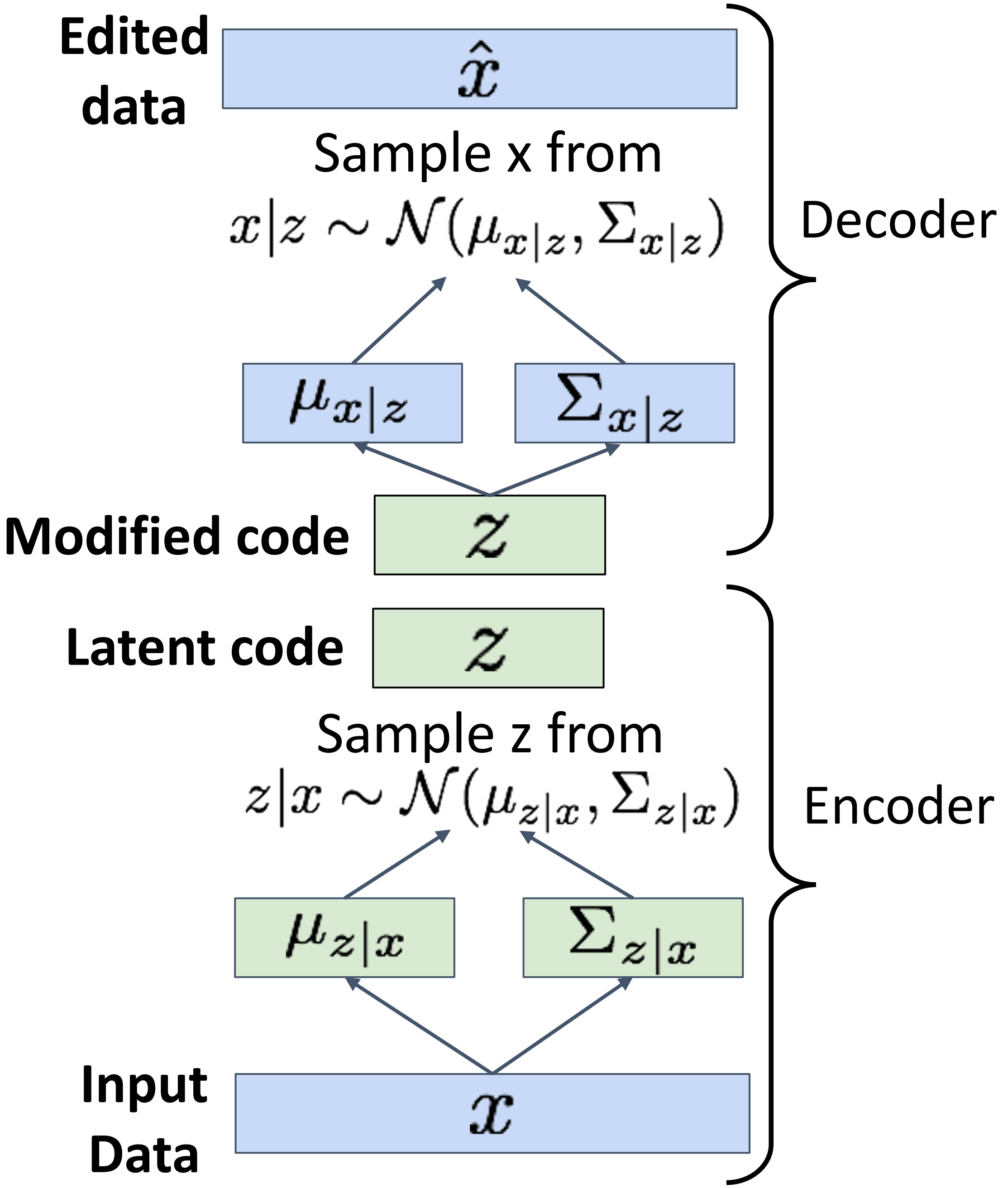
1. Run input data through **encoder** to get a distribution over latent codes
2. Sample code z from encoder output
3. Modify some dimensions of sampled code
4. Run modified z through **decoder** to get a distribution over data sample



Variational Autoencoders

After training we can **edit images**

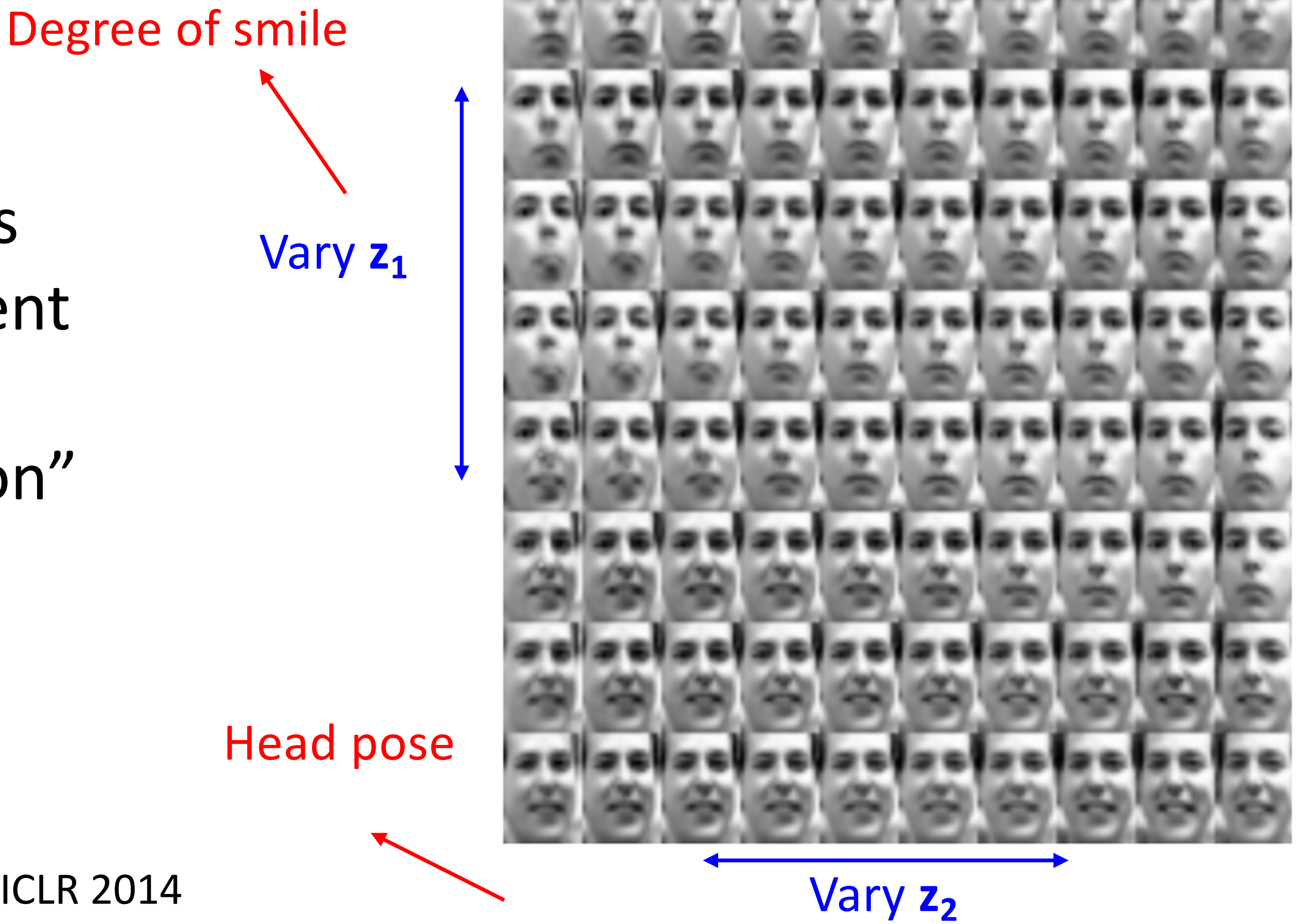
1. Run input data through **encoder** to get a distribution over latent codes
2. Sample code z from encoder output
3. Modify some dimensions of sampled code
4. Run modified z through **decoder** to get a distribution over data samples
5. Sample new data from (4)



Variational Autoencoders

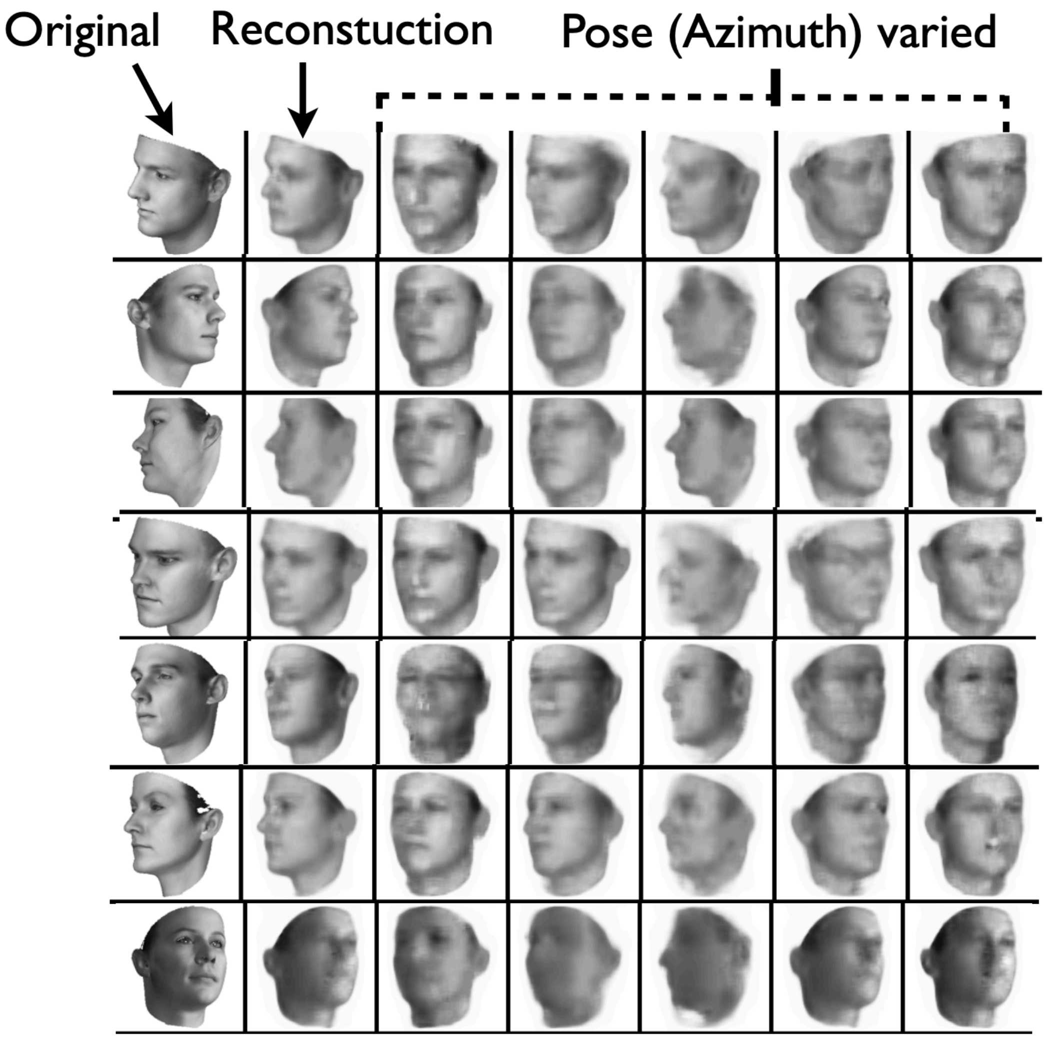
The diagonal prior on p(z) causes dimensions of z to be independent

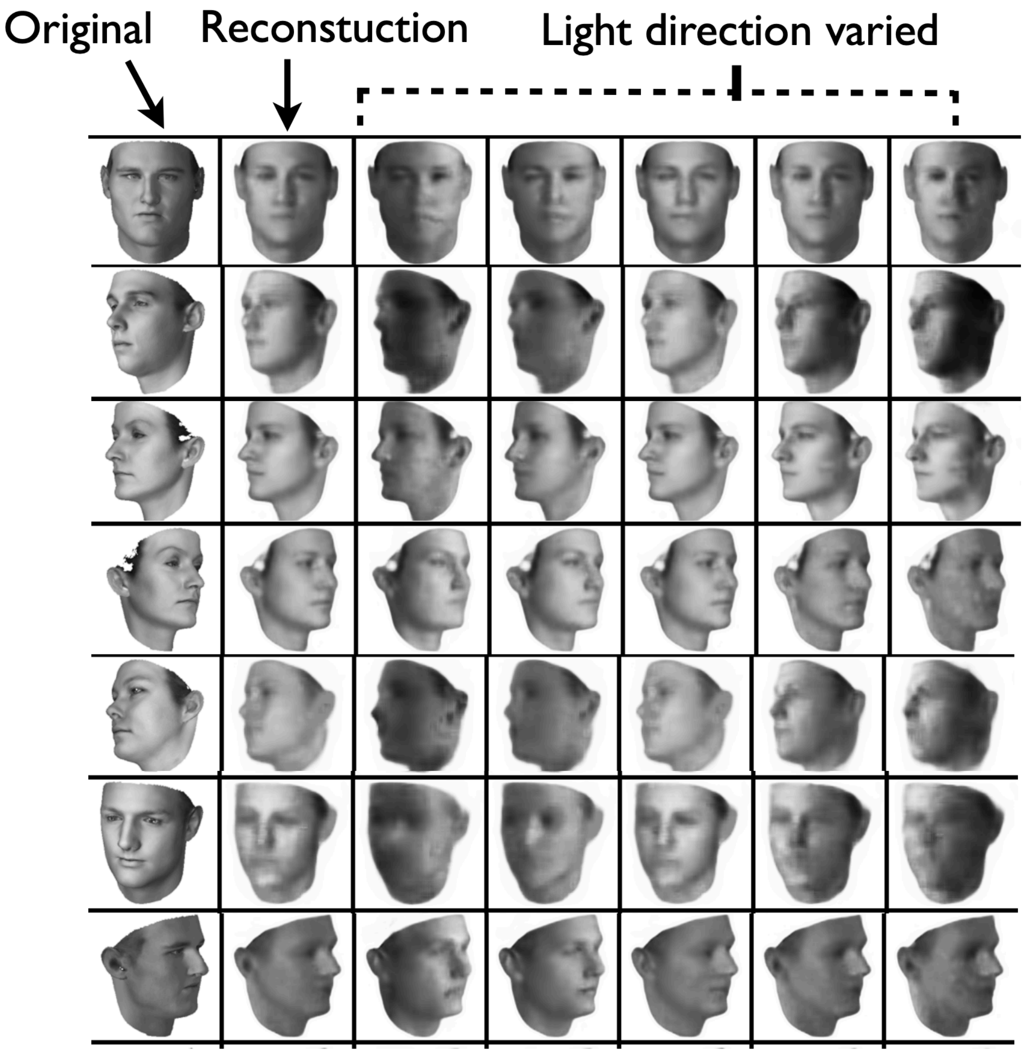
"Disentangling factors of variation"



Kingma and Welling, Auto-Encoding Variational Beyes, ICLR 2014

Variational Autoencoders: Image Editing





Kulkarni et al, “Deep Convolutional Inverse Graphics Networks”, NeurIPS 2014

Variational Autoencoder: Summary

Probabilistic spin to traditional autoencoders => allows generating data
Defines an intractable density => derive and optimize a (variational) lower bound

Pros:

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

Cons:

- Maximizes lower bound of likelihood: okay, but not as good evaluation as PixelRNN/PixelCNN
- Samples blurrier and lower quality compared to state-of-the-art (GANs)

Active areas of research:

- More flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian, e.g., Gaussian Mixture Models (GMMs)
- Incorporating structure in latent variables, e.g., Categorical Distributions

So far: Two types of generative models

Autoregressive models

- Directly maximize p(data)
- High-quality generated images
- Slow to generate images
- No explicit latent codes

Variational models

- Maximize lower-bound on p(data)
- Generated images often blurry
- Very fast to generate images
- Learn rich latent codes

So far: Two types of generative models

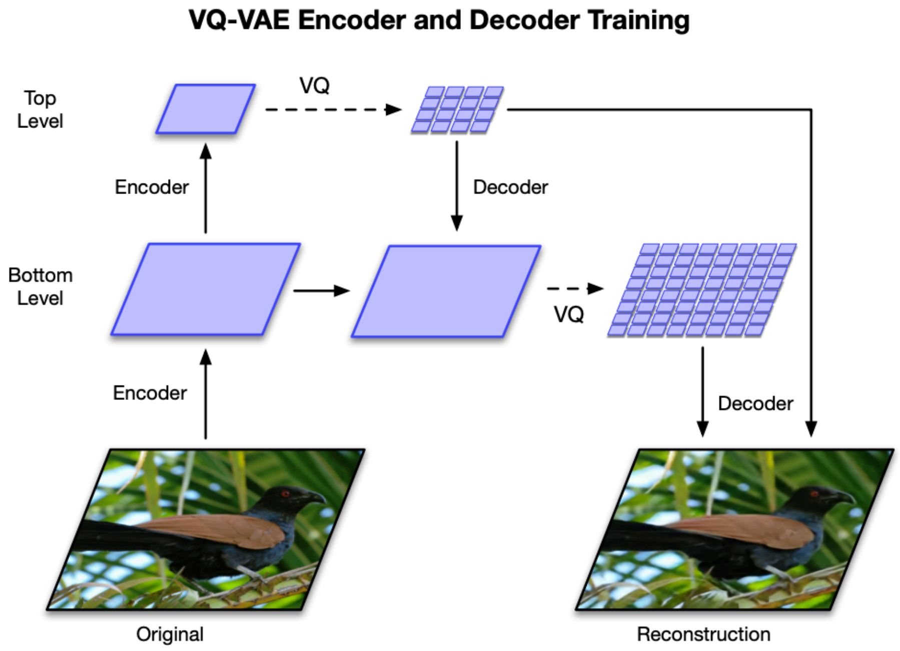
Autoregressive models
- Directly maximize p(data)
- High-quality generated images
- Slow to generate images
- No explicit latent codes

Variational models
- Maximize lower-bound on p(data)
- Generated images often blurry
- Very fast to generate images
- Learn rich latent codes

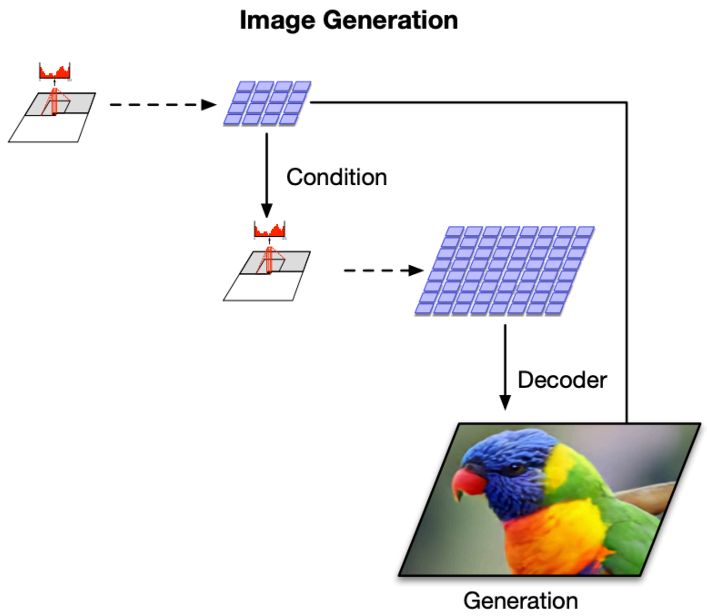
Can we combine them and get the best of both worlds?

Combining VAE + Autoregressive: Vector-Quantized Variational Autoencoder (VQ-VAE2)

Train a VAE-like model to generate multiscale grids of latent codes



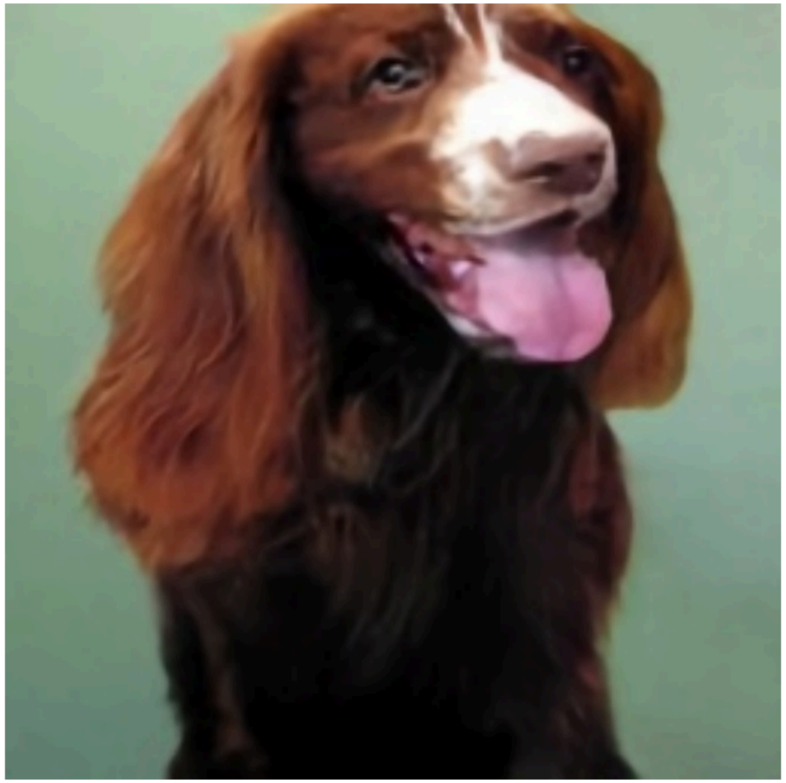
Use a multiscale PixelCNN to sample in latent code space



Razavi et al, "Generating Diverse High-Fidelity Images with VQ-VAE-2", NeurIPS 2019

Combining VAE + Autoregressive: VQ-VAE2

256 x 256 class-conditional samples, trained on ImageNet



Razavi et al, "Generating Diverse High-Fidelity Images with VQ-VAE-2", NeurIPS 2019

Combining VAE + Autoregressive: VQ-VAE2

256 x 256 class-conditional samples, trained on ImageNet

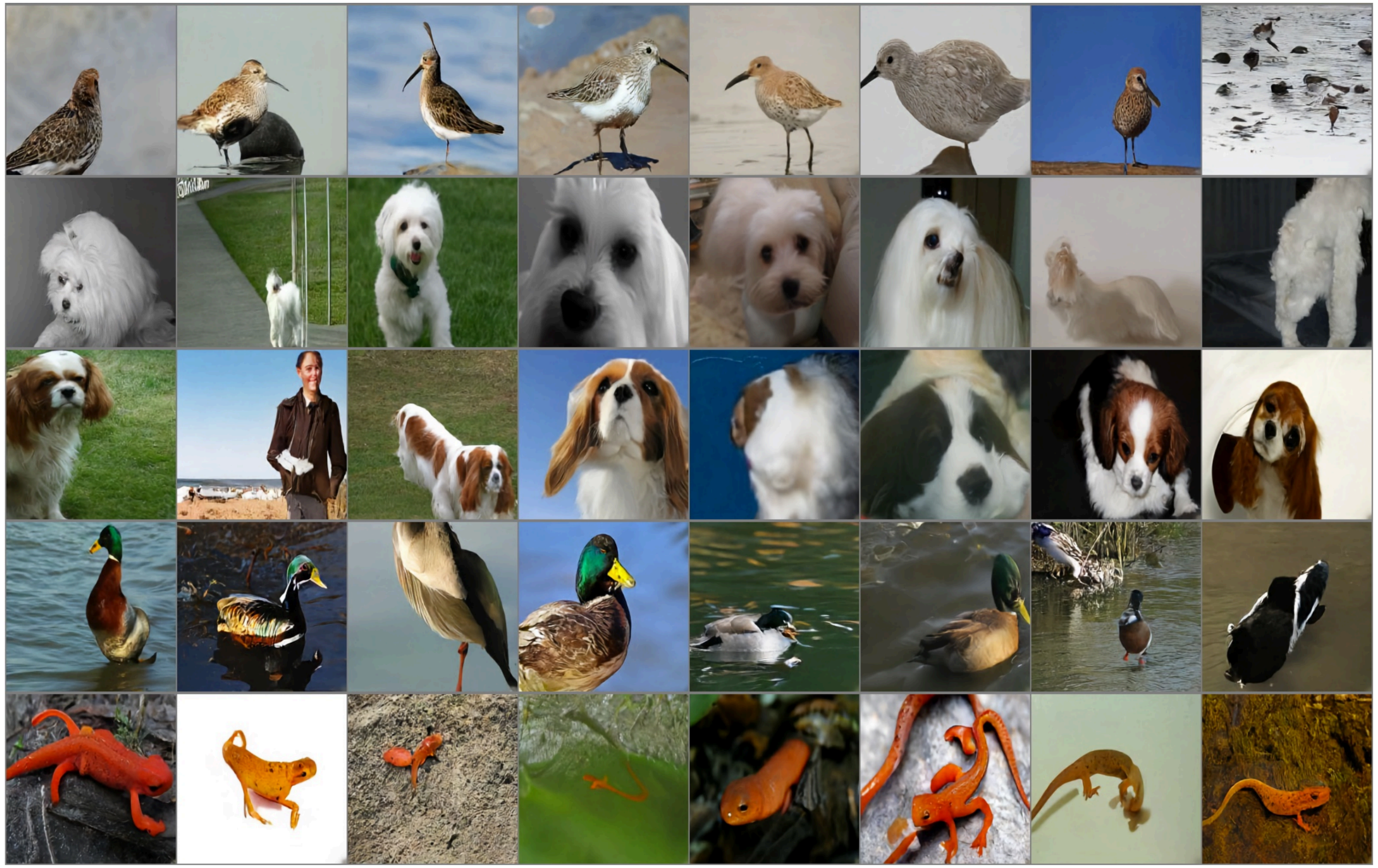
Redshank

Pekinese

Papillon

Drake

Spotted Salamander



Razavi et al, "Generating Diverse High-Fidelity Images with VQ-VAE-2", NeurIPS 2019

Combining VAE + Autoregressive: VQ-VAE2

1024 x 1024 generated faces, trained on FFHQ

Razavi et al, "Generating Diverse High-Fidelity Images with VQ-VAE-2", NeurIPS 2019

Combining VAE + Autoregressive: VQ-VAE2

1024 x 1024 generated faces, trained on FFHQ

Razavi et al, “Generating Diverse High-Fidelity Images with VQ-VAE-2”, NeurIPS 2019

Generative Models So Far:

Autoregressive Models directly maximize likelihood of training data:

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

Generative Models So Far:

Autoregressive Models directly maximize likelihood of training data:

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

Variational Autoencoders introduce a latent z, and maximize a lower bound:

$$p_\theta(x) = \int_Z p_\theta(x|z)p(z)dz \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

Generative Models So Far:

Autoregressive Models directly maximize likelihood of training data:

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

Variational Autoencoders introduce a latent z, and maximize a lower bound:

$$p_\theta(x) = \int_Z p_\theta(x|z)p(z)dz \geq E_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

Generative Adversarial Networks give up on modeling p(x), but allow us to draw samples from p(x)

Generative Adversarial Networks

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

Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks

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

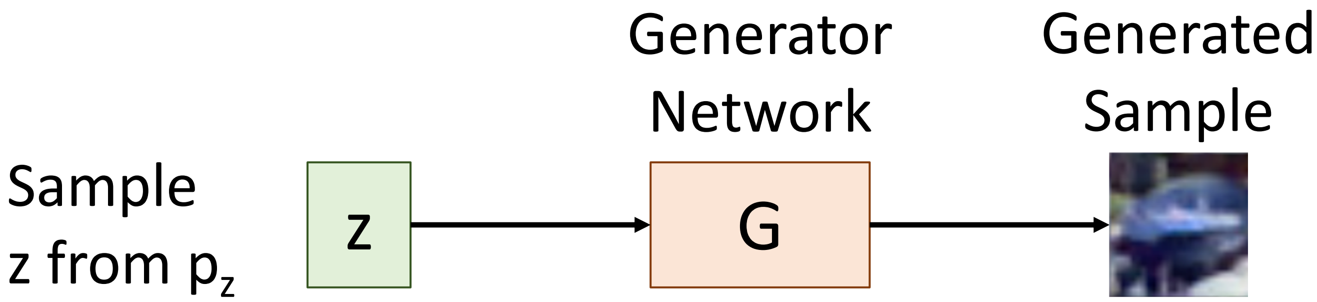
Idea: Introduce a latent variable z with simple prior p(z).
Sample $z \sim p(z)$ and pass to a **Generator Network** x = G(z)
Then x is a sample from the **Generator distribution** p_G. Want $p_G = p_{data}$!

Goodfellow et al, "Generative Adversarial Nets", 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 $p_G = p_{data}$!



Train **Generator Network** G to convert z into fake data x sampled from p_G

Goodfellow et al, "Generative Adversarial Nets", 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 $p_G = p_{data}$!

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)

Goodfellow et al, "Generative Adversarial Nets", 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 $p_G = p_{data}$!

Jointly train G and D. Hopefully p_G converges to p_{data}!

Generator Network

Generated Sample

Discriminator Network

Sample z from p_z

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

Real Sample

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

Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks: Training Objective

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

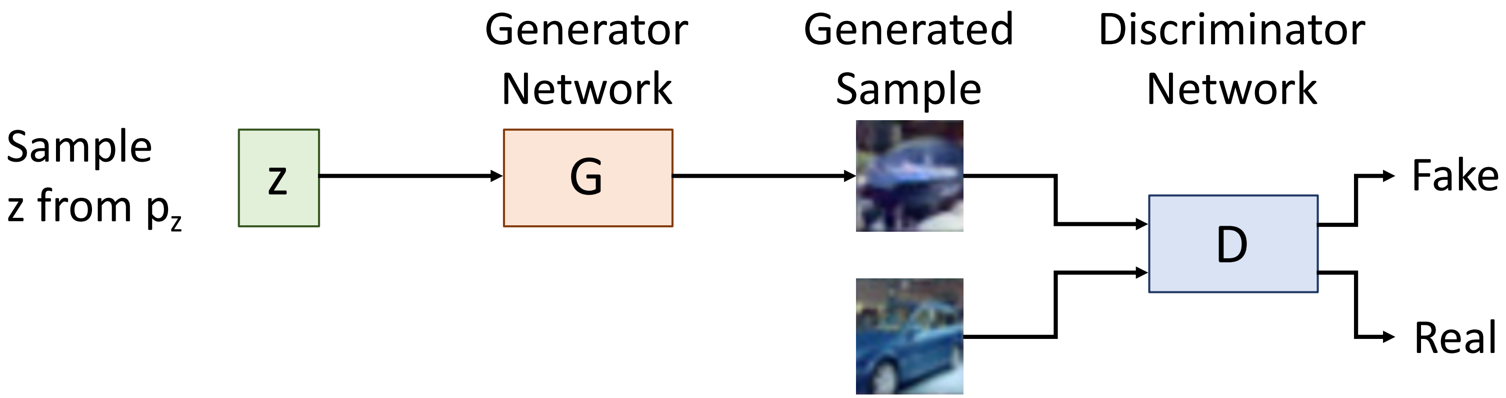
$$\min_G \max_D \left(E_{x \sim p_{data}}[\log D(x)] + E_{z \sim p(z)}\left[\log\left(1 - D\big(G(z)\big)\right)\right]\right)$$

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

Generative Adversarial Networks: Training Objective

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

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



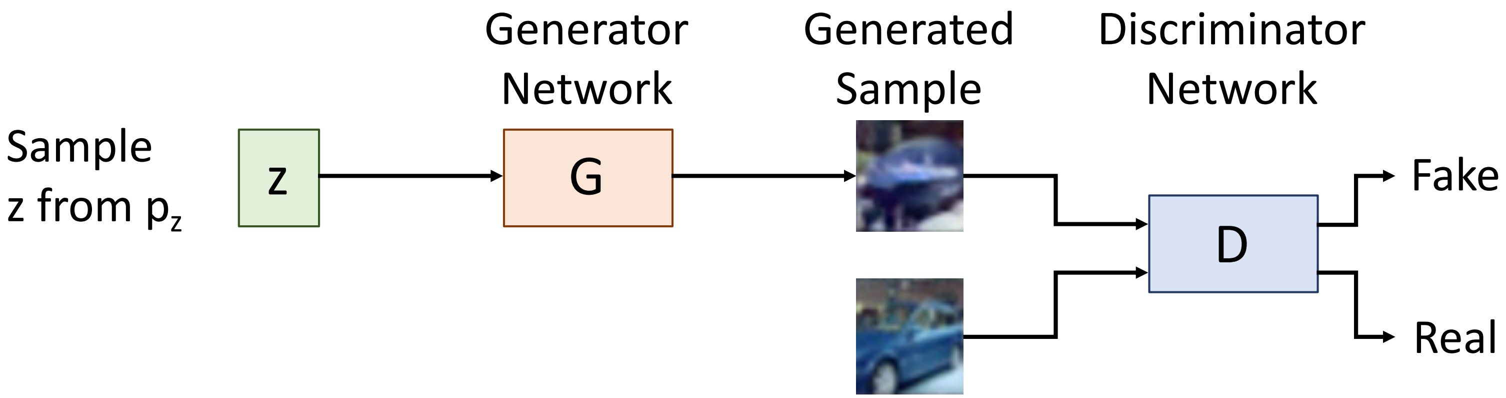
Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks: Training Objective

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

Discriminator wants D(x) = 1 for real data

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



Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

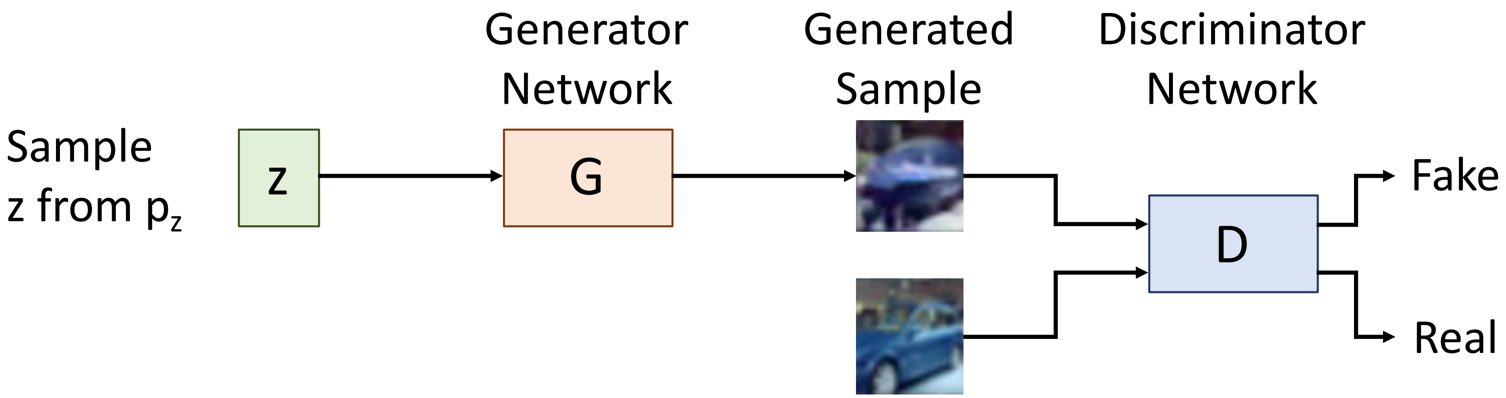
Generative Adversarial Networks: Training Objective

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

Discriminator wants D(x) = 1 for real data

Discriminator wants D(x) = 0 for fake data

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



Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks: Training Objective

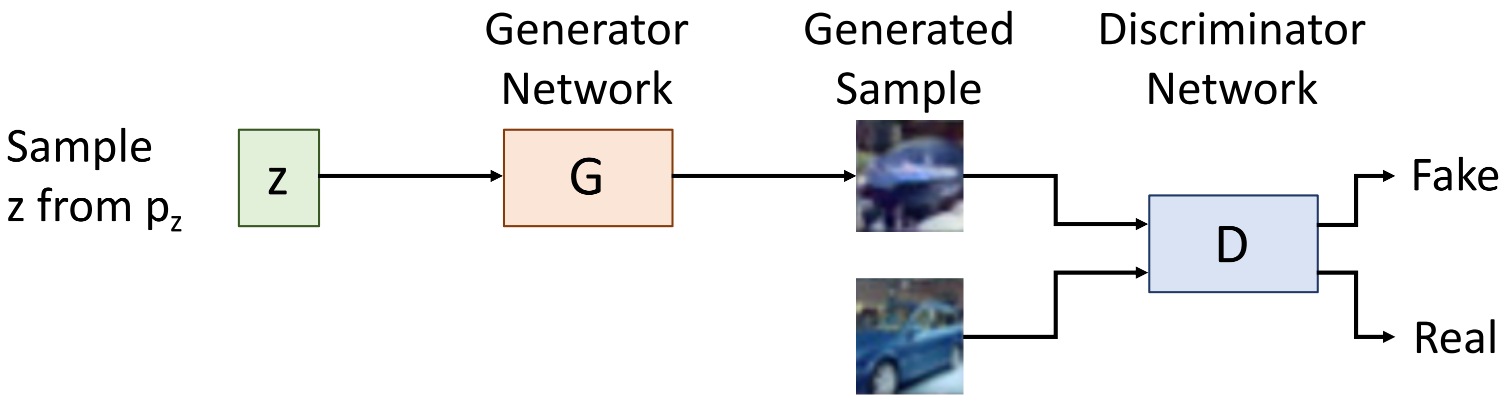
Jointly train generator G and discriminator D with a **minimax game**

Discriminator wants D(x) = 1 for real data

Discriminator wants D(x) = 0 for fake data

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

Generator wants D(x) = 1 for fake data



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

Generative Adversarial Networks: Training Objective

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

Train G and D using alternating gradient updates

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

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

Generative Adversarial Networks: Training Objective

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

Train G and D using alternating gradient updates

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

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

Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks: Training Objective

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

Train G and D using alternating gradient updates

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

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

For t in 1, … T:

1. (Update D) $D = D + \alpha_D \frac{\partial V}{\partial D}$
2. (Update G) $G = G - \alpha_G \frac{\partial V}{\partial G}$

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

Generative Adversarial Networks: Training Objective

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

Train G and D using alternating gradient updates

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

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

We are not minimizing any overall loss! No training curves to look at!

For t in 1, … T:

1. (Update D) $D = D + \alpha_D \frac{\partial V}{\partial D}$
2. (Update G) $G = G - \alpha_G \frac{\partial V}{\partial G}$

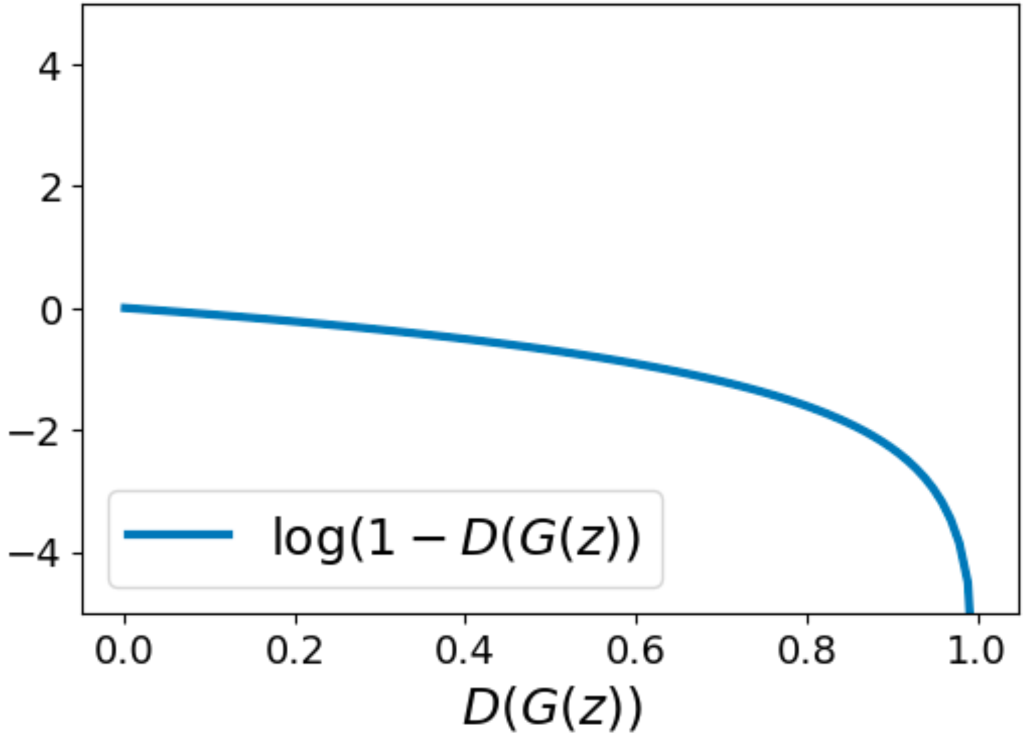
Goodfellow et al, “Generative Adversarial Nets”, NeurIPS 2014

Generative Adversarial Networks: Training Objective

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

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

At start of training, generator is very bad and discriminator can easily tell apart real/fake, so D(G(z)) close to 0



Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

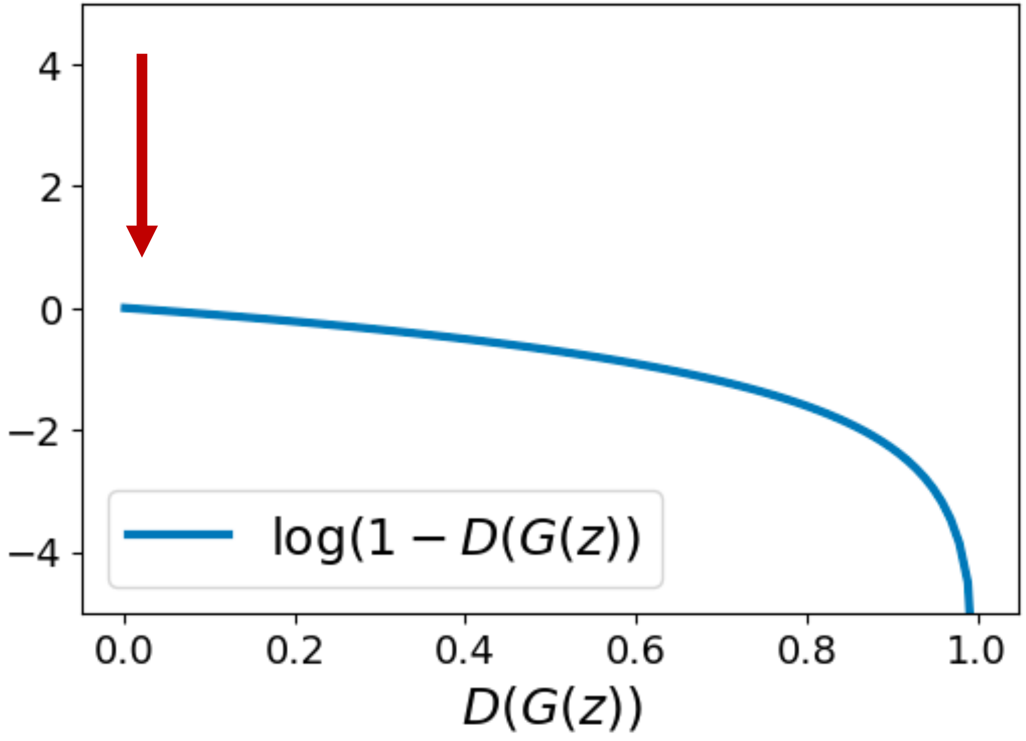
Generative Adversarial Networks: Training Objective

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

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

At start of training, generator is very bad and discriminator can easily tell apart real/fake, so D(G(z)) close to 0

Problem: Vanishing gradients for G



Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks: Training Objective

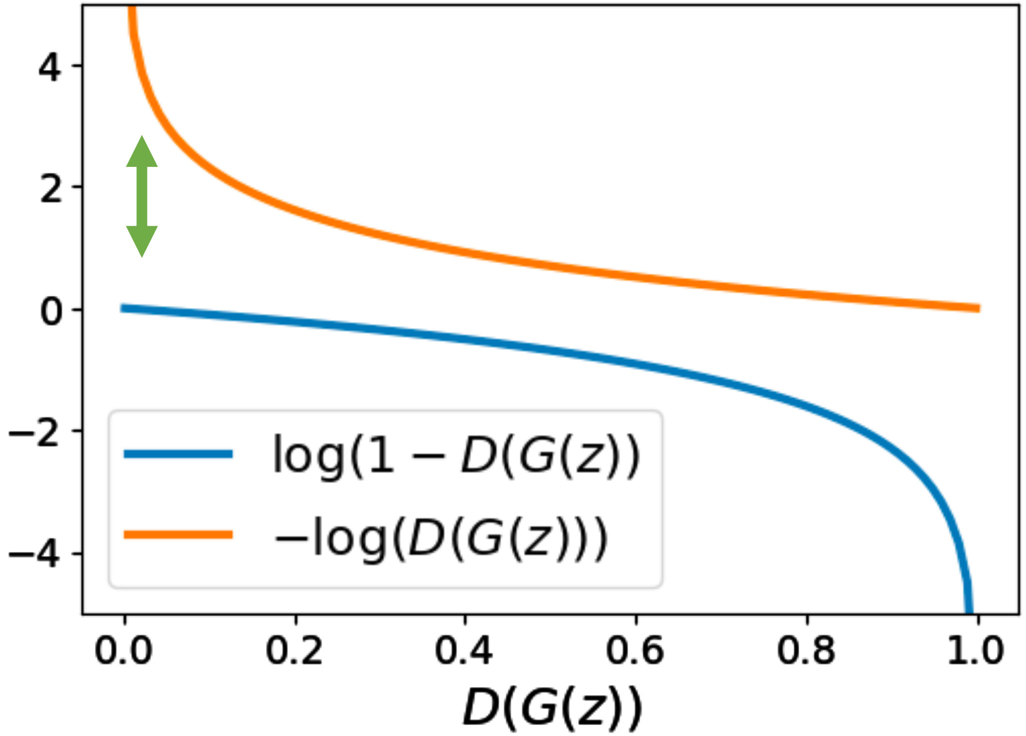
Jointly train generator G and discriminator D with a **minimax game**

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

At start of training, generator is very bad and discriminator can easily tell apart real/fake, so D(G(z)) close to 0

Problem: Vanishing gradients for G

Solution: Right now G is trained to minimize log(1-D(G(z)). Instead, train G to minimize –log(D(G(z)). Then G gets strong gradients at start of training!



Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks: Optimality

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

Why is this particular objective a good idea?

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

This minimax game achieves its global minimum when $p_G = p_{data}$!

Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks: Optimality

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

(Our objective so far)

Generative Adversarial Networks: Optimality

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

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

(Change of variables on second term)

Generative Adversarial Networks: Optimality

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

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

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

(Definition of expectation)

Generative Adversarial Networks: Optimality

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

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

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

(Push $\max_D$ inside integral)

Generative Adversarial Networks: Optimality

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

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

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

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

(Side computation to compute max)

Generative Adversarial Networks: Optimality

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

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

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

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

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

Generative Adversarial Networks: Optimality

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

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

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

$$f(y) = a \log y + b \log(1 - y) \qquad f'(y) = 0 \Leftrightarrow y = \frac{a}{a+b} \text{ (local max)}$$

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

Generative Adversarial Networks: Optimality

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

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

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

$$f(y) = a \log y + b \log(1 - y) \qquad f'(y) = 0 \Leftrightarrow y = \frac{a}{a+b} \text{ (local max)}$$

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

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

Generative Adversarial Networks: Optimality

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

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

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

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

Generative Adversarial Networks: Optimality

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

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

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

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

Generative Adversarial Networks: Optimality

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

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

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

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

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

Generative Adversarial Networks: Optimality

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

$$= \min_{G} \int_{X} \left(p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x) + p_{G}(x)} + p_{G}(x) \log \frac{p_{G}(x)}{p_{data}(x) + p_{G}(x)} \right) dx$$

Generative Adversarial Networks: Optimality

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

$$= \min_{G} \int_{X} \left(p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x) + p_G(x)} + p_G(x) \log \frac{p_G(x)}{p_{data}(x) + p_G(x)} \right) dx$$

$$= \min_{G} \left(E_{x \sim p_{data}} \left[\log \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}\right] + E_{x \sim p_G} \left[\log \frac{p_G(x)}{p_{data}(x) + p_G(x)}\right]\right)$$

(Definition of expectation)

Generative Adversarial Networks: Optimality

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

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

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

(Multiply by a constant)

Generative Adversarial Networks: Optimality

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

$$= \min_{G} \int_{X} \left(p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x) + p_G(x)} + p_G(x) \log \frac{p_G(x)}{p_{data}(x) + p_G(x)} \right) dx$$

$$= \min_{G} \left(E_{x \sim p_{data}} \left[\log \frac{2}{2} \frac{p_{data}(x)}{p_{data}(x) + p_G(x)} \right] + E_{x \sim p_G} \left[\log \frac{2}{2} \frac{p_G(x)}{p_{data}(x) + p_G(x)} \right] \right)$$

$$= \min_{G} \left(E_{x \sim p_{data}} \left[\log \frac{2 * p_{data}(x)}{p_{data}(x) + p_G(x)} \right] + E_{x \sim p_G} \left[\log \frac{2 * p_G(x)}{p_{data}(x) + p_G(x)} \right] - \log 4 \right)$$

Generative Adversarial Networks: Optimality

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

$$= \min_{G} \left(E_{x \sim p_{data}} \left[\log \frac{2 * p_{data}(x)}{p_{data}(x) + p_G(x)} \right] + E_{x \sim p_G} \left[\log \frac{2 * p_G(x)}{p_{data}(x) + p_G(x)} \right] - \log 4 \right)$$

Generative Adversarial Networks: Optimality

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

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

Kullback-Leibler Divergence:

$$KL(p, q) = E_{x \sim p}\left[\log \frac{p(x)}{q(x)}\right]$$

Generative Adversarial Networks: Optimality

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

$$= \min_{G} \left(E_{x \sim p_{data}} \left[\log \frac{2 * p_{data}(x)}{p_{data}(x) + p_G(x)} \right] + E_{x \sim p_G} \left[\log \frac{2 * p_G(x)}{p_{data}(x) + p_G(x)} \right] - \log 4 \right)$$

Kullback-Leibler Divergence:

$$KL(p, q) = E_{x \sim p} \left[\log \frac{p(x)}{q(x)} \right]$$

Generative Adversarial Networks: Optimality

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

$$= \min_{G} \left(E_{x \sim p_{data}} \left[\log \frac{2 * p_{data}(x)}{p_{data}(x) + p_G(x)} \right] + E_{x \sim p_G} \left[\log \frac{2 * p_G(x)}{p_{data}(x) + p_G(x)} \right] - \log 4 \right)$$

$$= \min_{G} \left(KL\left(p_{data}, \frac{p_{data} + p_G}{2} \right) + KL\left(p_G, \frac{p_{data} + p_G}{2} \right) - \log 4 \right)$$

Kullback-Leibler Divergence:

$$KL(p, q) = E_{x \sim p} \left[\log \frac{p(x)}{q(x)} \right]$$

Generative Adversarial Networks: Optimality

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

$$= \min_{G} \left(E_{x \sim p_{data}} \left[\log \frac{2 * p_{data}(x)}{p_{data}(x) + p_G(x)} \right] + E_{x \sim p_G} \left[\log \frac{2 * p_G(x)}{p_{data}(x) + p_G(x)} \right] - \log 4 \right)$$

$$= \min_{G} \left(KL\left(p_{data}, \frac{p_{data} + p_G}{2} \right) + KL\left(p_G, \frac{p_{data} + p_G}{2} \right) - \log 4 \right)$$

Generative Adversarial Networks: Optimality

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

$$= \min_{G} \left(E_{x \sim p_{data}} \left[\log \frac{2 * p_{data}(x)}{p_{data}(x) + p_G(x)} \right] + E_{x \sim p_G} \left[\log \frac{2 * p_G(x)}{p_{data}(x) + p_G(x)} \right] - \log 4 \right)$$

$$= \min_{G} \left(KL\left(p_{data}, \frac{p_{data} + p_G}{2} \right) + KL\left(p_G, \frac{p_{data} + p_G}{2} \right) - \log 4 \right)$$

Jensen-Shannon Divergence:

$$JSD(p, q) = \frac{1}{2} KL\left(p, \frac{p+q}{2} \right) + \frac{1}{2} KL\left(q, \frac{p+q}{2} \right)$$

Generative Adversarial Networks: Optimality

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

$$= \min_{G} \left(E_{x \sim p_{data}} \left[\log \frac{2 * p_{data}(x)}{p_{data}(x) + p_G(x)} \right] + E_{x \sim p_G} \left[\log \frac{2 * p_G(x)}{p_{data}(x) + p_G(x)} \right] - \log 4 \right)$$

$$= \min_{G} \left(KL\left(p_{data}, \frac{p_{data} + p_G}{2} \right) + KL\left(p_G, \frac{p_{data} + p_G}{2} \right) - \log 4 \right)$$

Jensen-Shannon Divergence:

$$JSD(p, q) = \frac{1}{2} KL\left(p, \frac{p+q}{2} \right) + \frac{1}{2} KL\left(q, \frac{p+q}{2} \right)$$

Generative Adversarial Networks: Optimality

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

$$= \min_{G} \left(E_{x \sim p_{data}} \left[\log \frac{2 * p_{data}(x)}{p_{data}(x) + p_G(x)} \right] + E_{x \sim p_G} \left[\log \frac{2 * p_G(x)}{p_{data}(x) + p_G(x)} \right] - \log 4 \right)$$

$$= \min_{G} \left(KL\left(p_{data}, \frac{p_{data} + p_G}{2} \right) + KL\left(p_G, \frac{p_{data} + p_G}{2} \right) - \log 4 \right)$$

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

Jensen-Shannon Divergence:

$$JSD(p, q) = \frac{1}{2} KL\left(p, \frac{p + q}{2} \right) + \frac{1}{2} KL\left(q, \frac{p + q}{2} \right)$$

Generative Adversarial Networks: Optimality

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

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

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

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

JSD is always nonnegative, and zero only when the two distributions are equal! Thus $p_{data} = p_G$ is the global min, QED

Jensen-Shannon Divergence:

$$JSD(p, q) = \frac{1}{2} KL\left(p, \frac{p+q}{2}\right) + \frac{1}{2} KL\left(q, \frac{p+q}{2}\right)$$

Generative Adversarial Networks: Optimality

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

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

Generative Adversarial Networks: Optimality

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

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

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)

Generative Adversarial Networks: Optimality

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

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

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

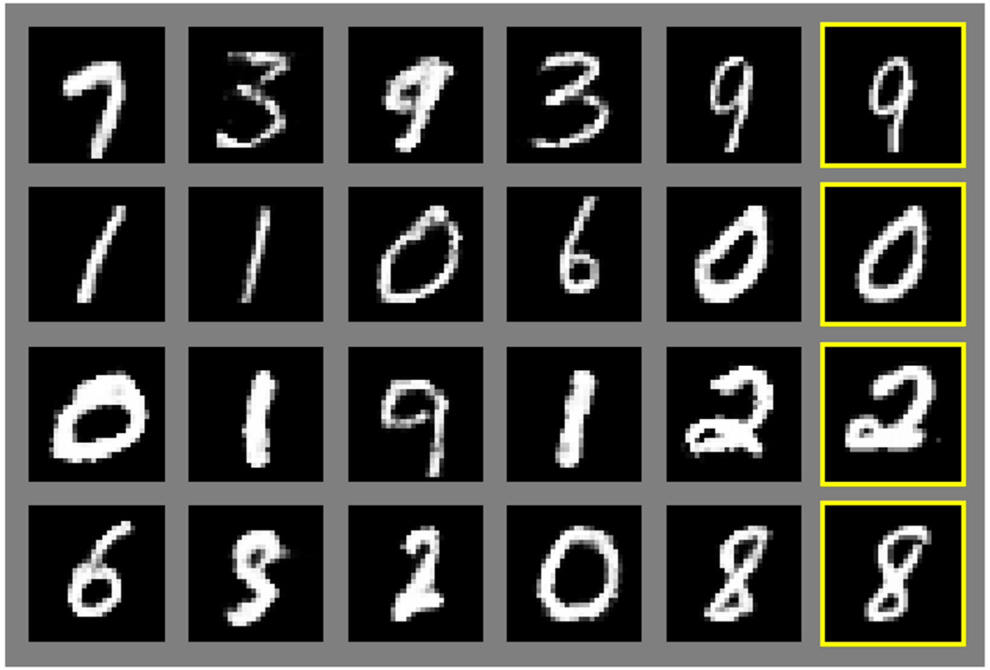
1. $D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$ (Optimal discriminator for any G)
2. $p_G(x) = p_{data}(x)$ (Optimal generator for optimal D)

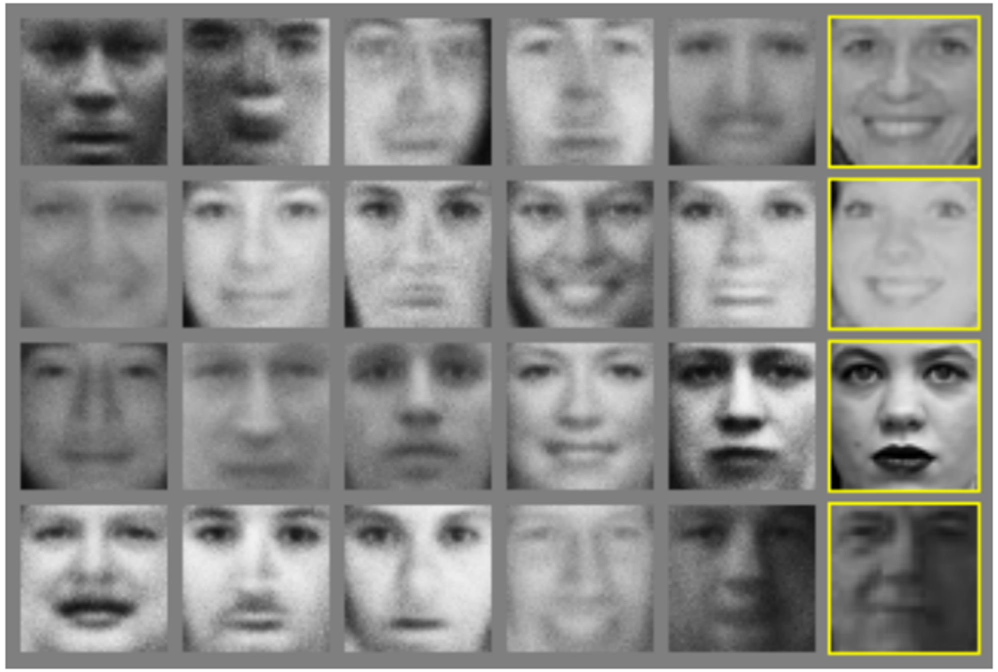
Caveats:

1. G and D are neural nets with fixed architecture. We don't know whether they can actually represent the optimal D and G.
2. This tells us nothing about convergence to the optimal solution

Generative Adversarial Networks: Results

Generated samples

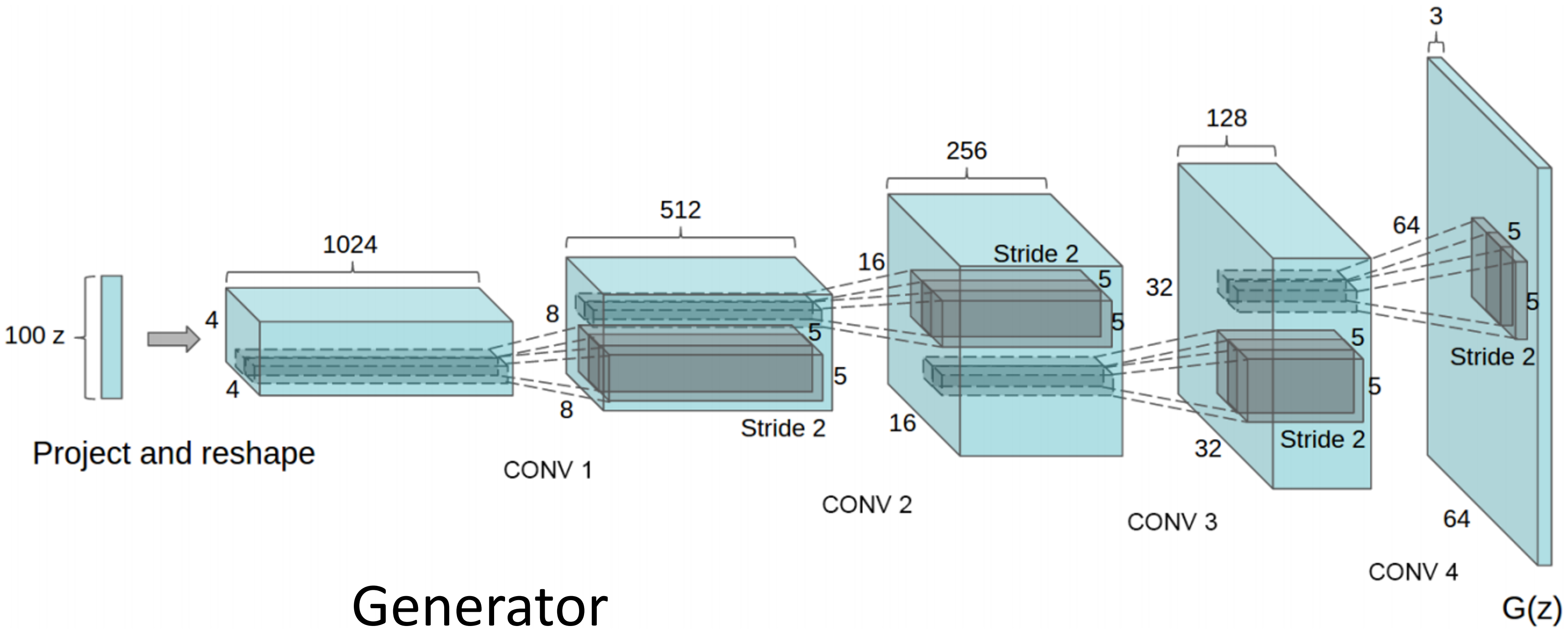




Nearest neighbor from training set

Goodfellow et al, "Generative Adversarial Nets", NeurIPS 2014

Generative Adversarial Networks: DC-GAN



Generator

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

Generative Adversarial Networks: DC-GAN

Samples from the model look much better!

Radford et al, ICLR 2016

Generative Adversarial Networks: Interpolation

Interpolating between points in latent z space

Radford et al, ICLR 2016

Generative Adversarial Networks: Vector Math

Samples from the model

Smiling woman

Neutral woman

Neutral man

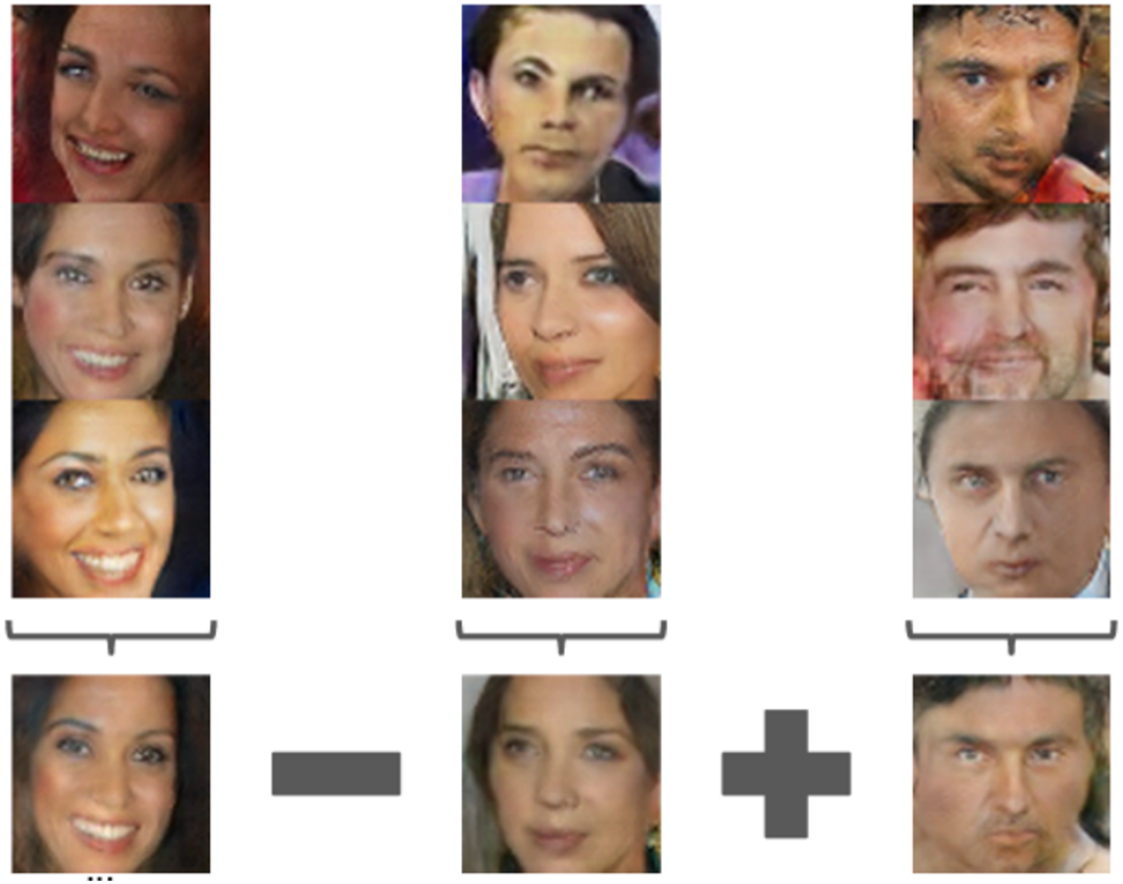
Radford et al, ICLR 2016

Generative Adversarial Networks: Vector Math

Smiling woman | Neutral woman | Neutral man

Samples from the model

Average Z vectors, do arithmetic



Radford et al, ICLR 2016

Generative Adversarial Networks: Vector Math

Smiling woman

Neutral woman

Neutral man

Samples from the model

Smiling Man

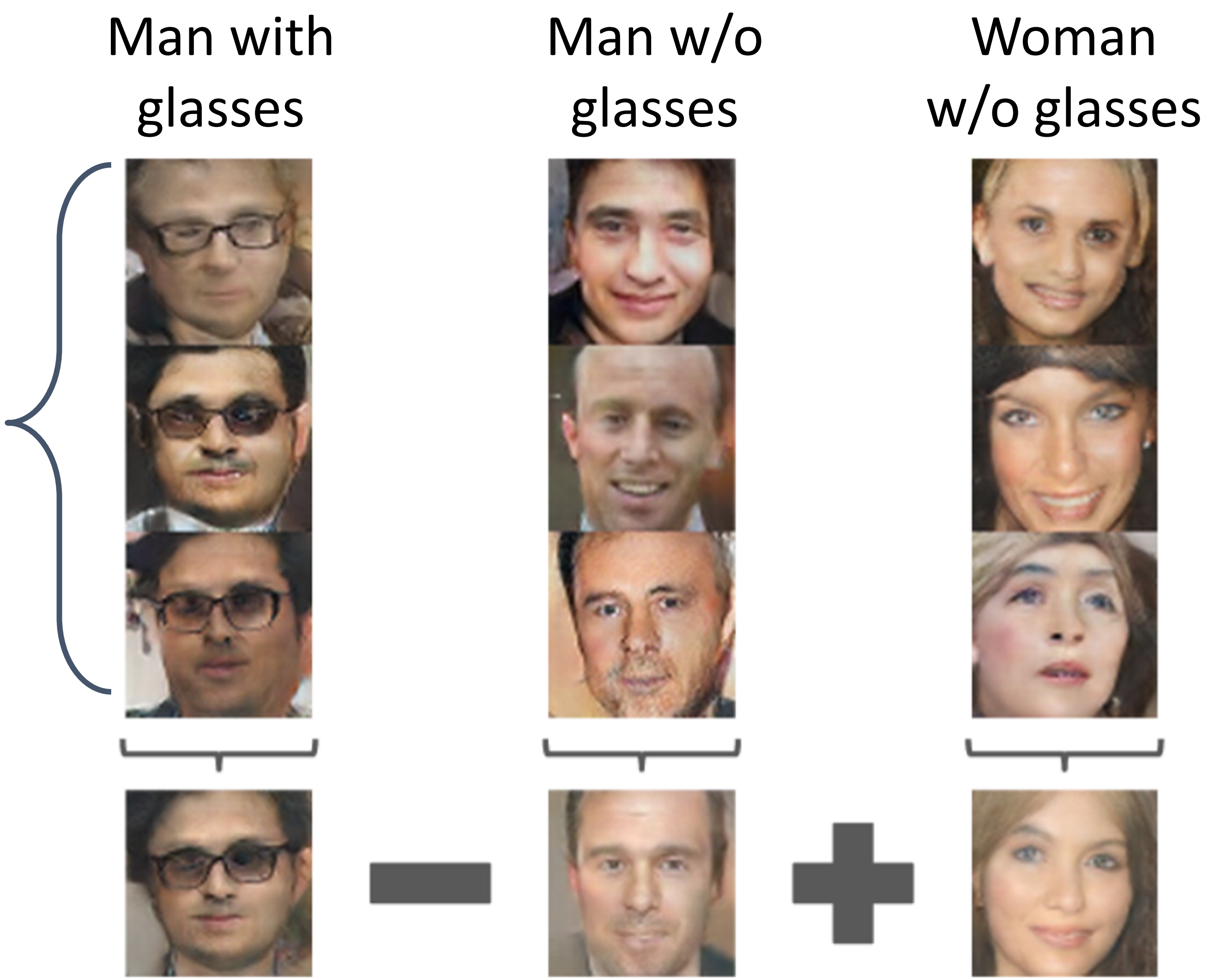
Average Z vectors, do arithmetic

Radford et al, ICLR 2016

Generative Adversarial Networks: Vector Math

Samples from the model

Average Z vectors, do arithmetic

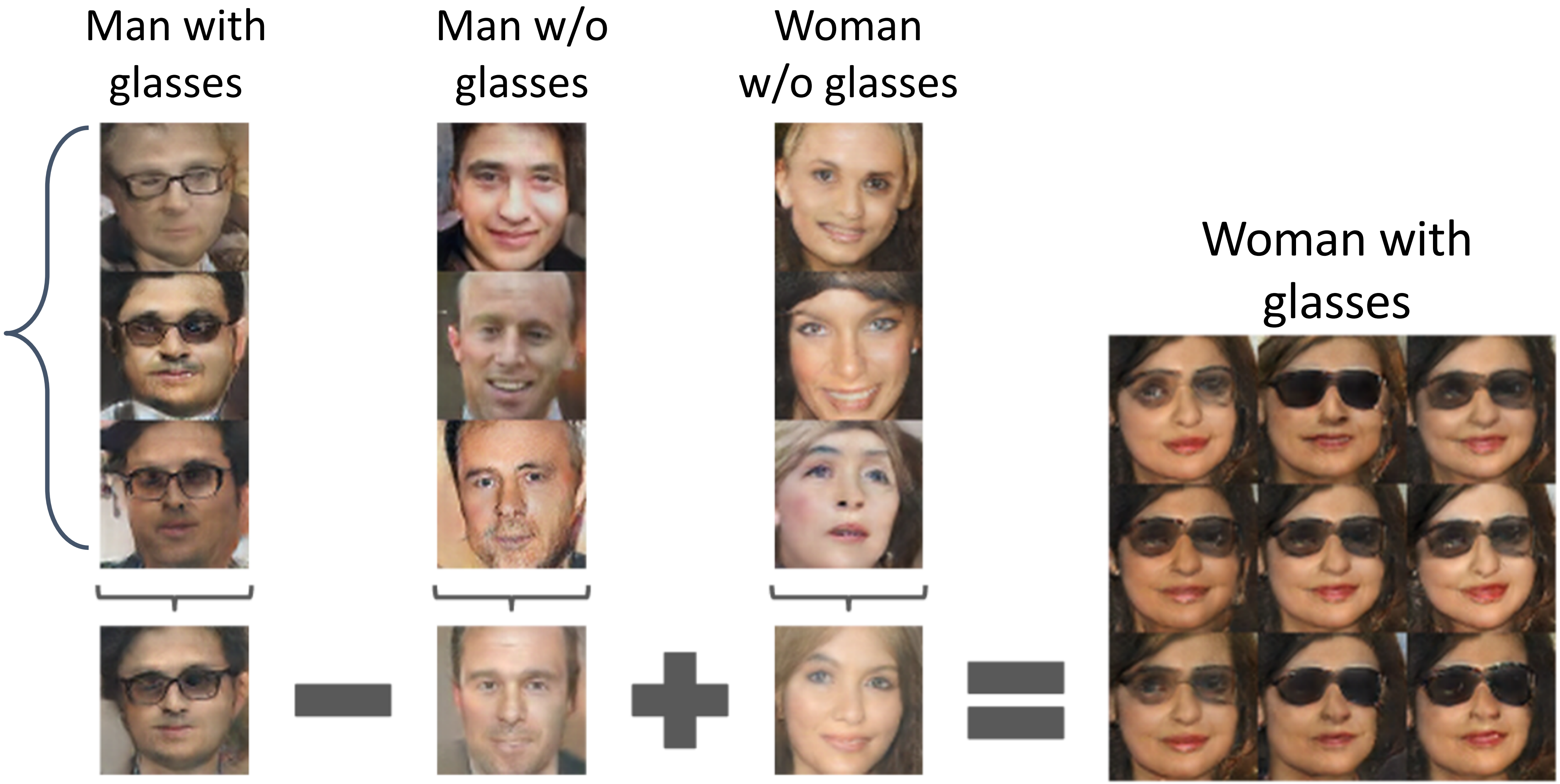


Radford et al, ICLR 2016

Generative Adversarial Networks: Vector Math

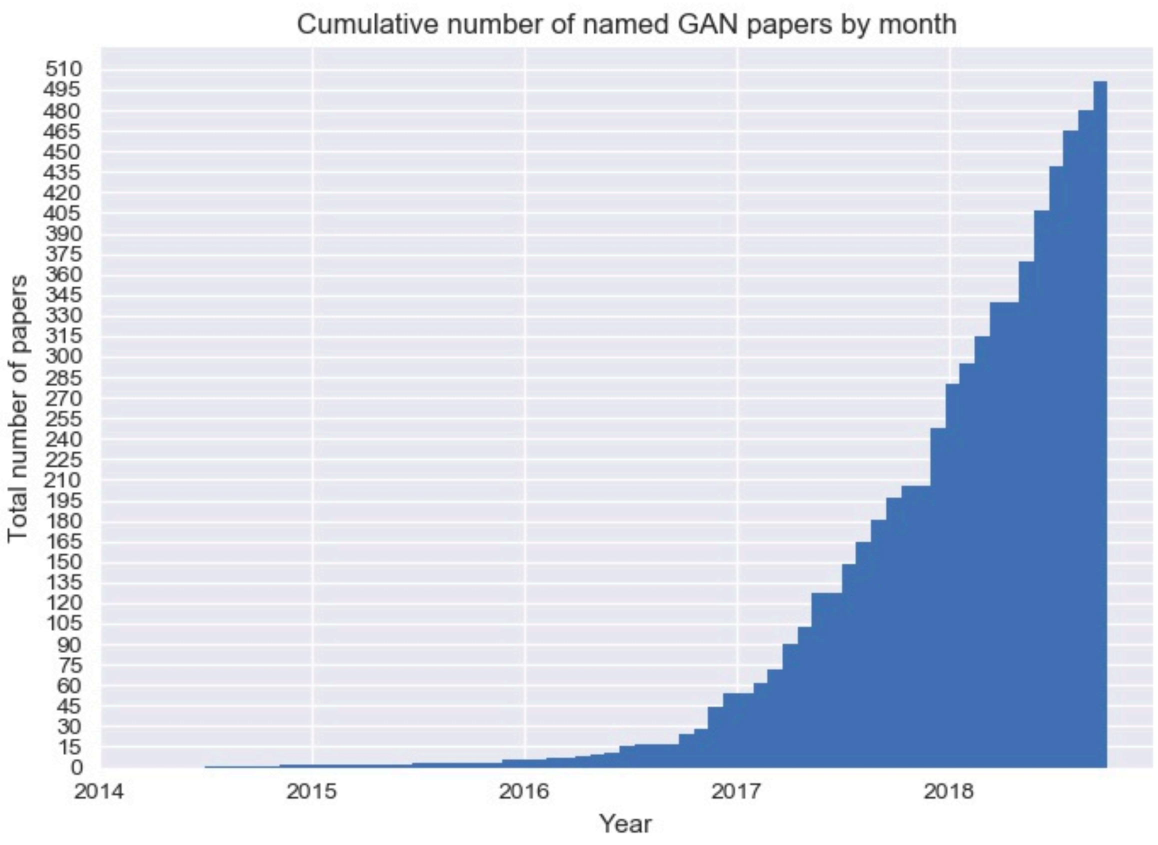
Samples from the model

Average Z vectors, do arithmetic



Radford et al, ICLR 2016

2017 to present: Explosion of GANs



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-PhysNet - 3D-PhysNet: Learning the Intuitive Physics of Non-Rigid Object Deformations
- 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
- acGAN - On-line Adaptative Curriculum Learning for GANs
- ACtuAL - ACtuAL: Actor-Critic Under Adversarial Learning
- AdaGAN - AdaGAN: Boosting Generative Models
- Adaptive GAN - Customizing an Adversarial Example Generator with Class-Conditional GANs
- AdvEntuRe - AdvEntuRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples
- AdvGAN - Generating adversarial examples with adversarial networks
- AE-GAN - AE-GAN: adversarial eliminating with GAN
- AE-OT - Latent Space Optimal Transport for Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AF-DCGAN - AF-DCGAN: Amplitude Feature Deep Convolutional GAN for Fingerprint Construction in Indoor Localization System
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AIM - Generating Informative and Diverse Conversational Responses via Adversarial Information Maximization
- 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
- AlphaGAN - AlphaGAN: Generative adversarial networks for natural image matting
- AM-GAN - Activation Maximization Generative Adversarial Nets
- AmbientGAN - AmbientGAN: Generative models from lossy measurements (github)
- AMC-GAN - Video Prediction with Appearance and Motion Conditions
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- APD - Adversarial Distillation of Bayesian Neural Network Posteriors
- 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
- ASDL-GAN - Automatic Steganographic Distortion Learning Using a Generative Adversarial Network
- 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 (github)
- AttnGAN - AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks (github)
- AVID - AVID: Adversarial Visual Irregularity Detection
- B-DCGAN - B-DCGAN:Evaluation of Binarized DCGAN for FPGA
- b-GAN - Generative Adversarial Nets from a Density Ratio Estimation Perspective
- BAGAN - BAGAN: Data Augmentation with Balancing GAN
- Bayesian GAN - Deep and Hierarchical Implicit Models
- Bayesian GAN - Bayesian GAN (github)
- BCGAN - Bayesian Conditional Generative Adverserial Networks
- BCGAN - Bidirectional Conditional Generative Adversarial networks
- BEAM - Boltzmann Encoded Adversarial Machines
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BEGAN-CS - Escaping from Collapsing Modes in a Constrained Space
- Bellman GAN - Distributional Multivariate Policy Evaluation and Exploration with the Bellman

- BGAN - Binary Generative Adversarial Networks for Image Retrieval (github)
- Bi-GAN - Autonomously and Simultaneously Refining Deep Neural Network Parameters by a Bi-Generative Adversarial Network Aided Genetic Algorithm
- BicycleGAN - Toward Multimodal Image-to-Image Translation (github)
- BiGAN - Adversarial Feature Learning
- BinGAN - BinGAN: Learning Compact Binary Descriptors with a Regularized GAN
- BourGAN - BourGAN: Generative Networks with Metric Embeddings
- BranchGAN - Branched Generative Adversarial Networks for Multi-Scale Image Manifold Learning
- BRE - Improving GAN Training via Binarized Representation Entropy (BRE) Regularization (github)
- BridgeGAN - Generative Adversarial Frontal View to Bird View Synthesis
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- BubGAN - BubGAN: Bubble Generative Adversarial Networks for Synthesizing Realistic Bubbly Flow Images
- BWGAN - Banach Wasserstein GAN
- C-GAN - Face Aging with Contextual Generative Adversarial Nets
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training (github)
- CA-GAN - Composition-aided Sketch-realistic Portrait Generation
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks (github)
- CAN - CAN: Creative Adversarial Networks, Generating Art by Learning About Styles and Deviating from Style Norms
- CapsGAN - CapsGAN: Using Dynamic Routing for Generative Adversarial Networks
- CapsuleGAN - CapsuleGAN: Generative Adversarial Capsule Network
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CatGAN - CatGAN: Coupled Adversarial Transfer for Domain Generation
- CausalGAN - CausalGAN: Learning Causal Implicit Generative Models with Adversarial Training
- CC-GAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks (github)
- cd-GAN - Conditional Image-to-Image Translation
- CDcGAN - Simultaneously Color-Depth Super-Resolution with Conditional Generative Adversarial Network
- CE-GAN - Deep Learning for Imbalance Data Classification using Class Expert Generative Adversarial Network
- CFG-GAN - Composite Functional Gradient Learning of Generative Adversarial Models
- CGAN - Conditional Generative Adversarial Nets
- CGAN - Controllable Generative Adversarial Network
- Chekhov GAN - An Online Learning Approach to Generative Adversarial Networks
- ciGAN - Conditional Infilling GANs for Data Augmentation in Mammogram Classification
- CinCGAN - Unsupervised Image Super-Resolution using Cycle-in-Cycle Generative Adversarial Networks
- CipherGAN - Unsupervised Cipher Cracking Using Discrete GANs
- ClusterGAN - ClusterGAN : Latent Space Clustering in Generative Adversarial Networks
- CM-GAN - CM-GANs: Cross-modal Generative Adversarial Networks for Common Representation Learning
- CoAtt-GAN - Are You Talking to Me? Reasoned Visual Dialog Generation through Adversarial Learning
- CoGAN - Coupled Generative Adversarial Networks
- ComboGAN - ComboGAN: Unrestrained Scalability for Image Domain Translation (github)
- ConceptGAN - Learning Compositional Visual Concepts with Mutual Consistency
- Conditional cycleGAN - Conditional CycleGAN for Attribute Guided Face Image Generation
- constrast-GAN - Generative Semantic Manipulation with Contrasting GAN
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- CorrGAN - Correlated discrete data generation using adversarial training
- Coulomb GAN - Coulomb GANs: Provably Optimal Nash Equilibria via Potential Fields
- Cover-GAN - Generative Steganography with Kerckhoffs' Principle based on Generative Adversarial Networks
- cowboy - Defending Against Adversarial Attacks by Leveraging an Entire GAN
- CR-GAN - CR-GAN: Learning Complete Representations for Multi-view Generation
- Cramèr GAN - The Cramer Distance as a Solution to Biased Wasserstein Gradients
- Cross-GAN - Crossing Generative Adversarial Networks for Cross-View Person Re-identification
- crVAE-GAN - Channel-Recurrent Variational Autoencoders
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CSG - Speech-Driven Expressive Talking Lips with Conditional Sequential Generative Adversarial Networks
- CT-GAN - CT-GAN: Conditional Transformation Generative Adversarial Network for Image Attribute Modification
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

GAN Improvements: Improved Loss Functions

Wasserstein GAN (WGAN)

Arjovsky, Chintala, and Bouttou, "Wasserstein GAN", 2017

WGAN with Gradient Penalty (WGAN-GP)

Gulrajani et al, "Improved Training of Wasserstein GANs", NeurIPS 2017

GAN Improvements: Higher Resolution

256 x 256 bedrooms

1024 x 1024 faces

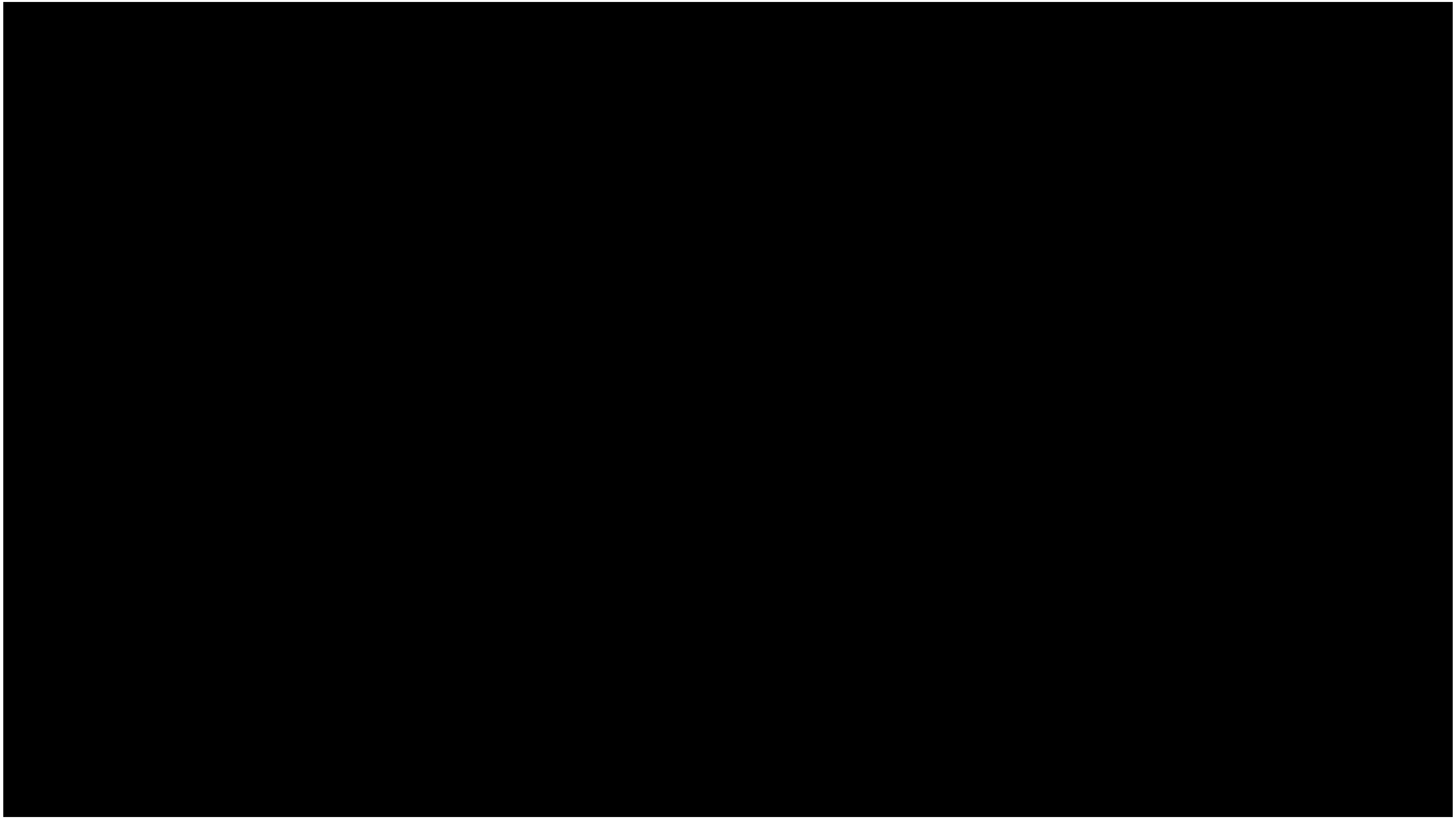
Karras et al, “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018

GAN Improvements: Higher Resolution

512 x 384 cars

1024 x 1024 faces

Karras et al, “A Style-Based Generator Architecture for Generative Adversarial Networks”, CVPR 2019



Karras et al, “A Style-Based Generator Architecture for Generative Adversarial Networks”, CVPR 2019

Current State-of-the-Art: StyleGAN2

Karras et al, “Analyzing and Improving the Image Quality of StyleGAN”, CVPR 2020

Conditional GANs

Recall: Conditional Generative Models learn p(x|y) instead of p(x)

Make generator and discriminator both take label y as an additional input!

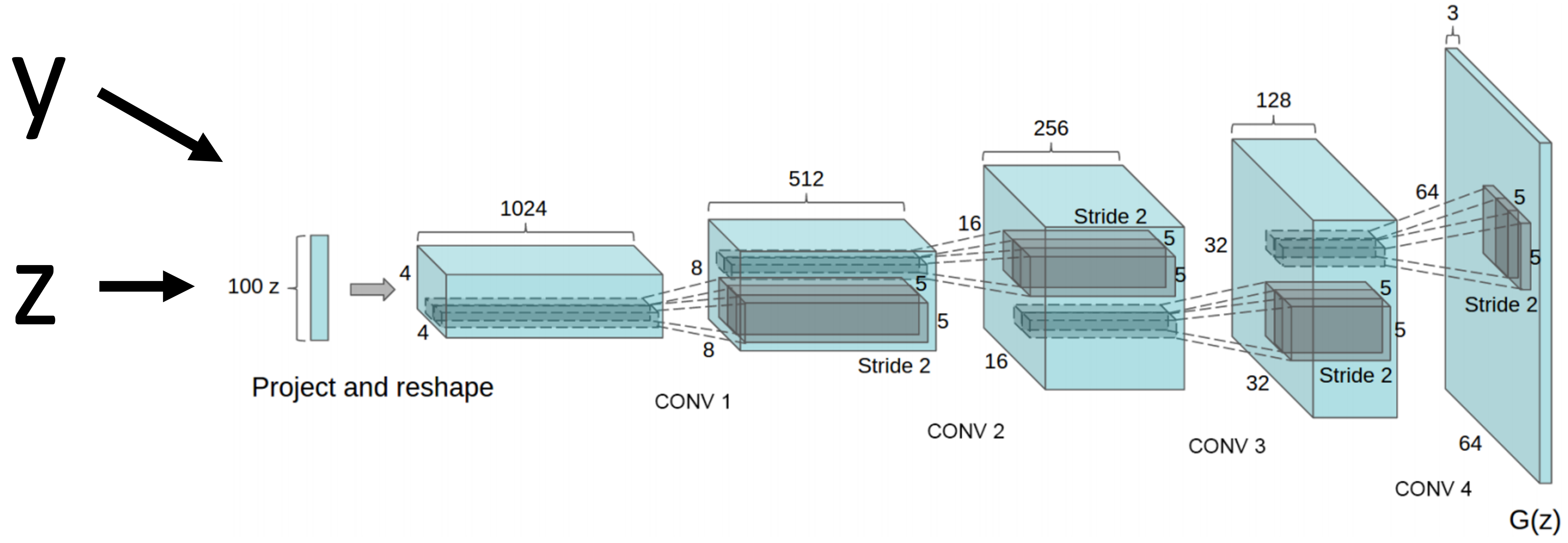


Figure credit: Radford et al, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR 2016

Conditional GANs: Conditional Batch Normalization

Batch Normalization

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N} \left(x_{i,j} - \mu_j\right)^2$$

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Learn a separate scale and shift for each different label y

Conditional Batch Normalization

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N} \left(x_{i,j} - \mu_j\right)^2$$

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

$$y_{i,j} = \boldsymbol{\gamma_j^y} \hat{x}_{i,j} + \boldsymbol{\beta_j^y}$$

Dumoulin et al, "A learned representation for artistic style", ICLR 2017

Conditional GANs: Spectral Normalization

Welsh springer spaniel

Fire truck

Daisy

Miyato et al, “Spectral Normalization for Generative Adversarial Networks”, ICLR 2018

128x128 images on ImageNet

Conditional GANs: Self-Attention

Goldfish

Indigo bunting

Redshank

Saint Bernard

Zhang et al, "Self-Attention Generative Adversarial Networks", ICML 2019

128x128 images on ImageNet

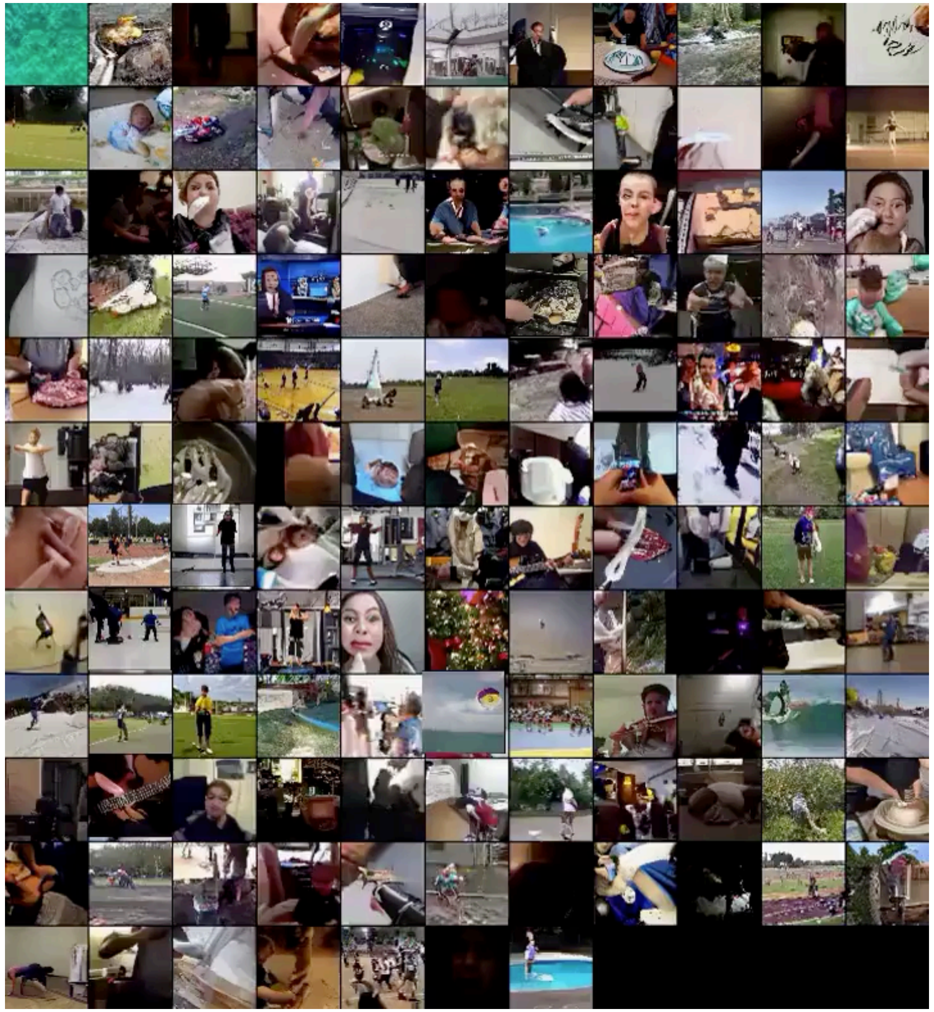
Conditional GANs: BigGAN

Brock et al, "Large Scale GAN Training for High Fidelity Natural Image Synthesis", ICLR 2019

512x512 images on ImageNet

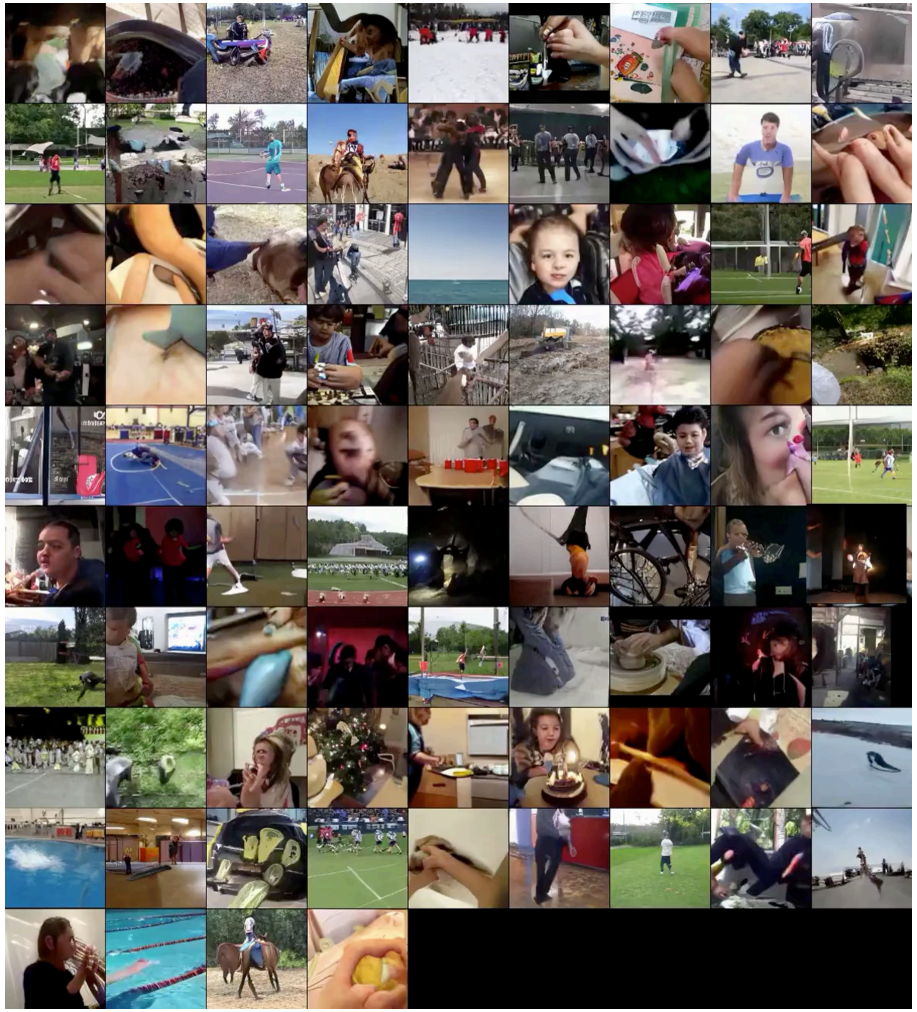
Generating Videos with GANs

Clark et al, "Adversarial Video Generation on Complex Datasets", arXiv 2019



64x64 images, 48 frames

https://drive.google.com/file/d/1FjOQYdUuxPXvS8yeOhXdPQMapUQaklLi/view



128x128 images, 12 frames

https://drive.google.com/file/d/165Yxuvvu3viOy-39LhhSDGtczbWphj_i/view

Conditioning on more than labels! Text to Image

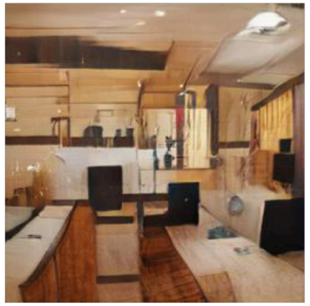
This bird is red and brown in color, with a stubby beak

The bird is short and stubby with yellow on its body

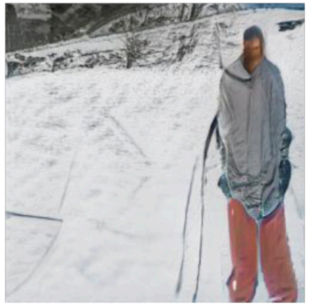
A bird with a medium orange bill white body gray wings and webbed feet

This small black bird has a short, slightly curved bill and long legs

A picture of a very clean living room



A group of people on skis stand in the snow

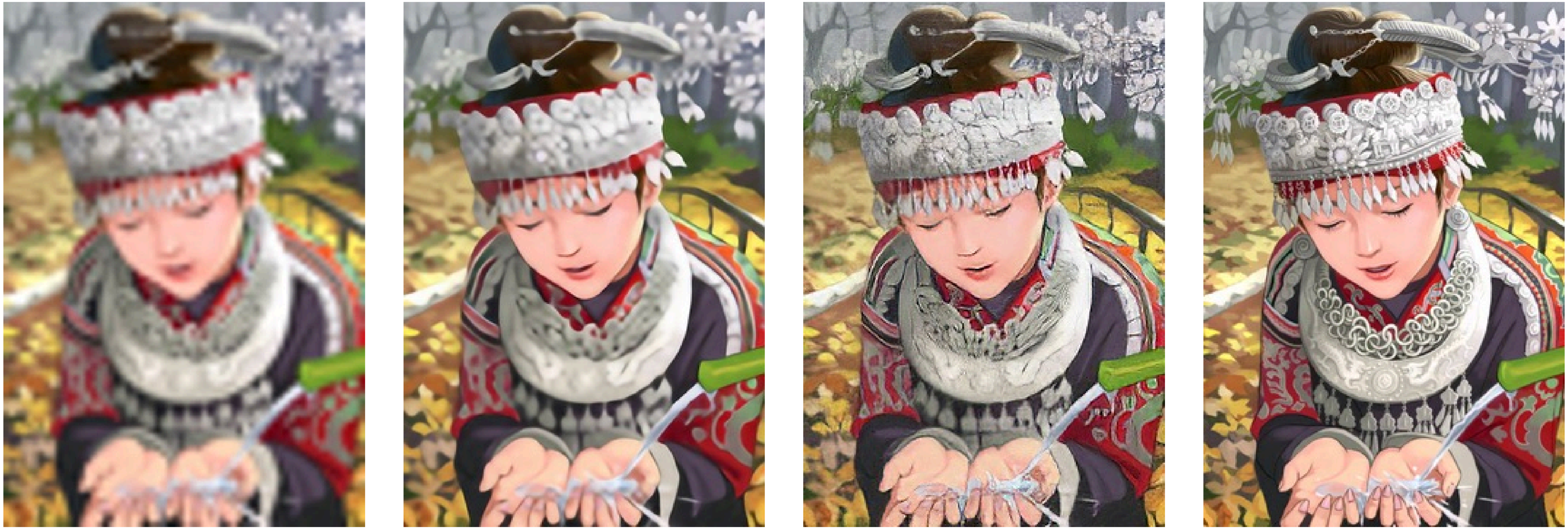


Eggs fruit candy nuts and meat served on white dish

A street sign on a stoplight pole in the middle of a day

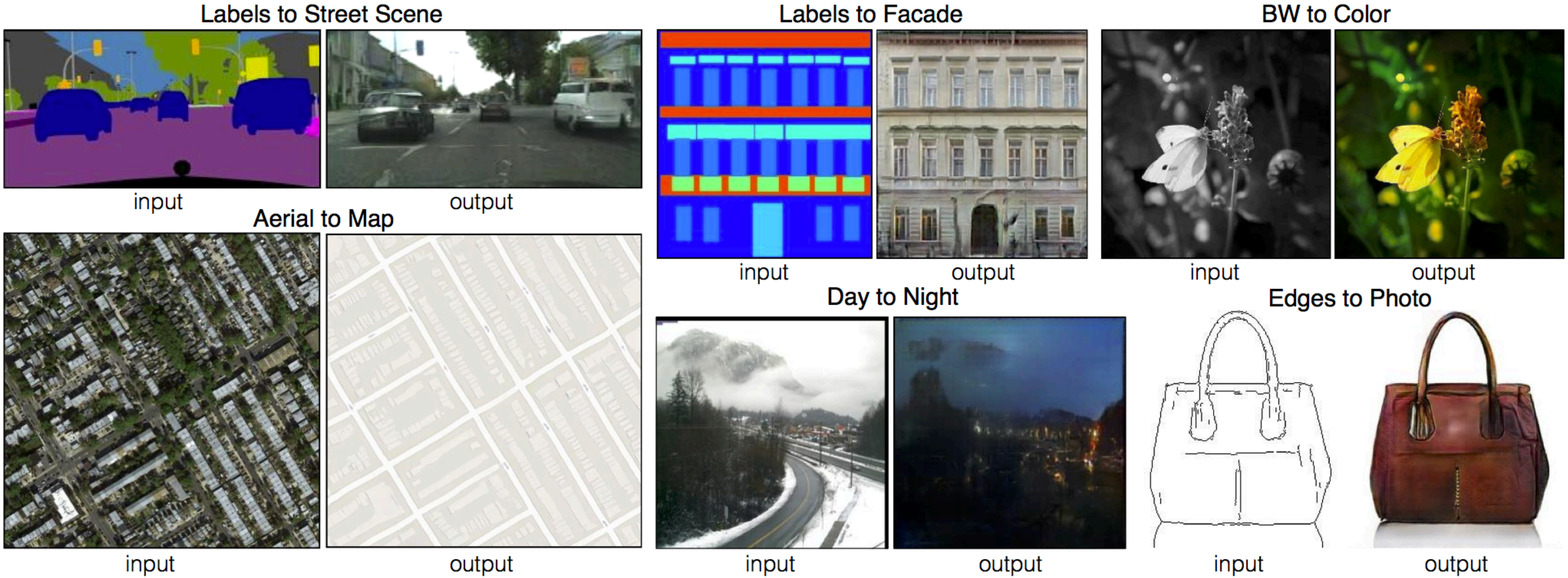
Zhang et al, “StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks.”, TPAMI 2018
Zhang et al, “StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks.”, ICCV 2017
Reed et al, “Generative Adversarial Text-to-Image Synthesis”, ICML 2016

Image Super-Resolution: Low-Res to High-Res



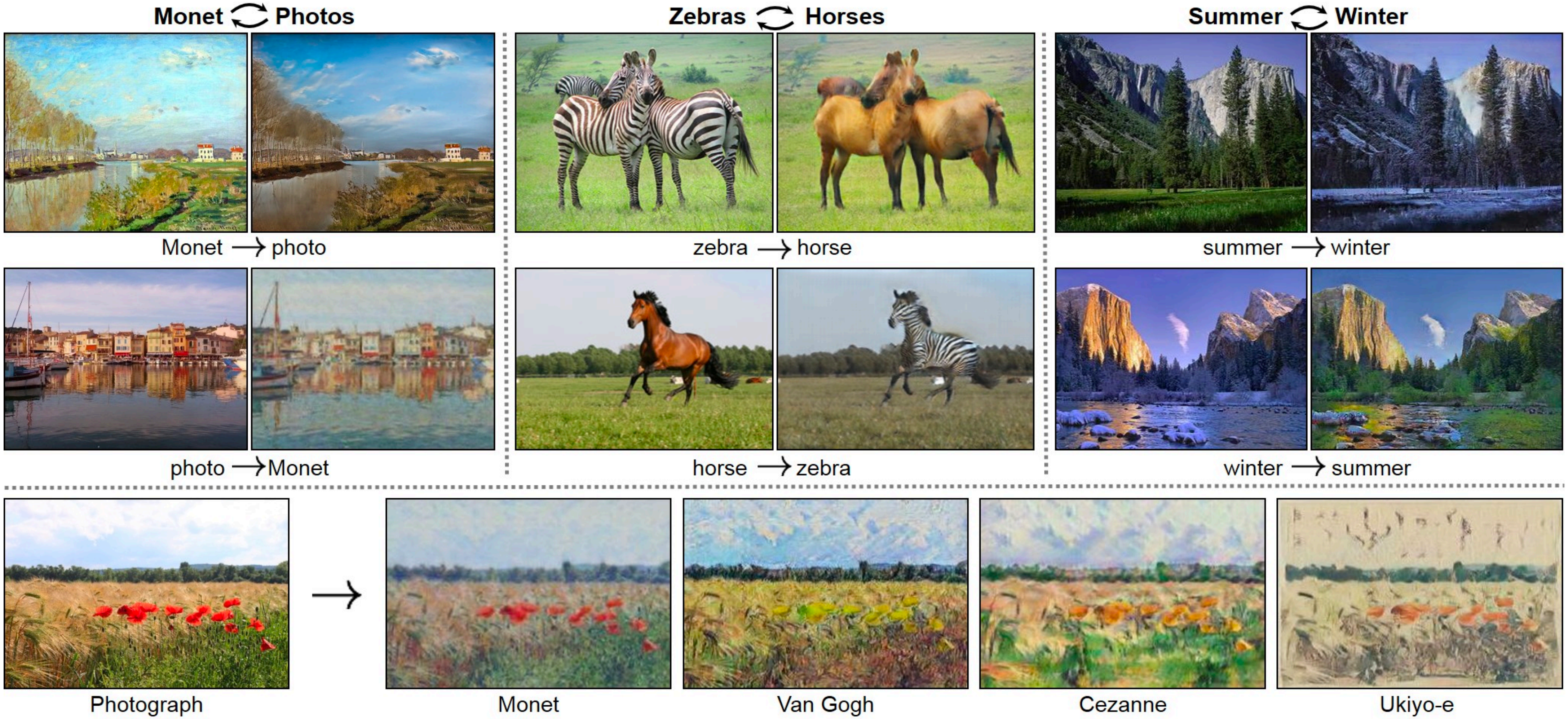
Ledig et al, “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network”, CVPR 2017

Image-to-Image Translation: Pix2Pix



Isola et al, "Image-to-Image Translation with Conditional Adversarial Nets", CVPR 2017

Unpaired Image-to-Image Translation: CycleGAN



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

Unpaired Image-to-Image Translation: CycleGAN

Input Video: Horse

Output Video: Zebra

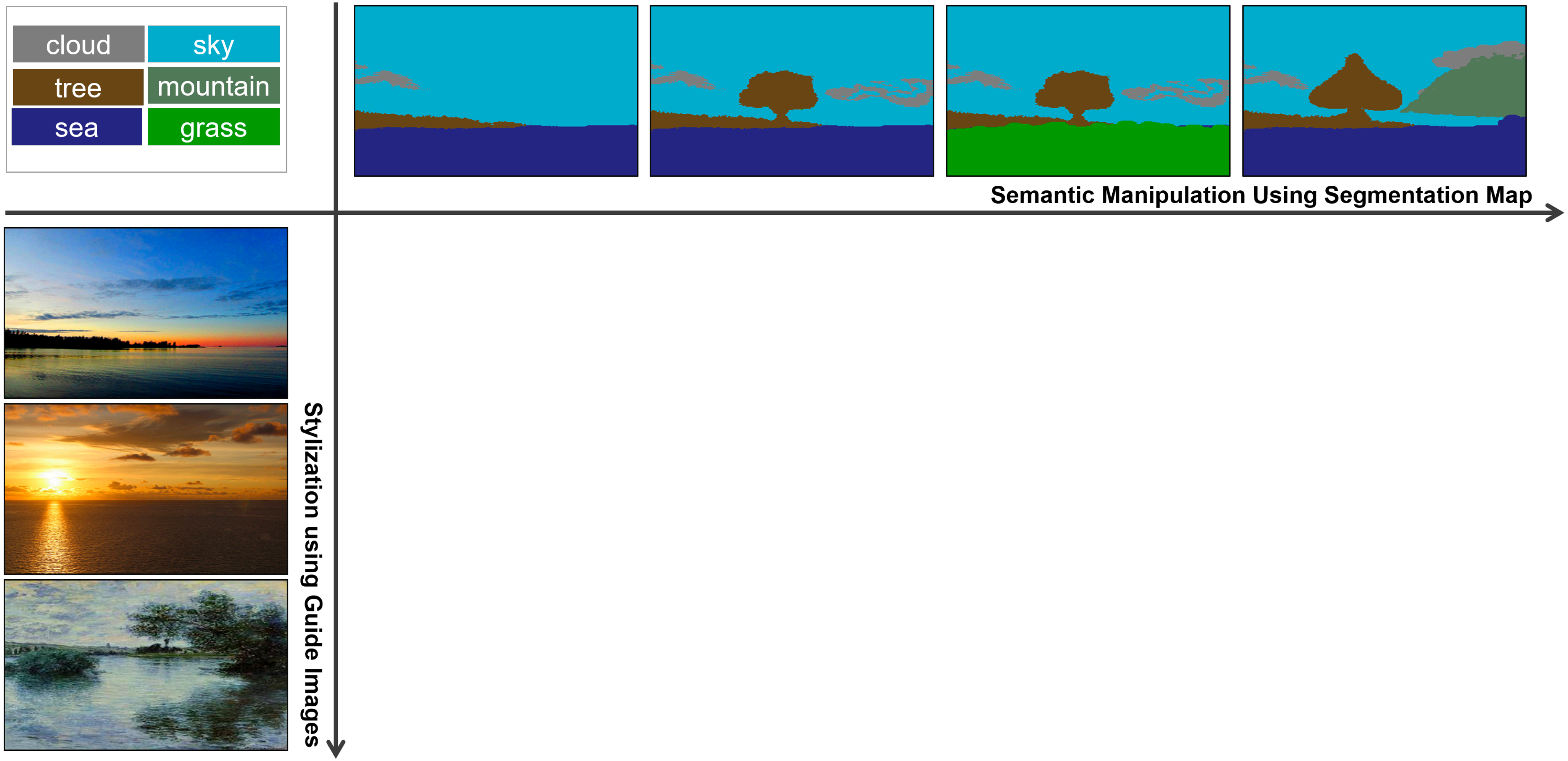
https://www.youtube.com/watch?v=9reHvktowLY

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

Label Map to Image

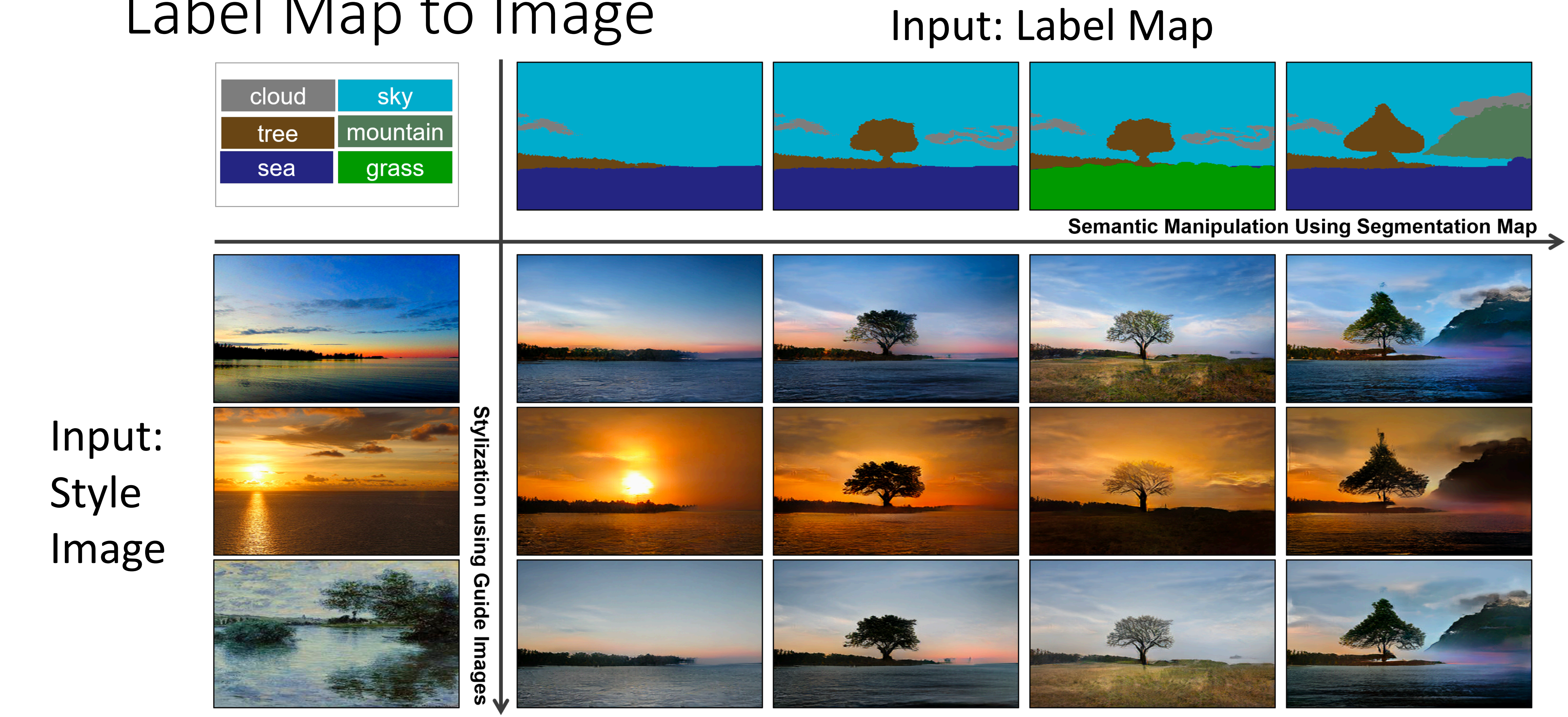
Input: Label Map

Input: Style Image



Park et al, "Semantic Image Synthesis with Spatially-Adaptive Normalization", CVPR 2019

Label Map to Image



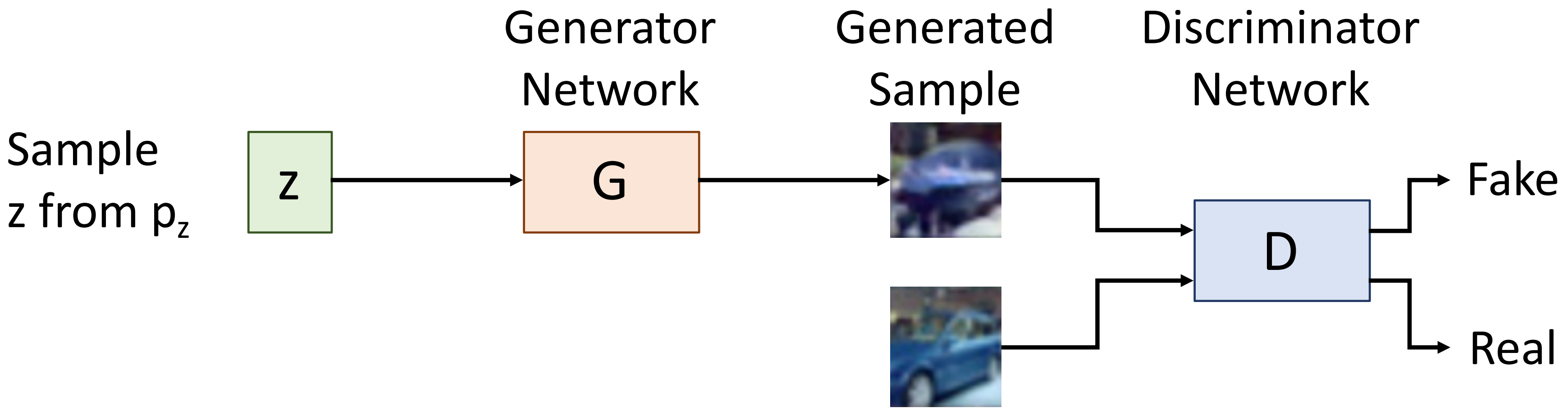
Park et al, "Semantic Image Synthesis with Spatially-Adaptive Normalization", CVPR 2019

GANs: Not just for images! Trajectory Prediction

Gupta, **Johnson**, Li, Savarese, Alahi, "Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks", CVPR 2018

GAN Summary

Jointly train two networks:
Discriminator: Classify data as real or fake
Generator: Generate data that fools the discriminator



Under some assumptions, generator converges to true data distribution
Many applications! Very active area of research!

Taxonomy of Generative Models

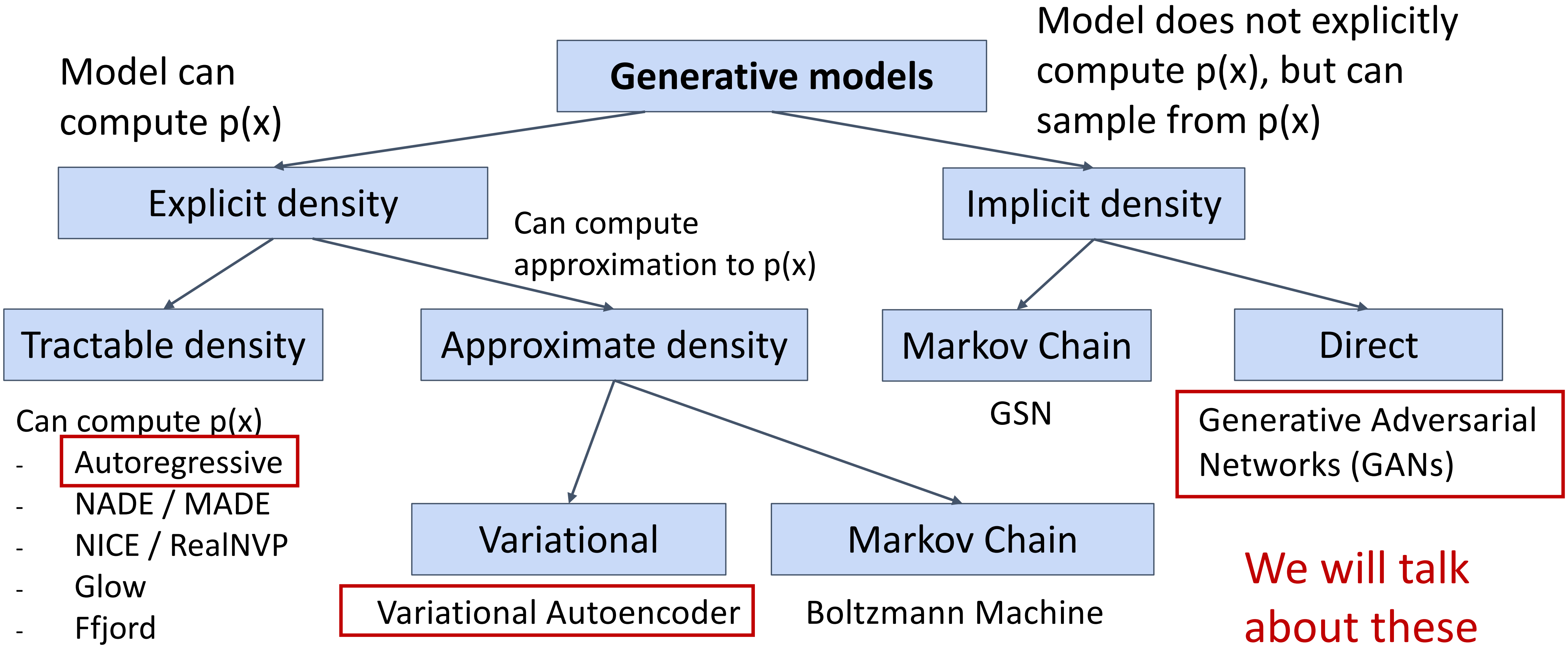


Figure adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Generative Models Summary

Autoregressive Models directly maximize likelihood of training data:

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

Good image quality, can evaluate with perplexity. Slow to generate data, needs tricks to scale up.

Variational Autoencoders introduce a latent z, and maximize a lower bound:

$$p_\theta(x) = \int_Z p_\theta(x|z)p(z)dz \geq E_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

Latent z allows for powerful interpolation and editing applications.

Generative Adversarial Networks give up on modeling p(x), but allow us to draw samples from p(x). Difficult to evaluate, but best qualitative results today

Next Time:
Guest Lecture: