# CS 1674: Generative Models

**PhD. Nils Murrugarra-Llerena**
nem177@pitt.edu



# 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.



→ Cat

Classification

This image is CC0 public domain

Slide credit: Fei-Fei Li

# 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.



A cat sitting on a suitcase on the floor

Image captioning

Caption generated using neuraltalk2
Image is CC0 Public domain

Slide credit: Fei-Fei Li

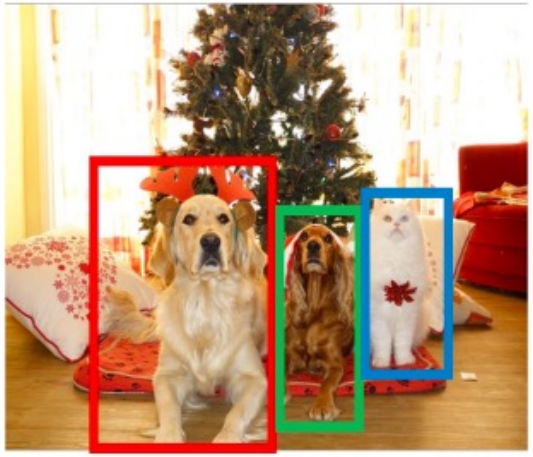
# Supervised vs Unsupervised Learning

**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.



DOG, DOG, CAT

Object Detection

This image is CC0 public domain

Slide credit: Fei-Fei Li

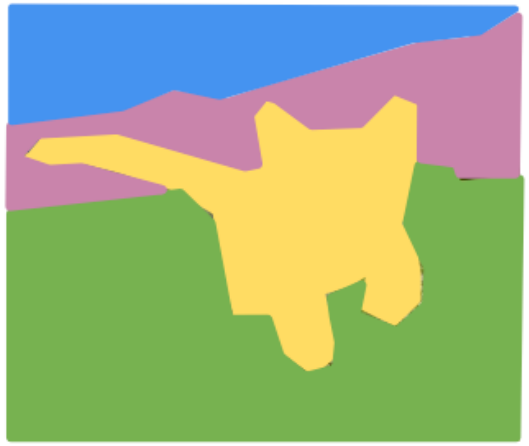
# Supervised vs Unsupervised Learning

**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.



GRASS, CAT, TREE, SKY

Semantic Segmentation

Slide credit: Fei-Fei Li

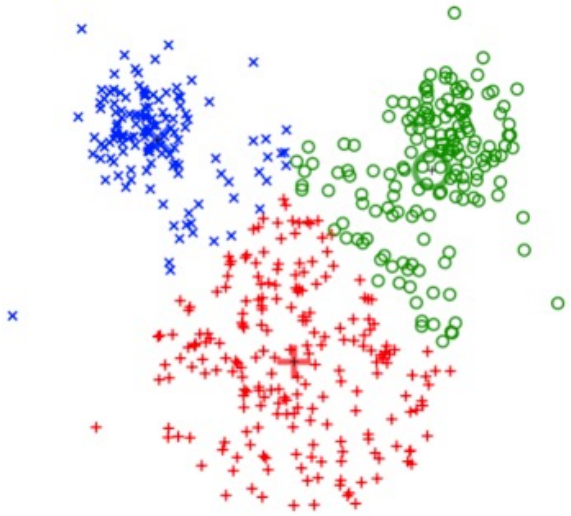
# Supervised vs Unsupervised Learning

**Unsupervised Learning**

**Data**: x
Just data, no labels!

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

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



K-means clustering

This image is CC0 public domain

Slide credit: Fei-Fei Li

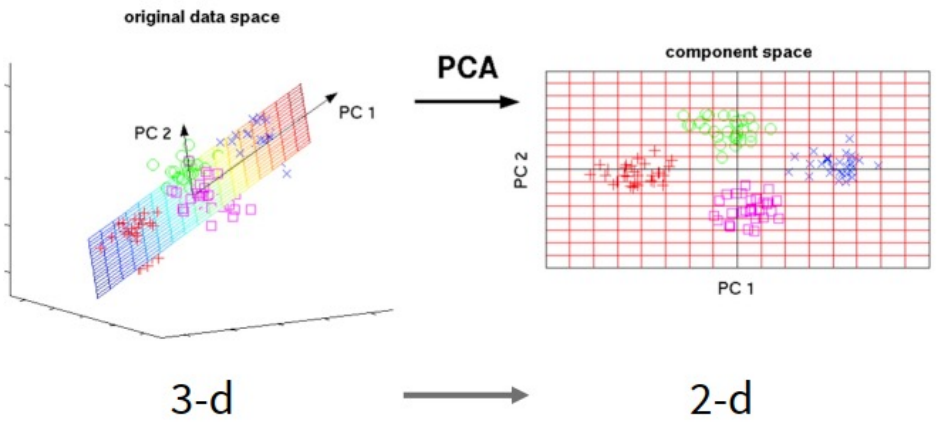
# Supervised vs Unsupervised Learning

**Unsupervised Learning**

**Data**: x
Just data, no labels!

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

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




Principal Component Analysis
(Dimensionality reduction)

This image from Matthias Scholz is
CC0 public domain

Slide credit: Fei-Fei Li

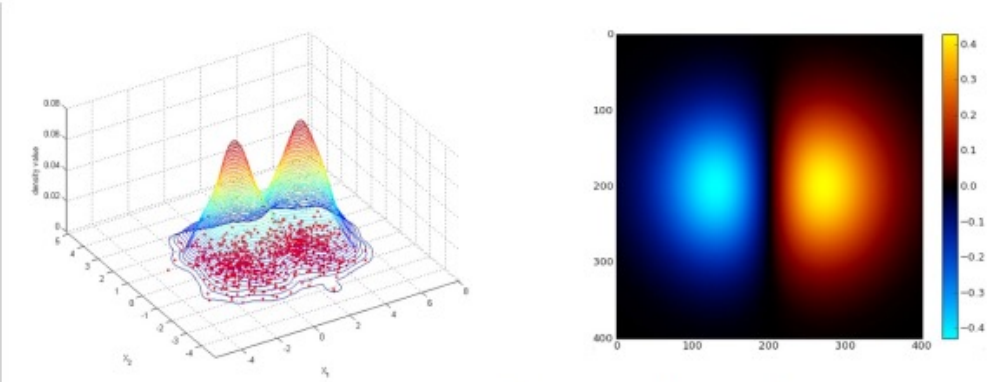
# Supervised vs Unsupervised Learning

**Unsupervised Learning**

**Data**: x
Just data, no labels!

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

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



2-d density estimation

Modeling P(x)

2-d density images left and right are CC0 public domain

Slide credit: Fei-Fei Li

# 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.

Slide credit: Fei-Fei Li

# Generative vs Discriminative 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)

**Data:** x



**Label:** y

Cat

**Probability Recap:**

**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

https://cs231n.stanford.edu/

Slide credit: Fei-Fei Li

# Generative vs Discriminative Models

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

**Generative Model**: Learn a probability distribution p(x)

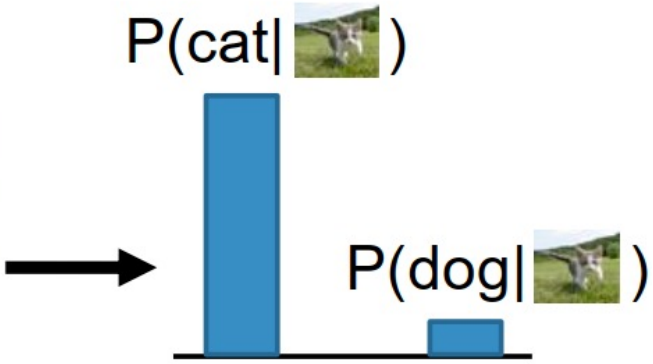
**Conditional Generative Model**: Learn p(x|y)

**Data:** x



**Label:** y

Cat




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

Different values of x **compete** for density

https://cs231n.stanford.edu/

Slide credit: Fei-Fei Li

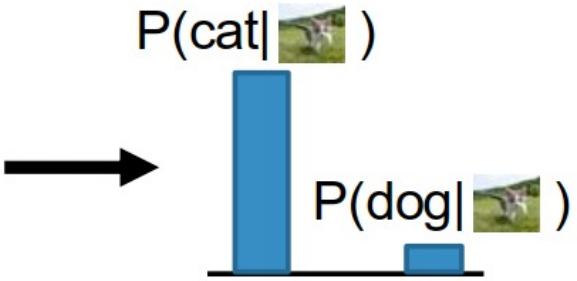
# Generative vs Discriminative Models

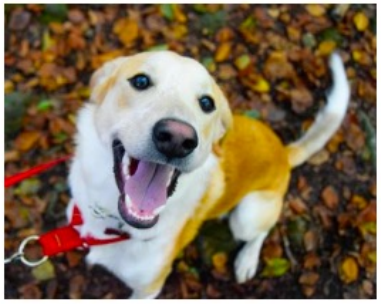
**Discriminative Model:** Learn a probability distribution p(y|x)

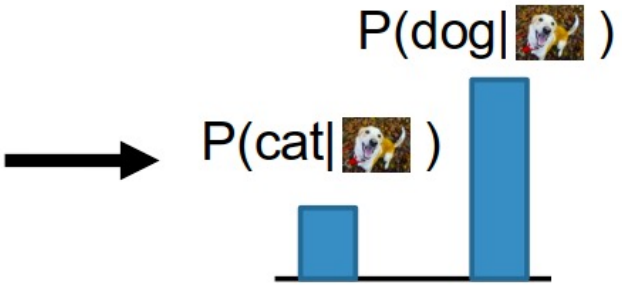
**Generative Model:** Learn a probability distribution p(x)

**Conditional Generative Model:** Learn p(x|y)








Possible **labels** for each image compete for probability.
No competition between **images**

https://cs231n.stanford.edu/

Slide credit: Fei-Fei Li

# Generative vs Discriminative Models

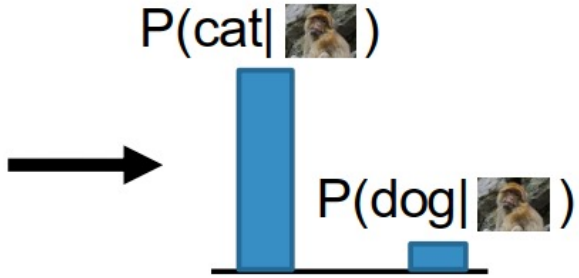
Monkey image is CC0 Public Domain
Abstract image is free to use under the Pixabay license

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

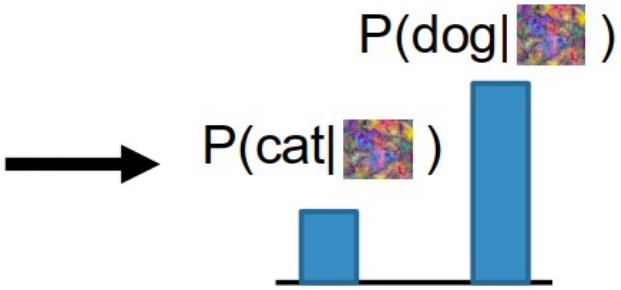
**Generative Model:** Learn a probability distribution p(x)

**Conditional Generative Model:** Learn p(x|y)








No way to handle unreasonable inputs; must give a label distribution for all possible inputs



Slide credit: Fei-Fei Li

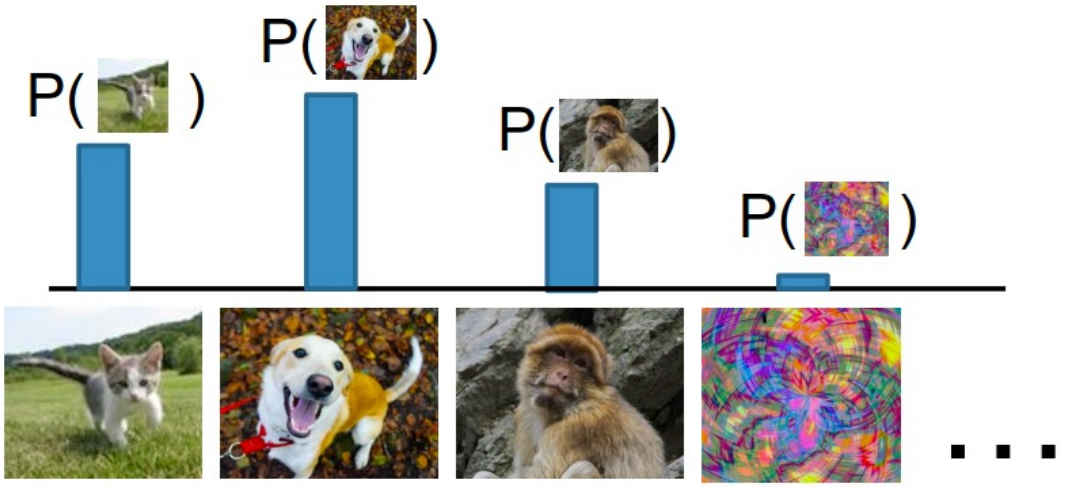
# Generative vs Discriminative Models

Cat image is CC0 public domain
Dog image is CC0 Public Domain
Monkey image is CC0 Public Domain
Abstract image is free to use under the Pixabay license

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

**Generative Model:** Learn a probability distribution p(x)

**Conditional Generative Model:** Learn p(x|y)




All possible **images** compete for probability mass

https://cs231n.stanford.edu/

Slide credit: Fei-Fei Li

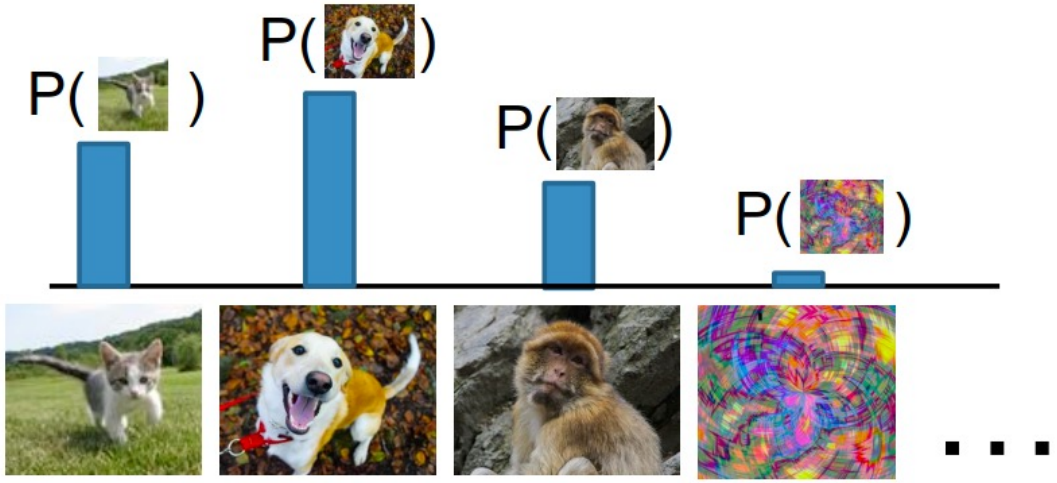
# Generative vs Discriminative Models

Cat image is CC0 public domain
Dog image is CC0 Public Domain
Monkey image is CC0 Public Domain
Abstract image is free to use under the Pixabay license

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

**Generative Model**: Learn a probability distribution p(x)

**Conditional Generative Model**: Learn p(x|y)




All possible **images** compete for probability mass

Requires deep understanding: Is a dog more likely to sit or stand? Is a 3-legged dog more likely than a 3-armed monkey?

https://cs231n.stanford.edu/

Slide credit: Fei-Fei Li

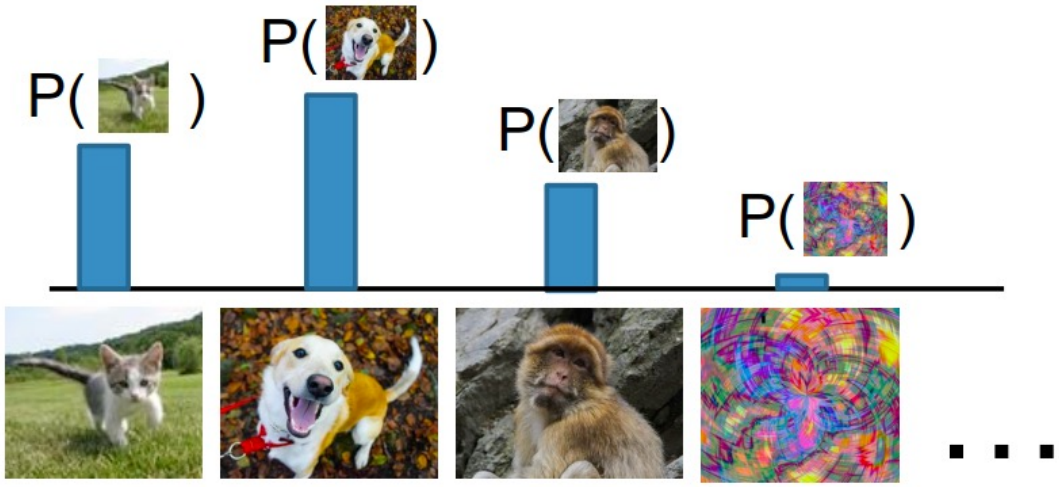
# Generative vs Discriminative Models

Cat image is CC0 public domain
Dog image is CC0 Public Domain
Monkey image is CC0 Public Domain
Abstract image is free to use under the Pixabay license

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

**Generative Model**: Learn a probability distribution p(x)

**Conditional Generative Model**: Learn p(x|y)




All possible **images** compete for probability mass

Model can "reject" unreasonable inputs by giving them small probability mass

https://cs231n.stanford.edu/

Slide credit: Fei-Fei Li

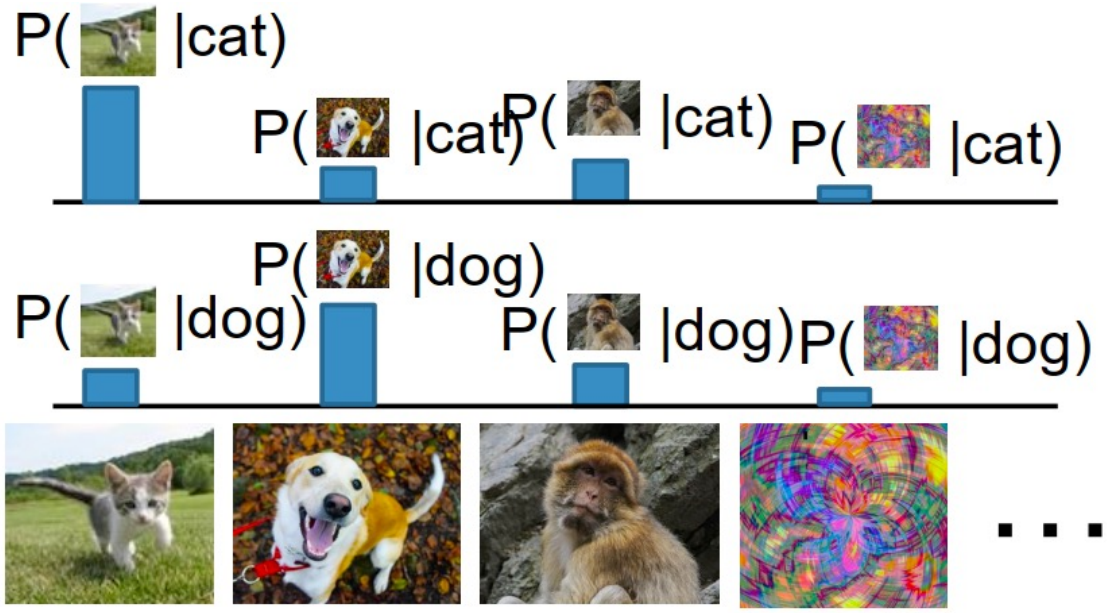
# Generative vs Discriminative Models

Cat image is CC0 public domain
Dog image is CC0 Public Domain
Monkey image is CC0 Public Domain
Abstract image is free to use under the Pixabay license

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

**Generative Model:** Learn a probability distribution p(x)

**Conditional Generative Model:** Learn p(x|y)




Each possible **label** induces a competition across all possible **images**

https://cs231n.stanford.edu/

Slide credit: Fei-Fei Li

# Generative vs Discriminative 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**:

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

https://cs231n.stanford.edu/

Slide credit: Fei-Fei Li

# Generative vs Discriminative 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**:

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

Discriminative Model

Conditional Generative Model

Prior over labels

(Unconditional) Generative Model

We can build a conditional generative model from other components … but not common in practice

https://cs231n.stanford.edu/

Slide credit: Fei-Fei Li

# Generative vs Discriminative Models

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

→ Assign labels to data
Feature learning (with labels)

**Generative Model**: Learn a probability distribution p(x)

→ Detect outliers
Feature learning (without labels)
**Sample** to generate new data

**Conditional Generative Model**: Learn p(x|y)

→ Assign labels while rejecting outliers
**Sample** to generate data from labels

https://cs231n.stanford.edu/

Slide credit: Fei-Fei Li

To join, go to: ahaslides.com/IKCMO

AhaSlides

# How do generative models differ from discriminative models in terms of probability distributions?

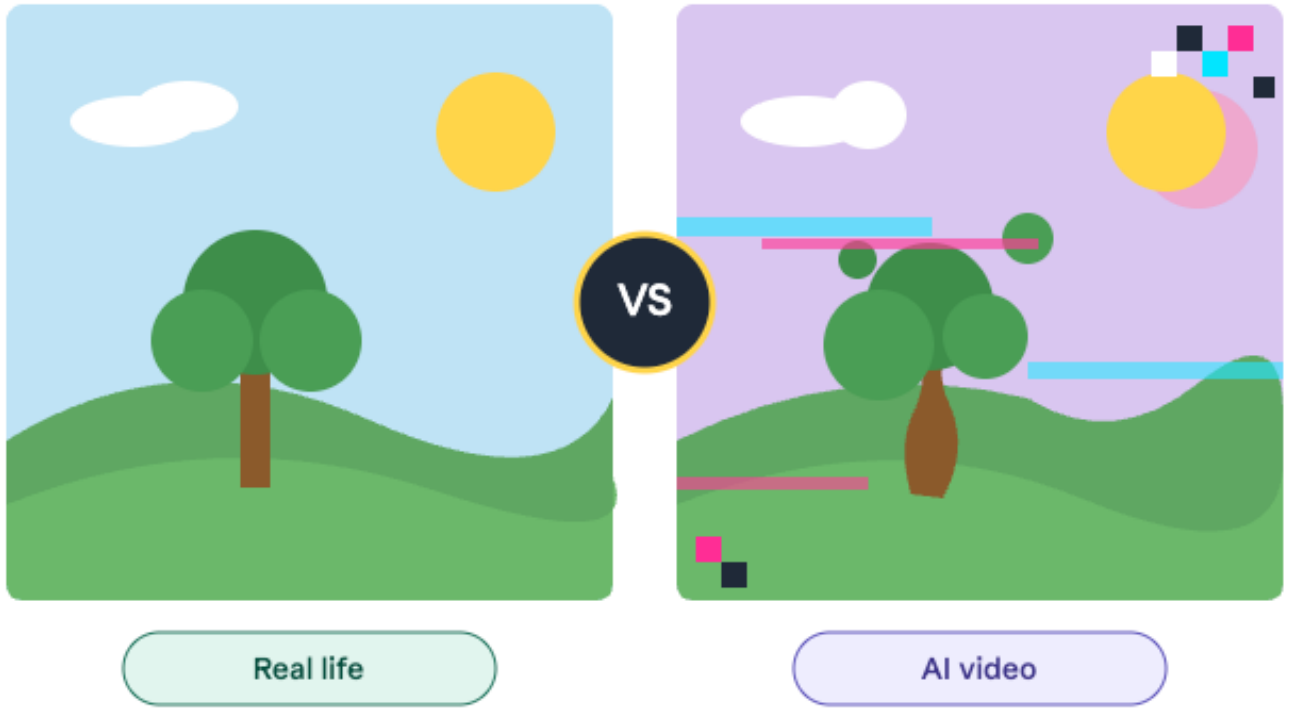
Edit slides

Add slides

11 0 /100

# Generative Modeling

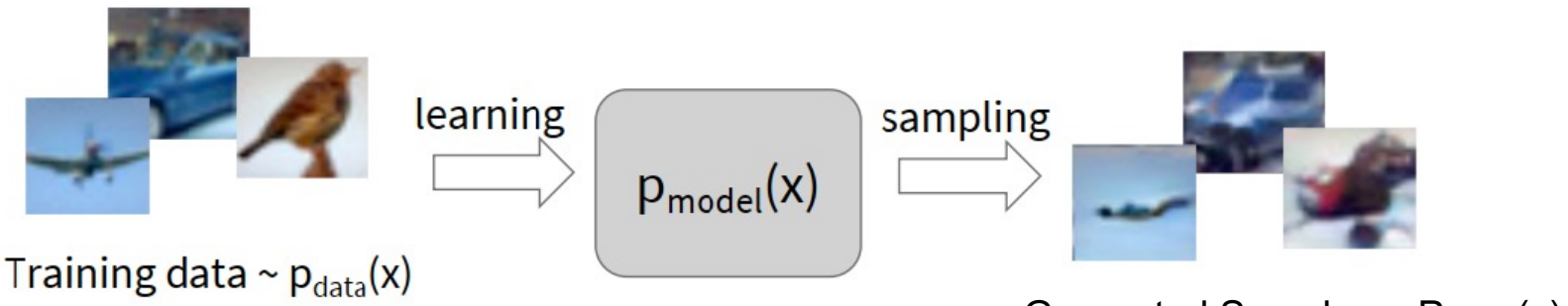
**Can You Spot the Fake? AI Video vs. Real Life Challenge!**




https://www.youtube.com/watch?v=oWLHAuUoYqQ

# Generative Modeling

Given training data, generate new samples from same distribution




Objectives:

1. Learn $p_{model}(x)$ that approximates $p_{data}(x)$
2. Sampling new x from $p_{model}(x)$

Slide credit: Fei-Fei Li

# Why Generative Models? Debiasing

Capable of uncovering **underlying features** in a dataset



Homogeneous skin color, pose

VS



Diverse skin color, pose, illumination

How can we use this information to create fair and representative datasets?

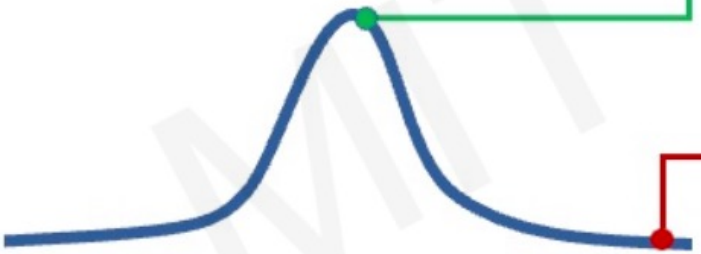
# Why Generative Models? Outlier Detection

- **Problem:** How can we detect when we encounter something new or rare?
- **Strategy:** Leverage generative models, detect outliers in the distribution
- Use outliers during training to improve even more!

**95% of Driving Data:**
(1) sunny, (2) highway, (3) straight road



**Detect outliers to avoid unpredictable behavior when training**





Edge Cases



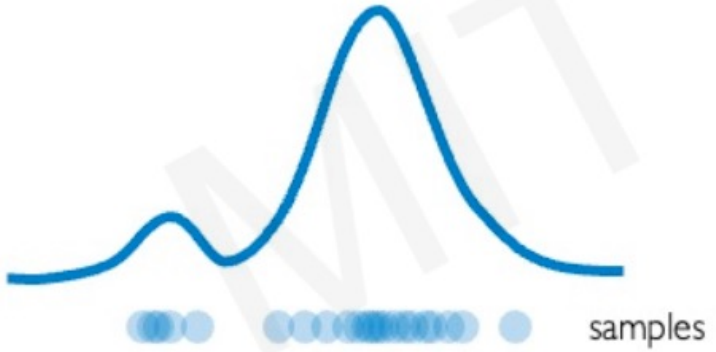
Harsh Weather



Pedestrians

# Why Generative Models? Sample Generation

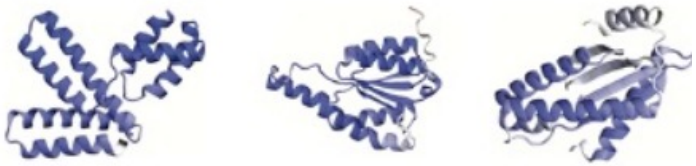
- **Generative models learn probability distributions**
- **Sampling** from that distribution → new data instances
- **Backbone of Generative AI:** generate language, images, and more





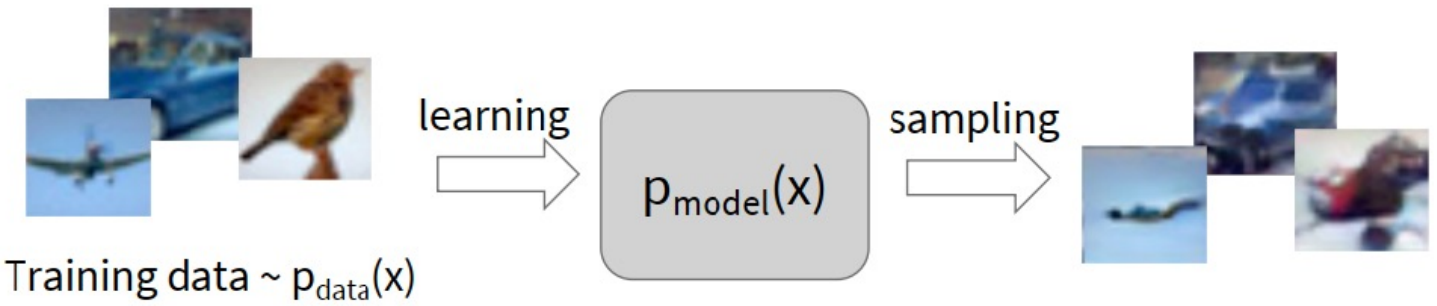






# Generative Modeling

Given training data, generate new samples from same distribution

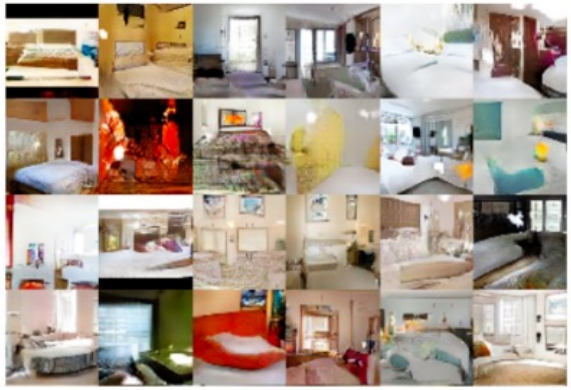
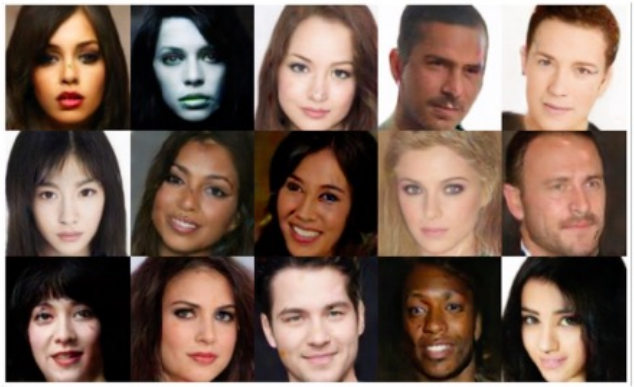



Formulate as density estimation problems:

- 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 credit: Fei-Fei Li

# Generative Modeling: Applications


- Realistic samples for artwork, super-resolution, colorization, etc.
- Learn useful features for downstream tasks such as classification.
- Getting insights from high-dimensional data (physics, medical imaging, etc.)
- Modeling physical world for simulation and planning (robotics and reinforcement learning applications)
- Many more ...

Figures from L-R are copyright: (1) Alec Radford et al. 2016; (2) Phillip Isola et al. 2017. Reproduced from (3) BAIR Blog.

Slide credit: Fei-Fei Li

# Taxonomy of Generative Models

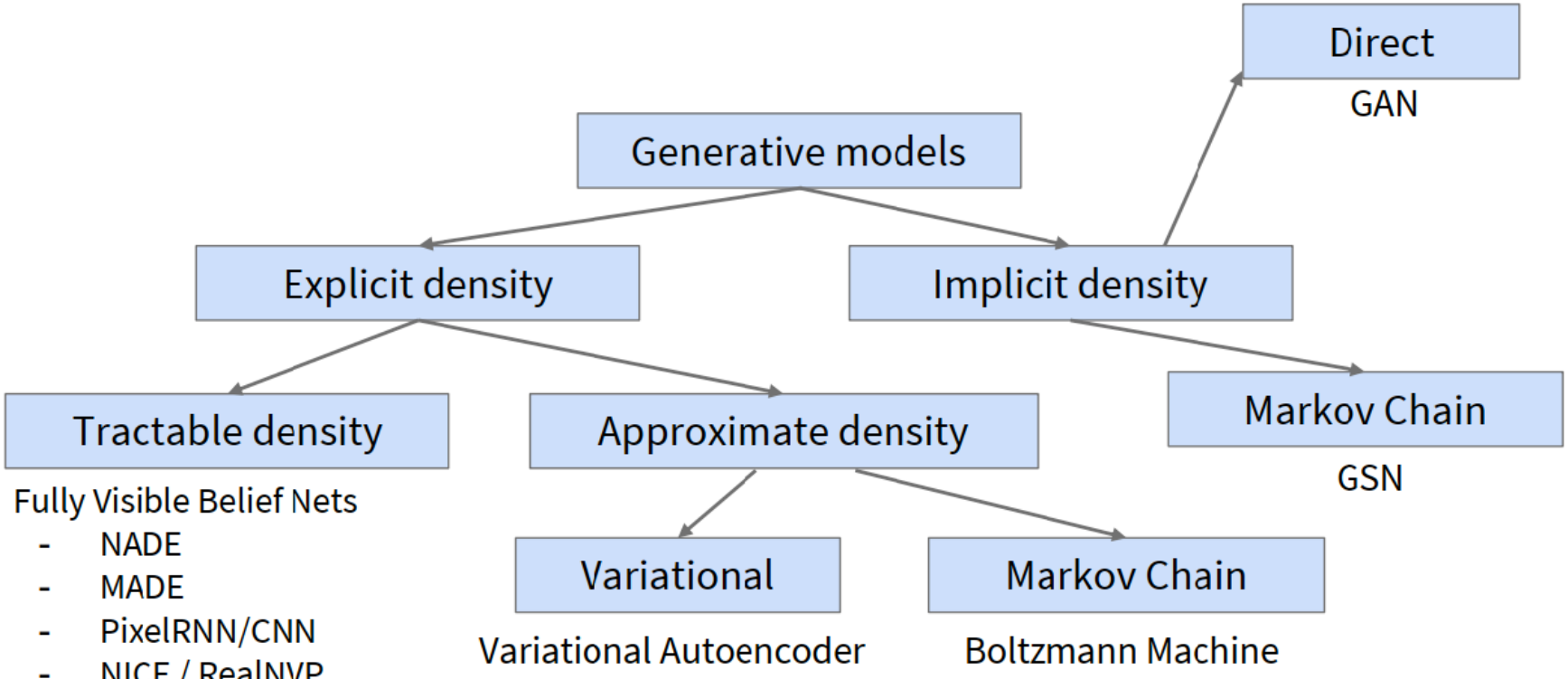



Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017

Slide credit: Fei-Fei Li

# PixelRNN and PixelCNN

# Fully visible belief network (FVBN)

Explicit density model

$$p(x) = p(x_1, x_2, \ldots, x_n)$$

Likelihood of image x

Joint likelihood of each pixel in the image

Slide credit: Fei-Fei Li

# Fully visible belief network (FVBN)

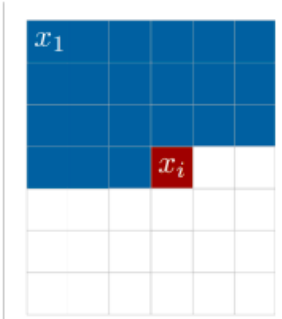
Explicit density model

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

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

Likelihood of image x

Probability of i'th pixel value given all previous pixels




Then maximize likelihood of training data

Slide credit: Fei-Fei Li

# Fully visible belief network (FVBN)

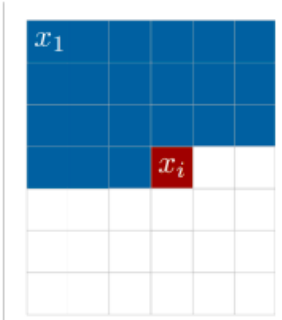
Explicit density model

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

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

Likelihood of image x

Probability of i'th pixel value given all previous pixels

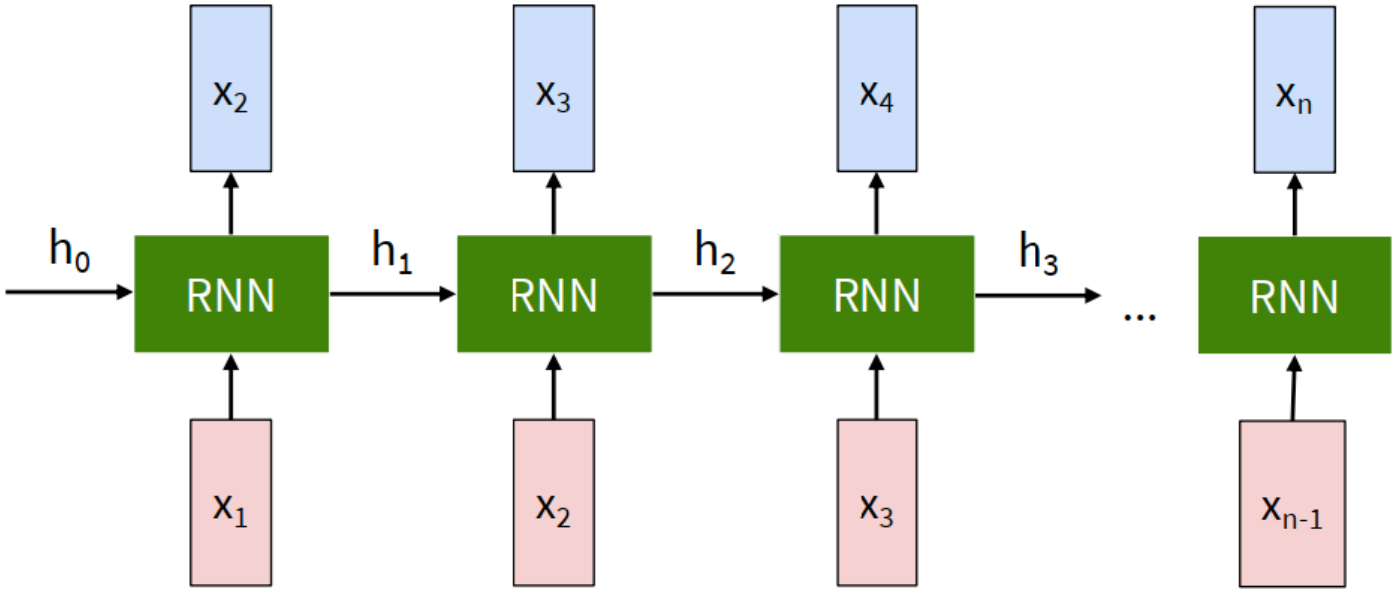



Complex distribution over pixel values => Express using a neural network!

Then maximize likelihood of training data

Slide credit: Fei-Fei Li

# Another solution: Recurrent Neural Network



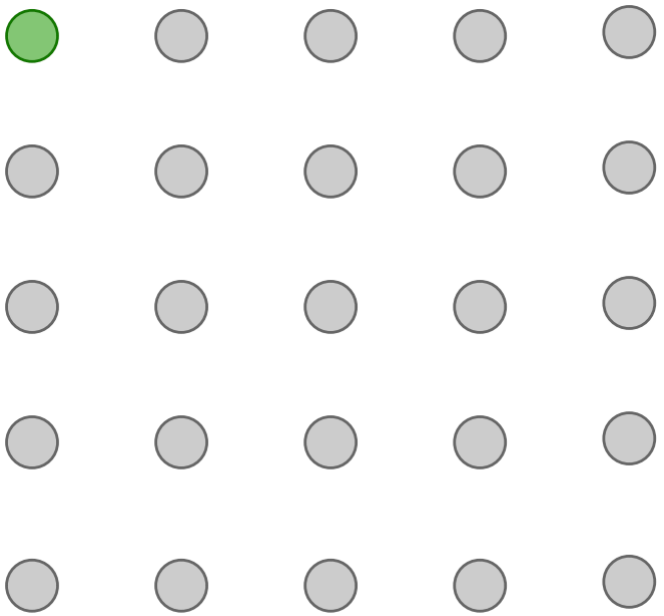

$$p(x_i|x_1, ..., x_{i-1})$$

Slide credit: Fei-Fei Li

# PixelRNN [van der Oord et al. 2016]

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

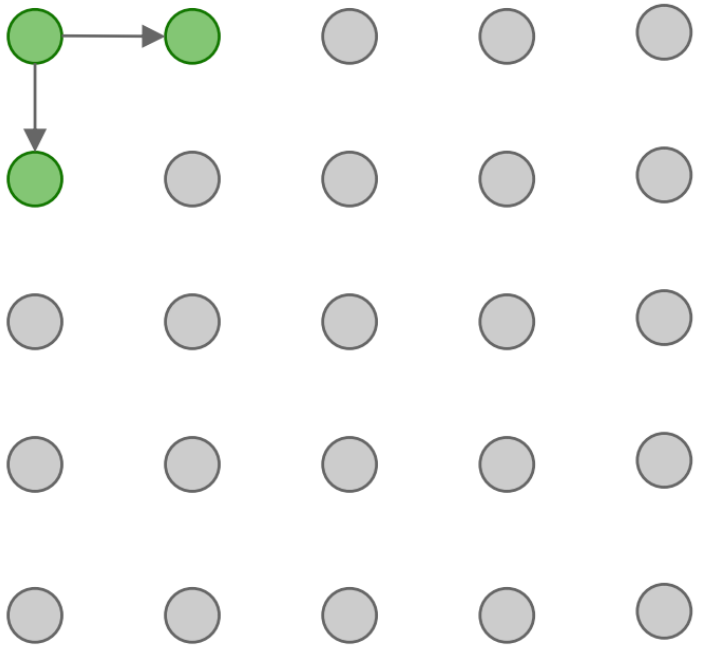


Slide credit: Fei-Fei Li

# PixelRNN [van der Oord et al. 2016]

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

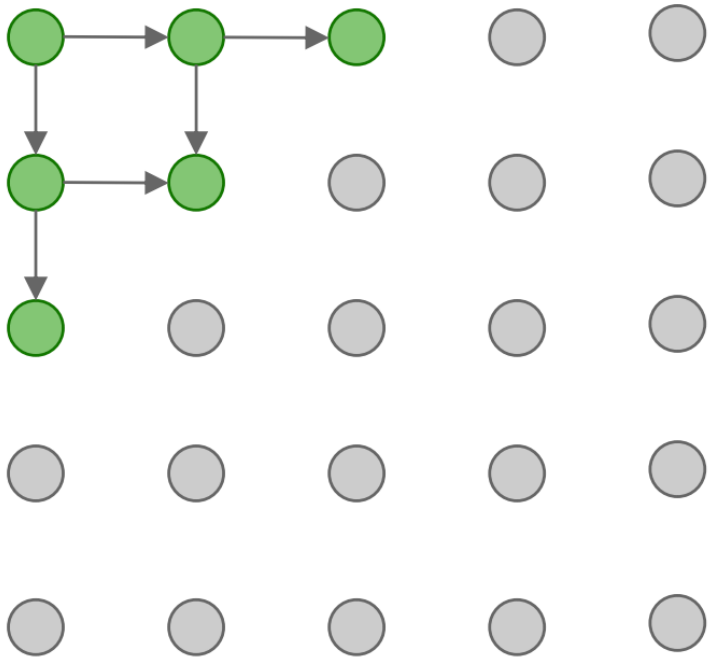


Slide credit: Fei-Fei Li

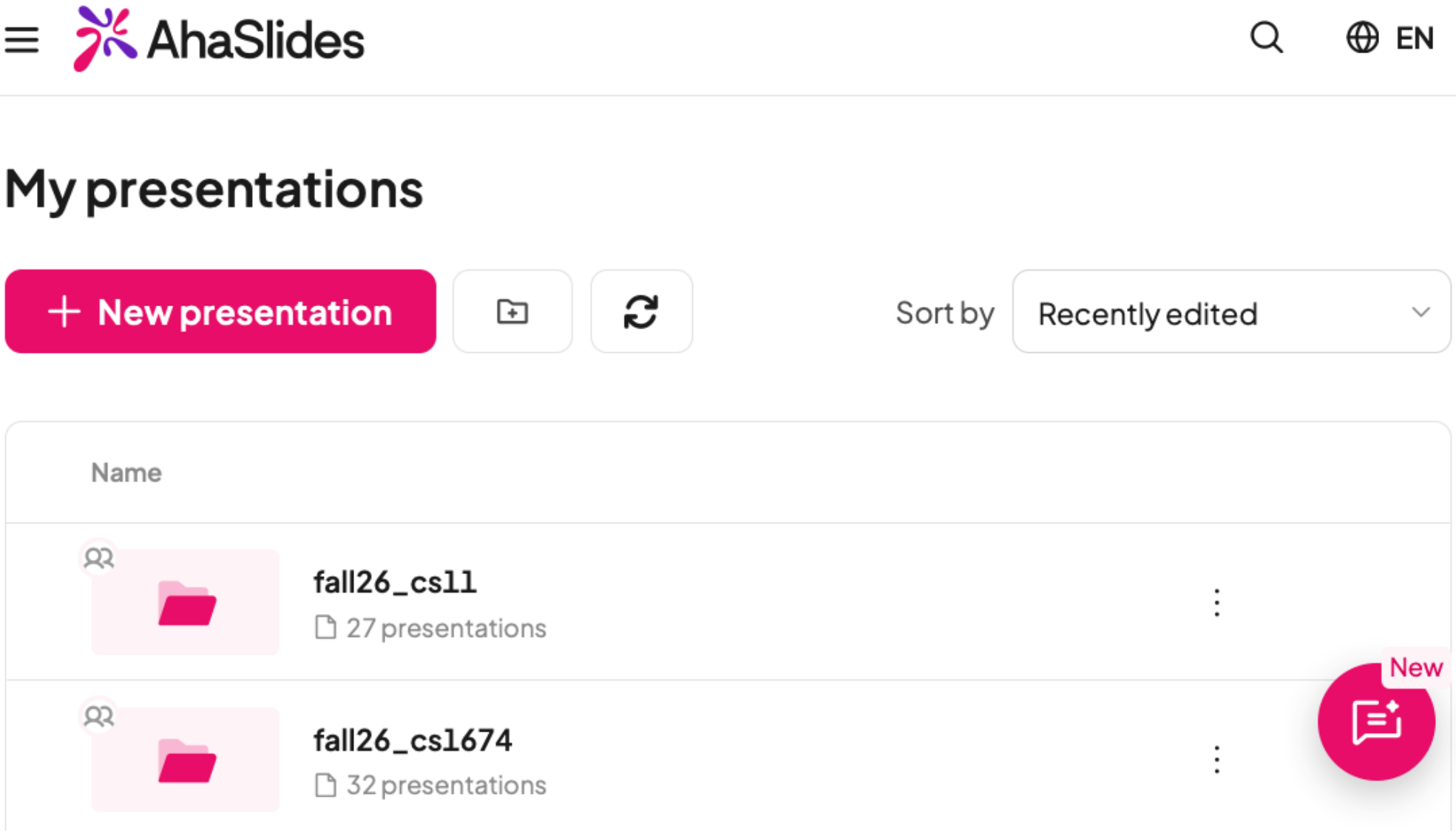
# PixelRNN [van der Oord et al. 2016]

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)



Slide credit: Fei-Fei Li

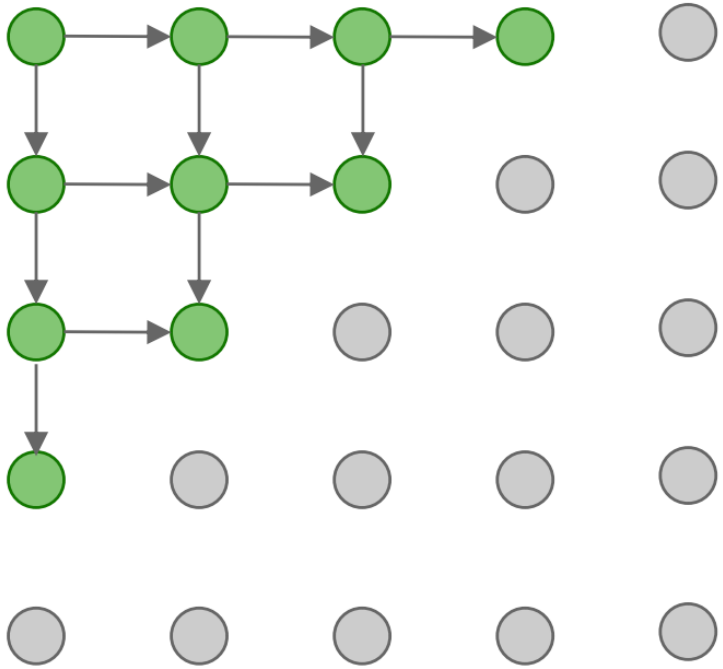

AhaSlides
EN
My presentations
New presentation
Sort by
Recently edited
Name
fall26_cs11
27 presentations
fall26_cs1674
32 presentations
New

# PixelRNN [van der Oord et al. 2016]

Generate image pixels starting from corner

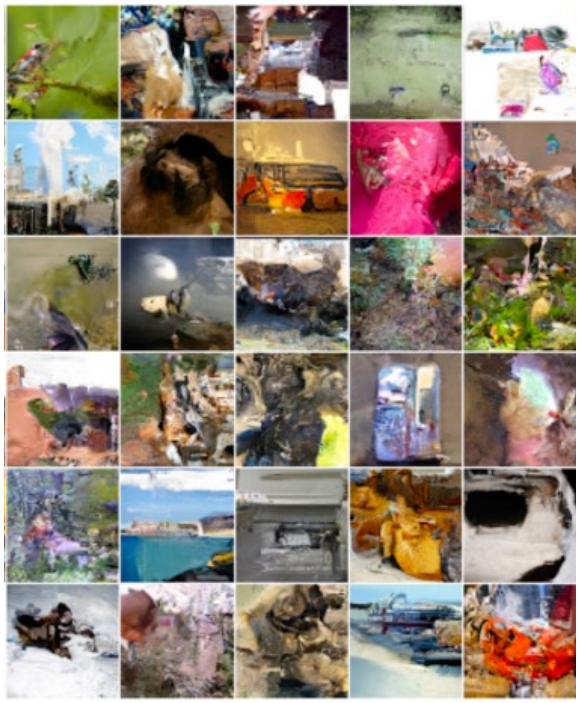
Dependency on previous pixels modeled using an RNN (LSTM)

Drawback: sequential generation is slow in both training and inference!



Slide credit: Fei-Fei Li

# PixelRNN [van der Oord et al. 2016]



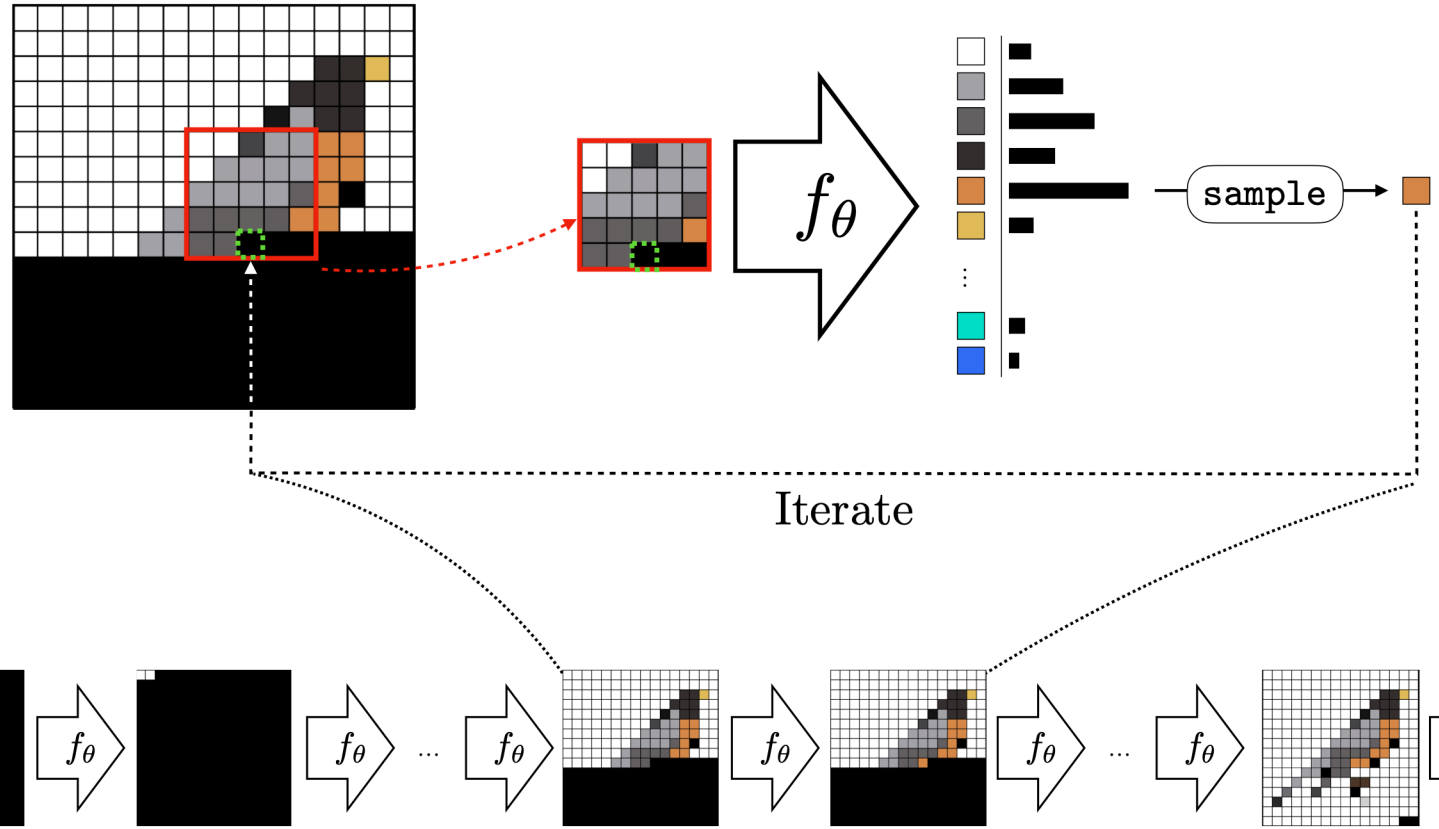
Samples from PixelRNN



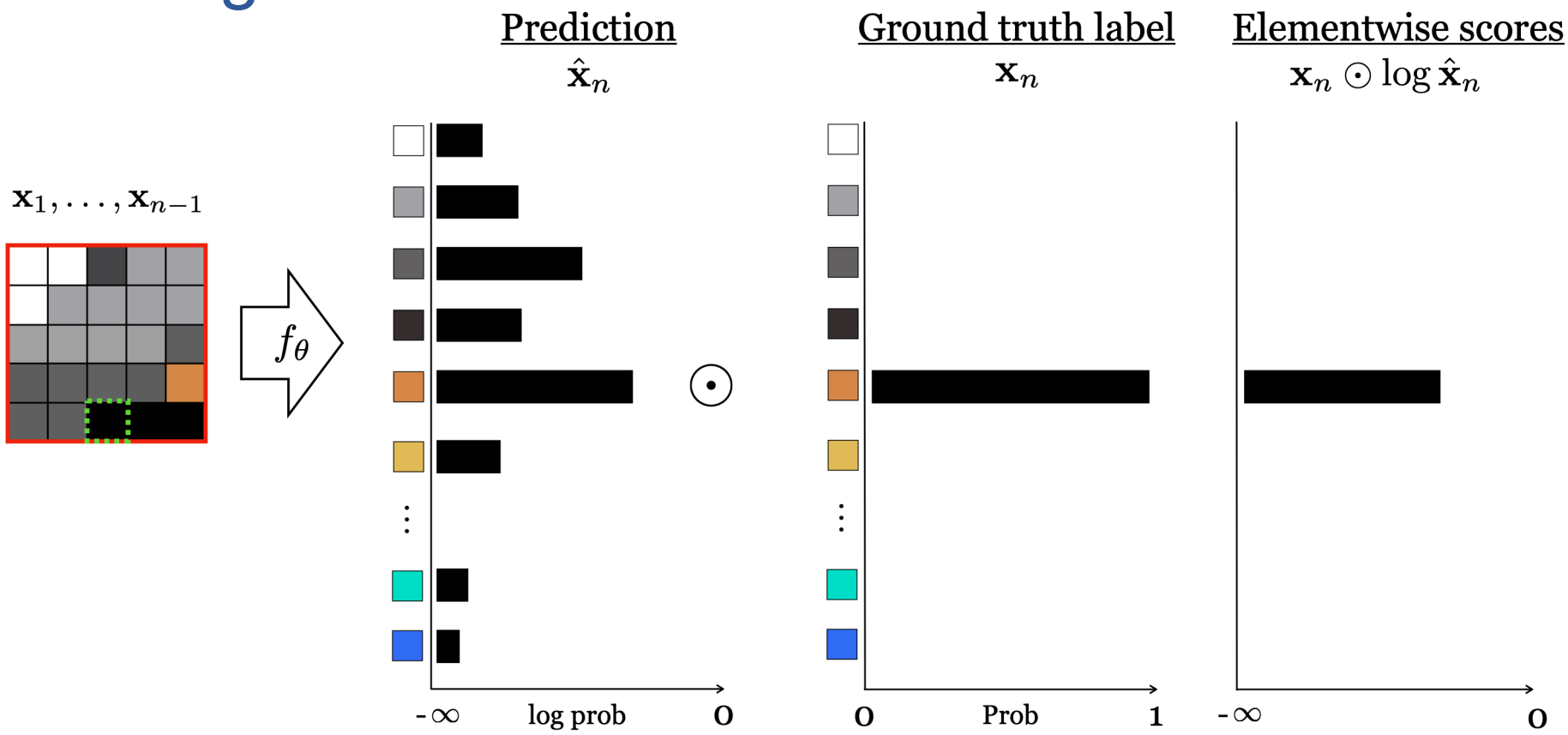

Image completions (conditional samples) from PixelRNN

[PixelRNN, van der Oord et al. 2016]

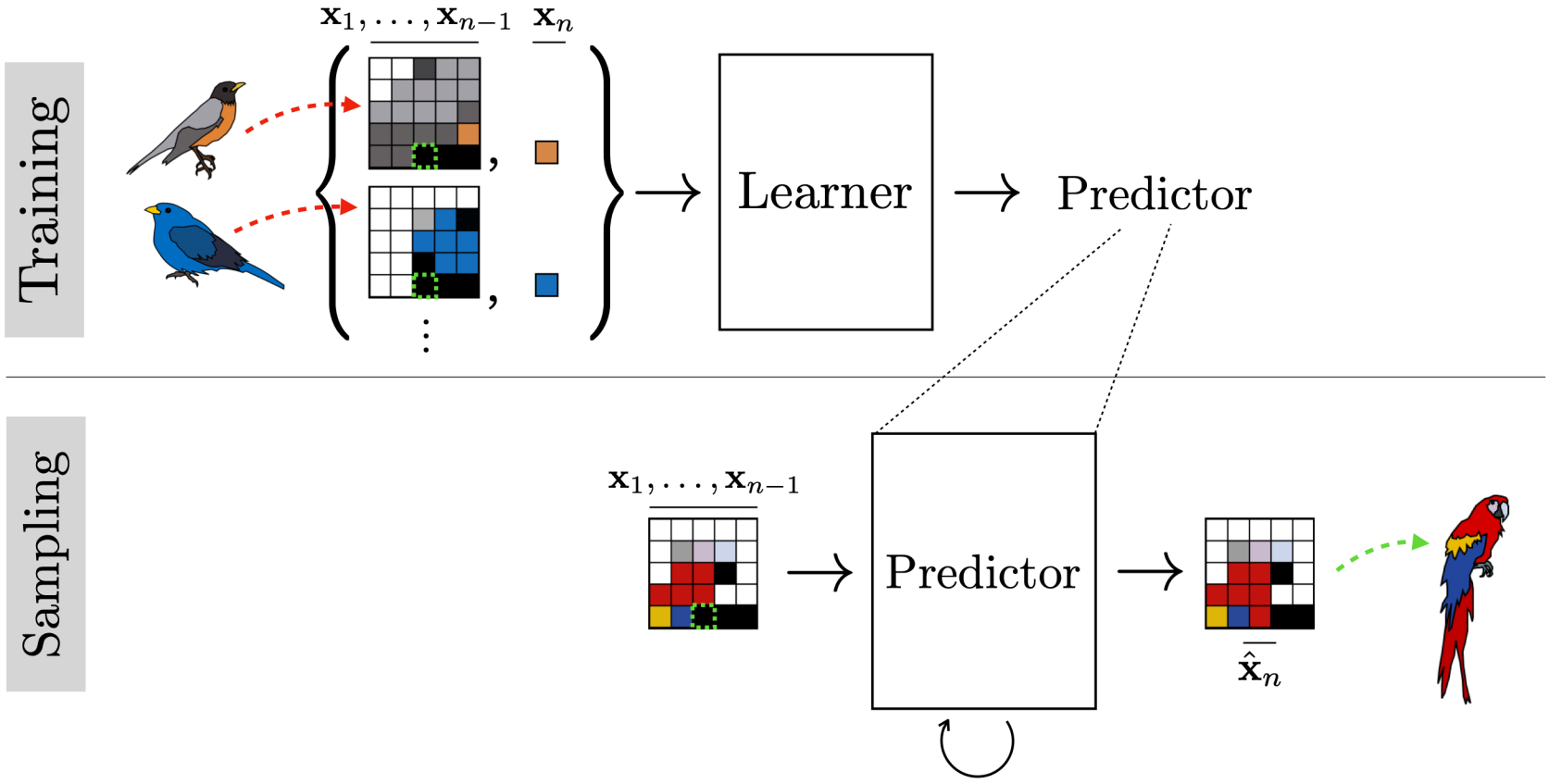
# Autoregressive Model of Pixels



# Autoregressive Model of Pixels



# Autoregressive Model of Pixels



# Taxonomy of Generative Models

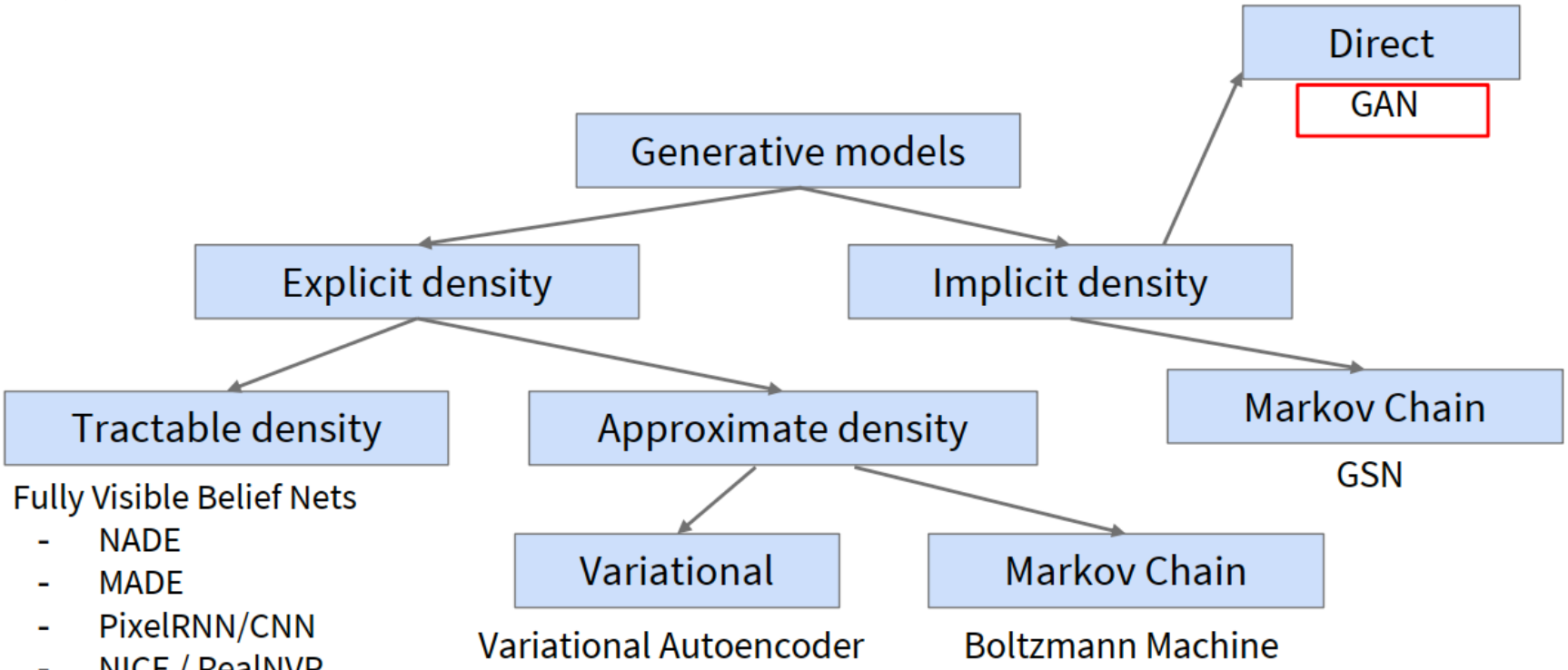



Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Slide credit: Fei-Fei Li

# Generative Adversarial Networks (GANs)

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?

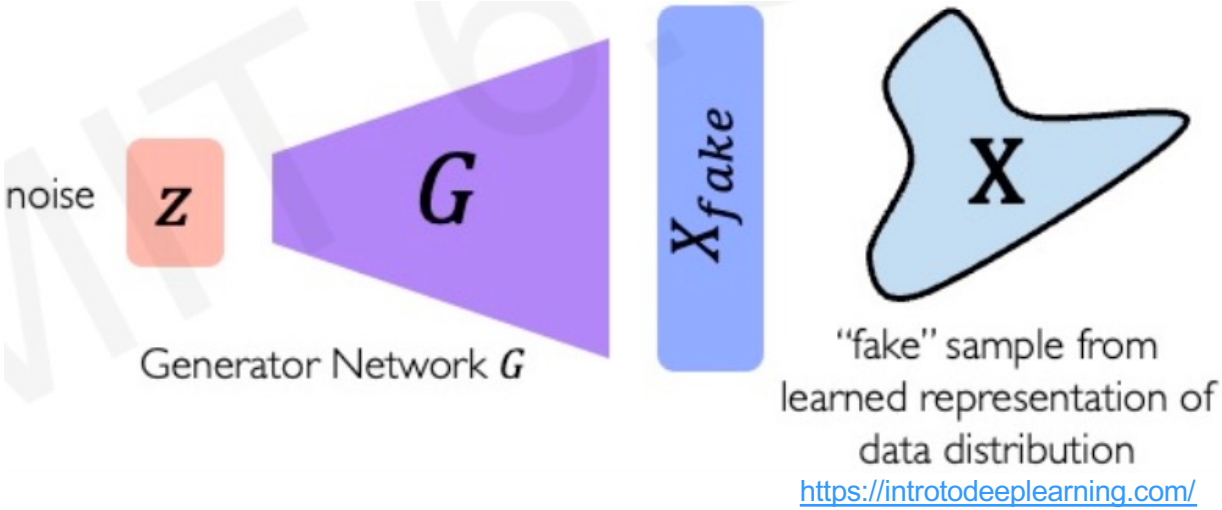
GANs: not modeling any explicit density function!

Slide credit: Fei-Fei Li

# Generative Adversarial Networks (GANs)

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.




https://introtodeeplearning.com/

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

Slide credit: Fei-Fei Li

# Generative Adversarial Networks (GANs)

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

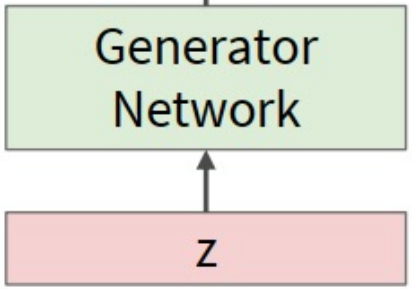
Solution: Sample from a simple distribution we can easily sample from, e.g. random noise. Learn transformation to training distribution.

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"



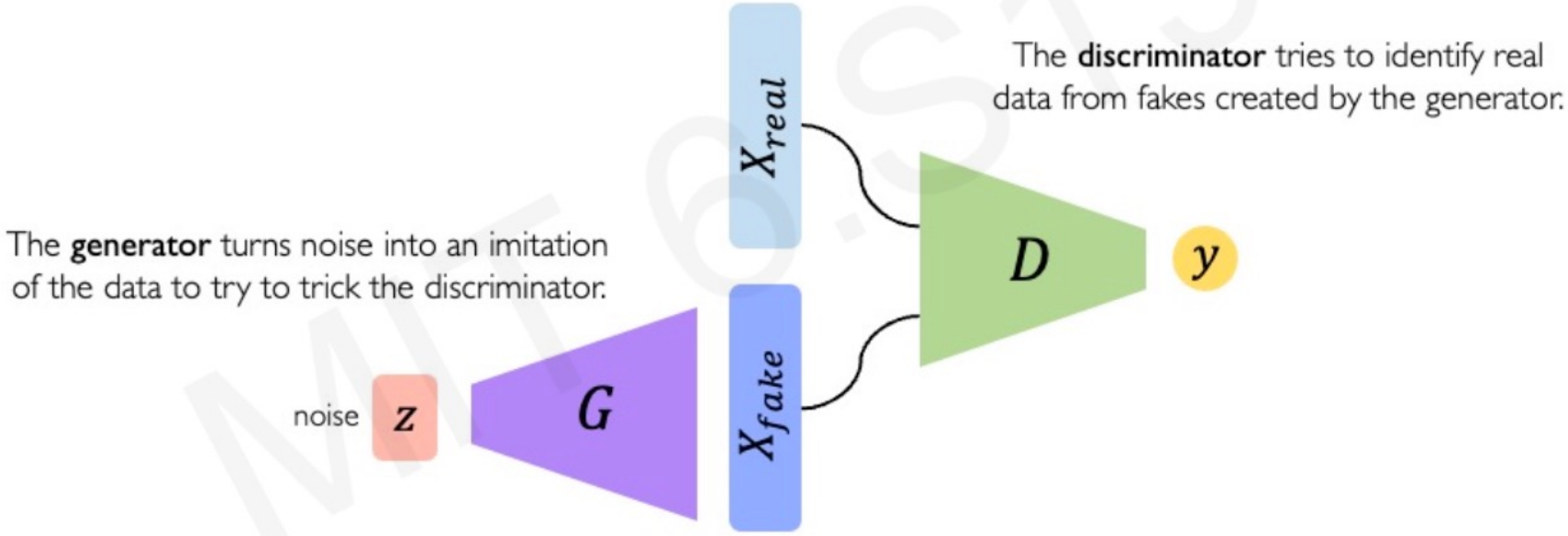

Input: Random noise

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

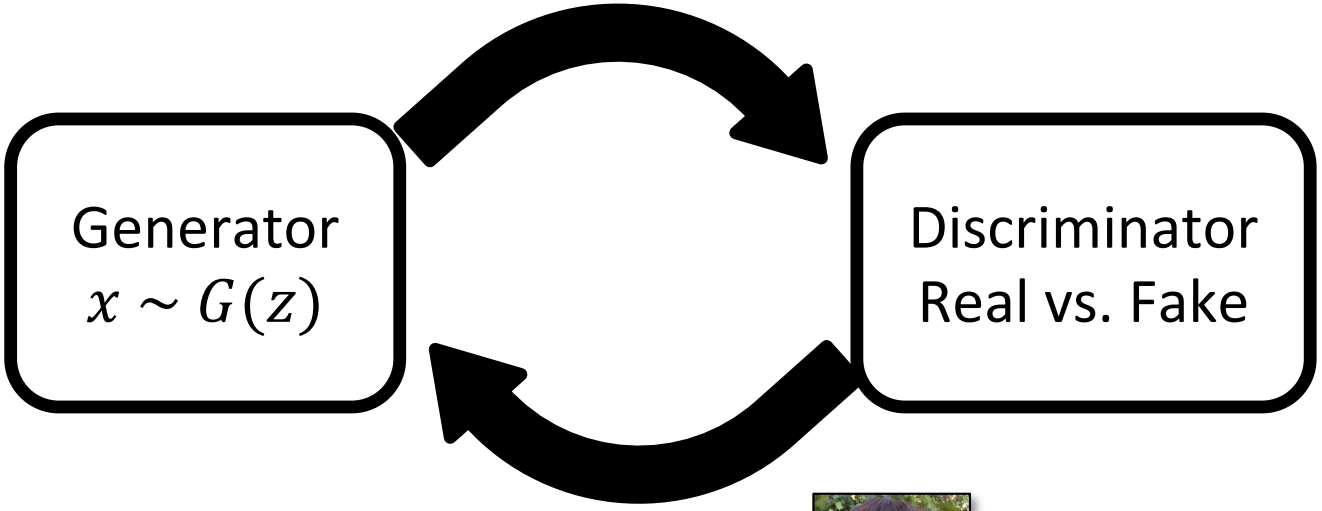
Slide credit: Fei-Fei Li

# Generative Adversarial Networks (GANs)

**Generative Adversarial Networks (GANs)** are a way to make a generative model by having two neural networks compete with each other.



# Adversarial Networks Framework

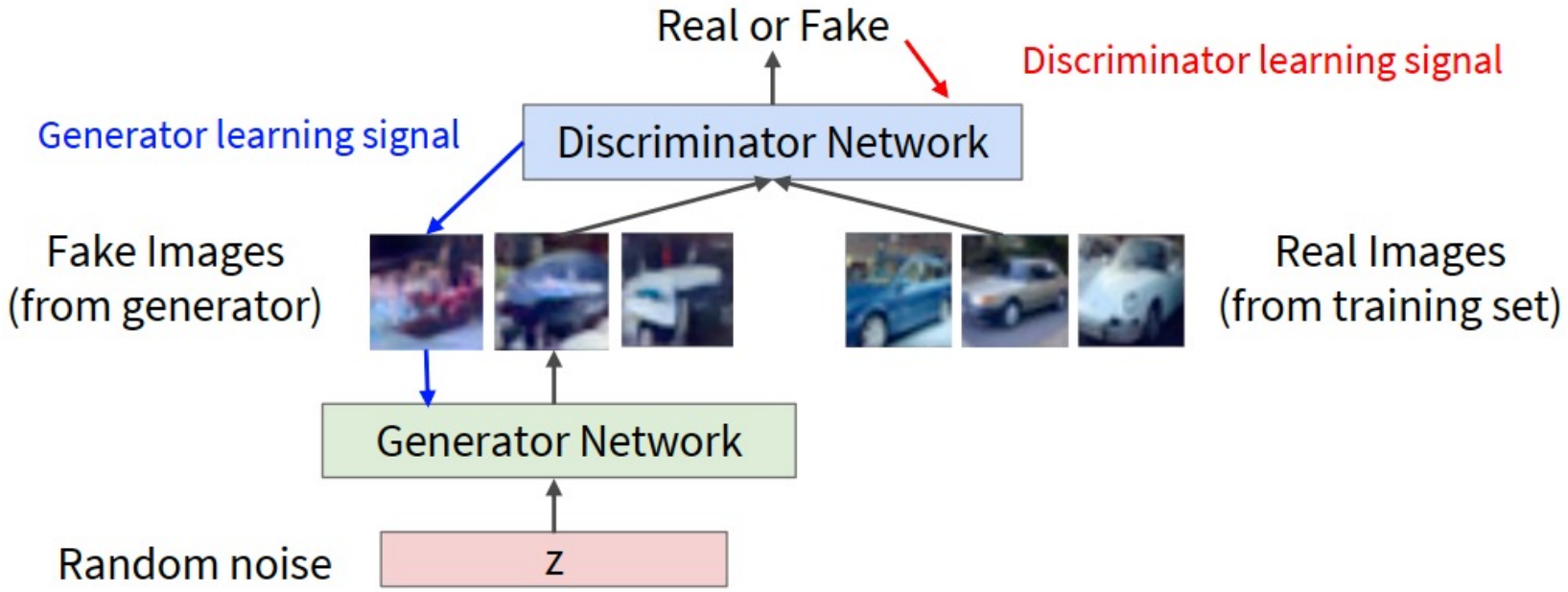





[Goodfellow et al. 2014]

# 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




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

Slide credit: Fei-Fei Li

# 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

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

Slide credit: Fei-Fei Li

# Training GANs: Two-player game

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

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

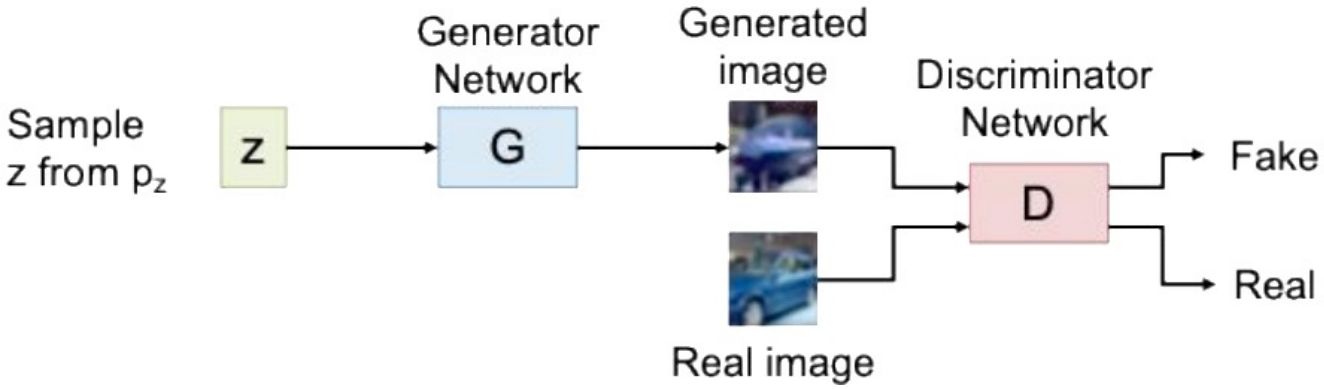
Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Slide credit: Fei-Fei Li

# Training GANs: Two-player game

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

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




D(x) = P(x is real)
D(x) = 0 => fake
D(x) = 1 => real

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

Slide credit: Fei-Fei Li

# Training GANs: Two-player game

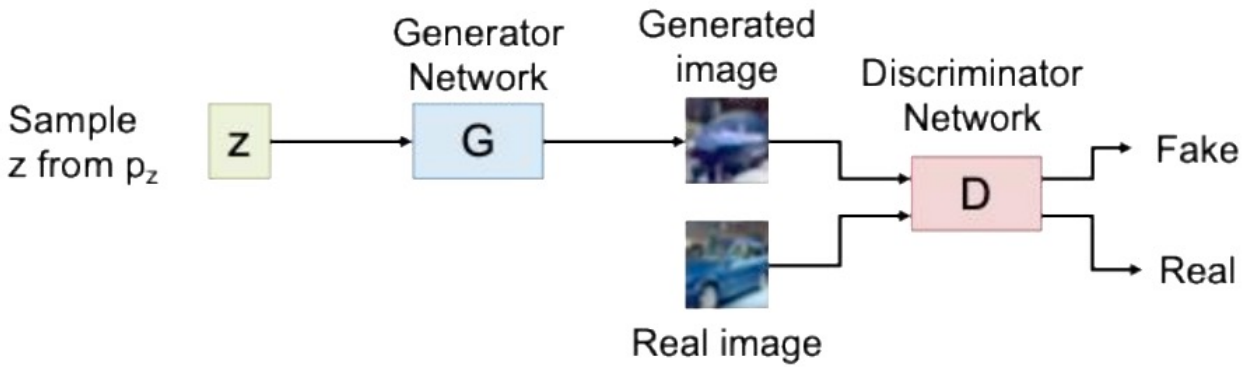
Jointly train generator G and discriminator D with a **minimax game**

Discriminator wants D(x) = 1 for real data

Discriminator wants D(x) = 0 for fake data

Imagine fixing G

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




D(x) = P(x is real)
D(x) = 0 => fake
D(x) = 1 => real

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

Slide credit: Fei-Fei Li

# Training GANs: Two-player game

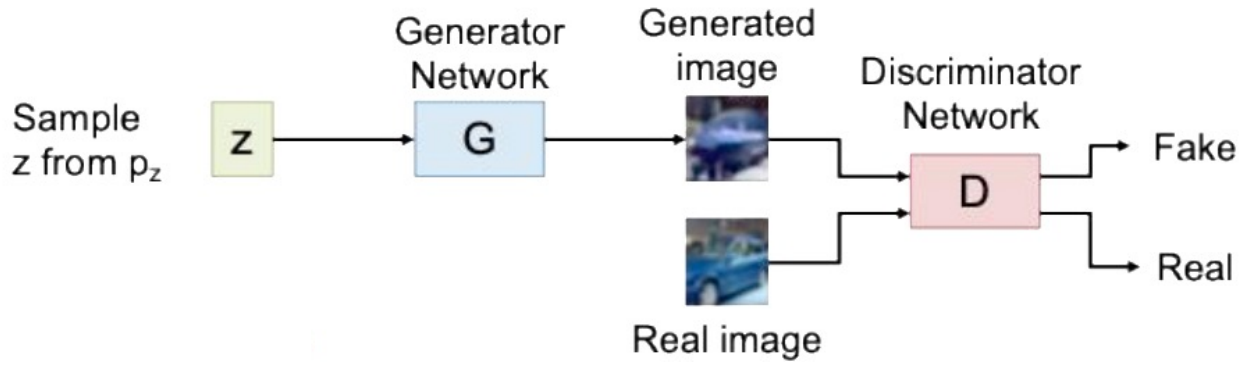
Jointly train generator G and discriminator D with a **minimax game**

This term does not depend on G

Generator wants D(x) = 1 for fake data

Imagine fixing **D**

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




D(x) = P(x is real)
D(x) = 0 => fake
D(x) = 1 => real

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

Slide credit: Fei-Fei Li

# Training GANs: Two-player game

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

Train **G** and **D** using alternating gradient updates

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

$$= \min_G \max_D V(G, D)$$

While True:

$$D = D + \alpha_D \frac{dV}{dD}$$

$$G = G - \alpha_G \frac{dV}{dG}$$

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

Slide credit: Fei-Fei Li

To join, go to: ahaslides.com/IKCMO

AhaSlides

# In the standard minimax objective function for Generative Adversarial Networks (GANs), what role does the discriminator network play?

Edit slides

Add slides

0 0 /100

# 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

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

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

Slide credit: Fei-Fei Li

# 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

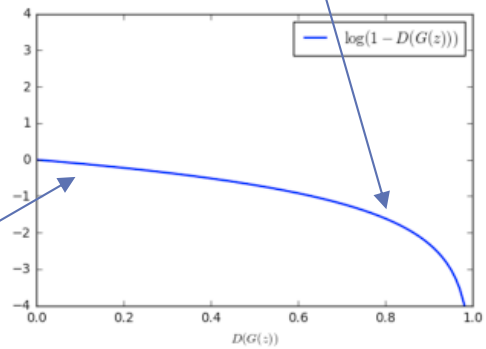
$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

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




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

Slide credit: Fei-Fei Li

# 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

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

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

Slide credit: Fei-Fei Li

To join, go to: ahaslides.com/IKCMO

AhaSlides

# Why is the original generator objective log(1−D(G(z))) often replaced in practice with log(D(G(z))) during GAN training?

Edit slides

Add slides

0 0 /100

# 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**

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!

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

Slide credit: Fei-Fei Li

# Training GANs: Other Divergence

Least Squares GAN

$$\min_D V_{\mathrm{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\mathrm{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-1)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z})))^2\right]$$

$$\min_G V_{\mathrm{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-1)^2\right].$$

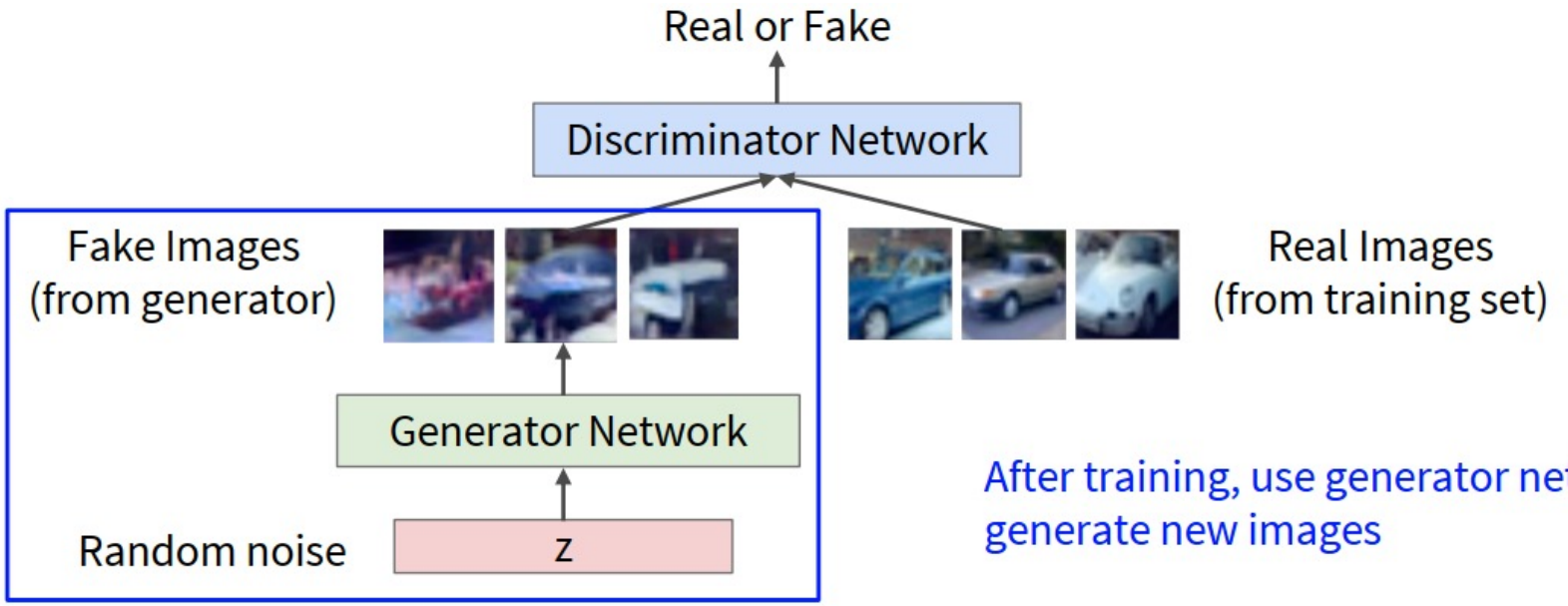
* Employed for our Assignment.

Xudong Mao et al., “Least Squares GAN”, ICCV 2017

https://arxiv.org/pdf/1611.04076
https://sh-tsang.medium.com/review-lsgan-least-squares-generative-adversarial-networks-gan-bec12167e915

# 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



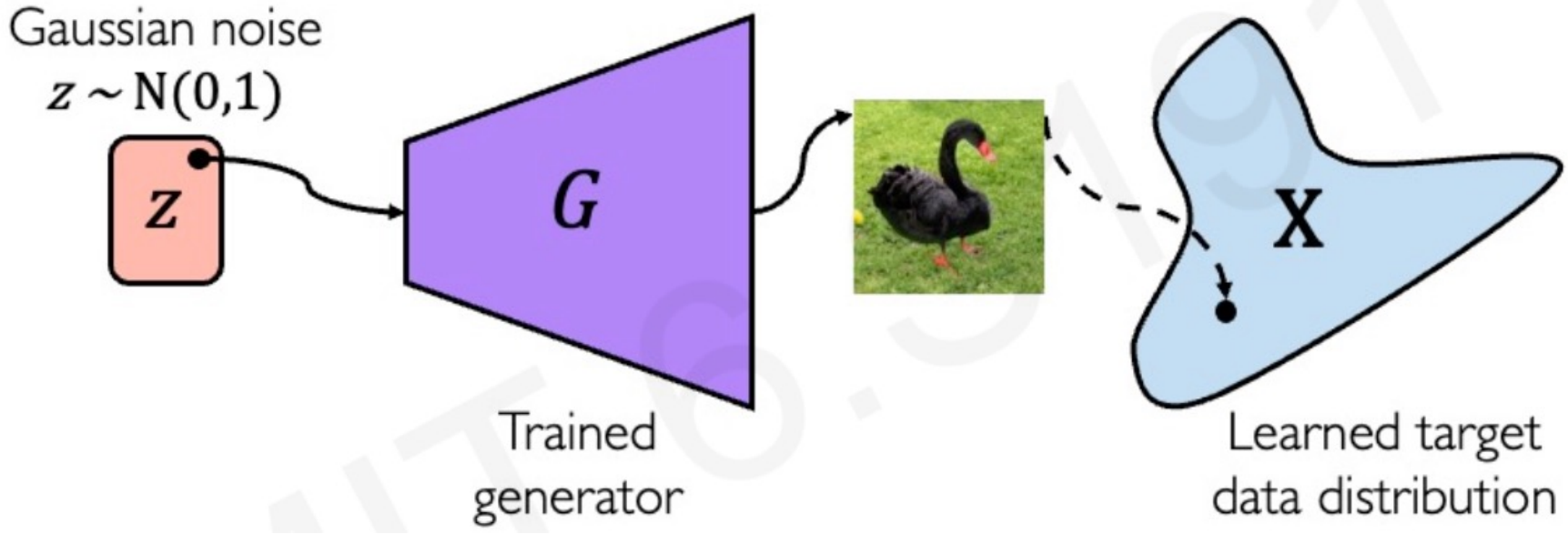

Fake and real images copyright Emily Denton et al. 2015

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

Slide credit: Fei-Fei Li

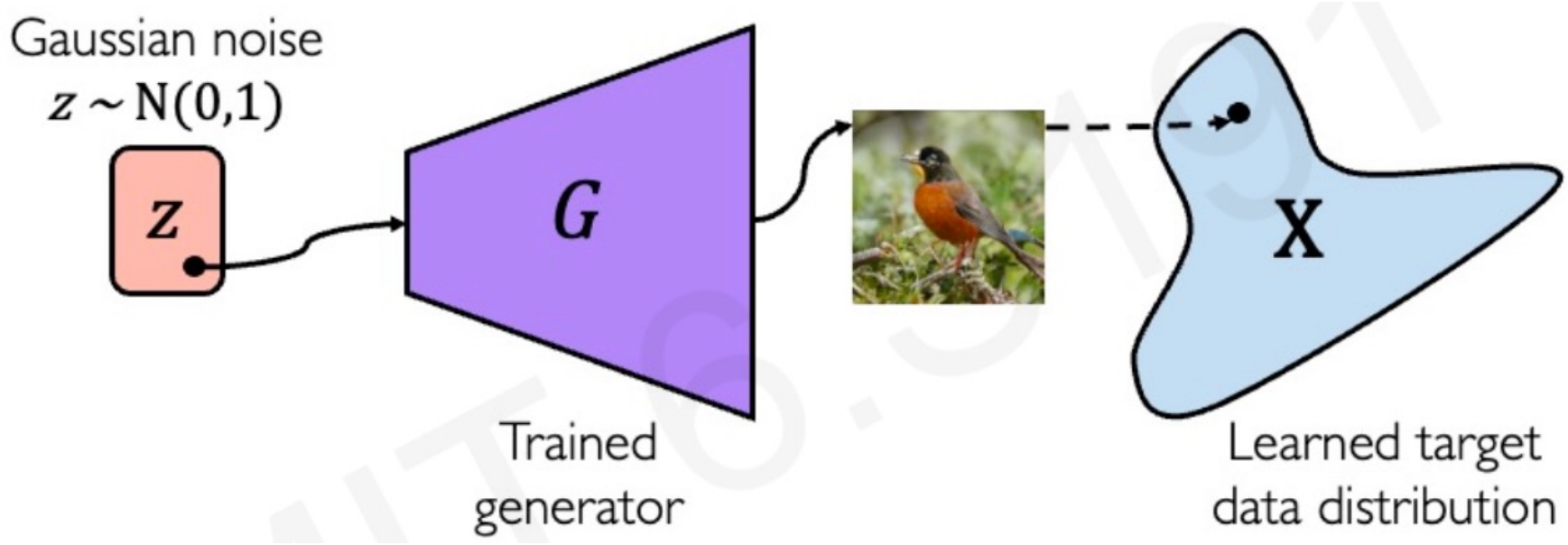
# GANs: Generating New Data

GANs are Distribution Transformers



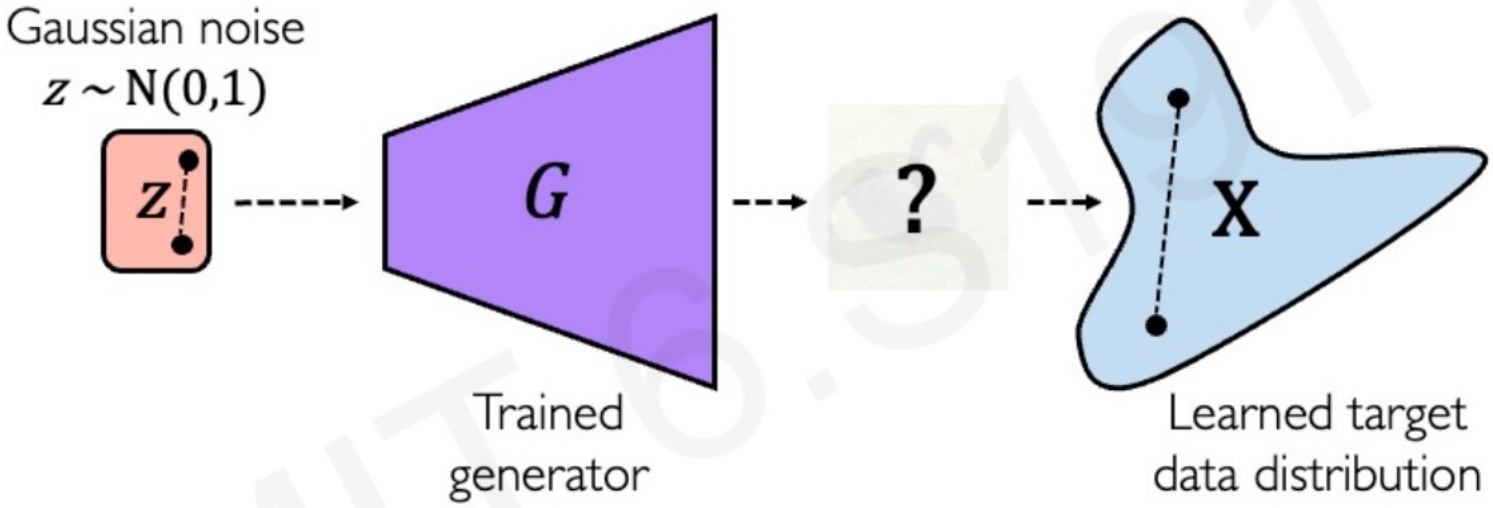
# GANs: Generating New Data

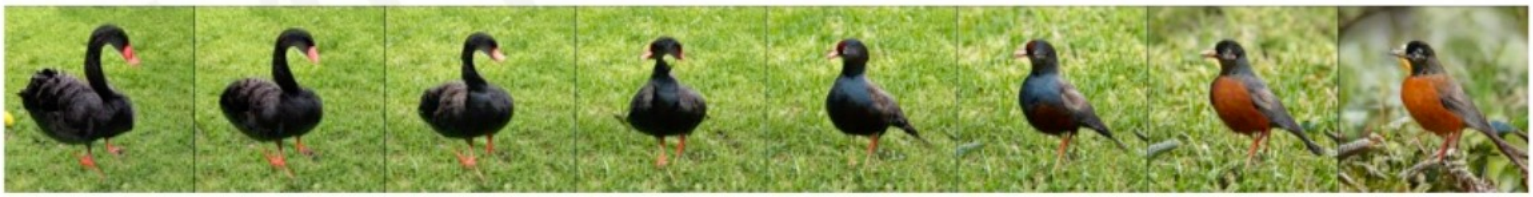
GANs are Distribution Transformers



# GANs: Generating New Data

GANs are Distribution Transformers



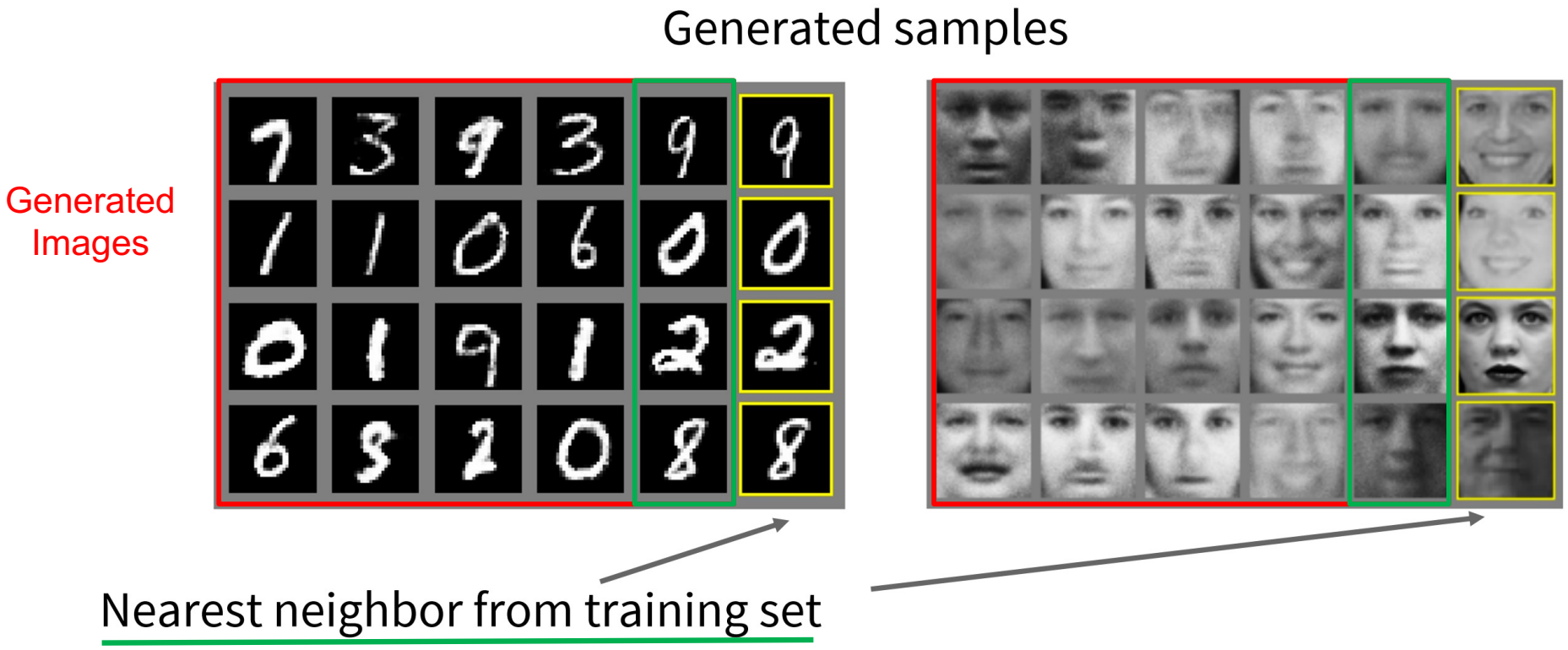



# Lab 10a: Generative Adversarial Networks (GANs) for Digits

Duration: 10 min



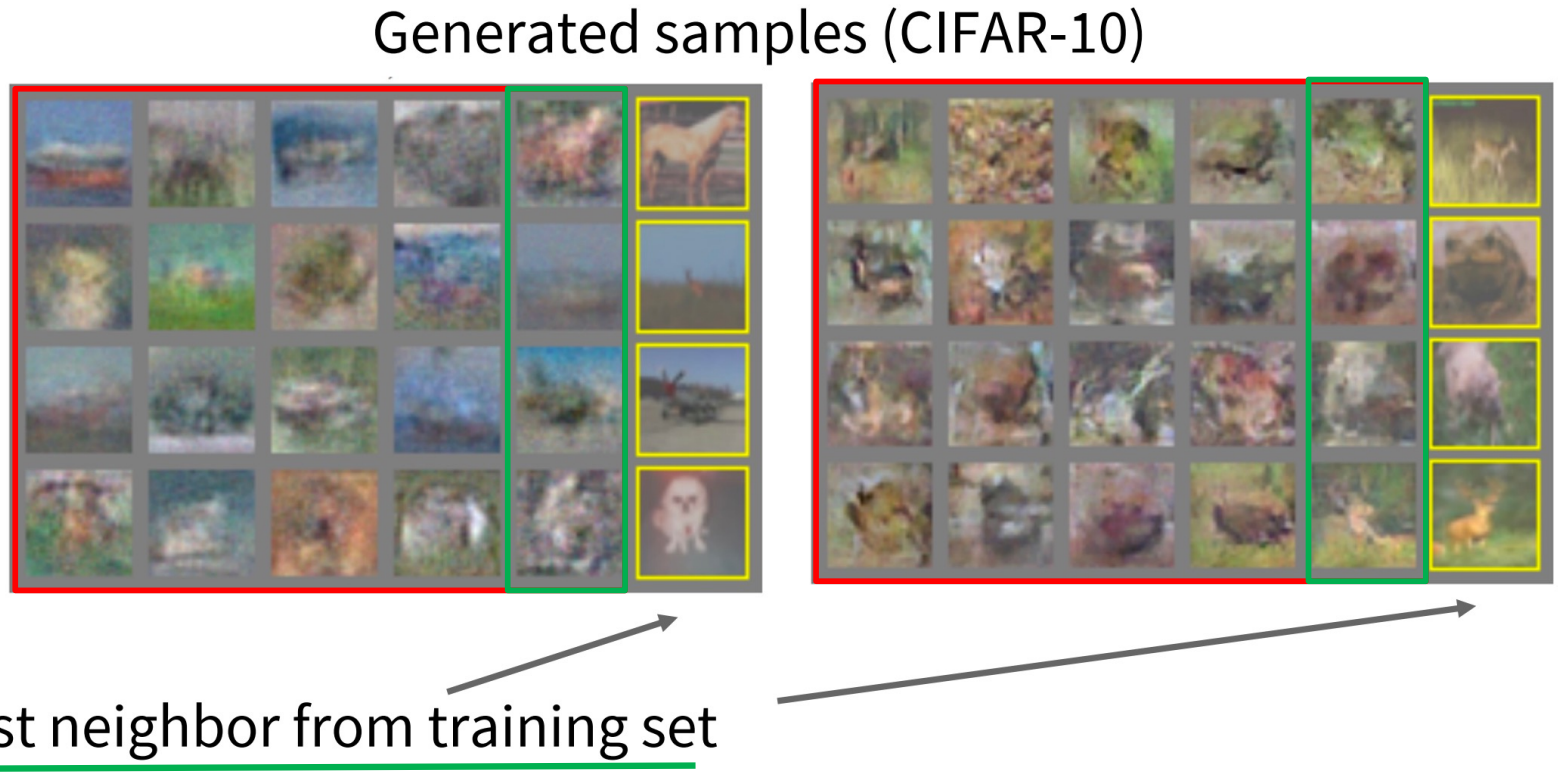
# Generative Adversarial Nets




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

Slide credit: Fei-Fei Li

# Generative Adversarial Nets




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

Slide credit: Fei-Fei Li

# Generative Adversarial Nets: DC-GAN

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

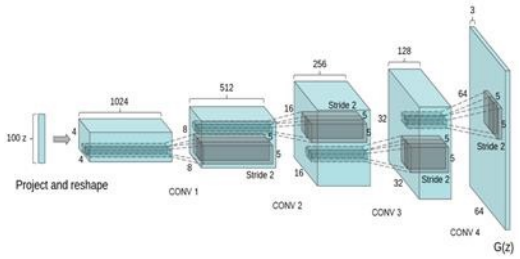
Slide credit: Fei-Fei Li

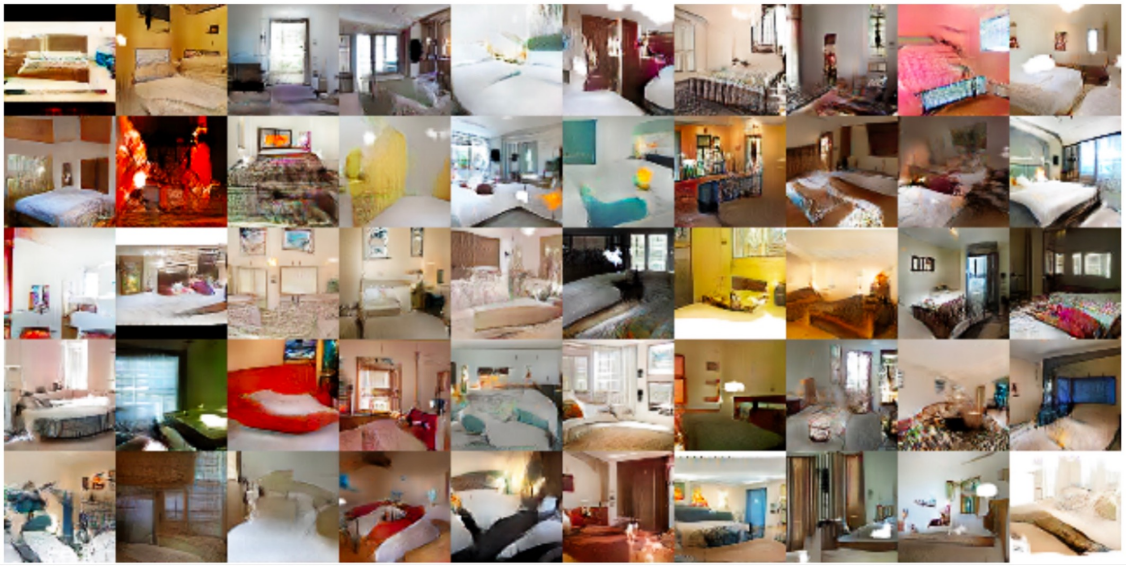
# Generative Adversarial Nets: DC-GAN

Generator G and discriminator D are both neural networks

Usually CNNs … GANs fell out of favor before ViT became popular

DC-GAN was the first GAN architecture that worked on non-toy data

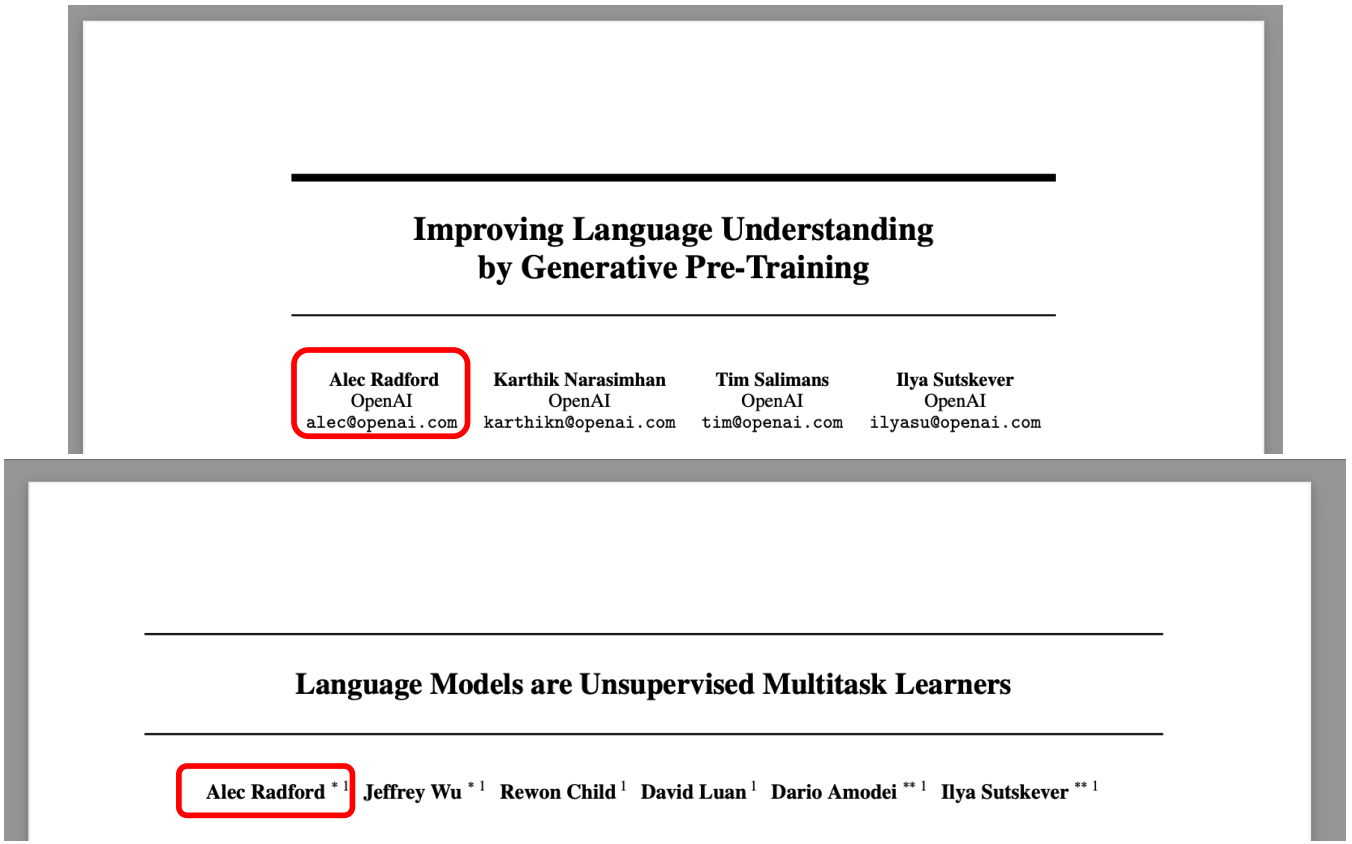


Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

Slide credit: Fei-Fei Li

# Generative Adversarial Nets: DC-GAN



**Improving Language Understanding by Generative Pre-Training**

**Alec Radford** OpenAI alec@openai.com
**Karthik Narasimhan** OpenAI karthikn@openai.com
**Tim Salimans** OpenAI tim@openai.com
**Ilya Sutskever** OpenAI ilyasu@openai.com

**Language Models are Unsupervised Multitask Learners**

**Alec Radford** [*1] **Jeffrey Wu** [*1] **Rewon Child** [1] **David Luan** [1] **Dario Amodei** [**1] **Ilya Sutskever** [**1]



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

Slide credit: Fei-Fei Li

# Generative Adversarial Nets: DC-GAN

Interpolating between random points in latent space



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

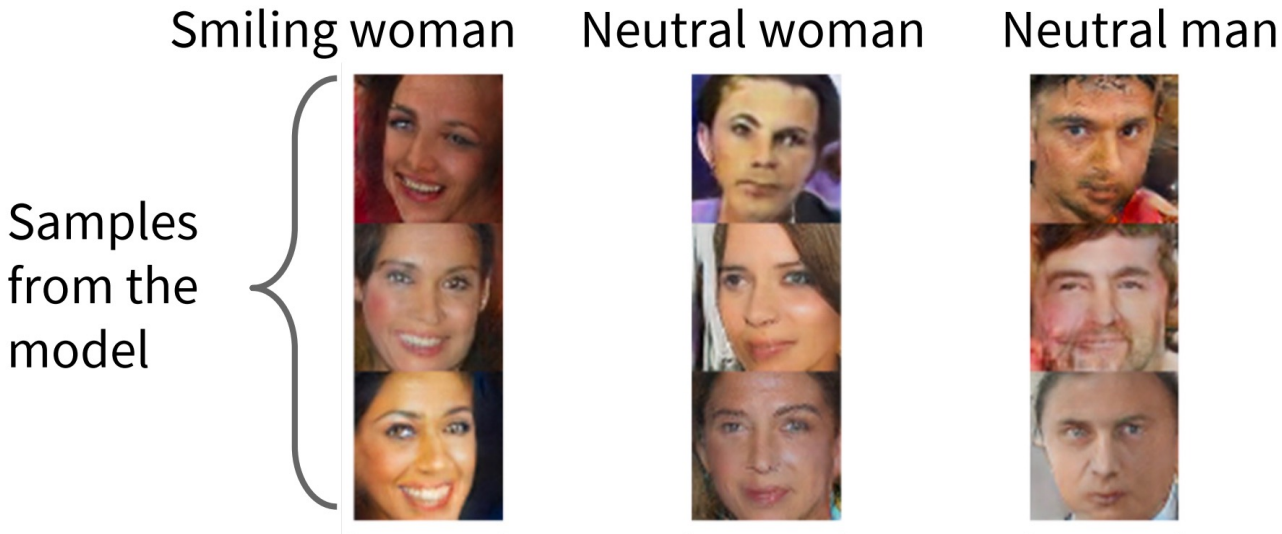
Slide credit: Fei-Fei Li

https://poloclub.github.io/ganlab/



# GANs: Interpretable Vector Math

Radford et al, ICLR 2016




Adapted from Serena Young

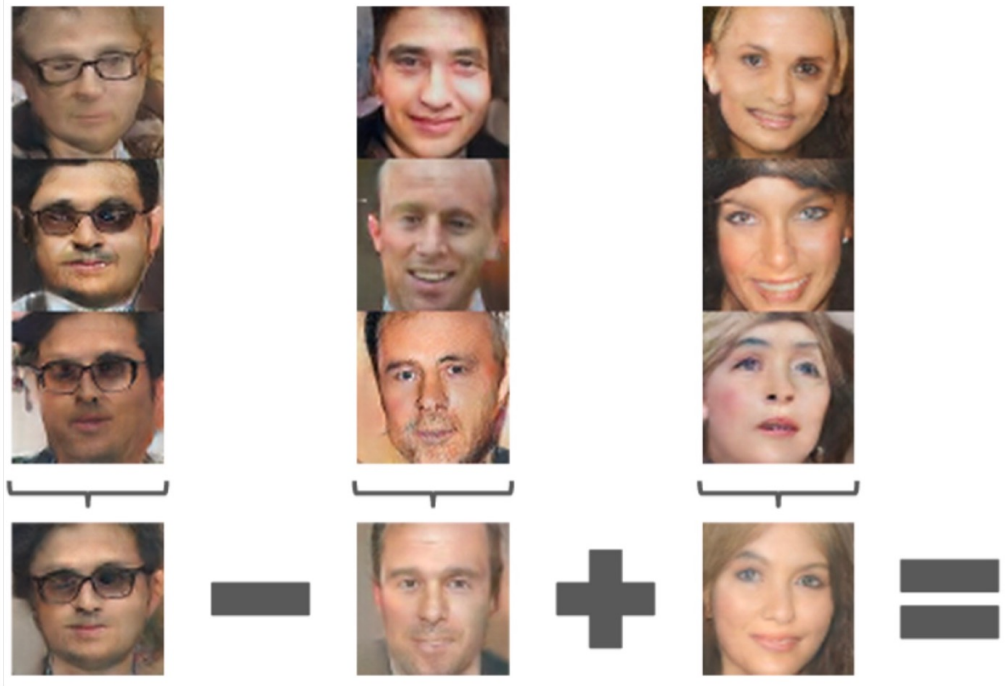
# GANs: Interpretable Vector Math

Glasses man

No glasses man

No glasses woman

Radford et al, ICLR 2016



Woman with glasses



Adapted from Serena Young

# 2017: Explosion of GANs

## "The GAN Zoo"

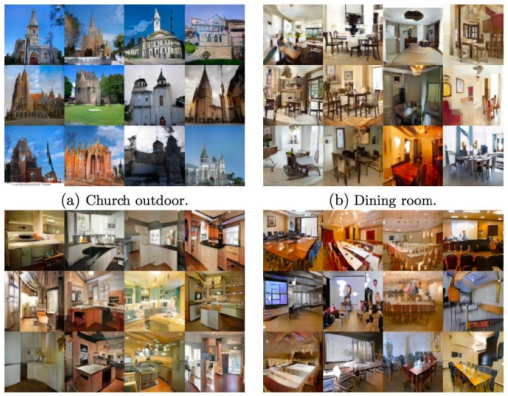
- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- 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
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

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

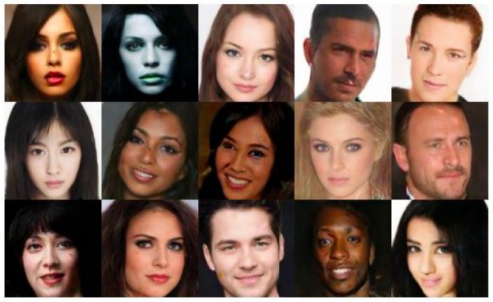
Adapted from Serena Young

# 2017: Explosion of GANs

## Better training and generation




LSGAN. Mao et al. 2017.



BEGAN. Bertholet et al. 2017.

Adapted from Serena Young

## Source->Target domain transfer




CycleGAN. Zhu et al. 2017.

## Text -> Image Synthesis

this small bird has a pink breast and crown, and black primaries and secondaries.

this magnificent fellow is almost all black with a red crest, and white cheek patch.



Reed et al. 2017.

## Many GAN applications



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

# 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

Adapted from Serena Young

# Lab 10b: Generative Adversarial Networks (GANs)

Duration: 10 min



To join, go to: **ahaslides.com/OB8VO**

AhaSlides

# Please, from Lab: GAN [Section 1.2], submit your interpolation plot result.

Get Feedback

Edit slides

Add slides

0 0 /100

# Summary: GANs - Scaling

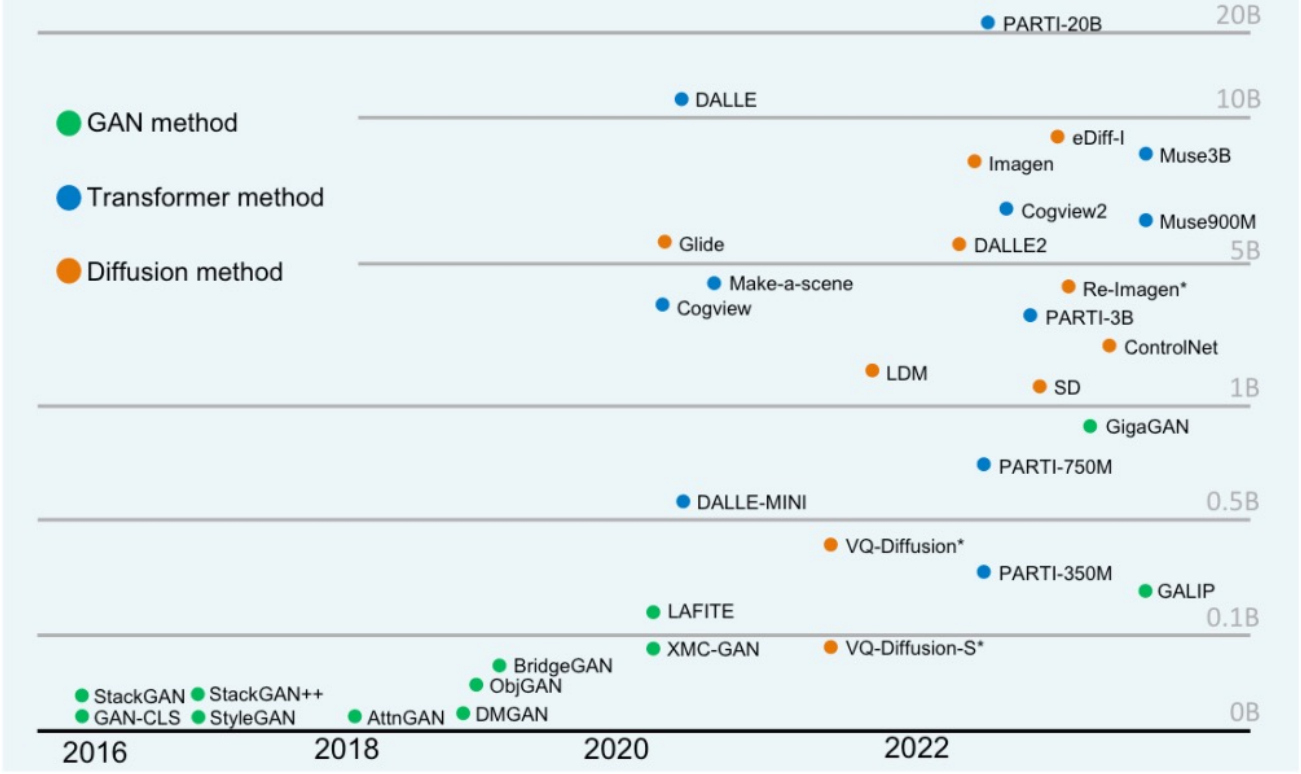



Fig. 5. Timeline of TTI model development, where green dots are GAN TTI models, blue dots are autoregressive Transformers and orange dots are Diffusion TTI models. Models are separated by their parameter, which are in general counted for all their components. Models with asterisk are calculated without the involvement of their text encoders.

https://www.cs.cmu.edu/~mgormley/courses/10423/schedule.html

Figure from Bie et al. (2023)

# Taxonomy of Generative Models

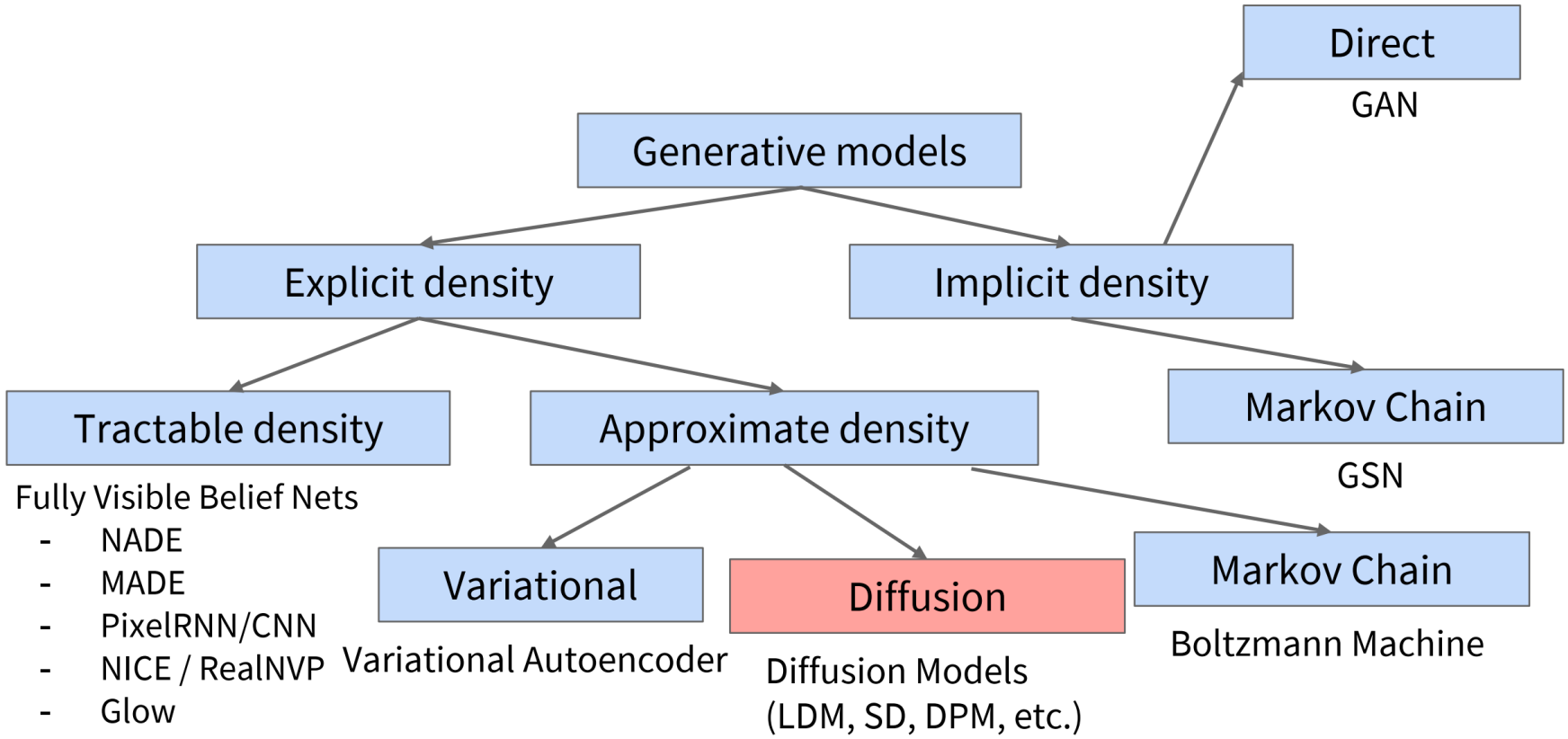



Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Network

Adapted from Fei-Fei Li

# Diffusion Models: Motivation

**How to create a sandcastle:**

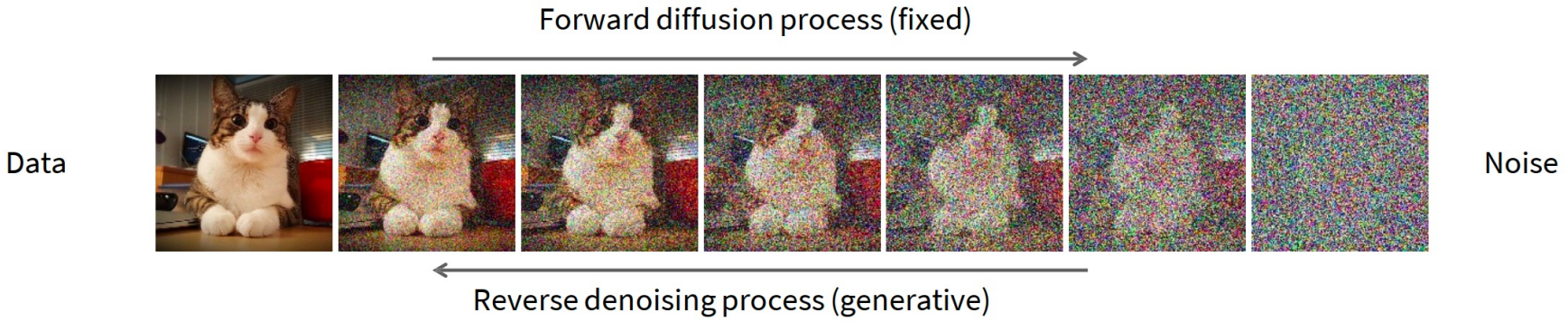
https://deeplearning.cs.cmu.edu/F23/
https://deeplearning.cs.cmu.edu/F23/document/slides/lec23.diffusion.updated.pdf

# Denoising Diffusion Models

Learning to generate by denoising

Denoising diffusion models consist of two processes:

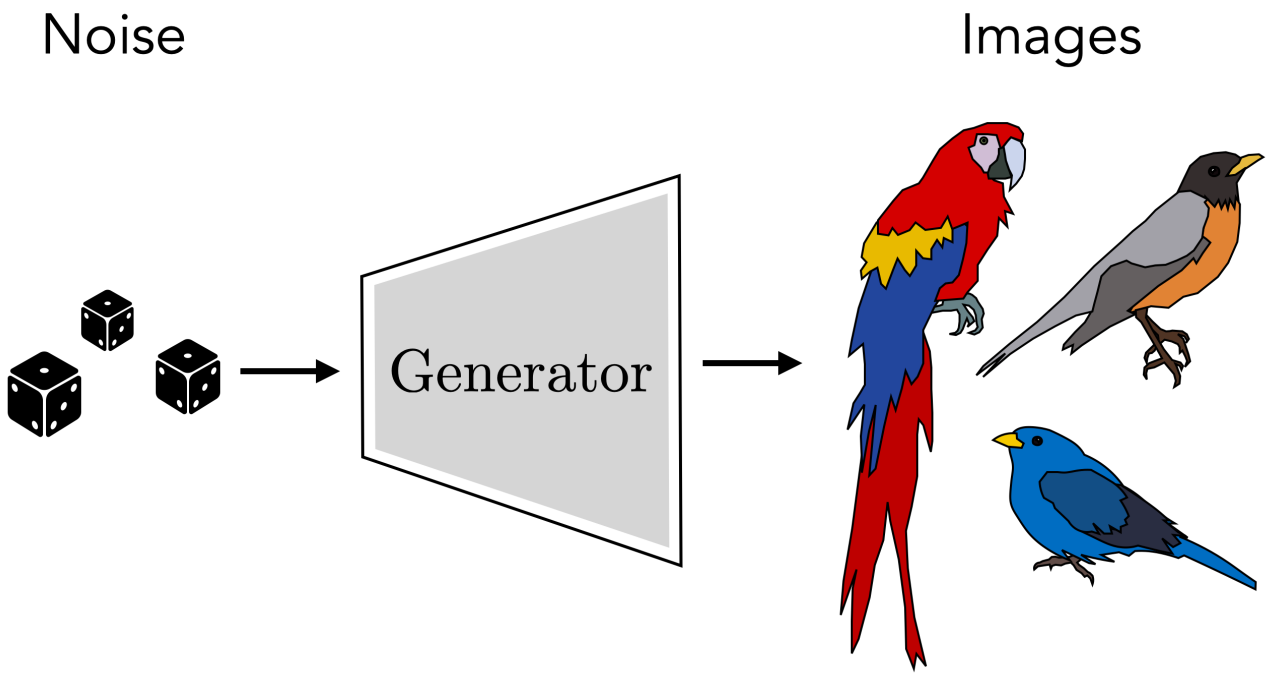
- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



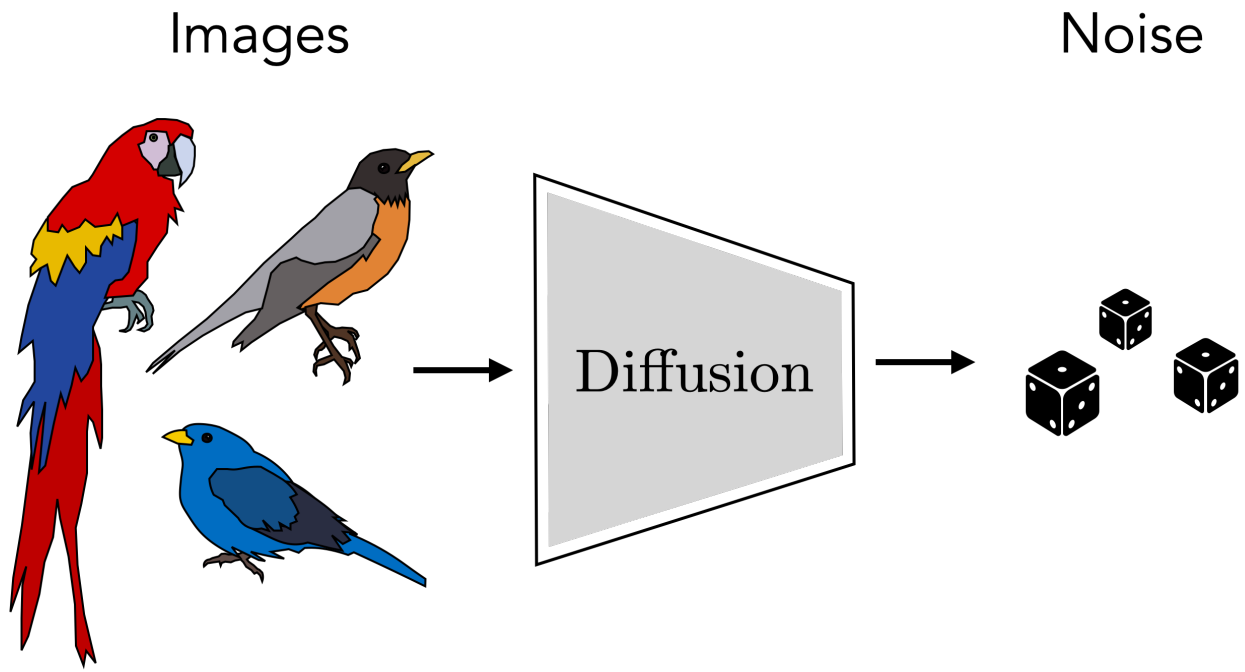

Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015 Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020
Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021

slide from https://cvpr2022-tutorial-diffusion-models.github.io/
Courtesy of Ruiqi Gao

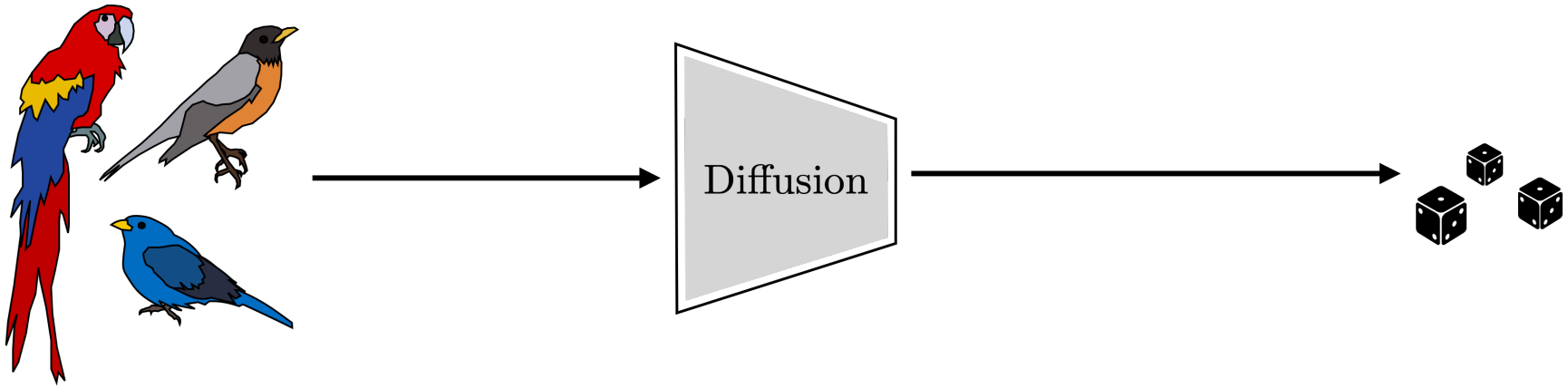
# Diffusion Models

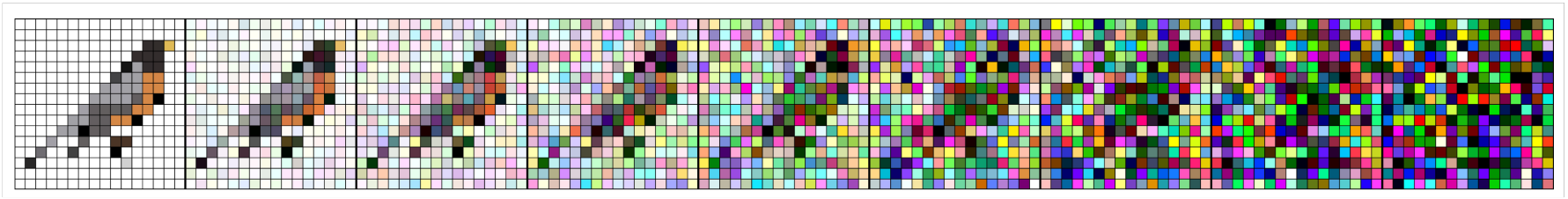


# Diffusion Models

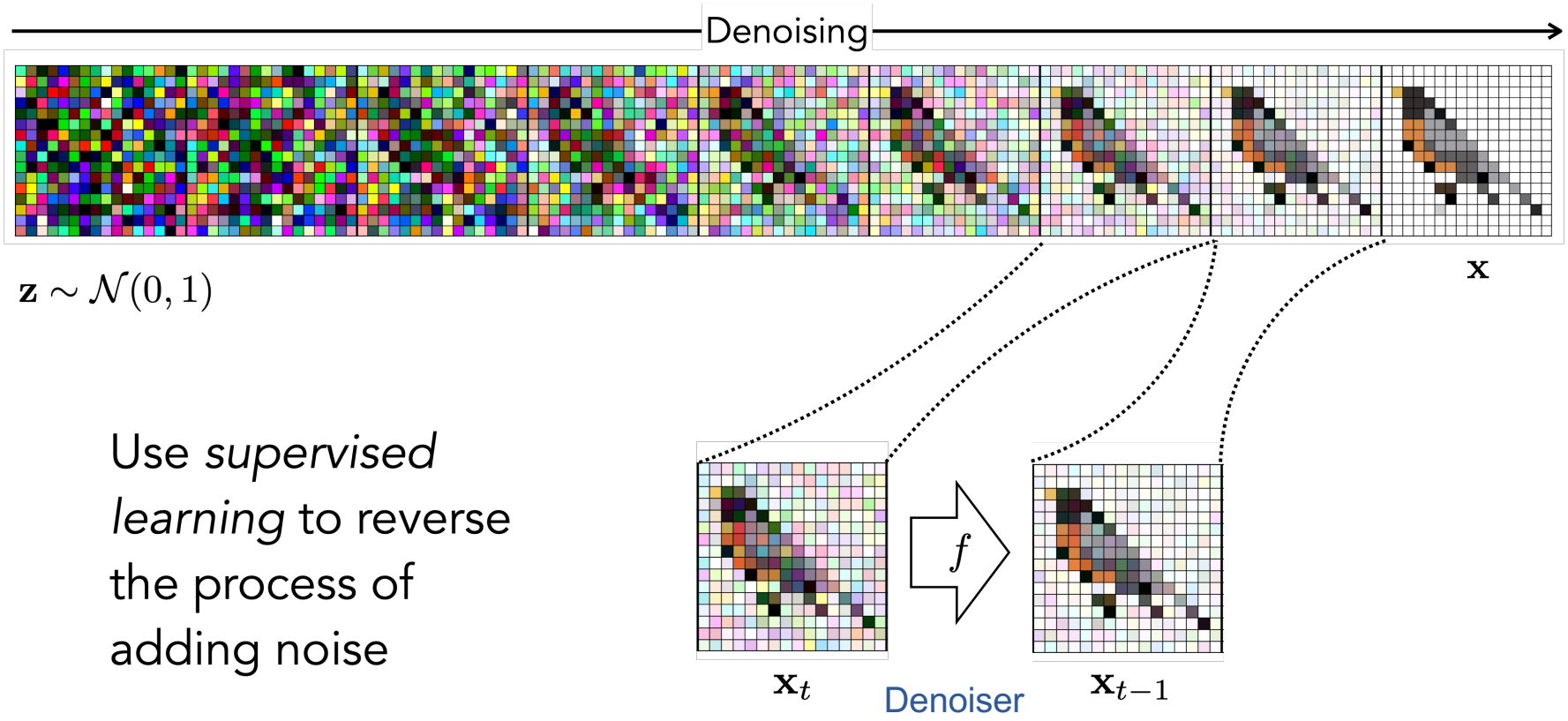


# Diffusion Models



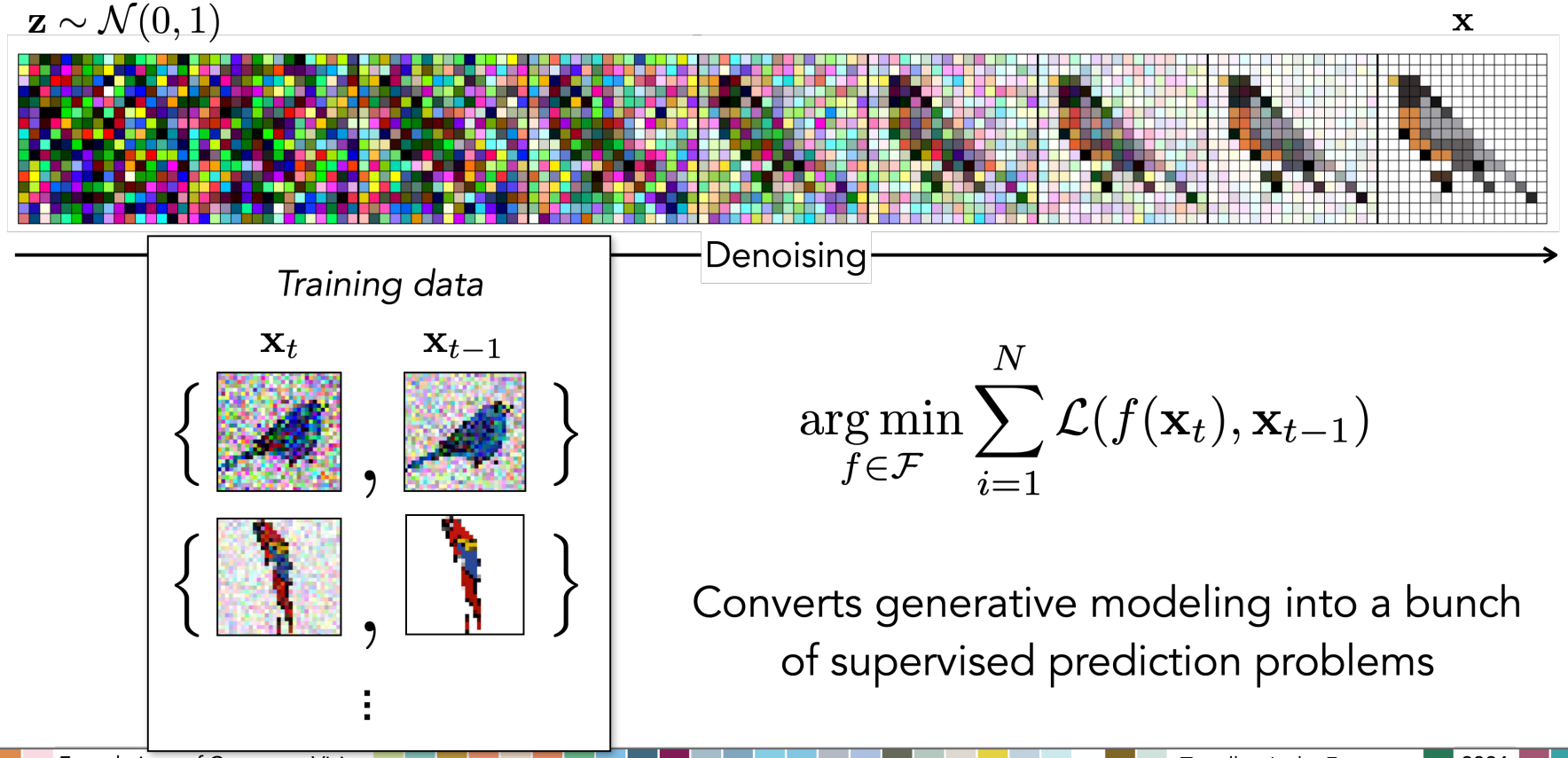



# Diffusion Models




Use *supervised learning* to reverse the process of adding noise

# Diffusion Models

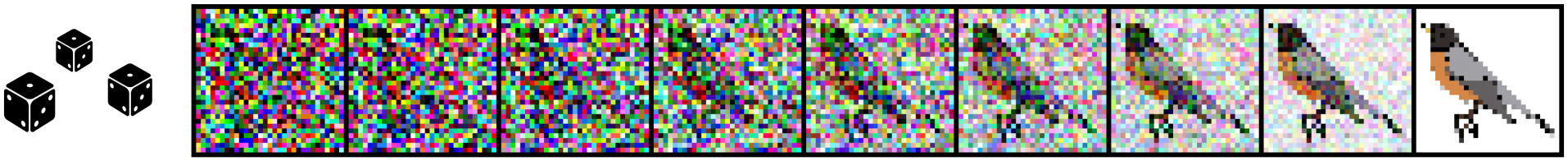



$$\arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}_t), \mathbf{x}_{t-1})$$

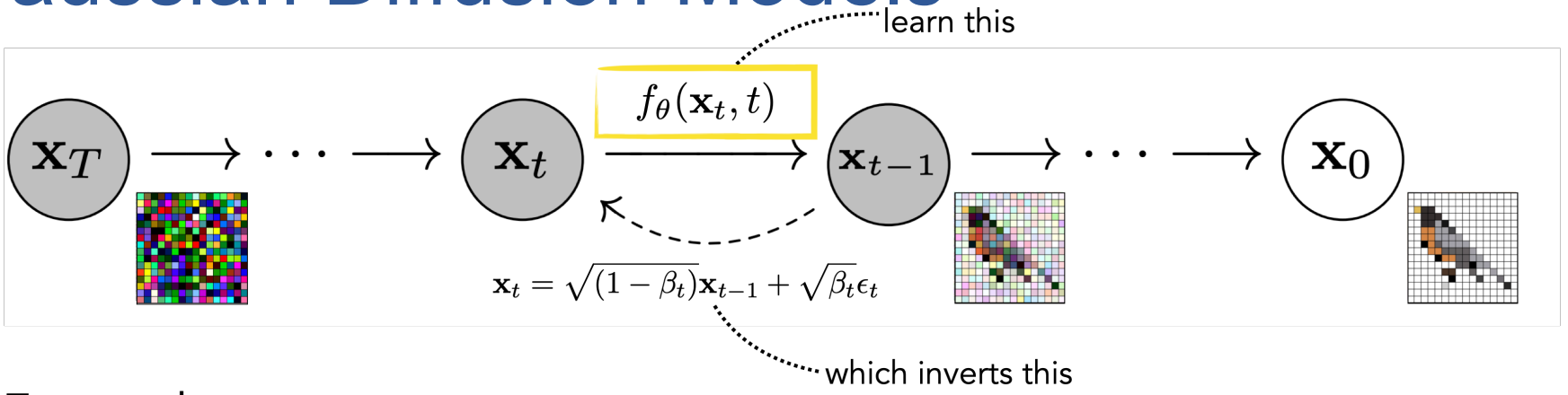
Converts generative modeling into a bunch of supervised prediction problems

# Diffusion Models

Different noise samples (dice rolls) result in different images



# Gaussian Diffusion Models




Forward process:

$$\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \qquad \mathbf{x}_t = \sqrt{(1-\beta_t)}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon_t$$

The variances, beta and sigma, are modeling choices. See Ho, Jain, and Abbeel for details.

Reverse process:

$$\mathbf{x}_{t-1} = f_\theta(\mathbf{x}_t, t)$$

# Diffusion Models: Training

1. Generate Training data by corrupting a bunch of images (forward process; noising)

2. Train a neural net to invert each step of corruption (reverse process; denoising)



# Diffusion Models: Training

**Algorithm 1.2**: Training a diffusion model.

**Input:** training data $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$
**Output:** trained model $f_\theta$
**Generate training sequences via diffusion:**
**for** $i = 1, \ldots, N$ **do**
  **for** $t = 1, \ldots, T$ **do**
    $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{x}_t^{(i)} \leftarrow \sqrt{(1-\beta_t)}\mathbf{x}_{t-1}^{(i)} + \sqrt{\beta_t}\epsilon_t$

Forward Process. Noising

**Train denoiser** $f_\theta$ **to reverse these sequences:**
$\theta^* = \arg\min_\theta \sum_{i=1}^{N} \sum_{t=1}^{T} \mathcal{L}(f_\theta(\mathbf{x}_t^{(i)}, t), \mathbf{x}_{t-1}^{(i)}))$
**Return:** $f_{\theta^*}$

Reverse Process. Denoising



Lab 10c

To join, go to: ahaslides.com/IKCMO

AhaSlides

# How do Denoising Diffusion Models train their neural network during the learning phase?

| 0 | 0 | 0 | 0 |
| --- | --- | --- | --- |
| By corrupting training …es with noise and … a net to invert each …orruption ste | By clustering unlabelled points using distance metrics in component space | By playing a minimax game where a discriminator classifies data as real or synthetic | By maximizing the exact intractable joint data density function directly via integration |

Edit slides

Add slides

0 0 /100

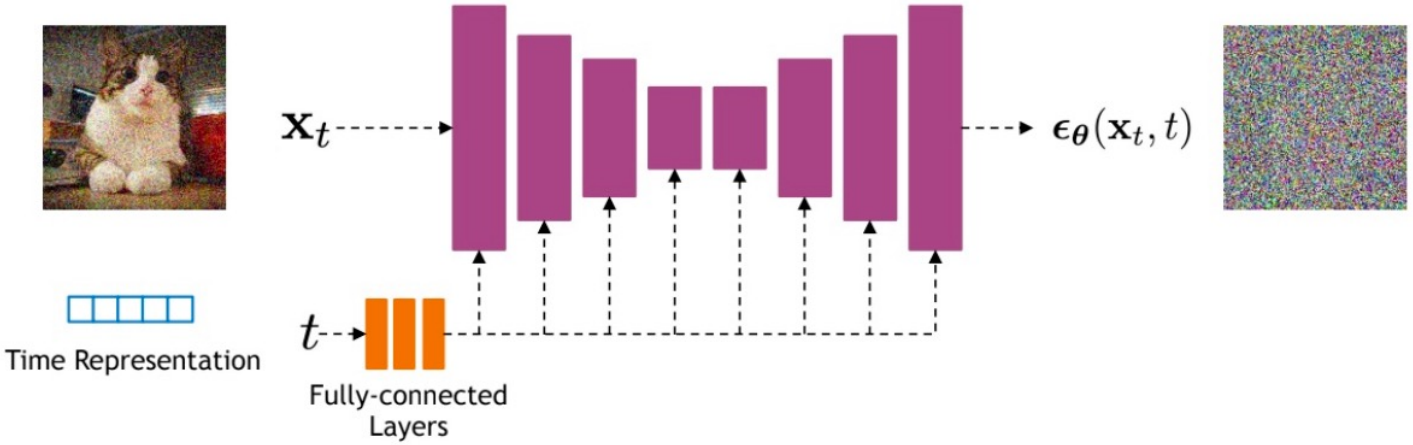
# Lab 10c: Diffusion Models

Duration: 10 min



# Diffusion Models: Training – Unet model transition

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$

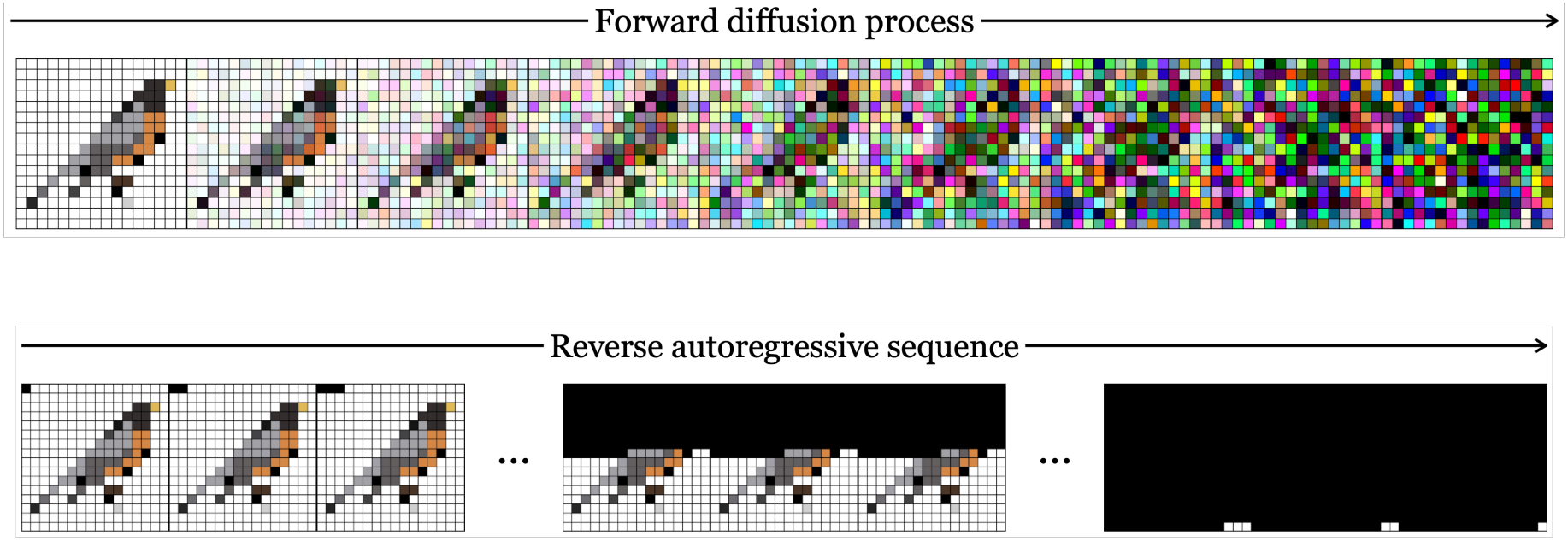



Time representation: sinusoidal positional embeddings or random Fourier features.

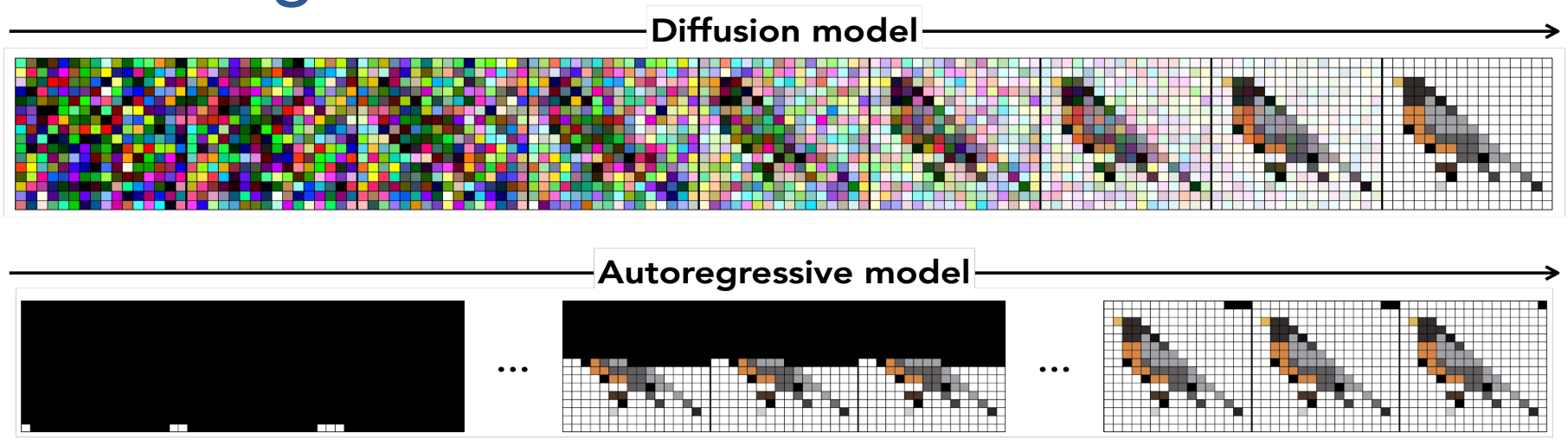
Time features are fed to the residual blocks using either simple spatial addition or using adaptive group normalization layers. (see Dharivwal and Nichol NeurIPS 2021)

https://cvpr2022-tutorial-diffusion-models.github.io/

# Autoregressive Modes vs Diffusion Models



# Autoregressive Modes vs Diffusion Models




A common strategy is to turn generative modeling into a sequence of supervised learning problems

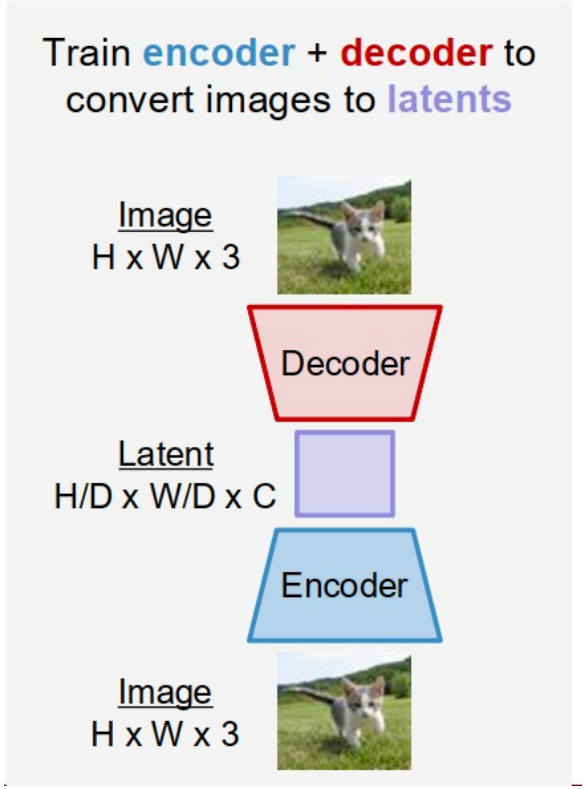
# Latent Diffusion Models: Motivation

- High-resolution image: 3 color channels, height 600, width 1000 → the resulting dimension is: 3*600*1000 = **1.8 million!**
- **This is a very high-dimensional space!!!!**



https://diffusion.csail.mit.edu/2026/

# Latent Diffusion Models (LDM)




Common setting: D=8, C=16

**Image**: 256 x 256 x 3

=> **Latent**: 32 x 32 x 16

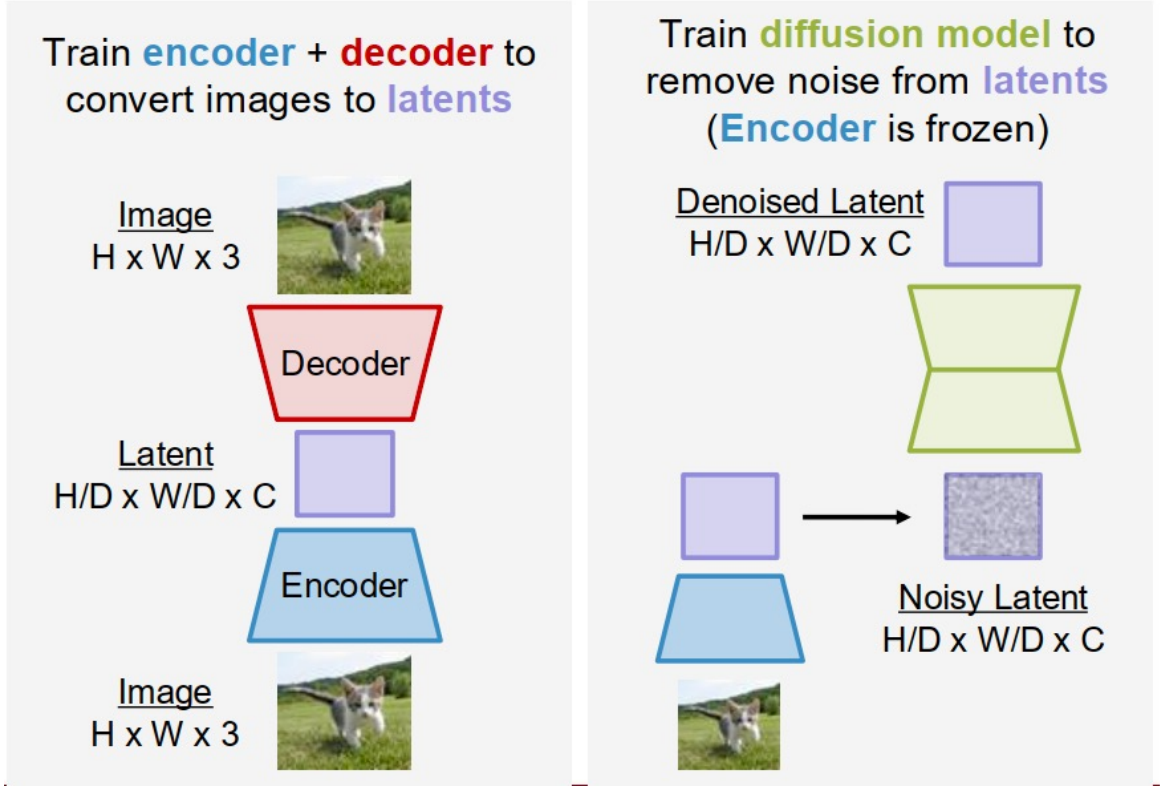
Encoder / Decoder are CNNs with attention



Rombach et al, "High-Resolution Image Synthesis with Latent Diffusion Models", CVPR 2022

# Latent Diffusion Models (LDM)

Latent diffusion is the most common form today!

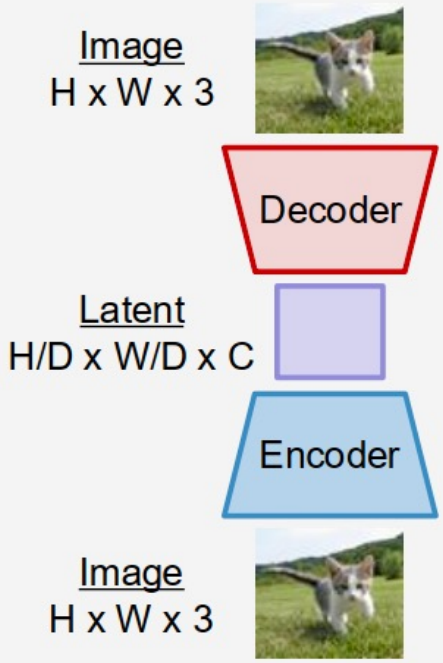



https://cs231n.stanford.edu/

Rombach et al, "High-Resolution Image Synthesis with Latent Diffusion Models", CVPR 2022

# Latent Diffusion Models (LDM)

Train encoder + decoder to convert images to latents



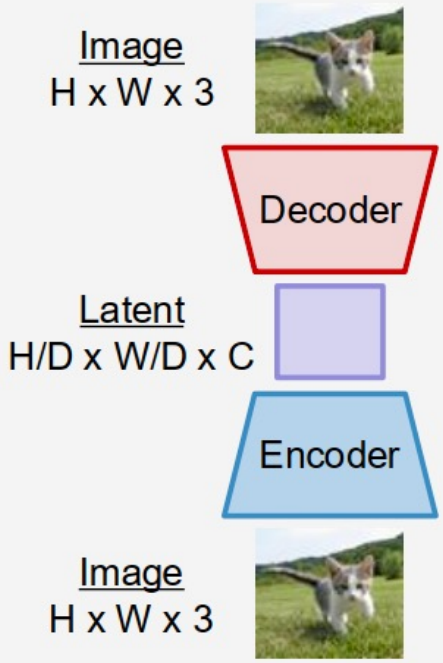

How do we train the encoder+decoder?



Rombach et al, "High-Resolution Image Synthesis with Latent Diffusion Models", CVPR 2022

# Latent Diffusion Models (LDM)

Train encoder + decoder to convert images to latents

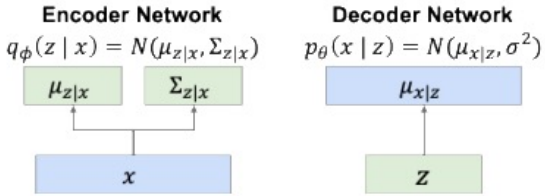



How do we train the encoder+decoder?

**Solution**: It's a VAE! Typically with very small KL prior weight

Recall: VAE

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

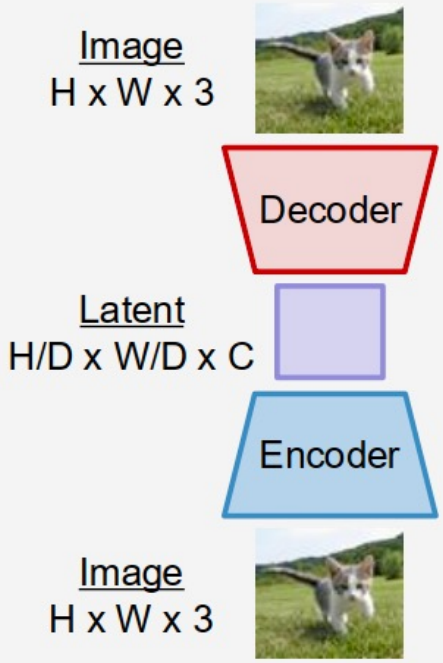



https://cs231n.stanford.edu/

Rombach et al, "High-Resolution Image Synthesis with Latent Diffusion Models", CVPR 2022

# Latent Diffusion Models (LDM)

Train **encoder** + **decoder** to convert images to **latents**



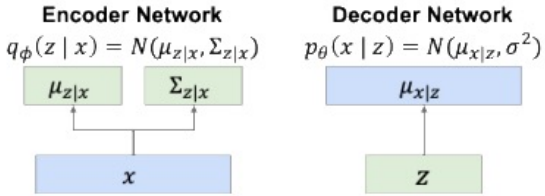

How do we train the encoder+decoder?

**Solution**: It's a VAE! Typically with very small KL prior weight

**Problem**: Decoder outputs often blurry

Recall: VAE

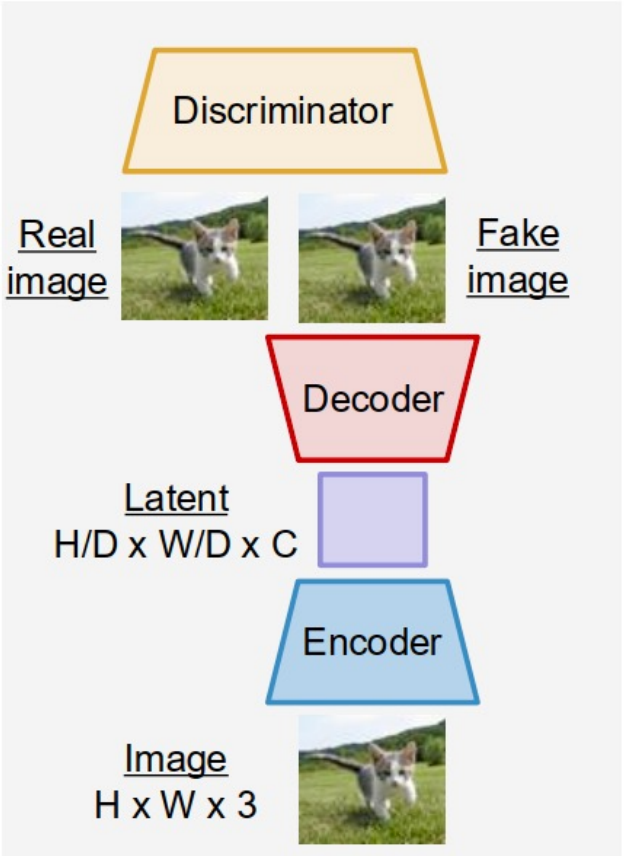
$$\log p_\theta(x) \geq E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$




https://cs231n.stanford.edu/

Rombach et al, "High-Resolution Image Synthesis with Latent Diffusion Models", CVPR 2022

# Latent Diffusion Models (LDM)




How do we train the encoder+decoder?

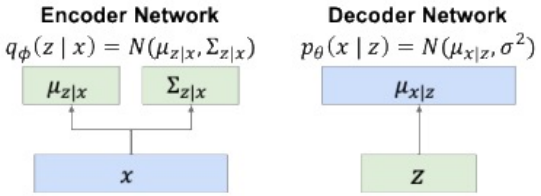
**Solution**: It's a VAE! Typically with very small KL prior weight

**Problem**: Decoder outputs often blurry

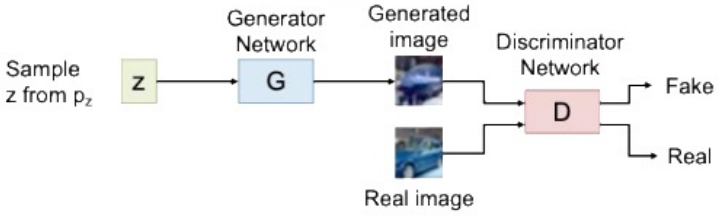
**Solution**: Add a discriminator!

Recall: VAE

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




Recall: GAN



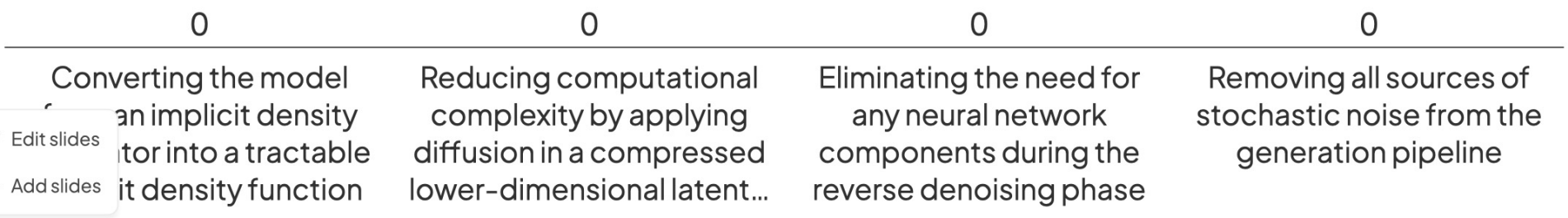

https://cs231n.stanford.edu/

Rombach et al, "High-Resolution Image Synthesis with Latent Diffusion Models", CVPR 2022

To join, go to: **ahaslides.com/IKCMO**

AhaSlides

# What is the primary motivation for introducing Latent Diffusion Models (LDMs) compared to operating directly in high-resolution pixel space?




Edit slides

Add slides

Get Feedback

0 0 /100

# 2022 / 2023 : The year of diffusion and generative modeling?










DALL·E 2

Stable Diffusion

slide from https://cvpr2022-tutorial-diffusion-models.github.io/
Courtesy of Ruiqi Gao

# Useful Resources on Generative Models

CS 236: Deep Generative Models [youtube](Stanford)

CS 294-158 Deep Unsupervised Learning [youtube] (Berkeley)

Introduction to Flow Matching and Diffusion Models (MIT)

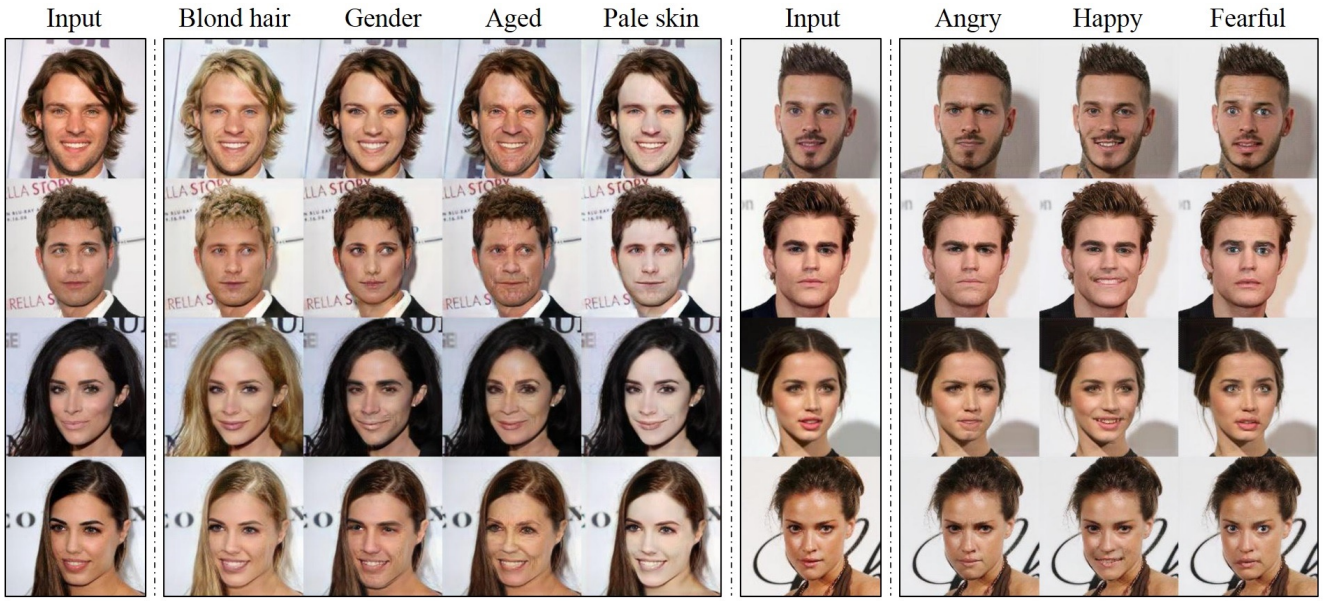


# Applications: Celebrities Who Never Existed



*Karras et al., "Progressive Growing of GANs for Improved Quality, Stability, and Variation", ICLR 2018*

# Applications: StarGAN




Choi et al., "StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation", CVPR 2018

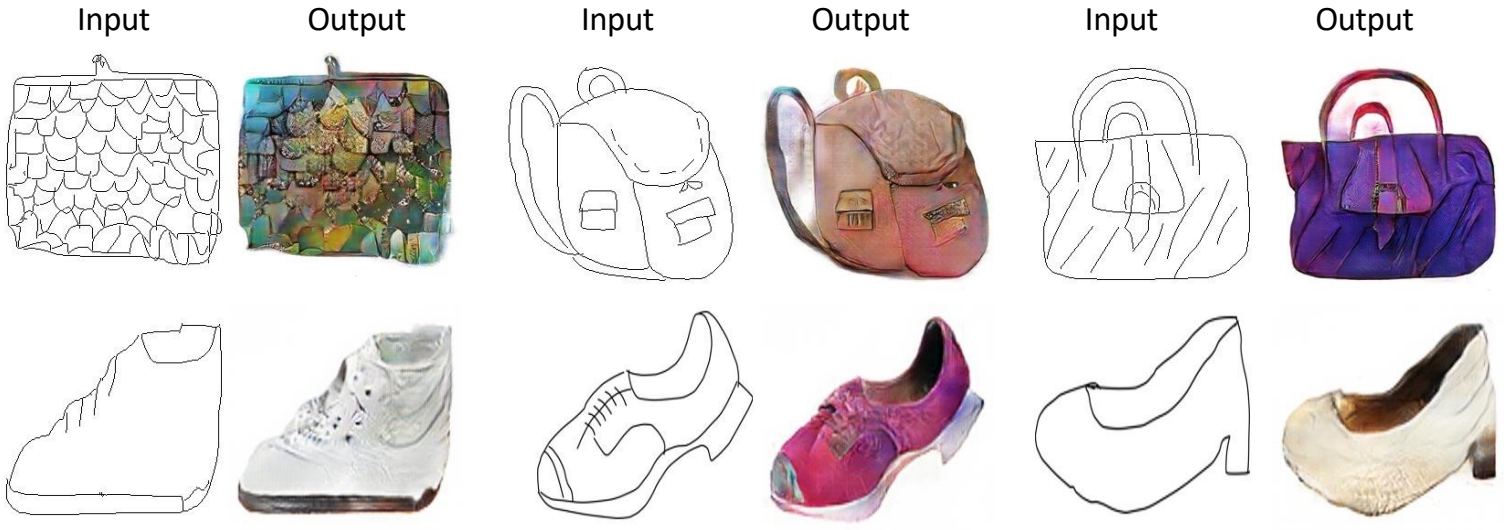
# Applications: Edges to Images




Edges from [Xie & Tu, 2015]

Pix2pix / CycleGAN

# Applications: Sketches to Images




Trained on Edges → Images

Data from [Eitz, Hays, Alexa, 2012]

Pix2pix / CycleGAN

# Applications: Changing Artistic Style




Pix2pix / CycleGAN

# Applications: Changing Seasons



Pix2pix / CycleGAN

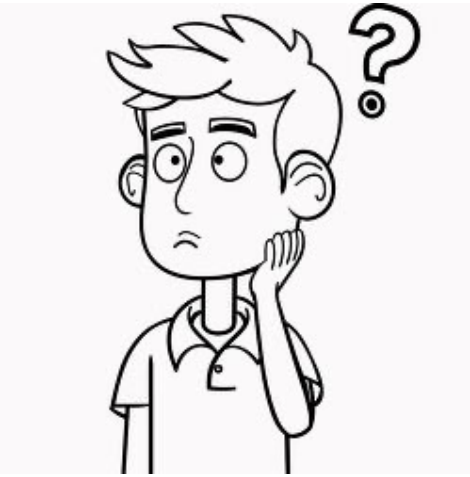
# Muddiest Point

What was the most confusing or 'muddiest' point from previous lectures? or What topic do you need more clarification on?



https://classroomclipart.com/

Submit on AhaSlides



https://classroomclipart.com/

To join, go to: ahaslides.com/ANLAD

AhaSlides

# What was the most confusing or 'muddiest' point from previous lectures? or What topic do you need more clarification on?

Get Feedback

Edit slides

Add slides

Group into themes

0 0 /100

# Extra

# Taxonomy of Generative Models

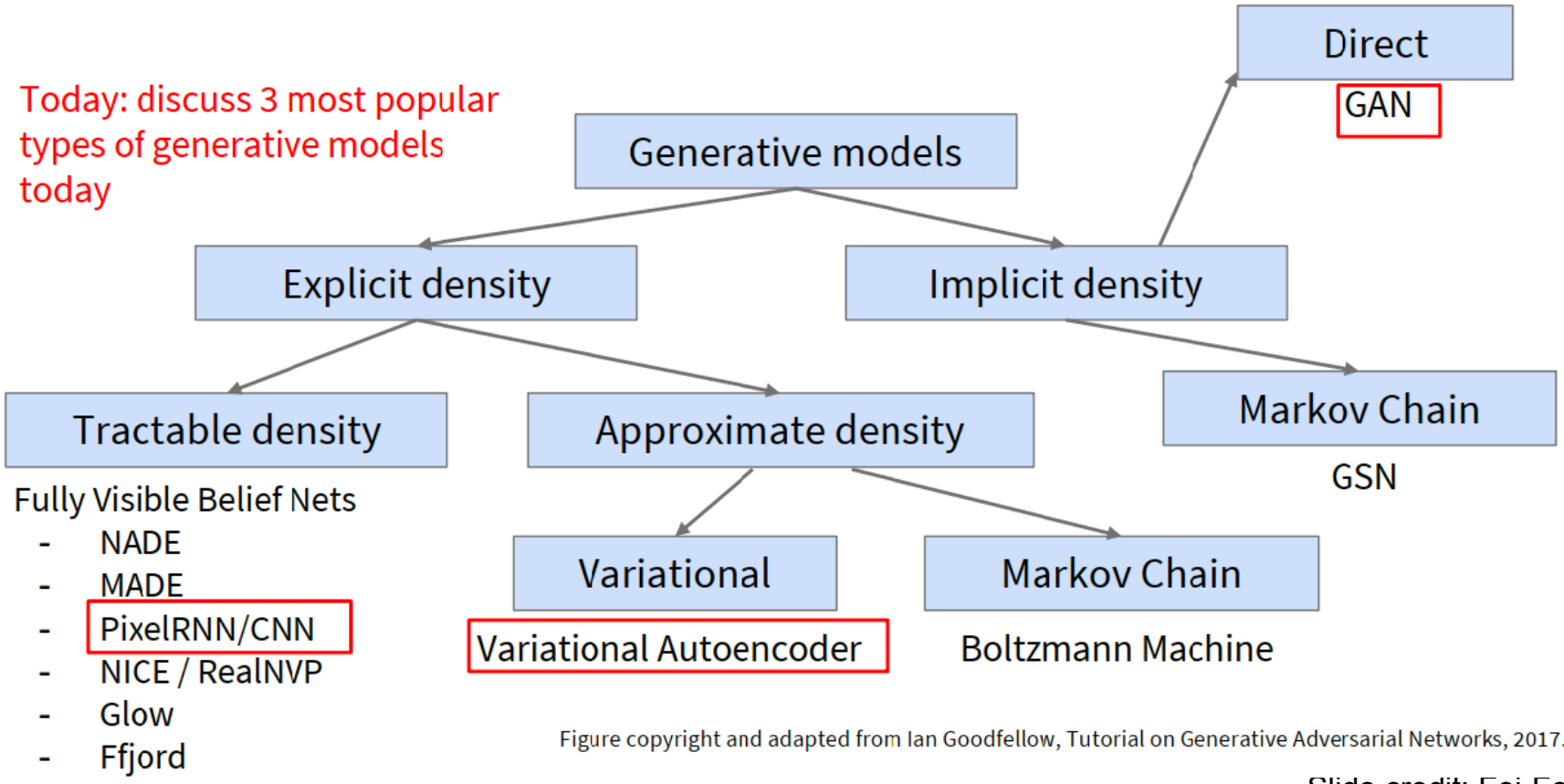



Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Slide credit: Fei-Fei Li

# PixelCNN [van der Oord et al. 2016]

Still generate image pixels starting from corner

Dependency on previous pixels now modeled using a CNN over context region (masked convolution)

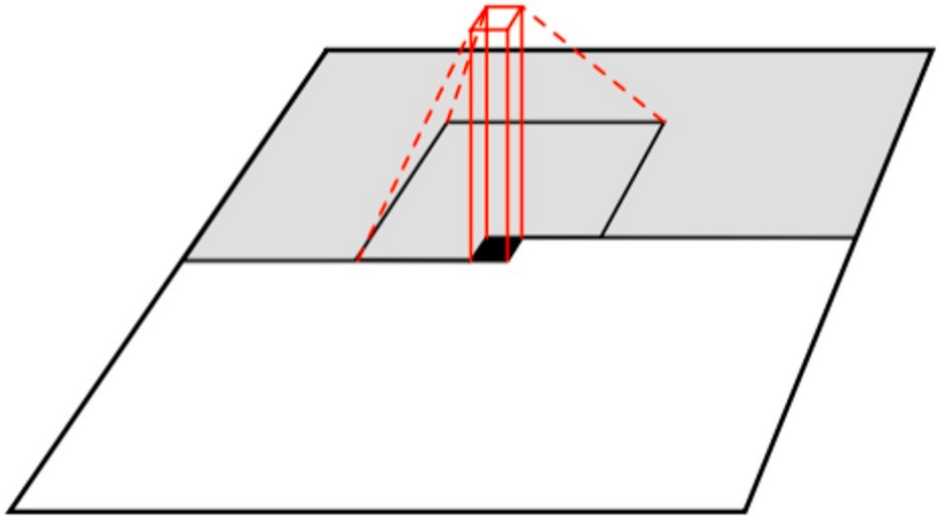


Figure copyright van der Oord et al., 2016

Slide credit: Fei-Fei Li

# PixelCNN [van der Oord et al. 2016]

Still generate image pixels starting from corner

Dependency on previous pixels now model using a CNN over context region (masked convolution)

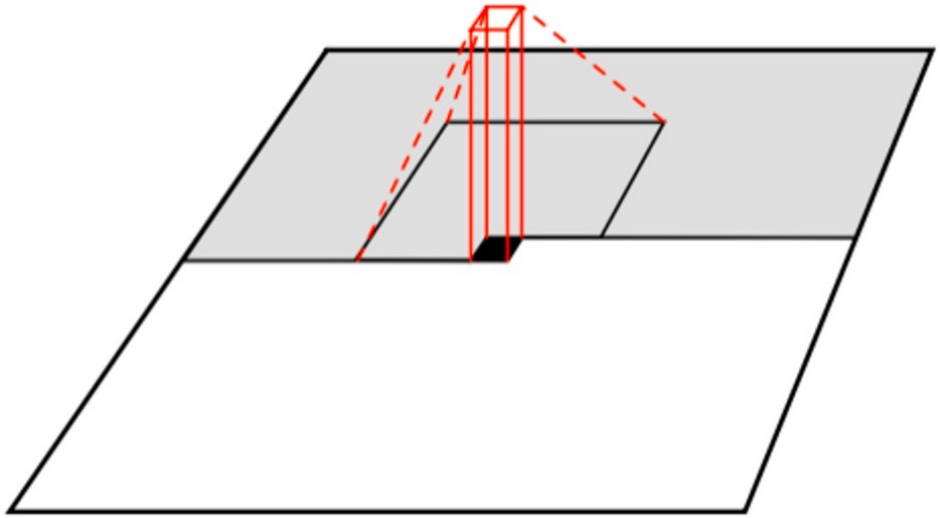


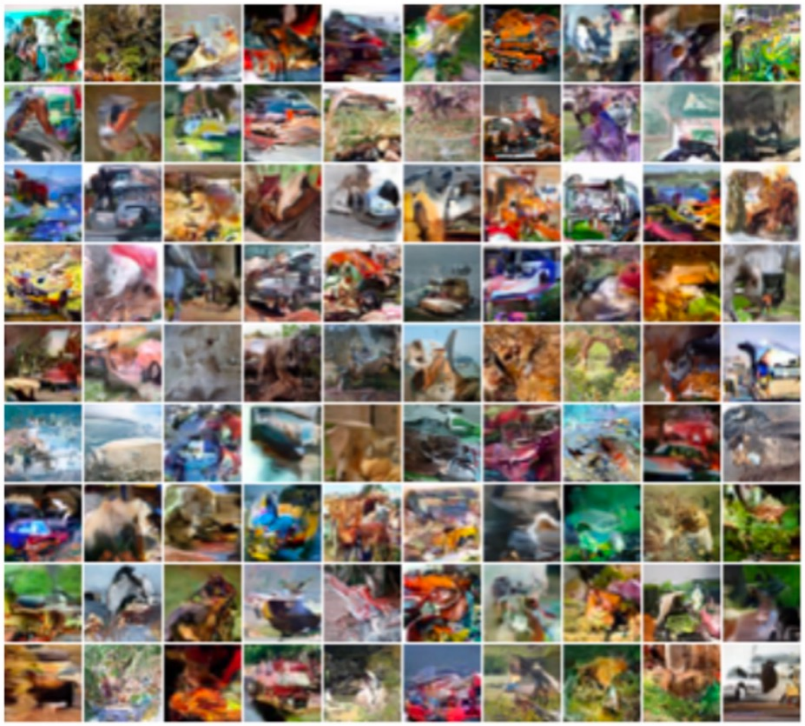
Figure copyright van der Oord et al., 2016

Training is faster than PixelRNN
(can parallelize convolutions since context Region values known from training images)

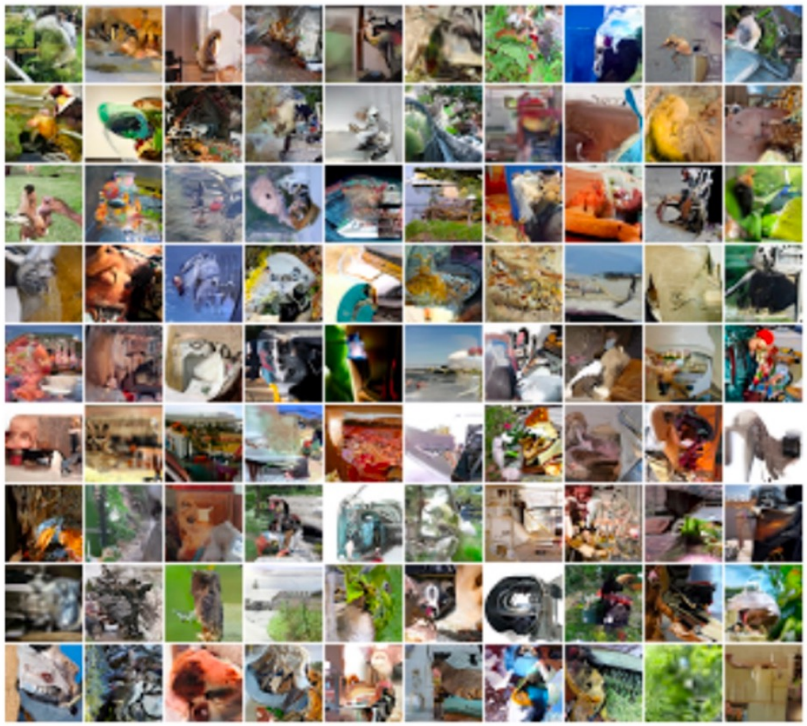
Generation is still slow:
For a 32x32 image, we need to do forward passes of the network 1024 times for a single image

Slide credit: Fei-Fei Li

# Generation Samples



32x32 CIFAR-10



32x32 ImageNet

Figures copyright Aaron van der Oord et al., 2016

Slide credit: Fei-Fei Li

# PixelRNN and PixelCNN

Pros:

- Can explicitly compute likelihood p(x)
- Easy to optimize
- Good samples

Cons:

- Sequential generation => slow

Improving PixelCNN performance

- Gated convolutional layers
- Short-cut connections
- Discretized logistic loss
- Multi-scale
- Training tricks
- Etc…

See

- Van der Oord et al. NIPS 2016
- Salimans et al. 2017 (PixelCNN++)

Slide credit: Fei-Fei Li

# How to improve PixelCNN?

What are the limitations of PixelCNN/RNN?

- Slow sampling time.
- May accumulate errors over multiple steps. (might not be a big issue for image completion)



How can we further improve results?

Slide credit: Jun-Yan Zhu - Learning-Based Image Synthesis

# VQ-VAE-2 :VAE+PixelCNN

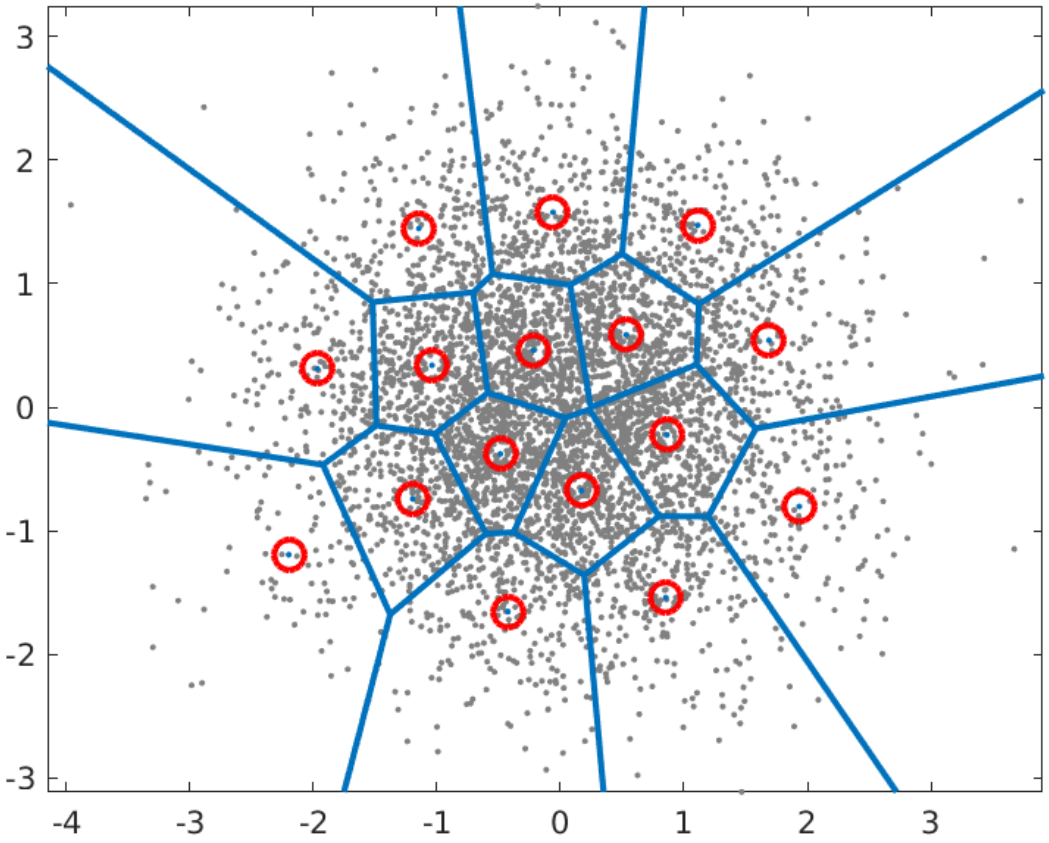


VQ (Vector quantization) maps continuous vectors into discrete codes

**Common methods**: clustering (e.g., k-means)

Generating Diverse High-Fidelity Images with VQ 45 -VAE-2 [Razavi et al., 2019]

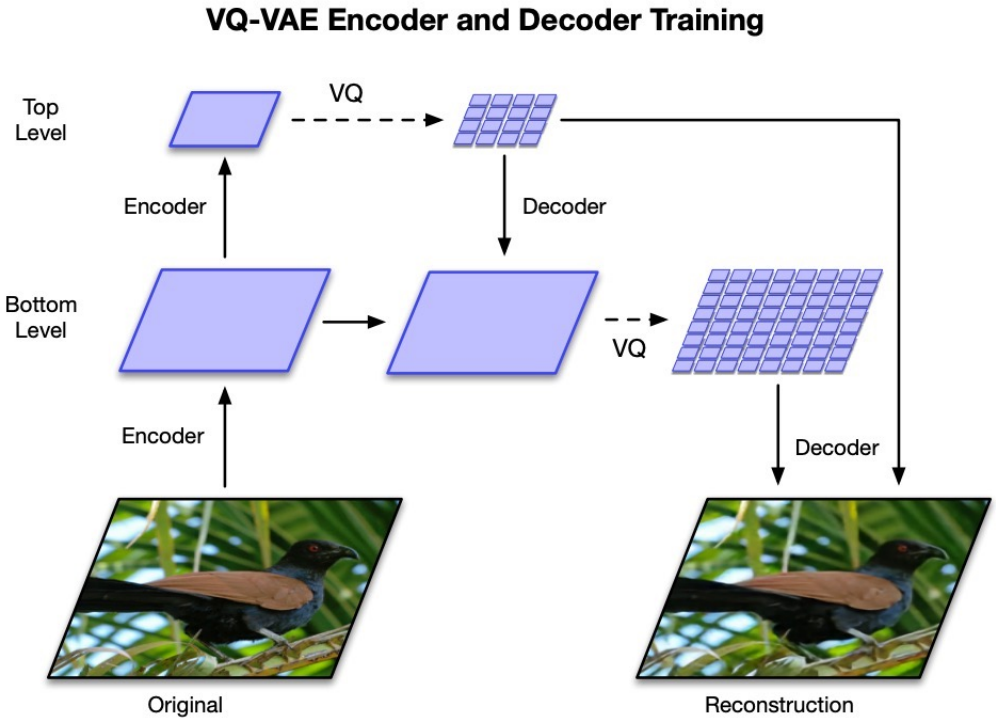
# VQ-VAE-2 : Vector Quantization




K-means, EM (GMM),
end-to-end learning

https://wiki.aalto.fi/pages/viewpage.action?pageId=149883153

# VQ-VAE-2 :VAE+PixelCNN




VAE+VQ: learn a more compact codebook for PixelCNN (instead of pixels)
PixelCNN: use a more expressive bottleneck for VAE (instead of Gaussian

Generating Diverse High-Fidelity Images with VQ 45 -VAE-2 [Razavi et al., 2019]

# Taxonomy of Generative Models

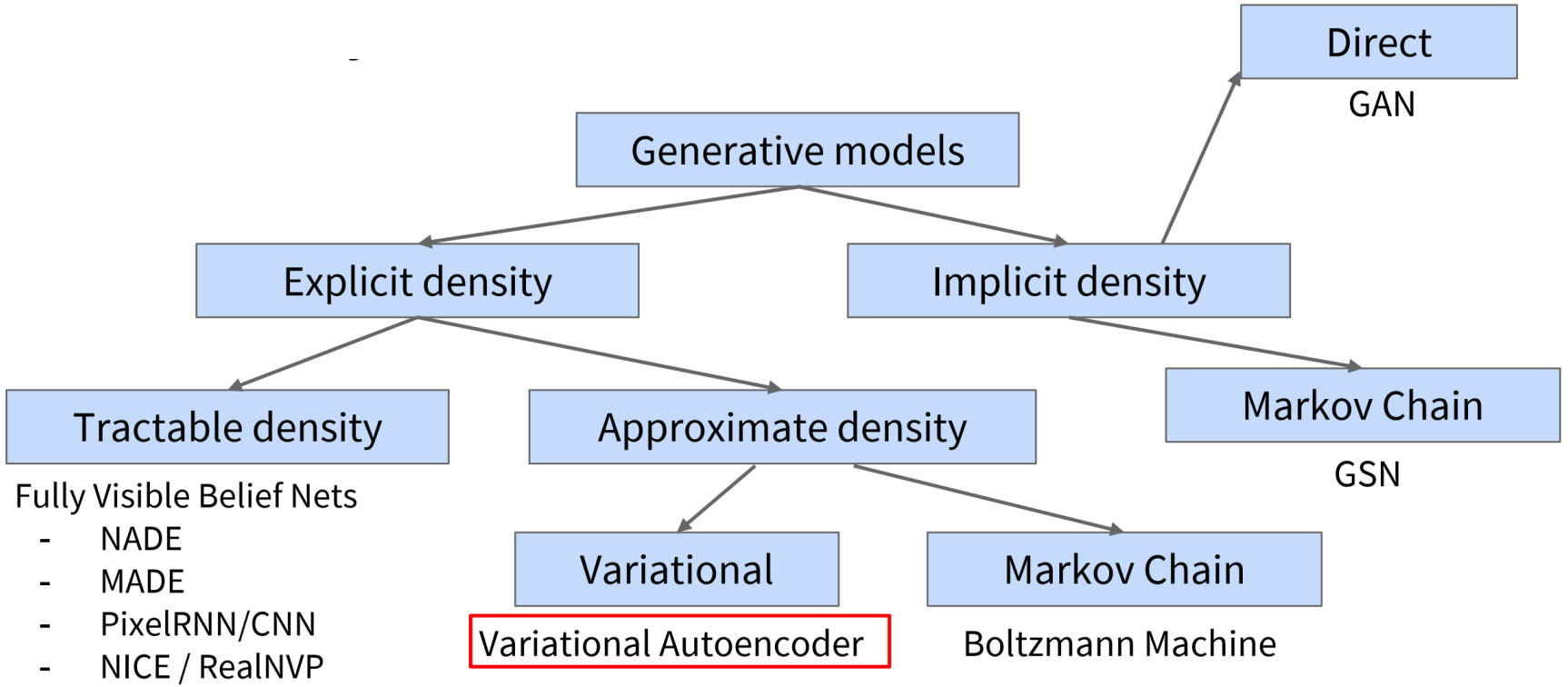



Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Slide credit: Fei-Fei Li

# The Myth of the Cave



https://www.youtube.com/watch?v=1RWOpQXTltA&ab_channel=TED-Ed

# What is a Latent Variable?



https://en.wikipedia.org/wiki/Myth_of_the_Cave

# The Myth of the Cave



https://en.wikipedia.org/wiki/Myth_of_the_Cave

Can we learn the **true explanatory factors** (e.g. latent variables) from only observed data?

# Variational AutoEncoders (VAE)

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

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

Variational Autoencoders (VAEs) define intractable density function with latent z:

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

No dependencies among pixels, can generate all pixels at the same time!

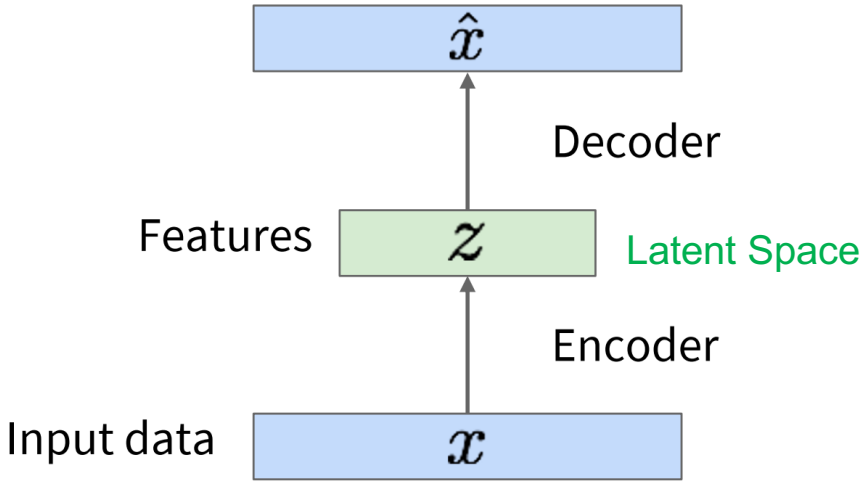
Cannot optimize directly, derive and optimize lower bound on likelihood instead

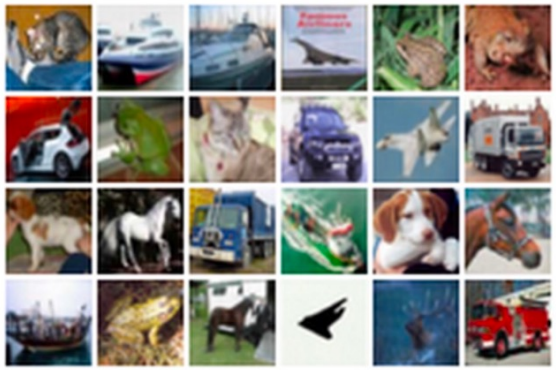
## Why latent z?

Slide credit: Fei-Fei Li

# Background: AutoEncoders

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data



Slide credit: Fei-Fei Li

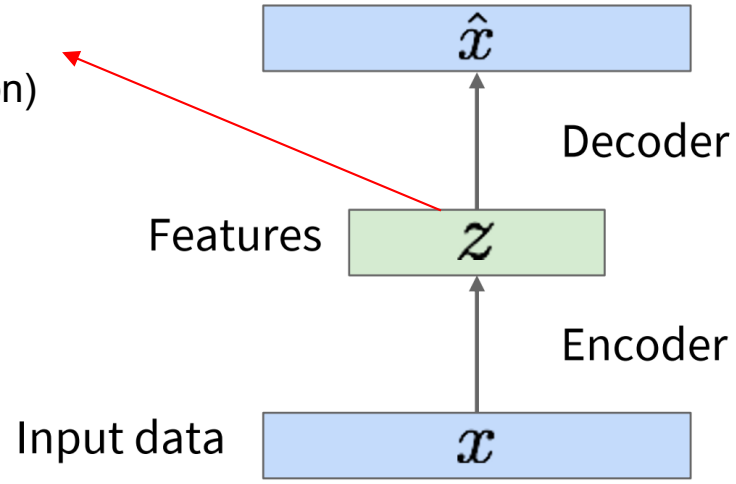
# Background: AutoEncoders

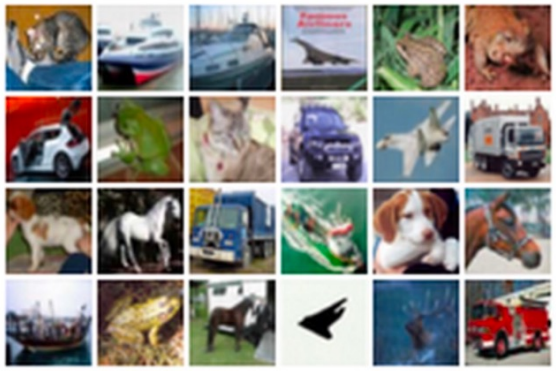
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

z usually smaller than x
(dimensionality reduction)

Q: Why dimensionality reduction?

A: Want features to capture meaningful factors of variation in data



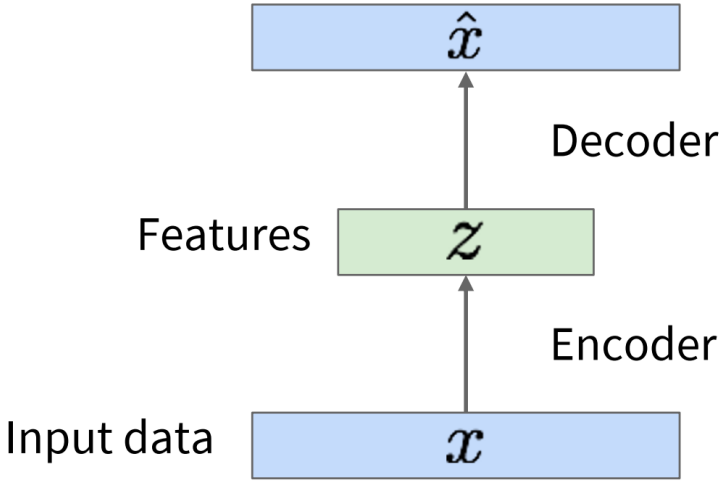



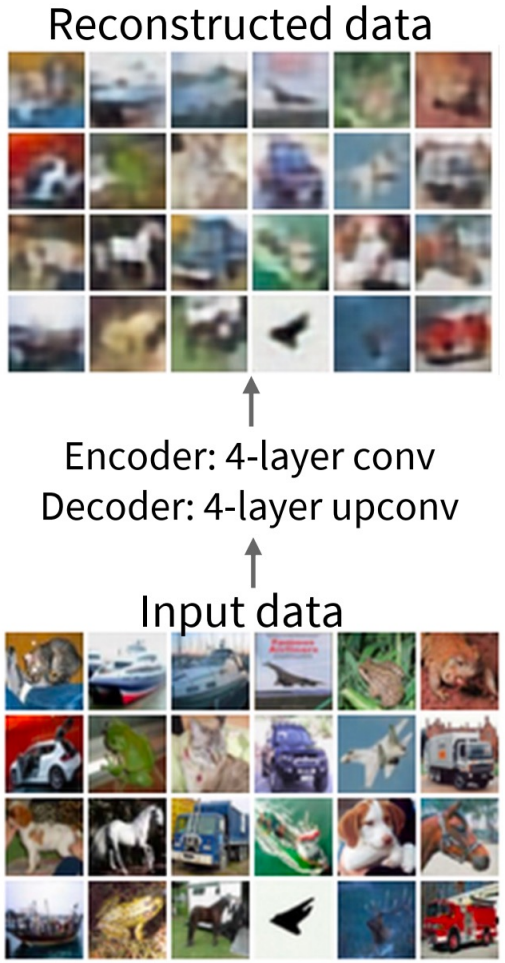
Slide credit: Fei-Fei Li

# Background: AutoEncoders

How to learn this feature representation?

Train such that features can be used to reconstruct original data
"Autoencoding" - encoding input itself



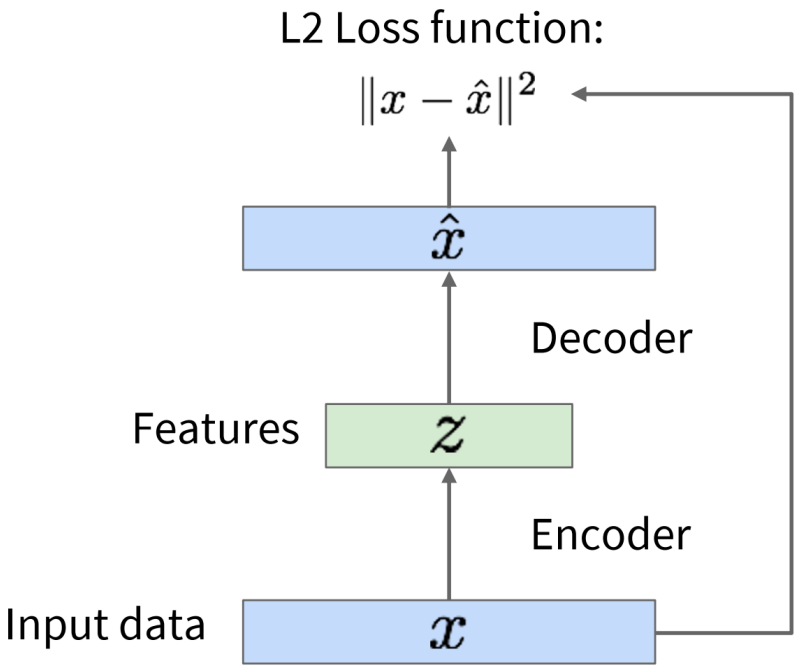



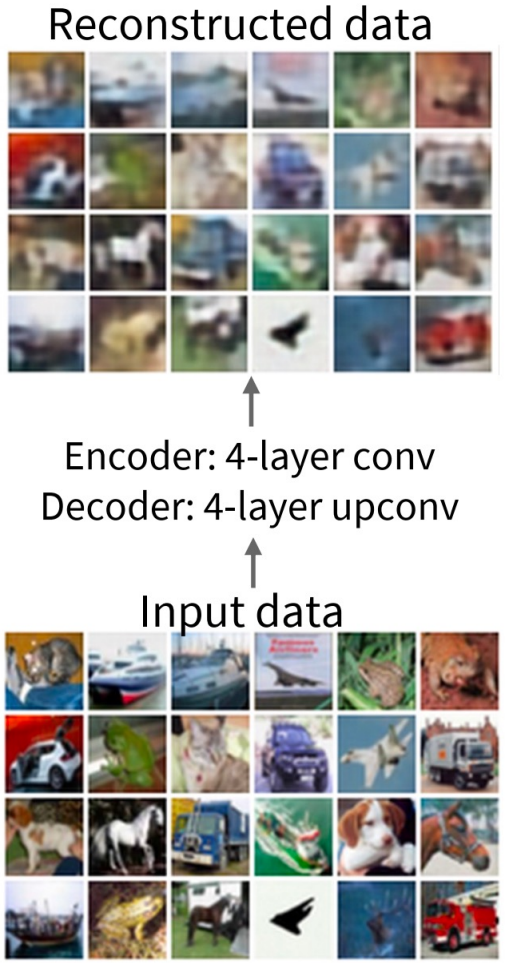

Slide credit: Fei-Fei Li

# Background: AutoEncoders

Train such that features can be used to reconstruct original data

Doesn't use labels!

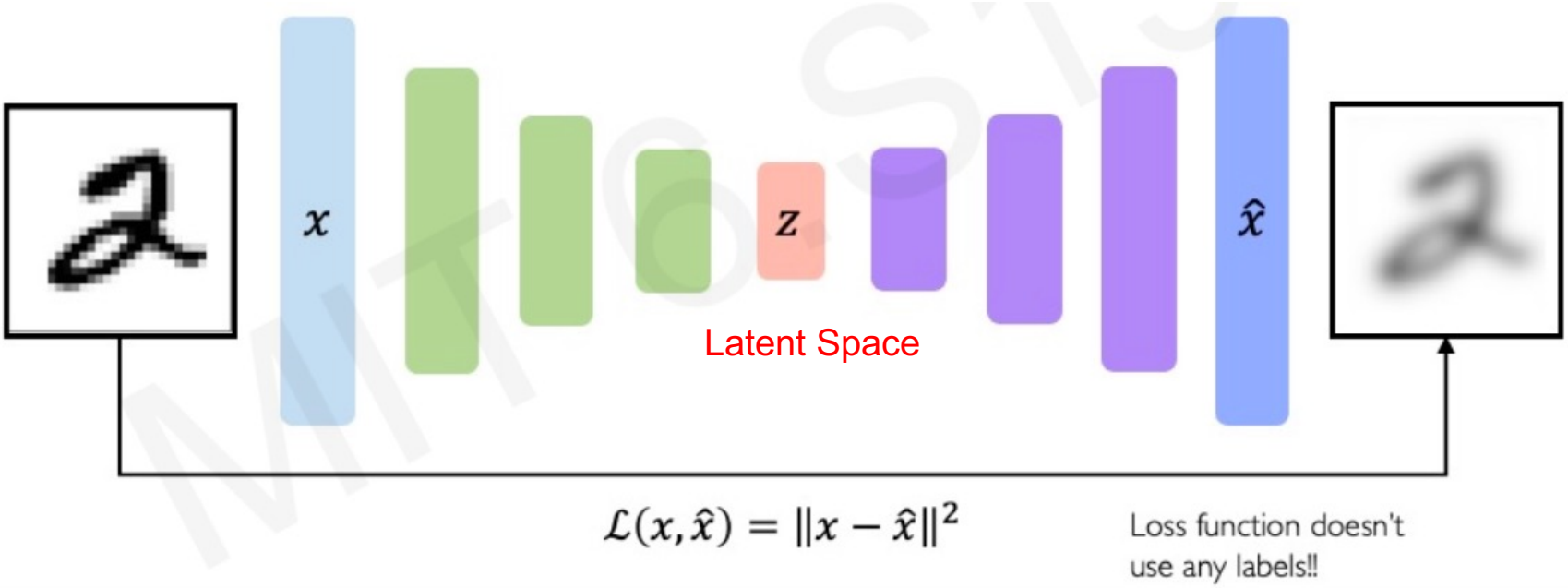


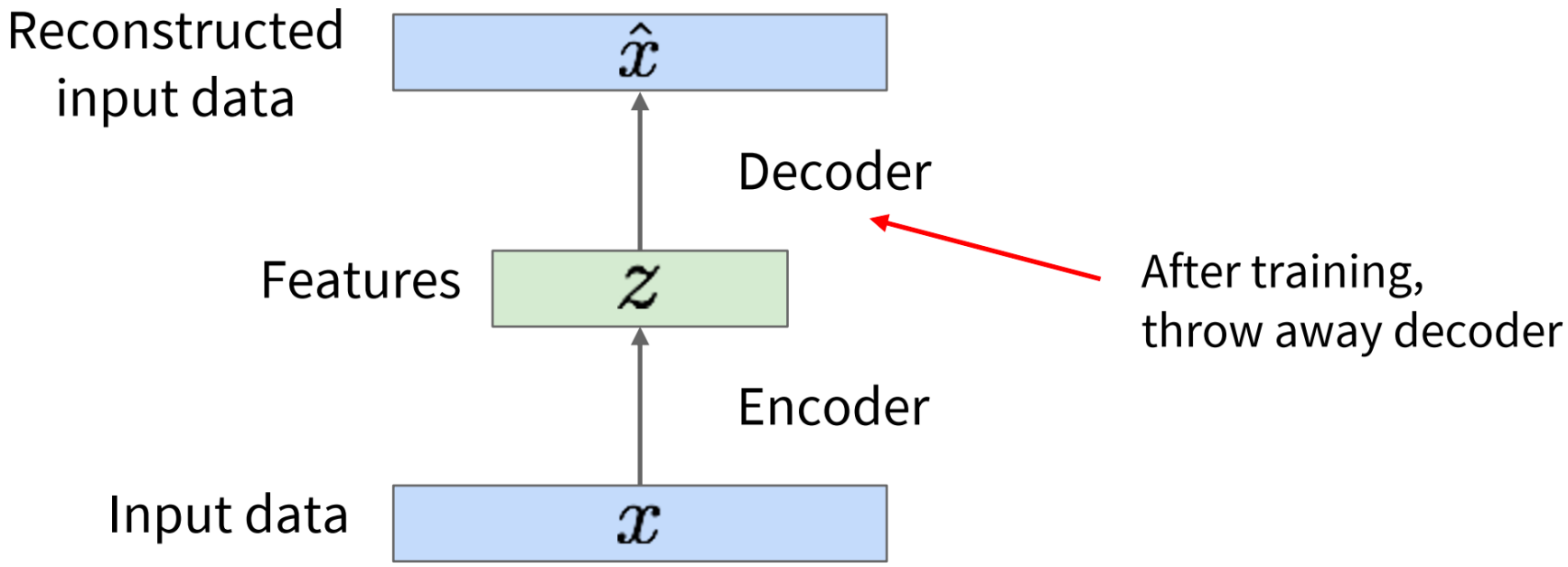

Slide credit: Fei-Fei Li

# Background: AutoEncoders

How can we learn this latent space?
Train the model to use these features to **reconstruct the original data.**



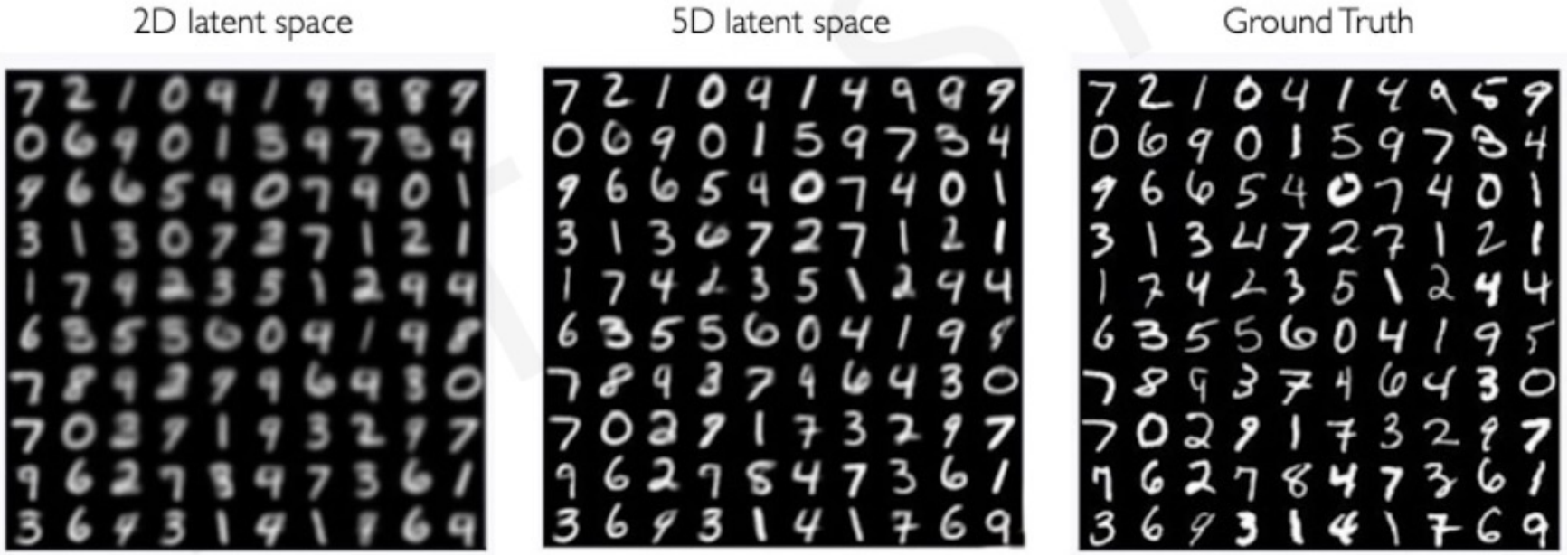
# Background: AutoEncoders




Slide credit: Fei-Fei Li

# Background: AutoEncoders – Reconstruction Quality

Autoencoding is a form of compression!
Smaller latent space will force a larger training bottleneck

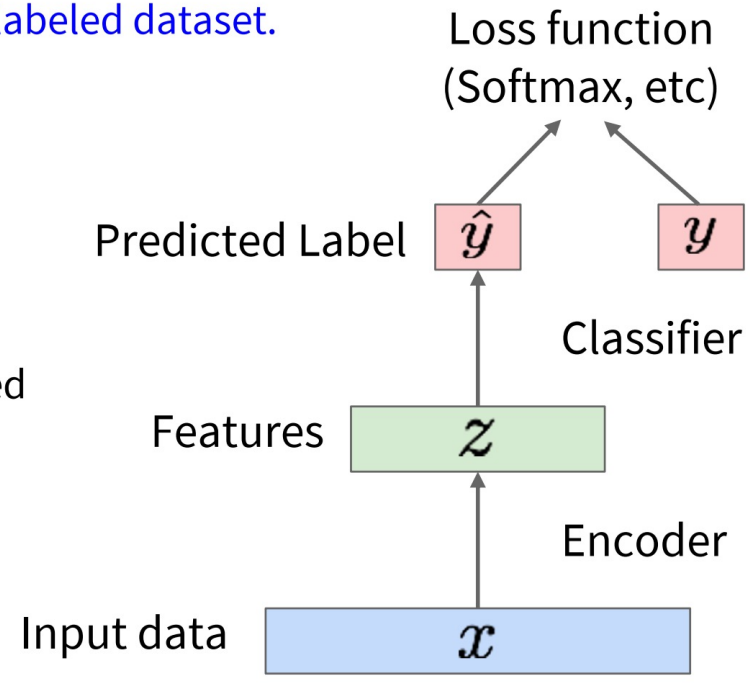



https://introtodeeplearning.com/

Slide credit: Ava Amini

# Background: AutoEncoders

Transfer from large, unlabeled dataset to small, labeled dataset.




Encoder can be used to initialize a supervised model

Fine-tune encoder jointly with classifier

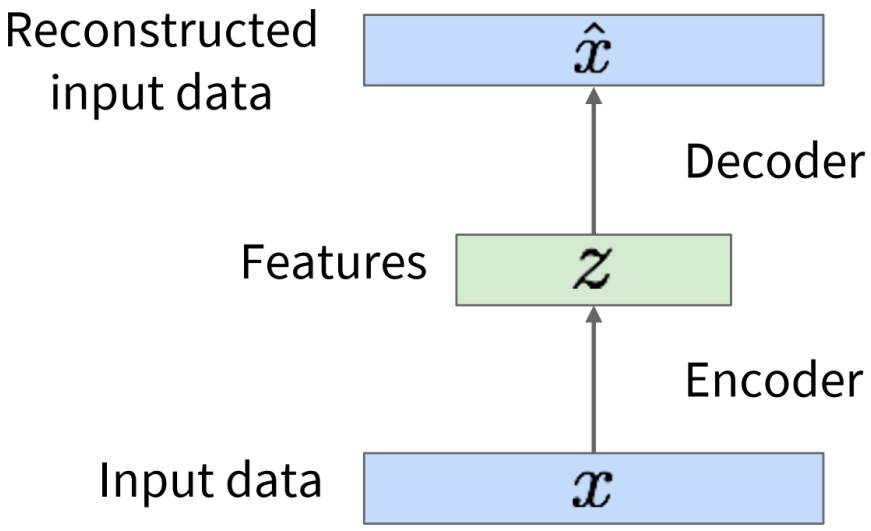
bird plane
dog deer truck

Train for final task (sometimes with small data)



Slide credit: Fei-Fei Li

# Background: AutoEncoders




Autoencoders can reconstruct data, and can learn features to initialize a supervised model

Features capture factors of variation in training data.

But we can't generate new images from an autoencoder because we don't know the space of z.

How do we make autoencoder a generative model?

Slide credit: Fei-Fei Li

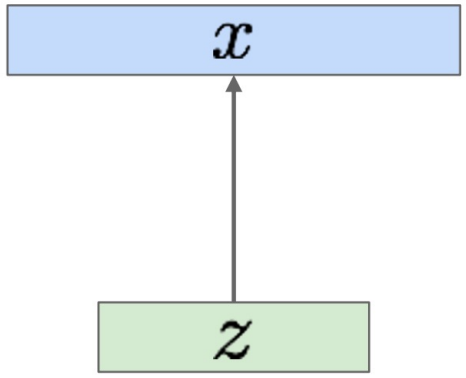
# Variational AutoEncoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from the distribution of unobserved (latent) representation z

Sample from true conditional $p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior $z^{(i)} \sim p_{\theta^*}(z)$

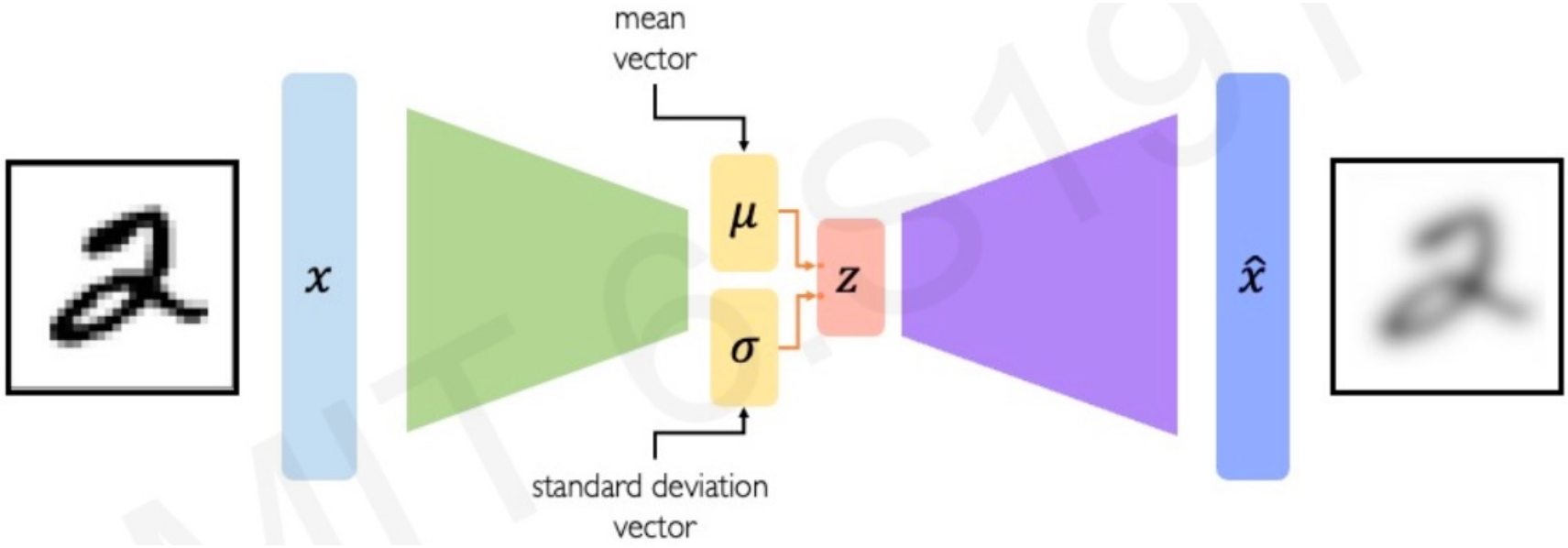



Intuition (remember from autoencoders!): x is an image, z is latent factors used to generate x: attributes, orientation, etc.

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Slide credit: Fei-Fei Li

# Variational AutoEncoders: Key Difference with AutoEncoders




**Variational autoencoders are a probabilistic twist on autoencoders!**
Sample from the mean and standard deviation to compute latent sample

# Variational AutoEncoders

We want to estimate the true parameters $\theta^*$ of this generative model given training data x.

How should we represent this model?

Choose prior p(z) to be simple, e.g. Gaussian. Reasonable for latent attributes, e.g. pose, how much smile.

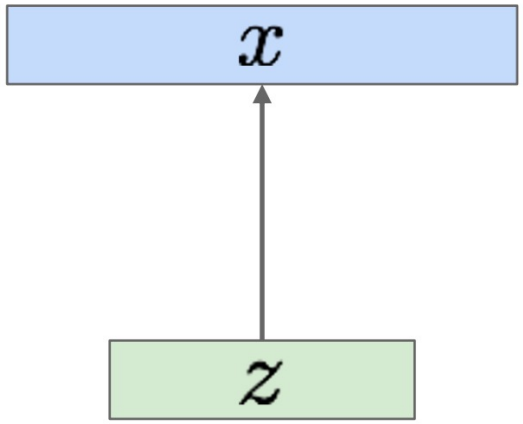
Conditional p(x|z) is complex (generates image) => represent with neural network

Sample from true conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$z^{(i)} \sim p_{\theta^*}(z)$




Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Slide credit: Fei-Fei Li

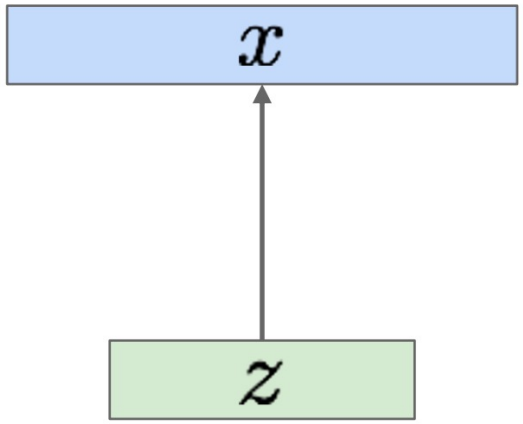
# Variational AutoEncoders

Sample from true conditional

$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$z^{(i)} \sim p_{\theta^*}(z)$




We want to estimate the true parameters $\theta^*$ of this generative model given training data x.

How to train the model?

Learn model parameters to maximize likelihood of training data

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

Q: What is the problem with this?

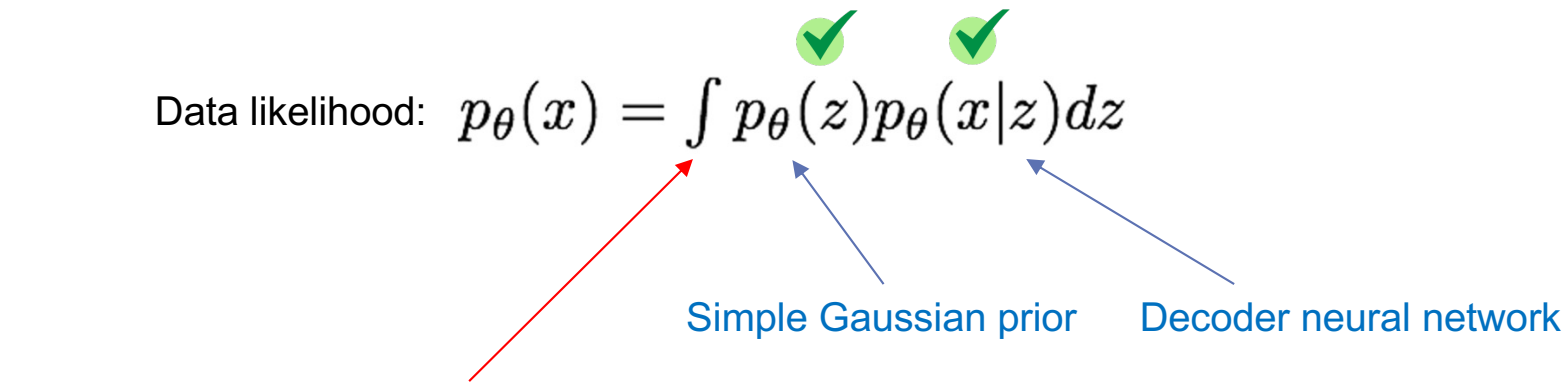
A: Intractable!

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Slide credit: Fei-Fei Li

# Variational AutoEncoders: Intractability

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






$$\log p(x) \approx \log \frac{1}{k} \sum_{i=1}^{k} p(x|z^{(i)}), \text{ where } z^{(i)} \sim p(z)$$

Monte Carlo estimation is too high variance

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Slide credit: Serena Young

# Variational AutoEncoders: Intractability

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

Posterior density: $p_\theta(z|x) = p_\theta(x|z) p_\theta(z) / p_\theta(x)$

Intractable data likelihood

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Slide credit: Fei-Fei Li

# Variational AutoEncoders: Intractability

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

Posterior density also intractable: $p_\theta(z|x) = p_\theta(x|z) p_\theta(z) / p_\theta(x)$

**Solution:** In addition to modeling $p_\theta(x|z)$, learn $q_\phi(z|x)$ that approximates the true posterior $p_\theta(z|x)$.

Will see that the approximate posterior allows us to derive a lower bound on the data likelihood that is tractable, which we can optimize.

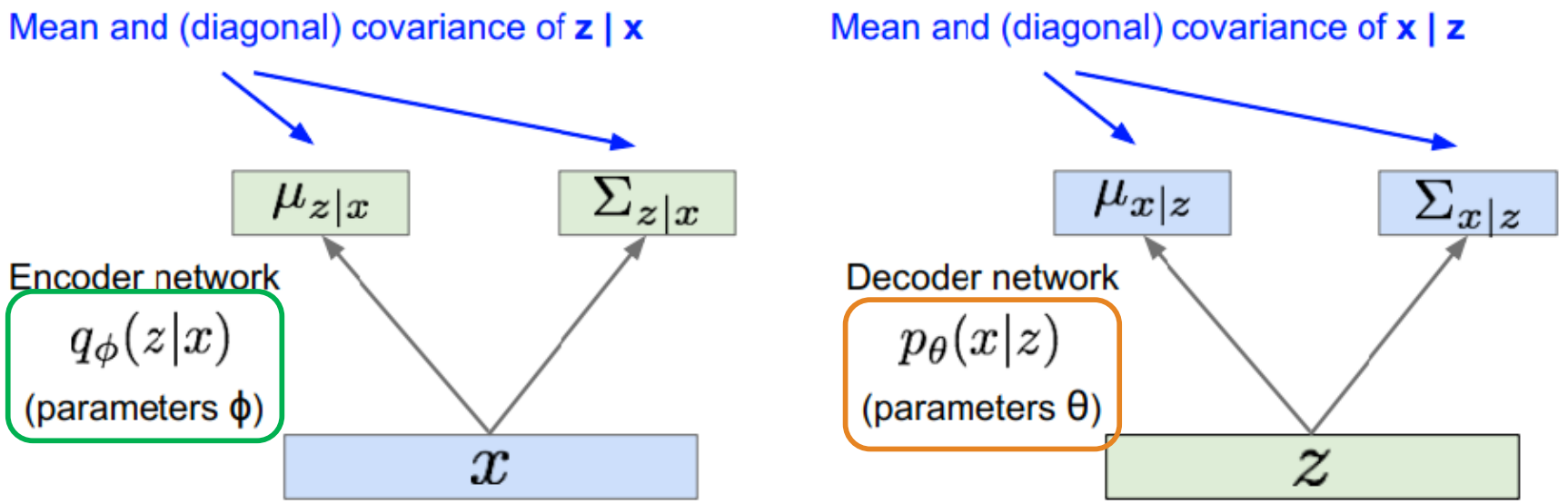
Variational inference is to approximate the unknown posterior distribution from only the observed data x

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Slide credit: Fei-Fei Li

# Variational AutoEncoders

Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic

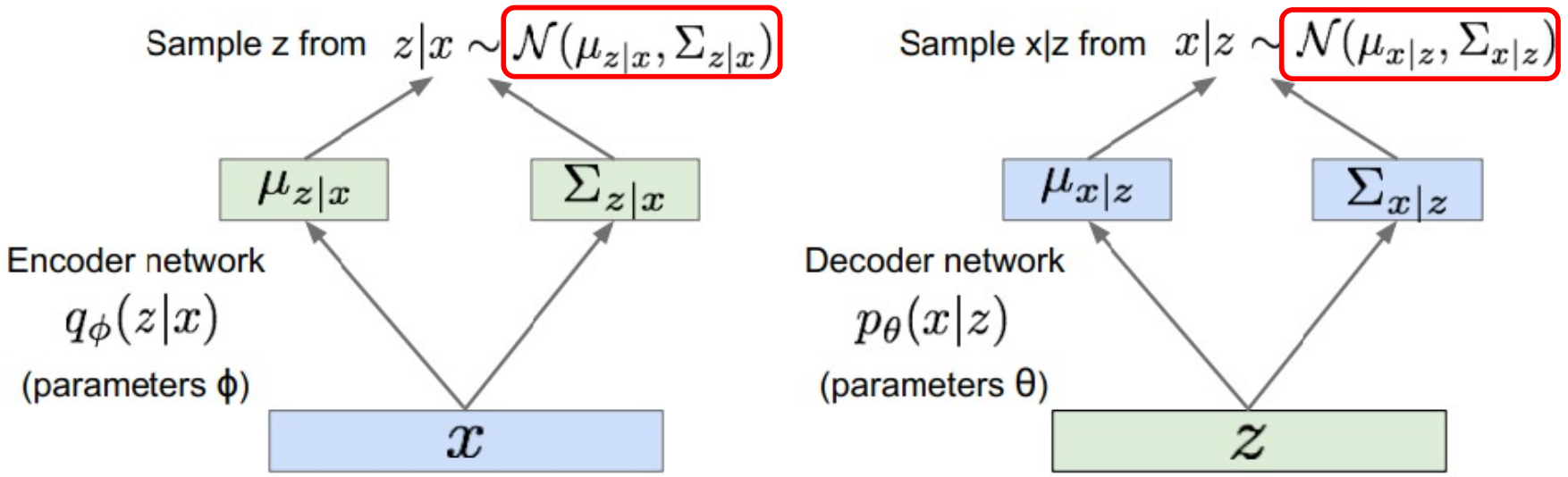



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Slide credit: Fei-Fei Li

# Variational AutoEncoders

Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic

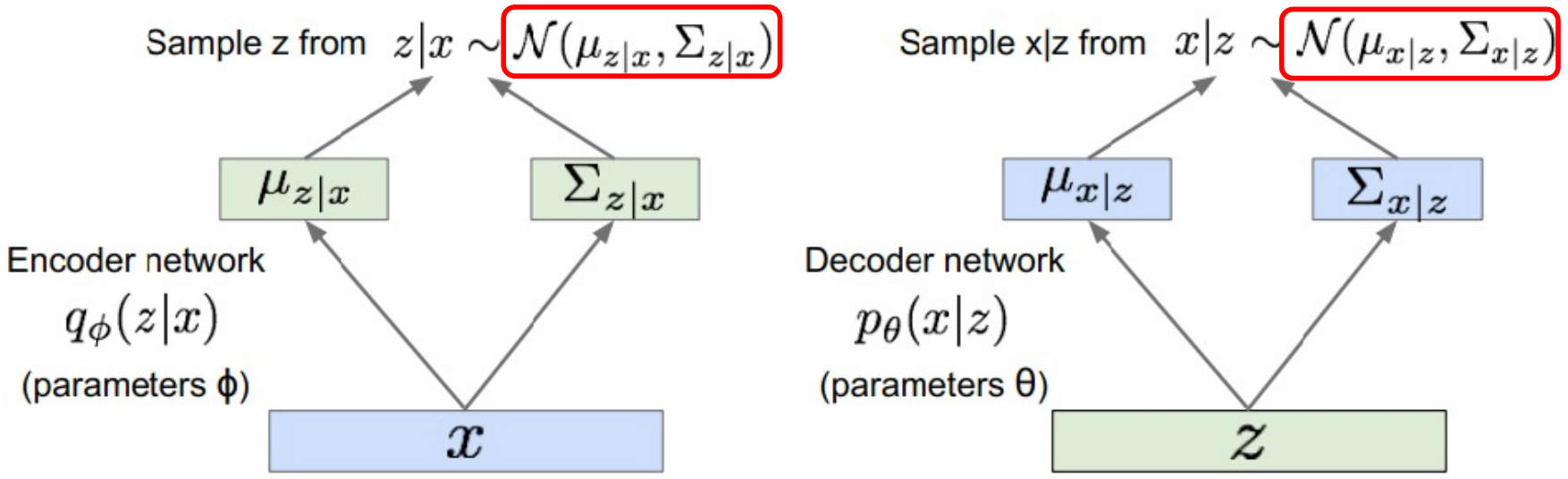



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Slide credit: Fei-Fei Li

# Variational AutoEncoders

Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic




Encoder and decoder networks also called "recognition"/"inference" and "generation" networks

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Slide credit: Fei-Fei Li

# Variational AutoEncoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

Taking expectation wrt. z (using encoder network) will come in handy later

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

The expectation wrt. z (using encoder network) let us write nice KL terms

Slide credit: Fei-Fei Li

# Variational AutoEncoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))$$

We want to maximize the Data likelihood

Decoder network gives pθ(x|z), can compute estimate of this term through sampling (need some trick to differentiate through sampling).

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

pθ(z|x) intractable (saw earlier), can't compute this KL term :( But we know KL divergence always >= 0[1]

[1] https://statproofbook.github.io/P/kl-nonneg.html

Slide credit: Fei-Fei Li

# Variational AutoEncoders

Now equipped with our encoder and decoder networks, let's work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Bayes' Rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad \text{(Multiply by constant)}$$

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad \text{(Logarithms)}$$

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{L(x(i),\,\theta,\,\phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{>=0}$$

We want to maximize the Data likelihood

Decoder: reconstruct the input data

Encoder: make approximate posterior distribution close to prior

Tractable lower bound which we can take gradient of and optimize! ($p_\theta(x|z)$ differentiable, KL term differentiable)

Slide credit: Fei-Fei Li

# Variational AutoEncoders

Putting it all together: maximizing the likelihood lower bound

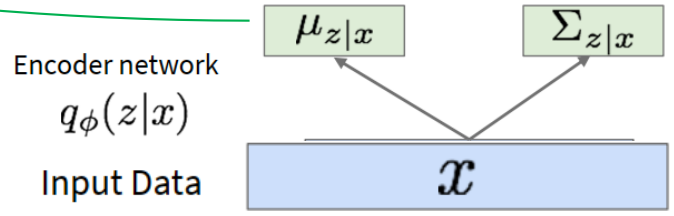
$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Let's look at computing the KL divergence between the estimated posterior and the prior given some data

Make approximate posterior distribution close to prior

$$D_{KL}(\mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) || \mathcal{N}(0, I))$$

Have analytical solution




Slide credit: Fei-Fei Li

# Variational AutoEncoders: KL Divergence

$$D_{KL}(\mathcal{N}(\mu_{z|x}, \Sigma_{z|x})||\mathcal{N}(0, I))$$

Inferred Latent Distribution

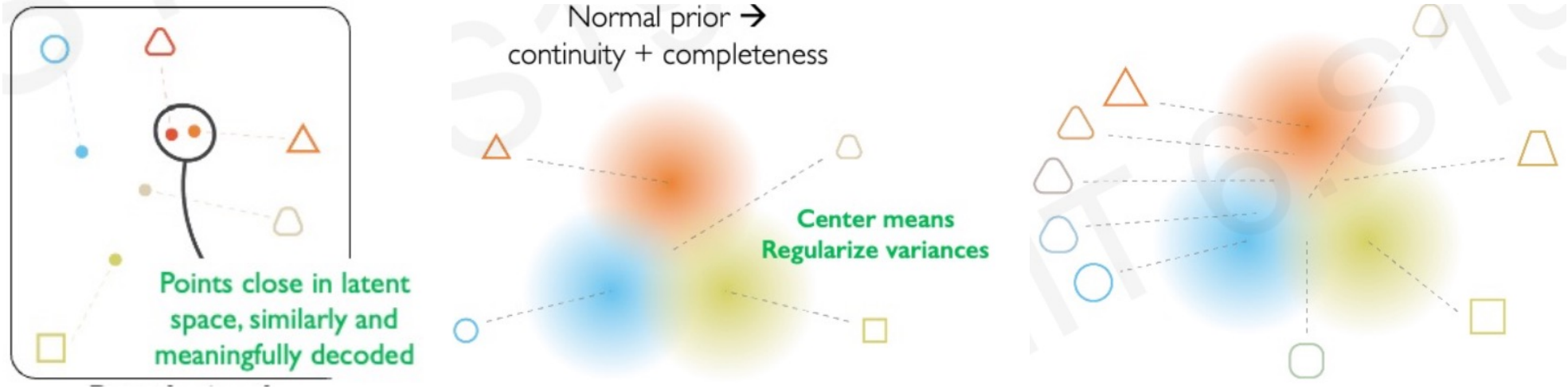
Fixed Prior on Latent Distribution

- Encourage encodings to distribute them evenly around the center of the latent space
- Penalize the network when it tries to "cheat" by clustering points in specific regions (i.e. memorizing data)

# Variational AutoEncoders: KL Divergence

What properties do we want to achieve from KL Divergence?

1. Continuity: Points that are close in latent space → similar content after decoding
2. Completeness: sampling from latent space → "meaningful" content after decoding



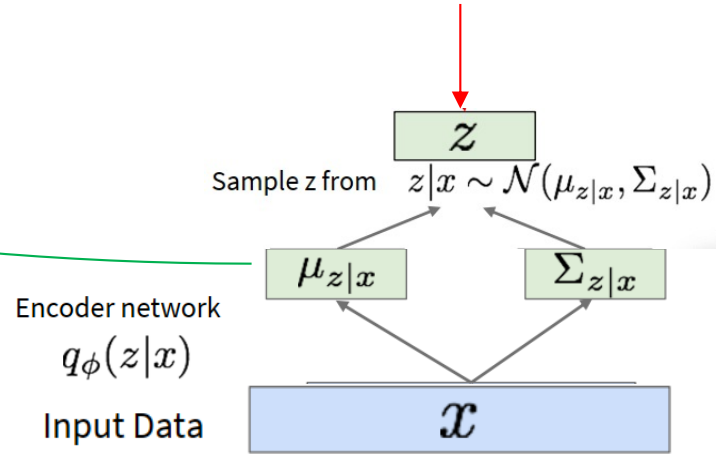
# Variational AutoEncoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Make approximate posterior distribution close to prior

Not part of the computation graph!




Slide credit: Fei-Fei Li

# Variational AutoEncoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

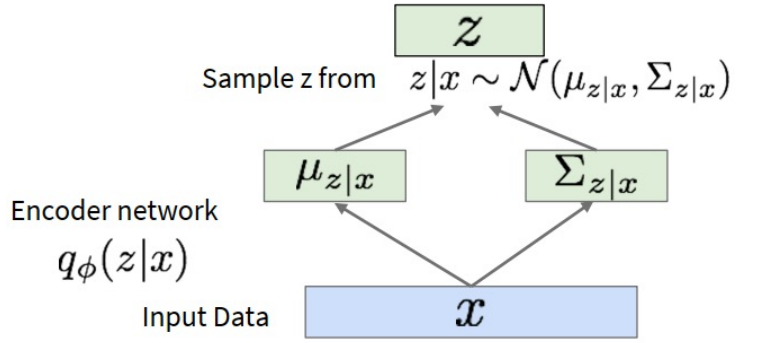
Reparameterization trick to make sampling differentiable:

$$\text{Sample } \epsilon \sim \mathcal{N}(0, I)$$

$$z = \mu_{z|x} + \epsilon \sigma_{z|x}$$

Part of computation graph

Input to the graph

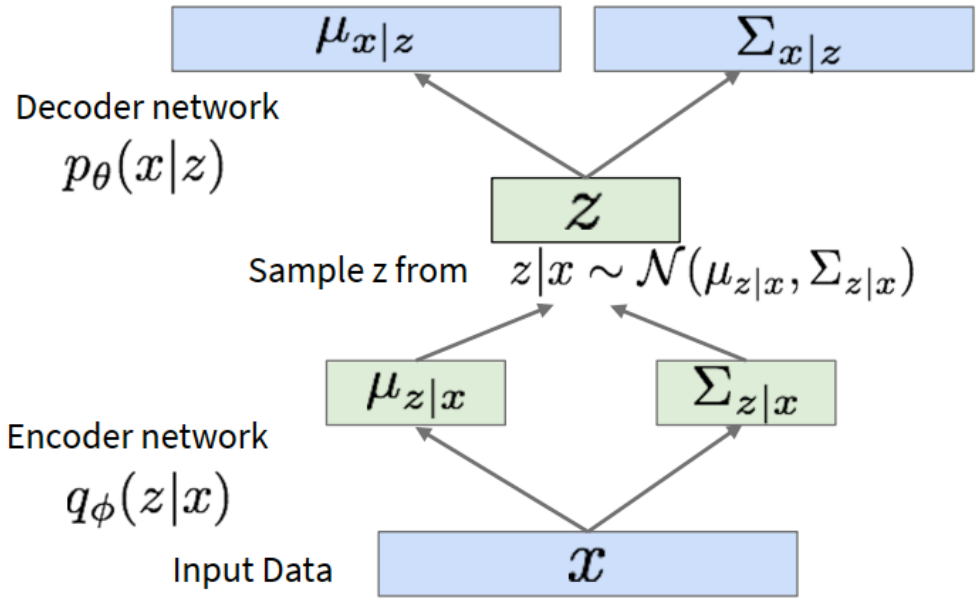



Slide credit: Fei-Fei Li

# Variational AutoEncoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$



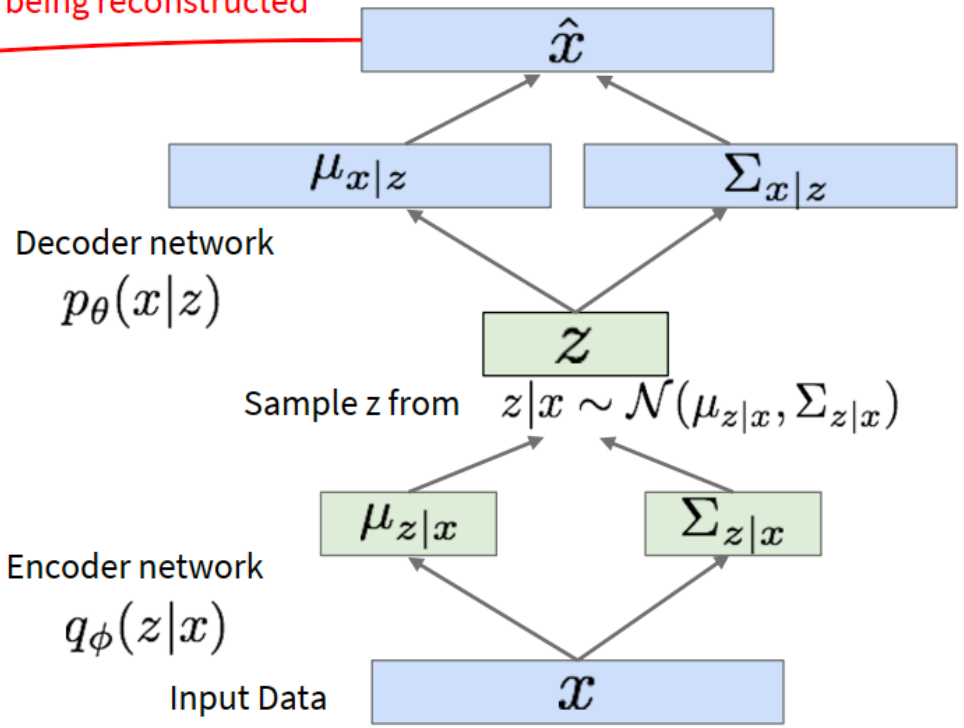

Slide credit: Fei-Fei Li

# Variational AutoEncoders

Putting it all together: maximizing the likelihood lower bound

Maximize likelihood of original input being reconstructed

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$



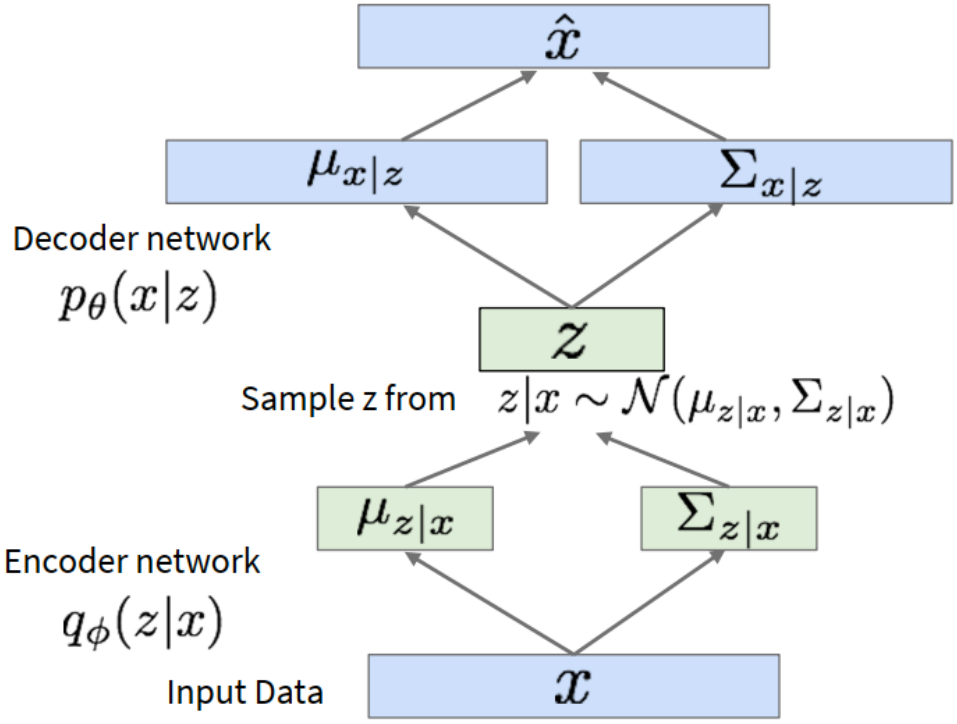

Slide credit: Fei-Fei Li

# Variational AutoEncoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

For every minibatch of input data: compute this forward pass, and then backprop!




Slide credit: Fei-Fei Li

# Variational AutoEncoders: Generating Data!

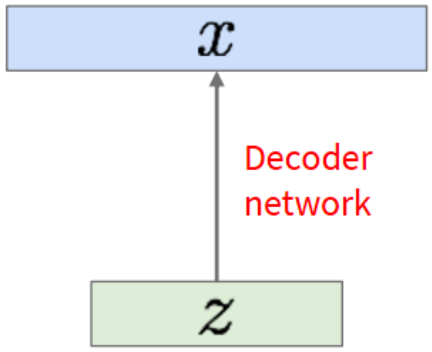
Our assumption about data generation process

Sample from true conditional

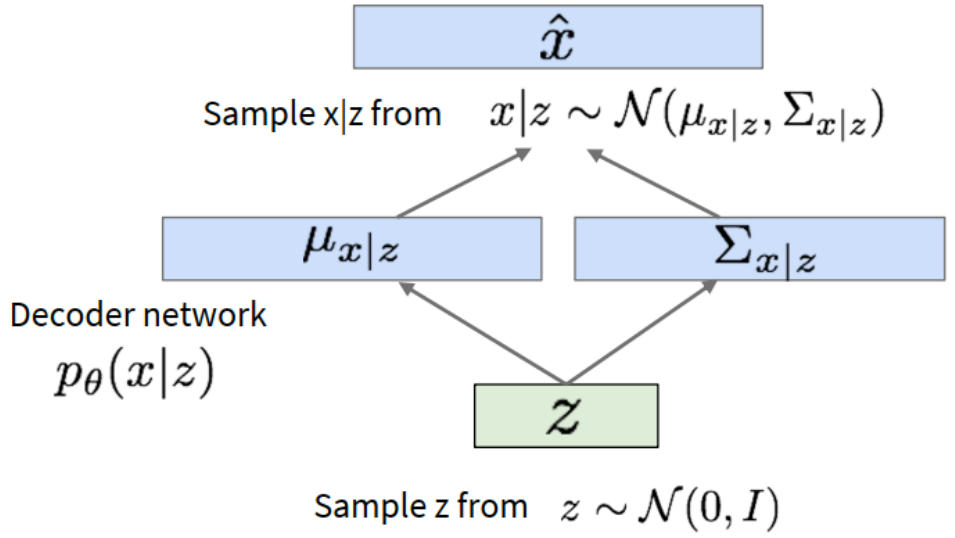
$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

$z^{(i)} \sim p_{\theta^*}(z)$




Now given a trained VAE:
use decoder network & sample z from prior!

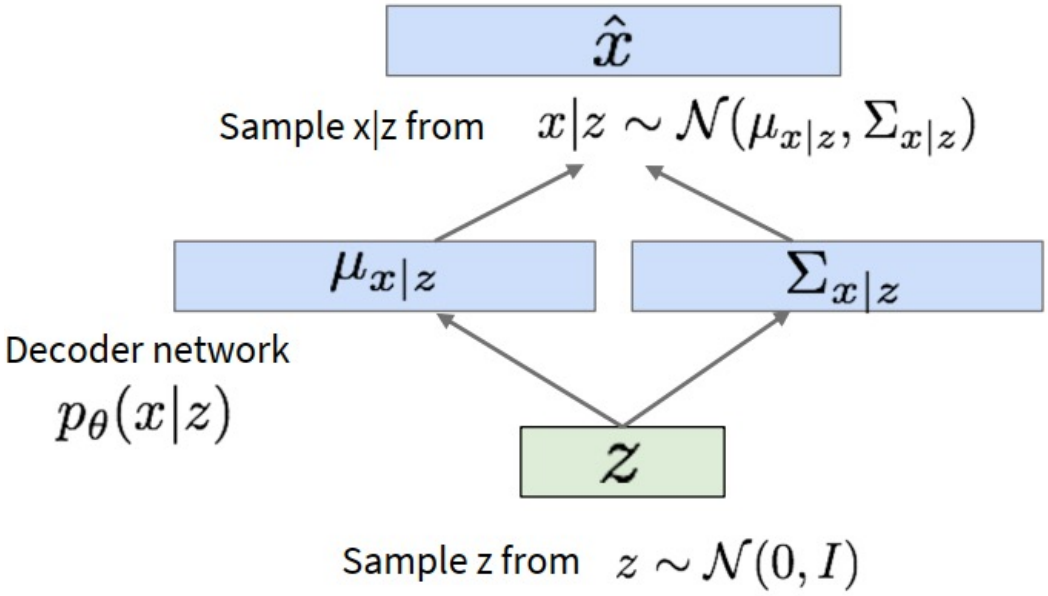


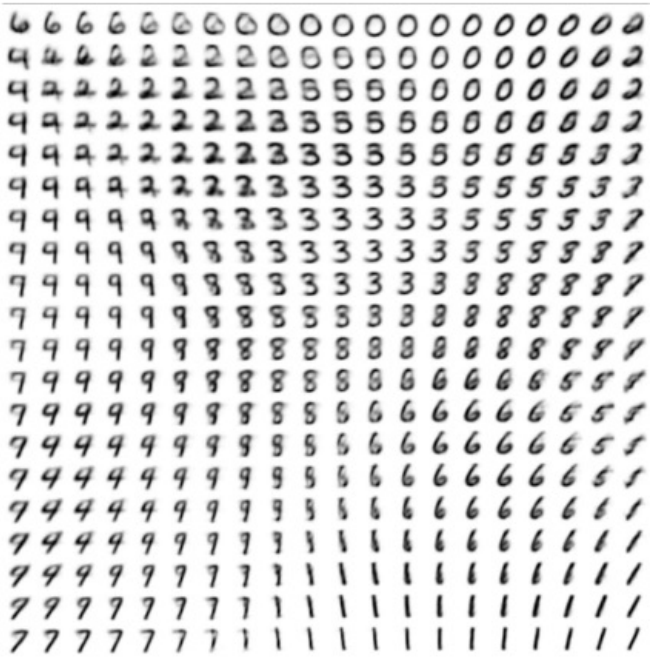

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Slide credit: Fei-Fei Li

# Variational AutoEncoders: Generating Data!

Use decoder network. Now sample z from prior!



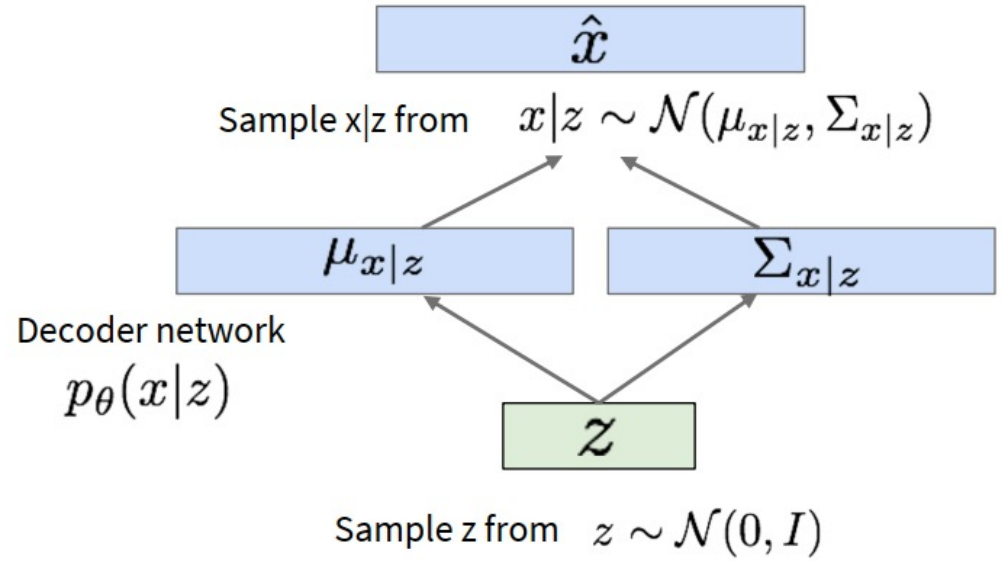



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Slide credit: Fei-Fei Li

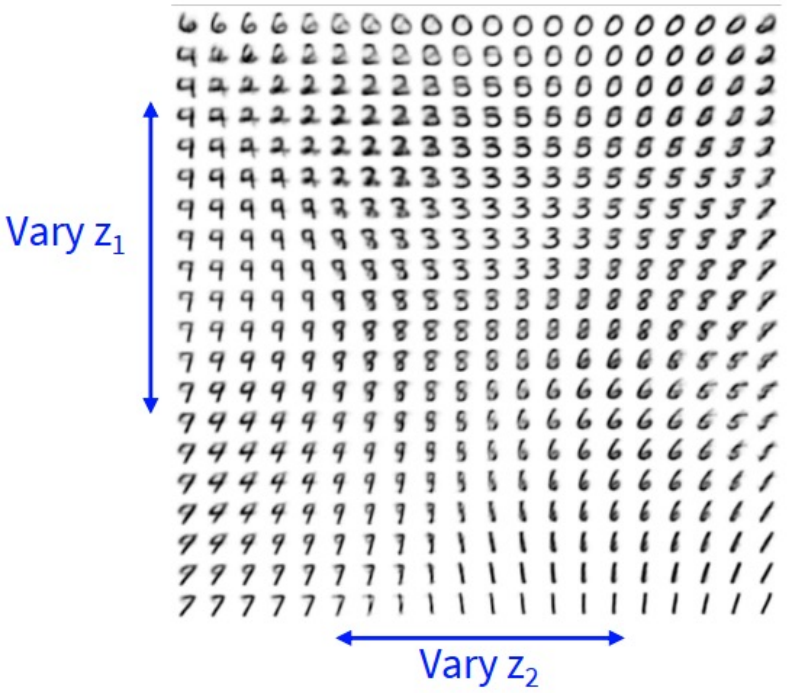
# Variational AutoEncoders: Generating Data!

Use decoder network. Now sample z from prior!




Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Data manifold for 2-d z



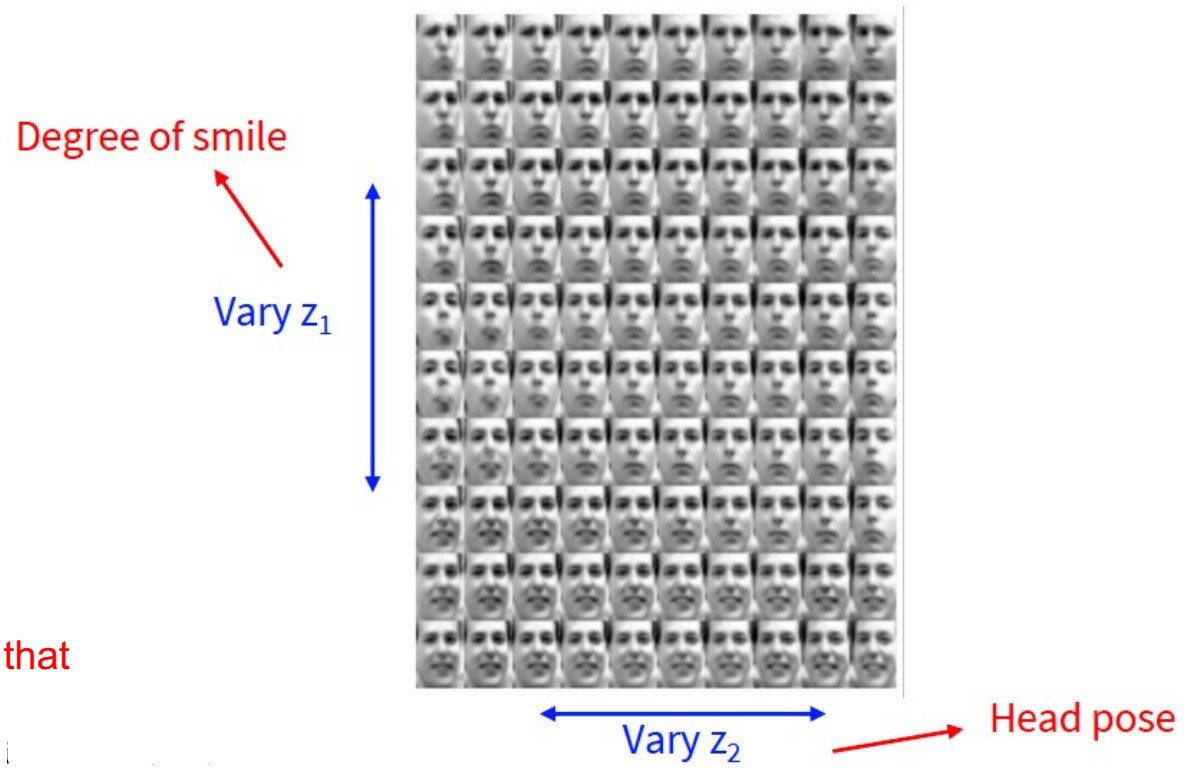

Slide credit: Fei-Fei Li

# Variational AutoEncoders: Generating Data!

Diagonal prior on z
=> independent
latent variables

Different dimensions
of z encode
interpretable factors
of variation

Also good feature representation that
can be computed using $q_\phi(z|x)$!



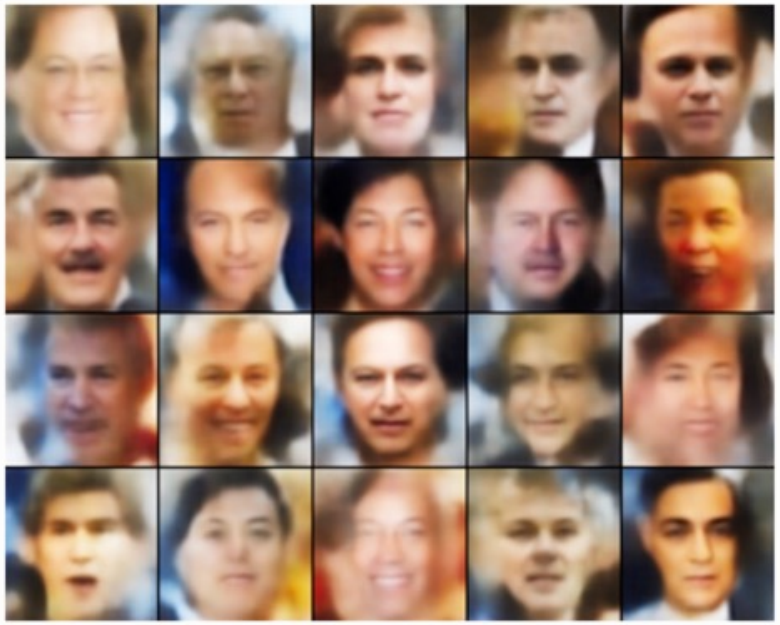

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Slide credit: Fei-Fei Li

# Variational AutoEncoders: Generating Data!



32x32 CIFAR-10



Labeled Faces in the Wild

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

Slide credit: Fei-Fei Li

# Music VAE

https://magenta.tensorflow.org/music-vae

# Variational AutoEncoders: Generating Data!

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
- Interpretable latent space.
- 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), Categorical Distributions.
- Learning disentangled representations.

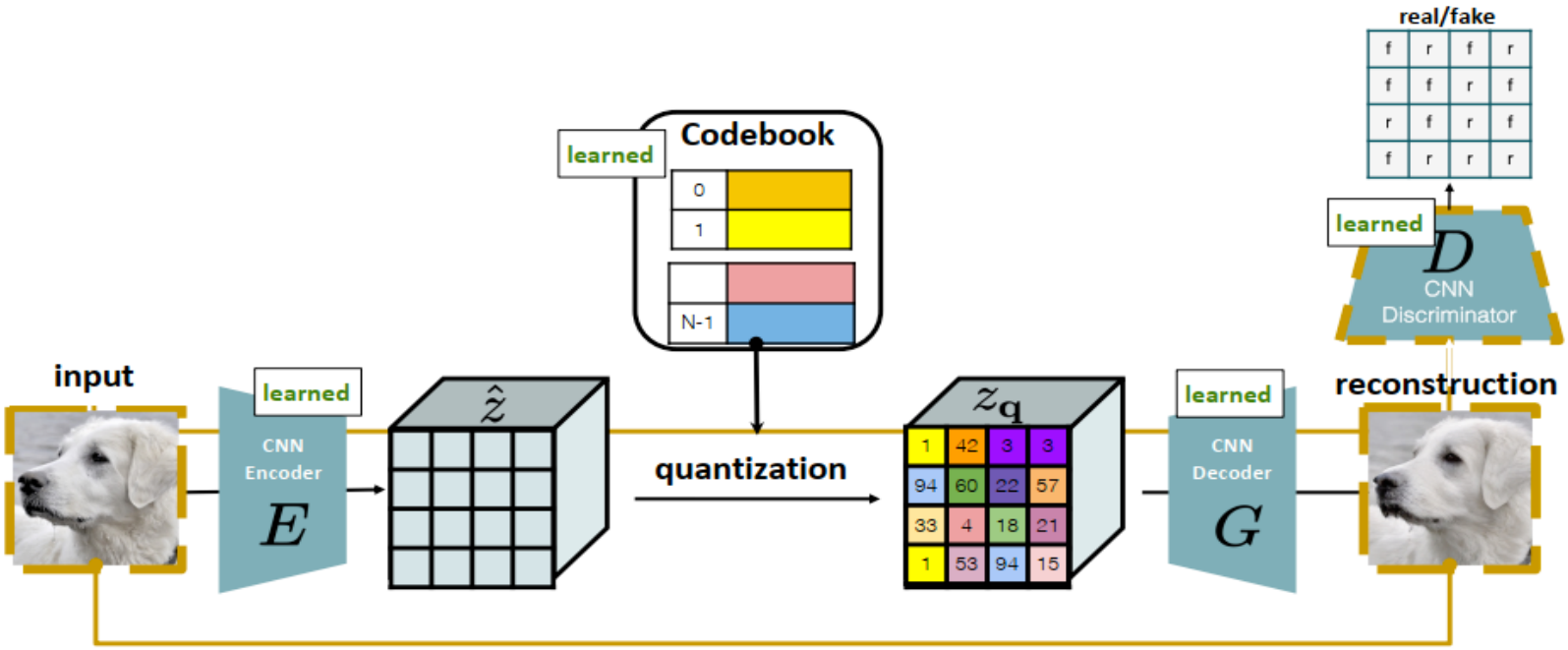
Slide credit: Fei-Fei Li

# GAN Example for Digits on Numpy

https://nbviewer.org/url/www.cs.toronto.edu/~rgrosse/courses/csc321_2018/tutorials/tut9_GAN.ipynb#



# From VQ-VAE to VQGAN




$$\mathcal{L}_{\text{total}} = \mathcal{L}_{rec} + \mathcal{L}_{VQ} + \lambda \mathcal{L}_{GAN} \quad \text{where} \quad \lambda = \frac{\nabla_{G_L}[\mathcal{L}_{rec}]}{\nabla_{G_L}[\mathcal{L}_{GAN}] + \delta}$$

Slide credit: Robin Rombach

# From VQ-VAE to VQGAN




Slide credit: Robin Rombach

# From VQ-VAE to VQGAN



Slide credit: Robin Rombach