Lecture 13: Generative Models (part 1)

Administrative

- Today 5/14: Assignment 3 released
- Tomorrow 5/15: Milestone 1 Check-In due

Last Time: Self-Supervised Learning

Dataset
(no labels)

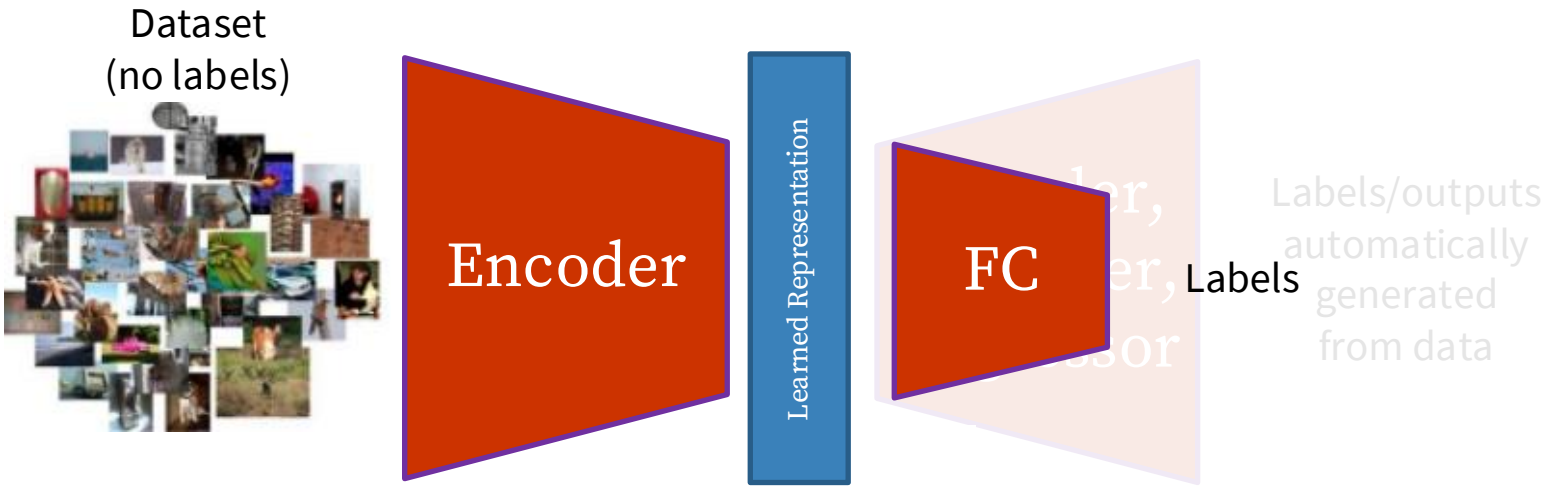
Encoder

Learned Representation

Decoder,
Classifier,
Regressor

Labels/outputs
automatically
generated
from data

Last Time: Self-Supervised Learning

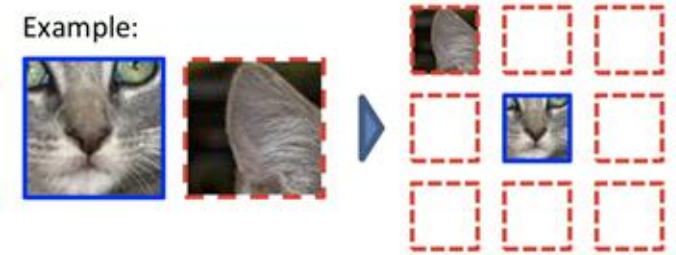


Last Time: Self-Supervised Learning

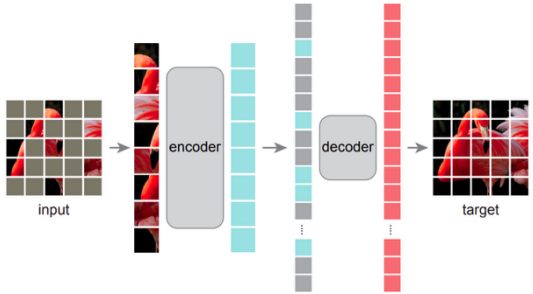
Pretext tasks from image transformations

- Rotation, inpainting, rearrangement, coloring
- Reconstruction-based learning (MAE)

Rotation



Rearrangement



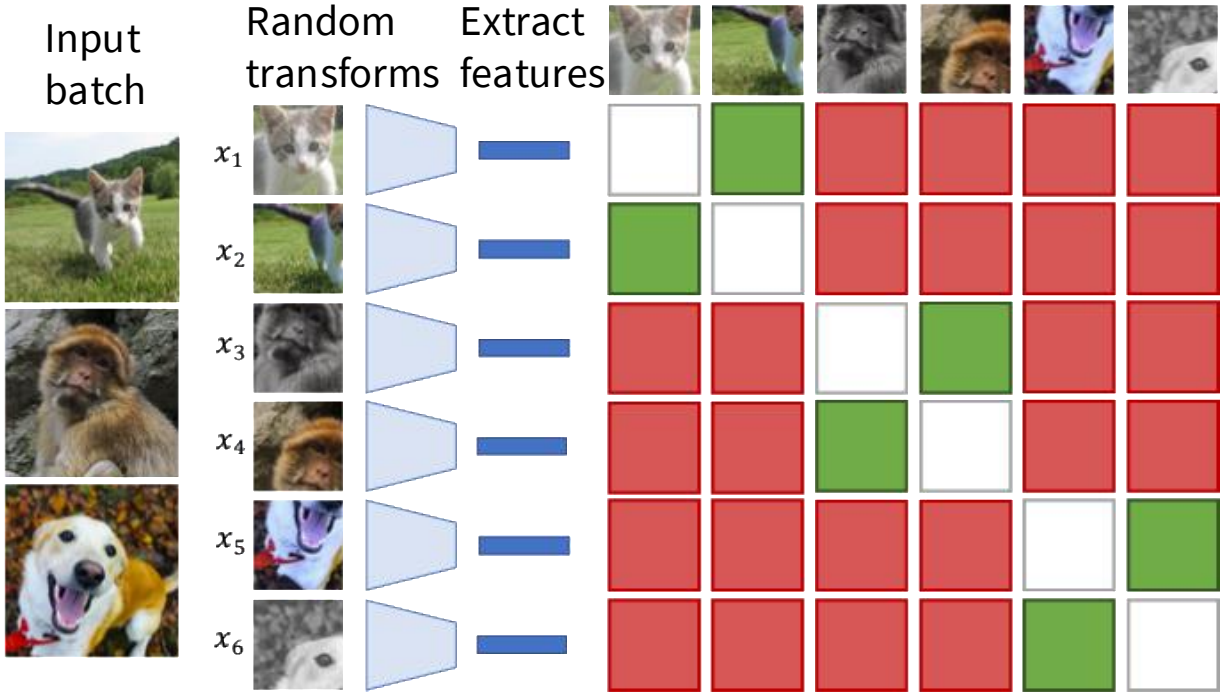
Reconstruction

Last Time: Self-Supervised Learning

Contrastive representation learning

- Intuition and formulation
- Instance contrastive learning: SimCLR and MOCO
- Sequence contrastive learning: CPC
- Self-Distillation Without Labels, DINO

Last Time: Contrastive Learning (SimCLR)

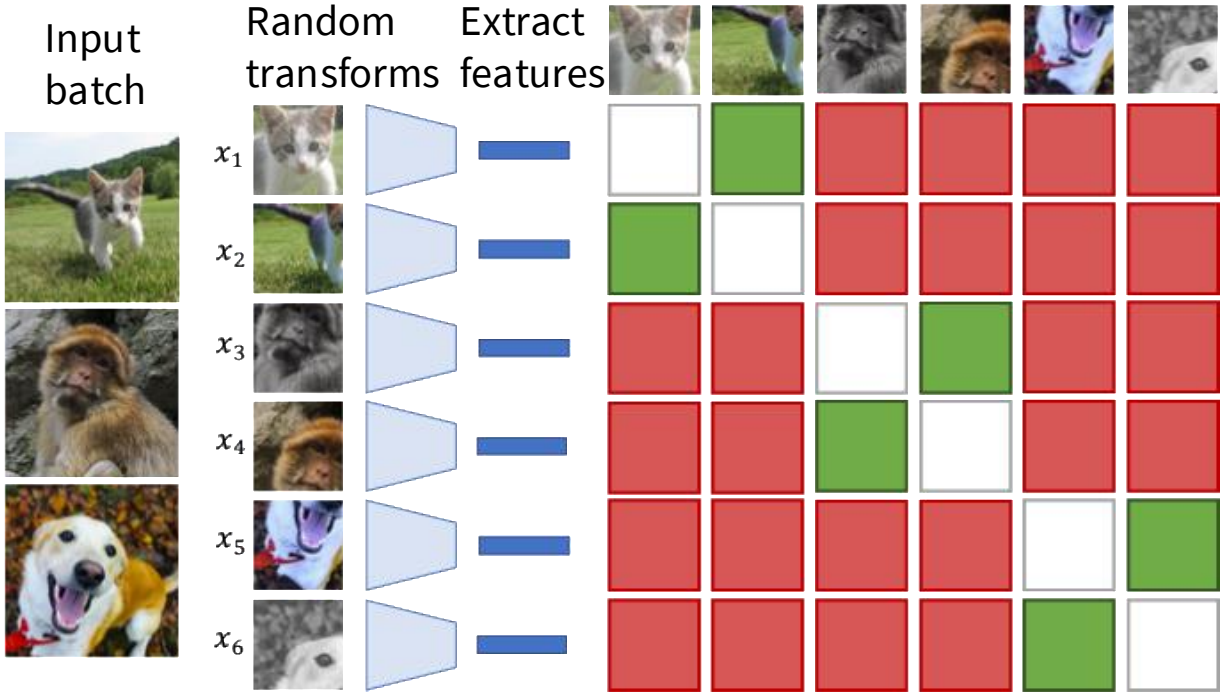


Corresponding pairs should have similar features

Other pairs should have dissimilar features

Chen et al, "A simple framework for contrastive learning of visual representations", ICML 2020

Last Time: Contrastive Learning (SimCLR)



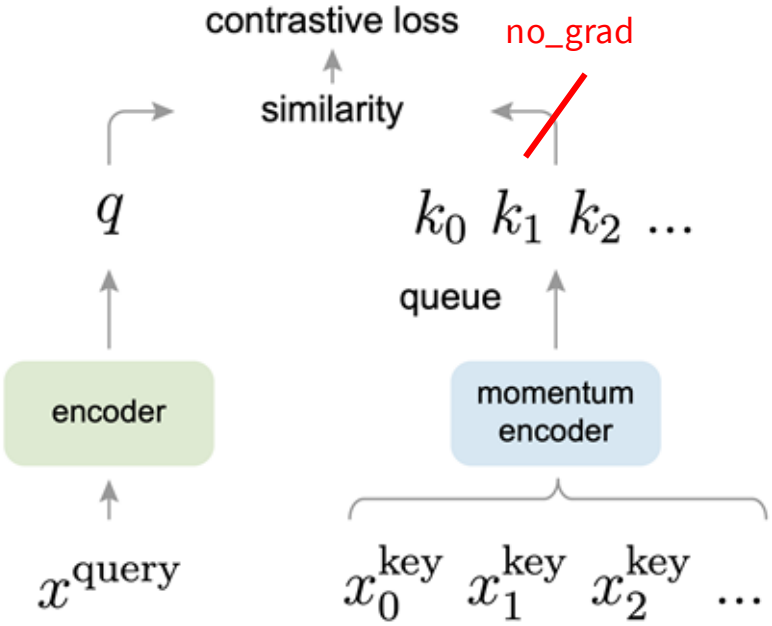
Corresponding pairs should have similar features

Other pairs should have dissimilar features

Problem: Need large batch size with lots of negatives

Chen et al, "A simple framework for contrastive learning of visual representations", ICML 2020

Contrastive Learning: MoCo



Key differences to SimCLR:

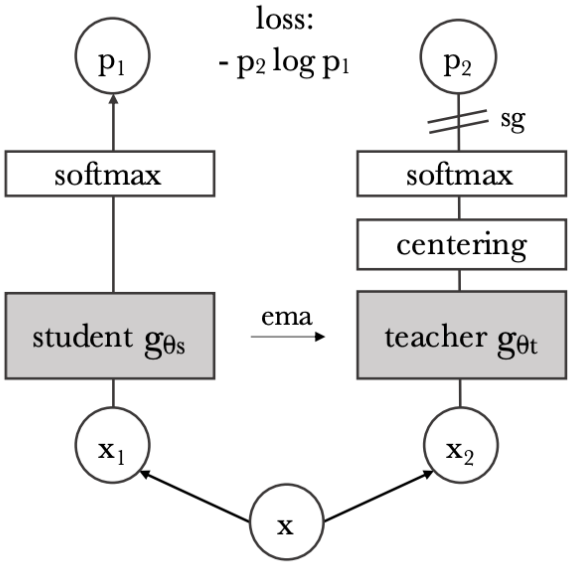
- Keep a running queue of keys (negative samples).
- Compute gradients and update the encoder only through the queries.
- Decouple min-batch size with the number of keys: can support a large number of negative samples.
- The key encoder is slowly progressing through the momentum update rules:

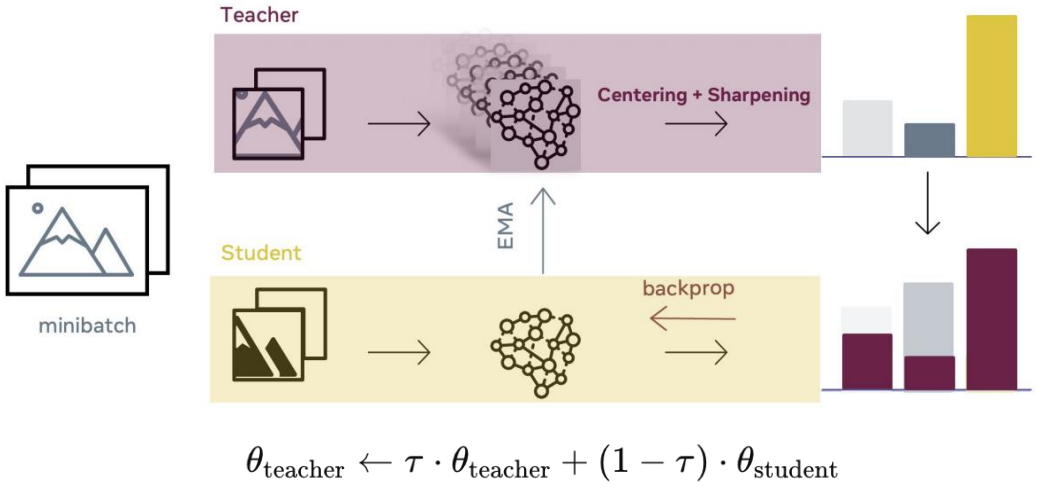
$$\theta_{\text{k}} \leftarrow m\theta_{\text{k}} + (1-m)\theta_{\text{q}}$$

He et al, "Momentum Contrast for Unsupervised Visual Representation Learning", CVPR 2020

Self-Supervised Learning: DINO

Similar in spirit to MoCo, but matches features using KL divergence instead of dot product, and uses Vision Transformers instead of ResNets



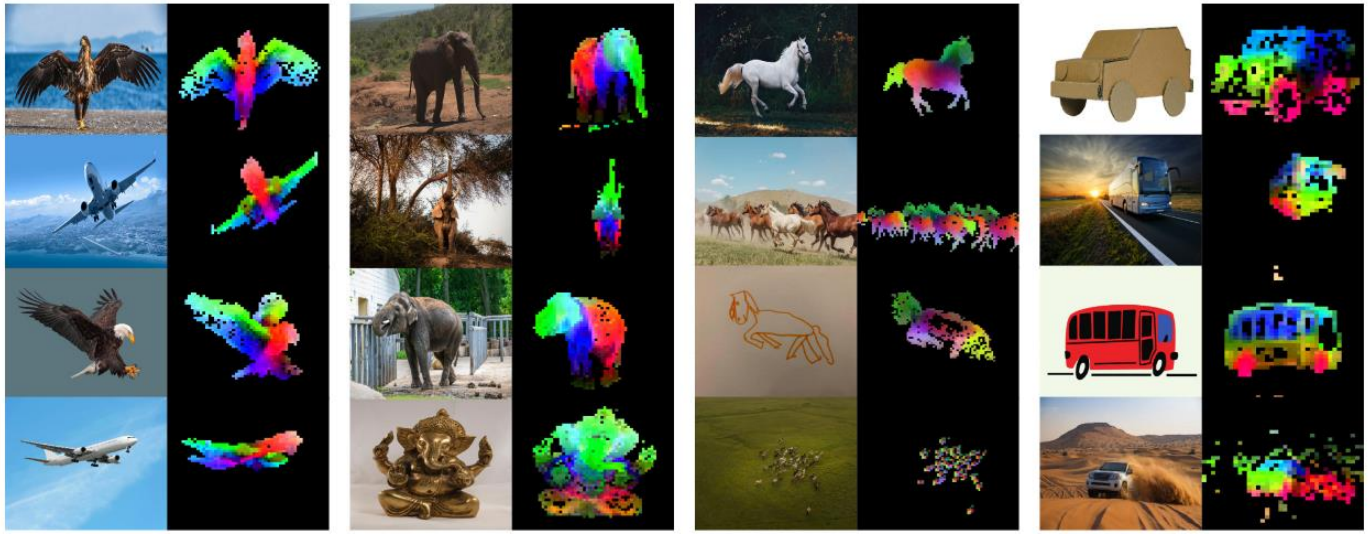


$$\theta_{\text{teacher}} \leftarrow \tau \cdot \theta_{\text{teacher}} + (1 - \tau) \cdot \theta_{\text{student}}$$

Caron et al. 2021 Emerging Properties in Self-Supervised Vision Transformers

Self-Supervised Learning: DINOv2

Scales up training data from 1M ImageNet images to 142M images
Very strong image features, commonly used in practice



PCA feature visualization

Oquab et al, “DINOv2: Learning Robust Visual Features without Supervision”, arXiv 2023; Darcet et al, “Vision Transformers Need Registers”, arXiv 2023

Today:
Generative Models (part 1)

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.

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

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

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

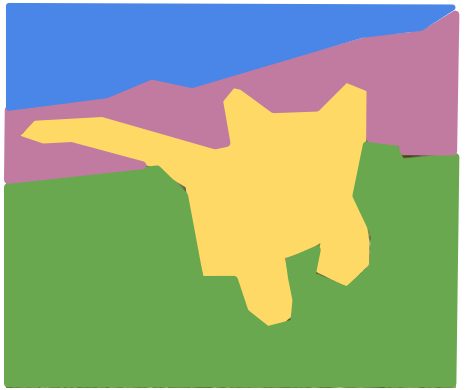
Supervised vs Unsupervised Learning

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

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.

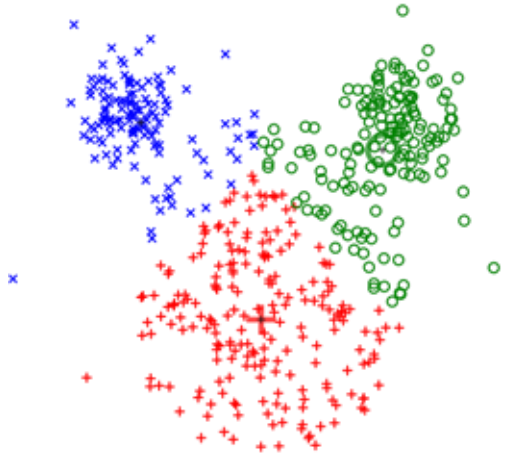
Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn hidden structure in data

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

Supervised vs Unsupervised Learning



K-means clustering

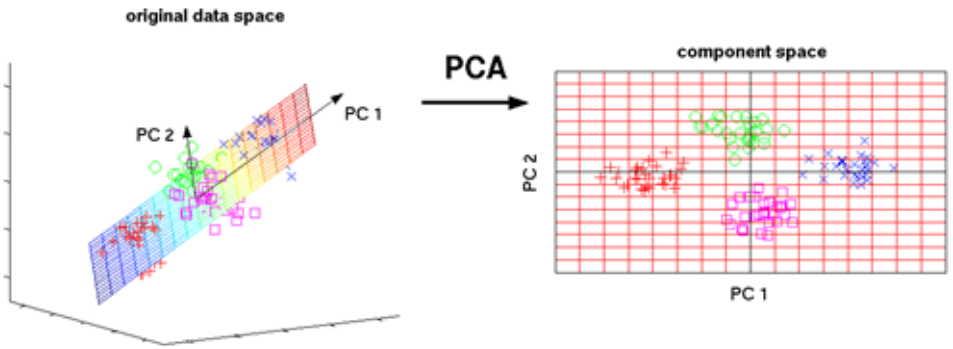
Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn hidden structure in data

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

Supervised vs Unsupervised Learning



Principal Component Analysis
(Dimensionality reduction)

Unsupervised Learning

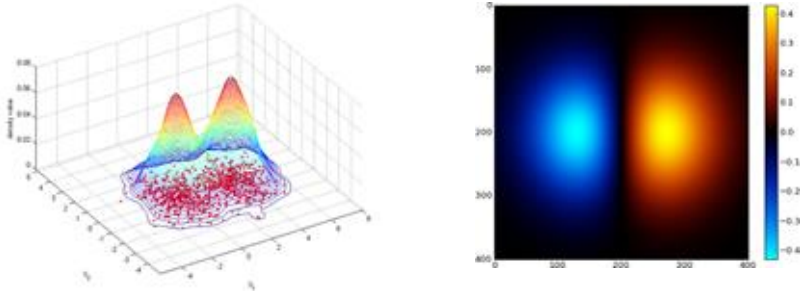
Data: x
Just data, no labels!

Goal: Learn hidden structure in data

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

Supervised vs Unsupervised Learning

1-d density estimation



2-d density estimation

Modeling p(x)

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn hidden structure in data

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

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

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

Generative vs Discriminative Models

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

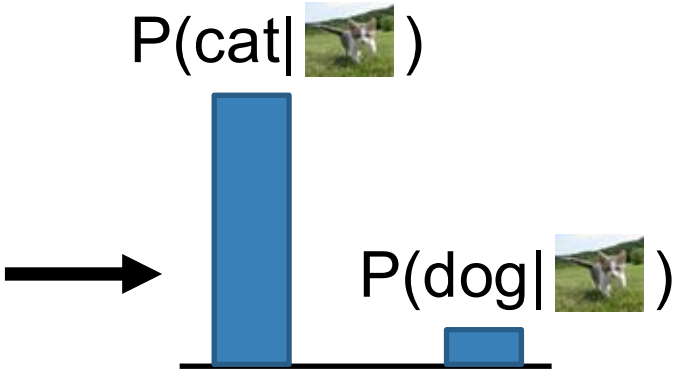
Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)

Data: x

Label: y

Cat



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

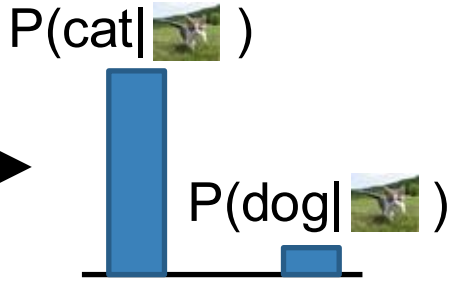
Different values of x **compete** for density

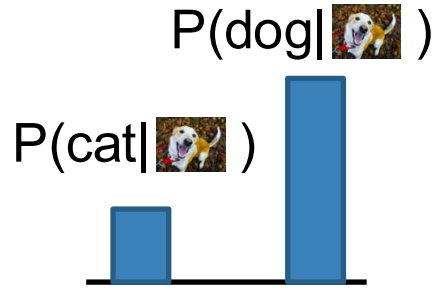
Generative vs Discriminative Models

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

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)





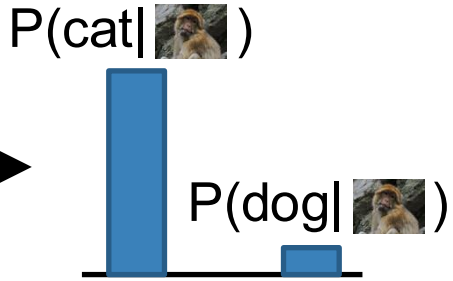
Possible **labels** for each image compete for probability.
No competition between **images**

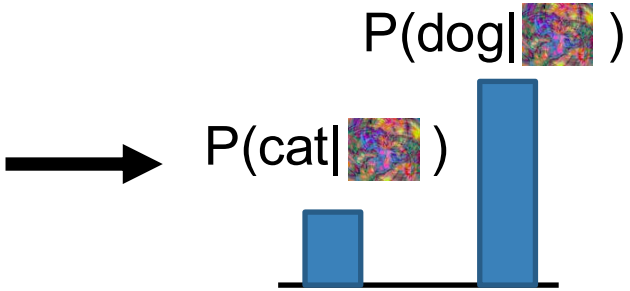
Generative vs Discriminative Models

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

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)





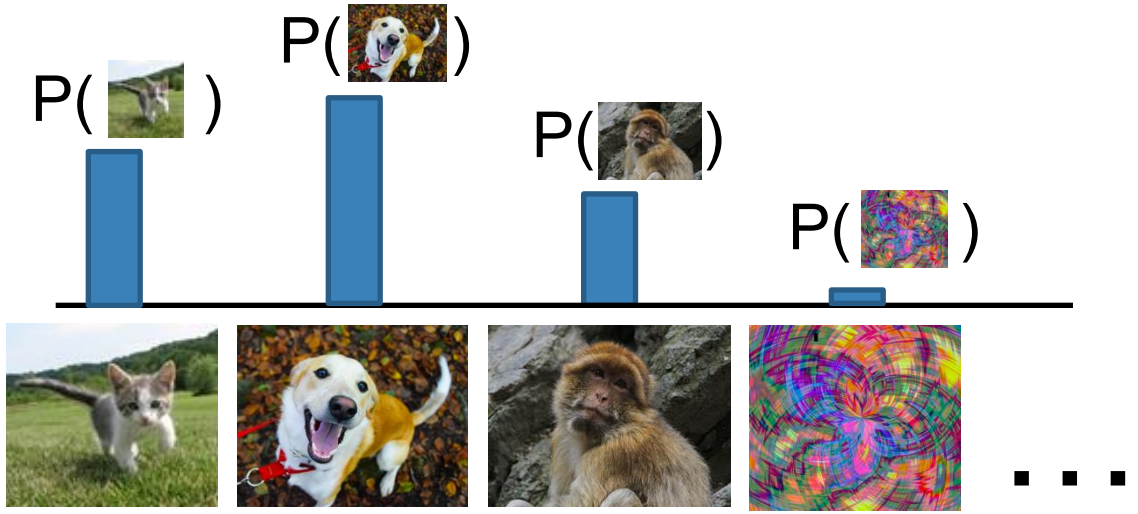
No way to handle unreasonable inputs; must give a label distribution for all possible inputs

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)



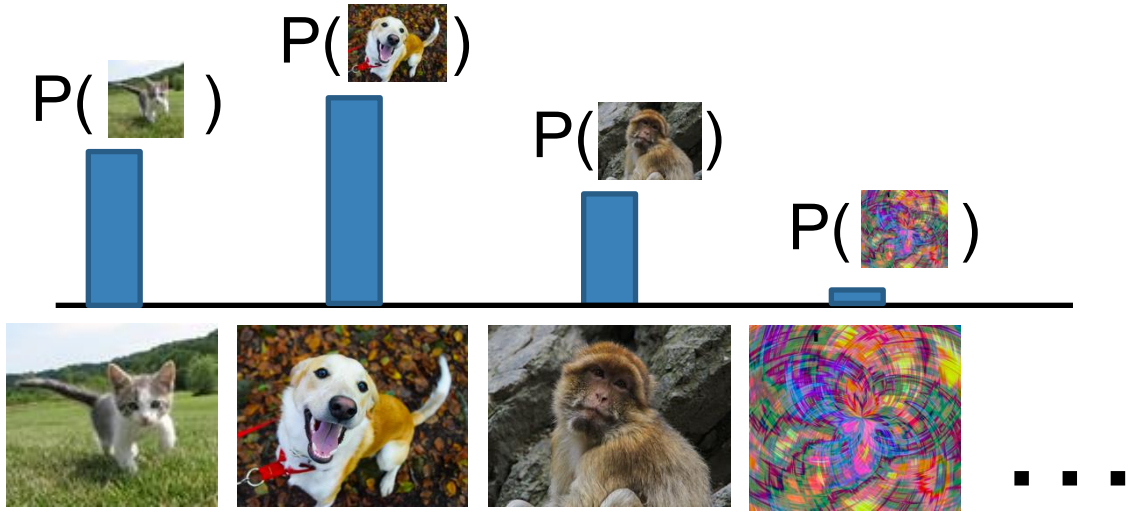
All possible **images** compete for probability mass

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)



All possible **images** compete for probability mass

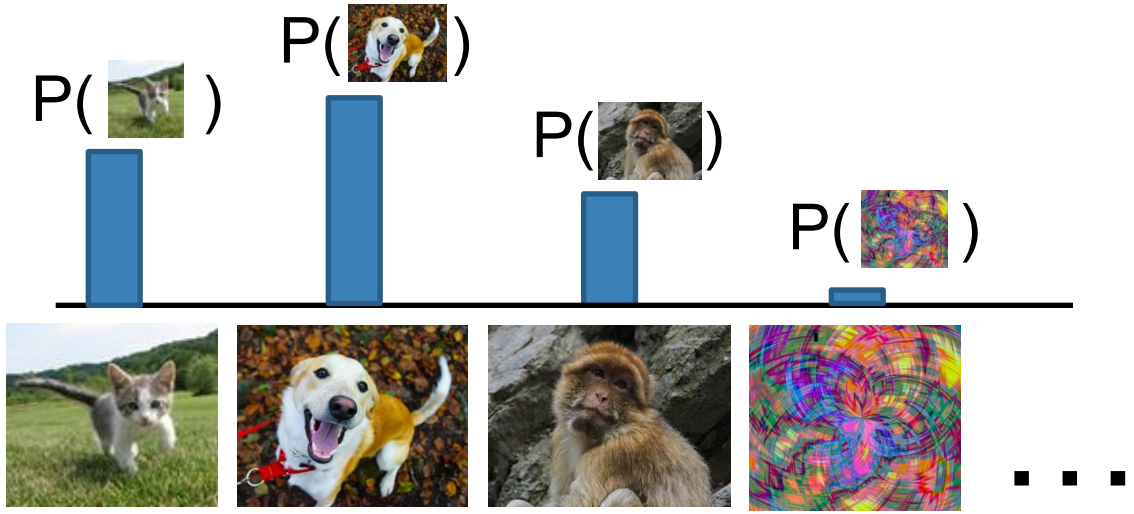
Requires deep understanding: Is a dog more likely to sit or stand? Is a 3-legged dog more likely than a 3-armed monkey?

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)



All possible **images** compete for probability mass

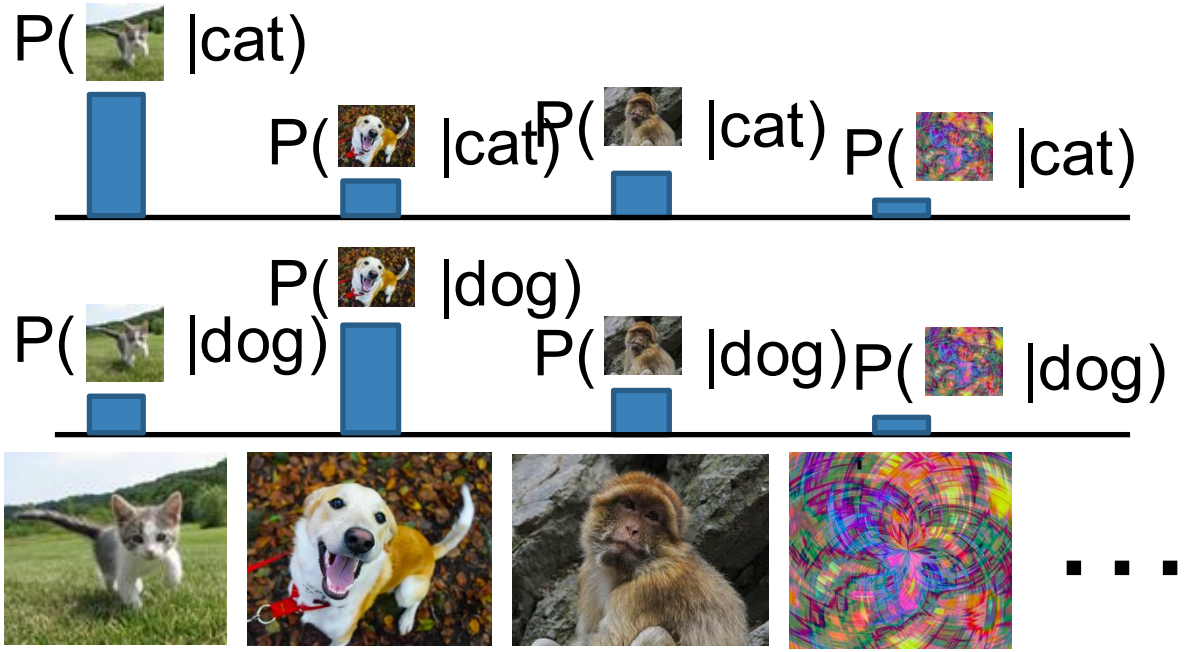
Model can “reject” unreasonable inputs by giving them small probability mass

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)



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

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**:

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

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

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)

Conditional Generative Model: Learn p(x|y)

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)

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)

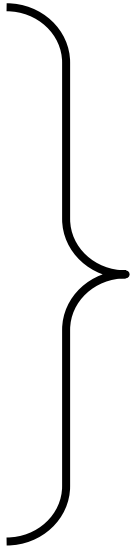
→ Assign labels while rejecting outliers
Sample to generate data from labels

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)



”Generative models” means either of these; conditional generative models are most common in practice

Why Generative Models?

Modeling ambiguity: If there are many possible outputs x for an input y, we want to model P(x | y)

Language Modeling: Produce output text x from input text y

Write me a short rhyming poem about generative models

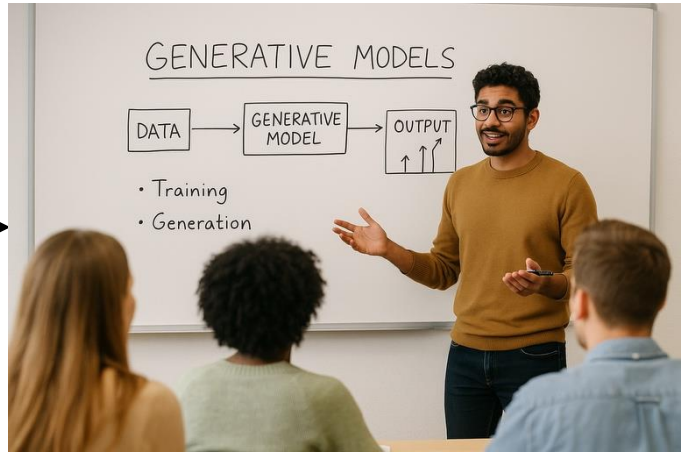
*They sample from a learned **P**,*
A distribution—structured, free.
Each token comes conditionally,
On all the ones that used to be.

Why Generative Models?

Modeling ambiguity: If there are many possible outputs x for an input y, we want to model P(x | y)

Text to Image: Produce output image x from input text y

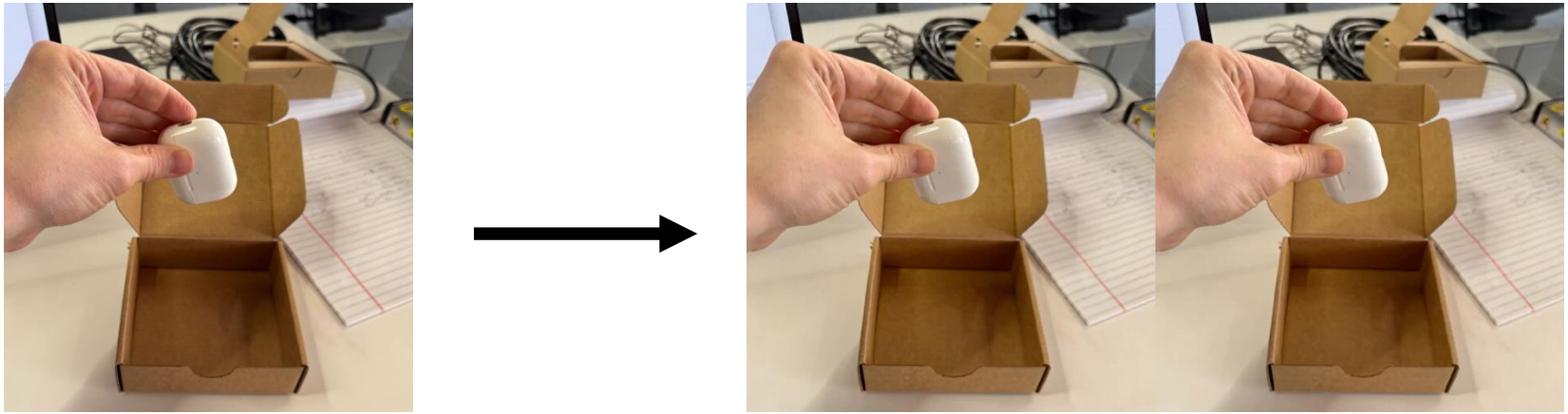
Make me an image showing a person teaching a class on generative models in front of a whiteboard



Why Generative Models?

Modeling ambiguity: If there are many possible outputs x for an input y, we want to model P(x | y)

Image to Video: What happens next?



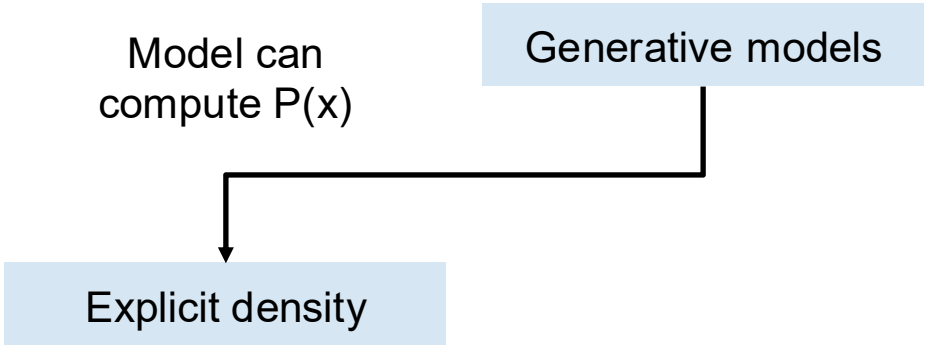
Taxonomy of Generative Models

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

Generative models

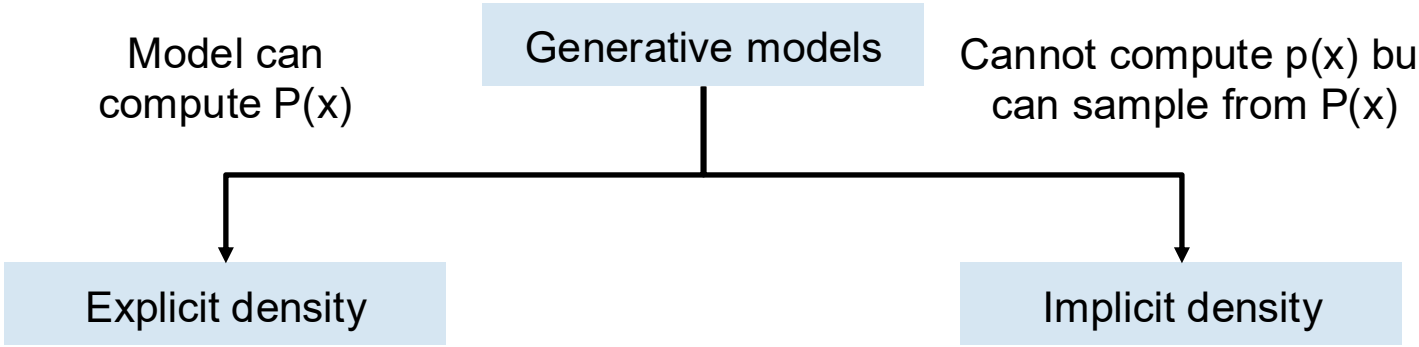
Taxonomy of Generative Models

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



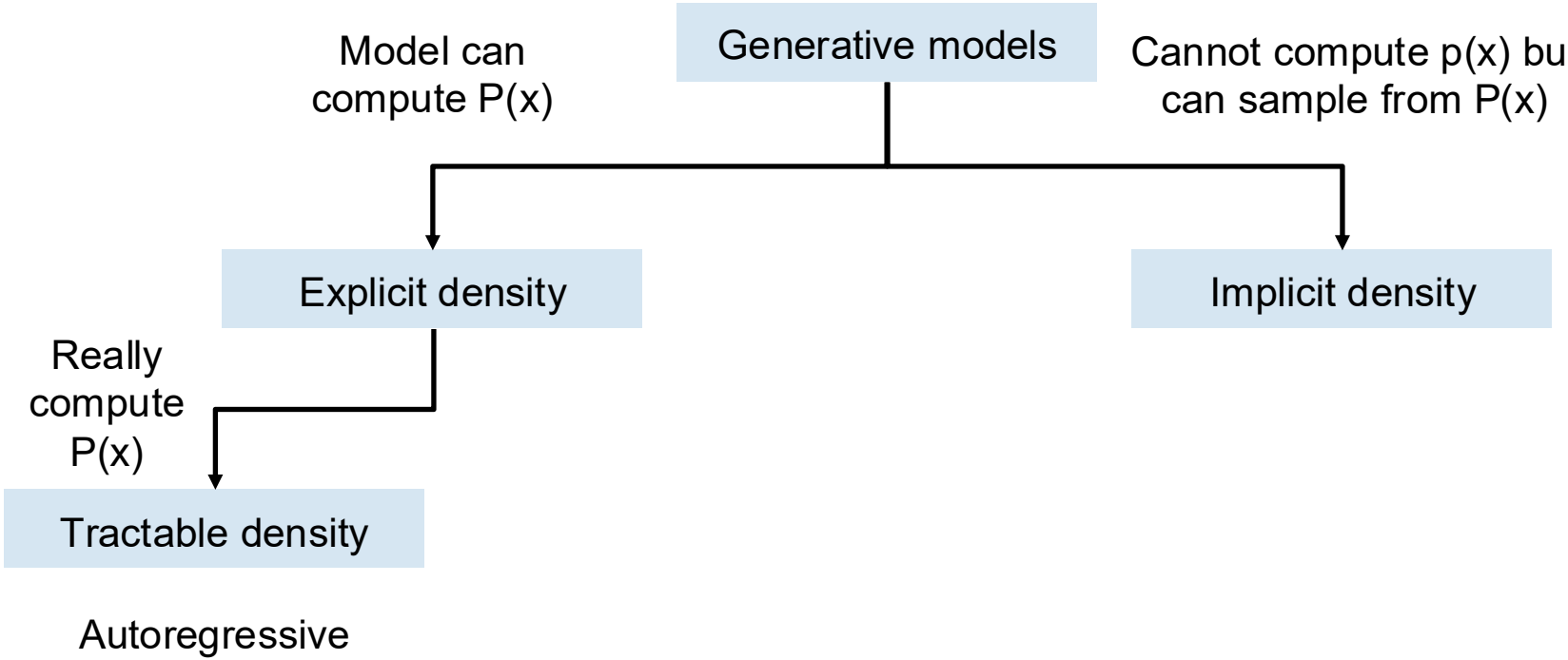
Taxonomy of Generative Models

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



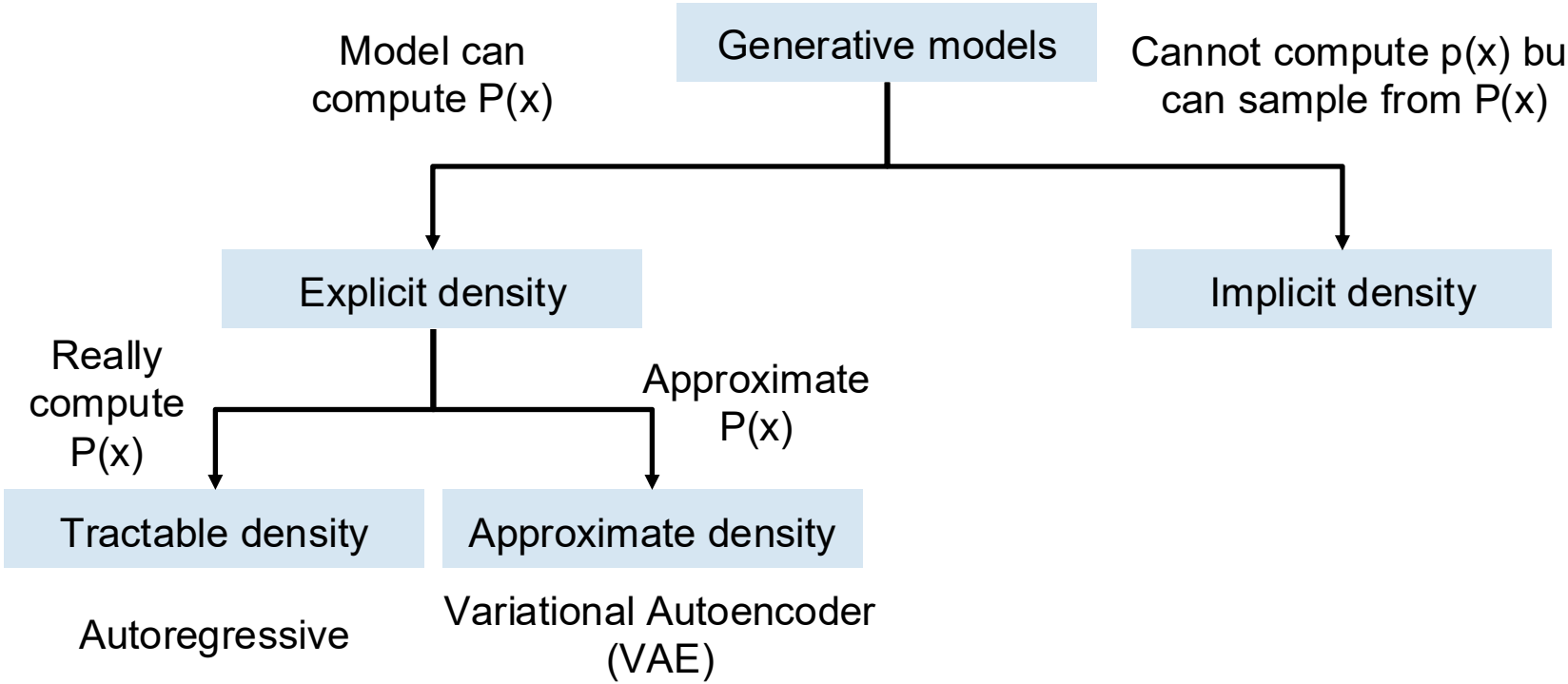
Taxonomy of Generative Models

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



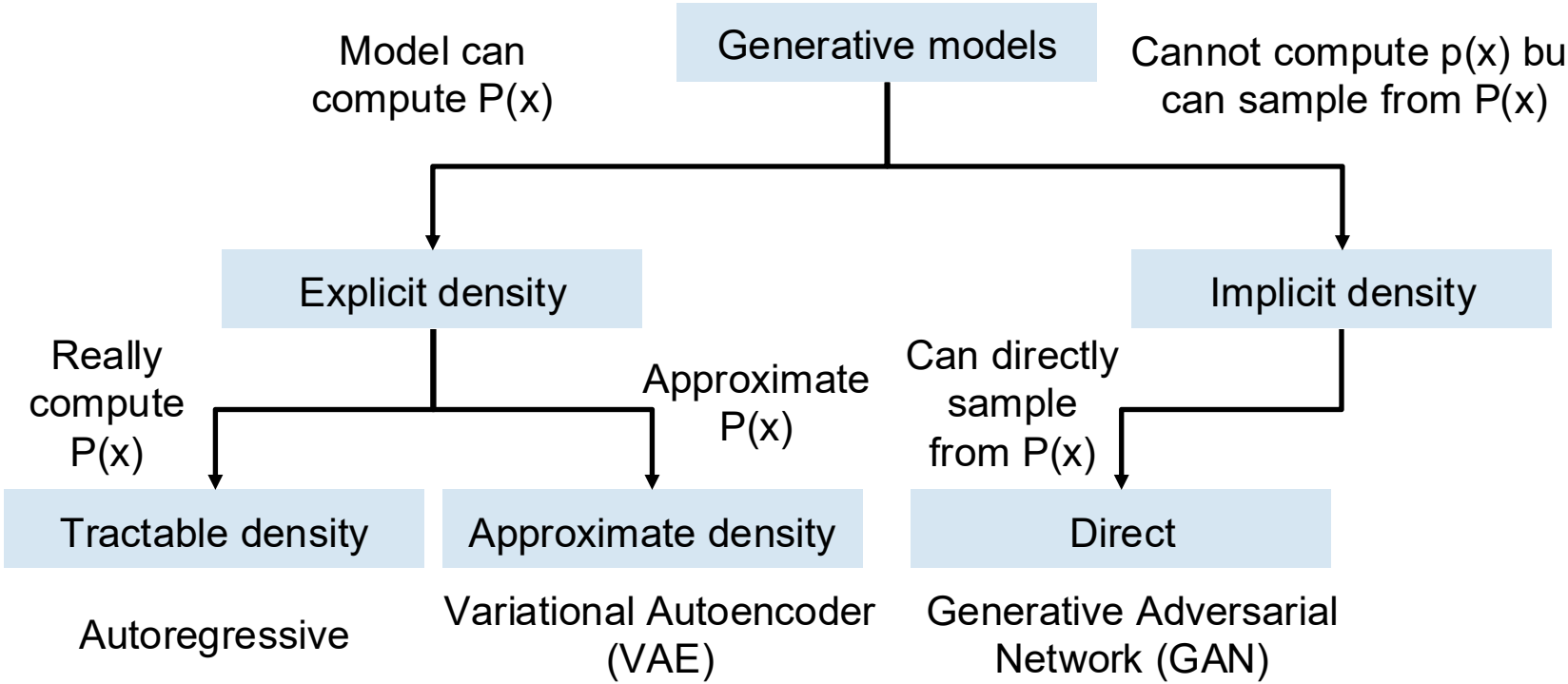
Taxonomy of Generative Models

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



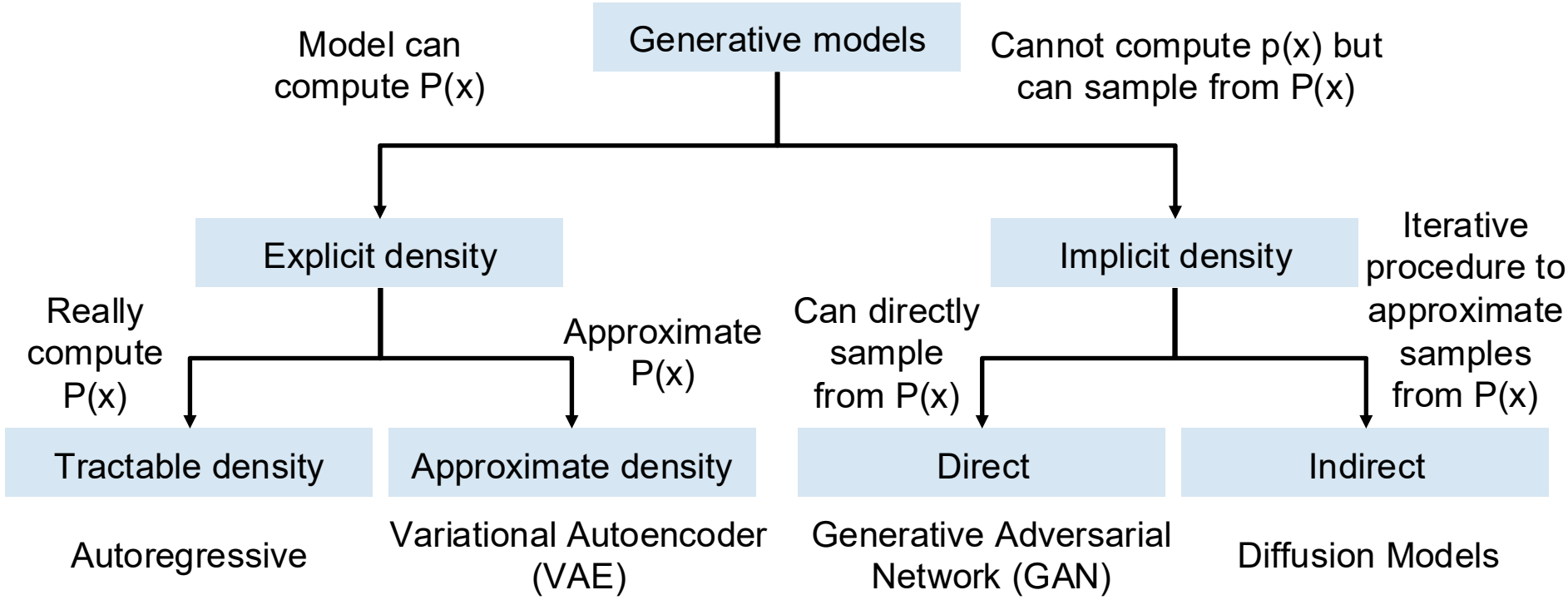
Taxonomy of Generative Models

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



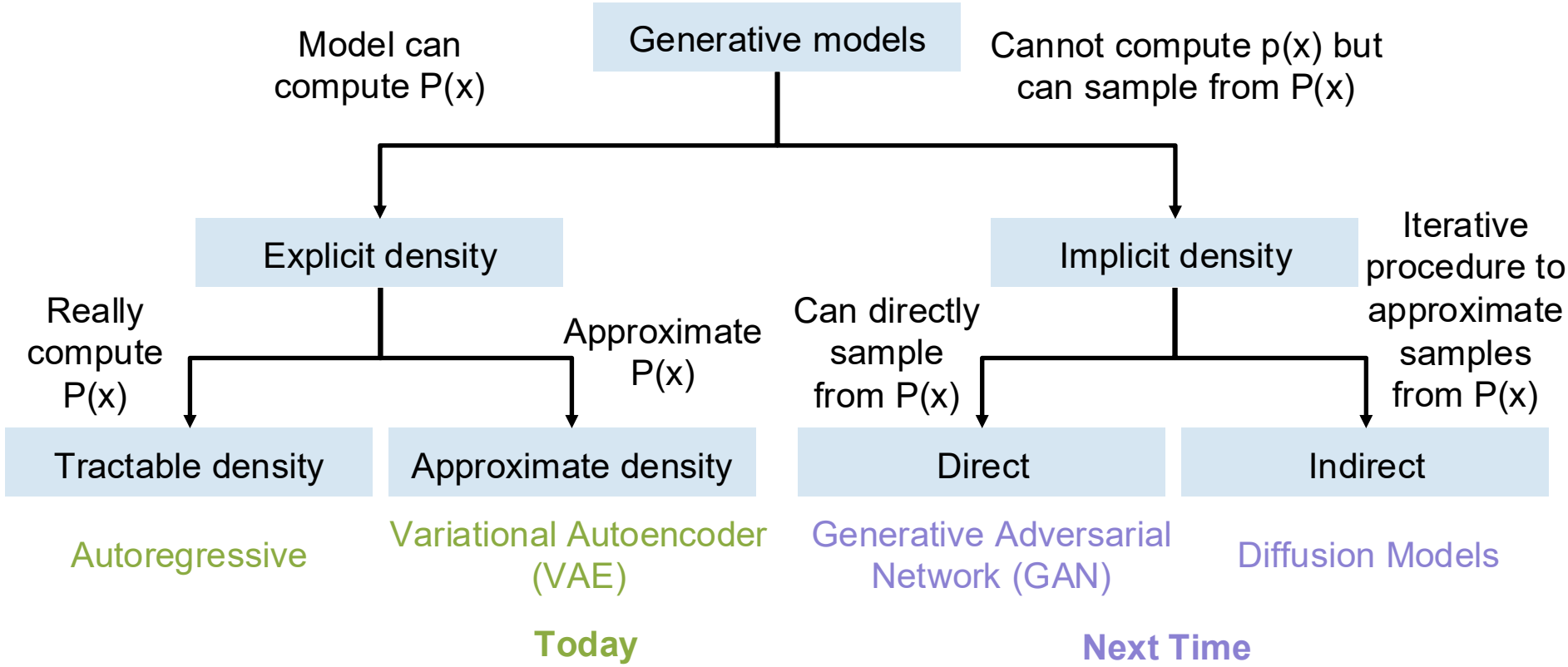
Taxonomy of Generative Models

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



Taxonomy of Generative Models

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



Autoregressive Models

Maximum Likelihood Estimation

Goal: Write down an explicit function for $p(x) = f(x, W)$

Maximum Likelihood Estimation

Goal: Write down an explicit function for $p(x) = f(x, W)$

Given dataset $x^{(1)}, x^{(2)}, \ldots x^{(N)}$, train the model by solving:

$$W^* = \arg\max_W \prod_i p(x^{(i)})$$

Maximize probability of training data
(Maximum likelihood estimation)

Maximum Likelihood Estimation

Goal: Write down an explicit function for $p(x) = f(x, W)$

Given dataset $x^{(1)}, x^{(2)}, \ldots x^{(N)}$, train the model by solving:

$$W^* = \arg\max_W \prod_i p(x^{(i)})$$

Maximize probability of training data (Maximum likelihood estimation)

$$= \arg\max_W \sum_i \log p(x^{(i)})$$

Log trick: Swap product and sum

Maximum Likelihood Estimation

Goal: Write down an explicit function for $p(x) = f(x, W)$

Given dataset $x^{(1)}, x^{(2)}, \ldots x^{(N)}$, train the model by solving:

$$W^* = \arg\max_W \prod_i p(x^{(i)})$$

Maximize probability of training data (Maximum likelihood estimation)

$$= \arg\max_W \sum_i \log p(x^{(i)})$$

Log trick: Swap product and sum

$$= \arg\max_W \sum_i \log f(x^{(i)}, W)$$

This is our loss function. maximize it with gradient descent

Autoregressive Models

Goal: Write down an explicit function for $p(x) = f(x, W)$

Assume x is a sequence: $x = (x_1, x_2, \ldots, x_T)$

Autoregressive Models

Goal: Write down an explicit function for $p(x) = f(x, W)$

Assume x is a sequence: $x = (x_1, x_2, \ldots, x_T)$

Use the chain rule of probability:

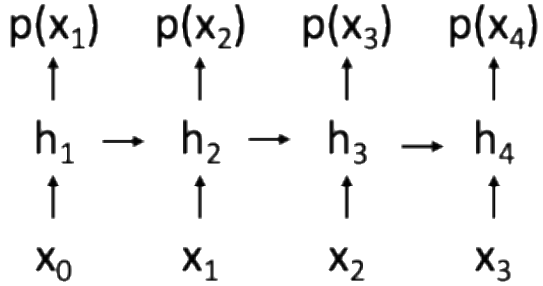
$$\begin{aligned} p(x) &= p(x_1, x_2, x_3, \ldots, x_T) \\ &= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \ldots \\ &= \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1}) \end{aligned}$$

Autoregressive Models

Goal: Write down an explicit function for $p(x) = f(x, W)$

Assume x is a sequence: $x = (x_1, x_2, \ldots, x_T)$

We have already seen this!



Language modeling with RNN

Use the chain rule of probability:

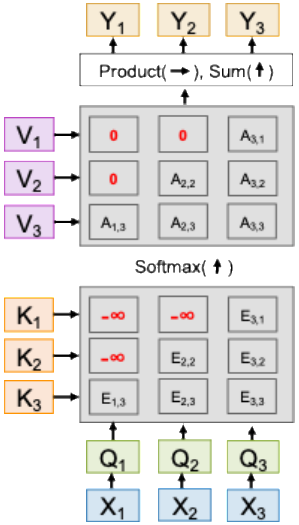
$$
\begin{aligned}
p(x) &= p(x_1, x_2, x_3, \ldots, x_T) \\
&= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \ldots \\
&= \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})
\end{aligned}
$$

LLMs are Autoregressive Models

Goal: Write down an explicit function for $p(x) = f(x, W)$

Assume x is a sequence: $x = (x_1, x_2, \ldots, x_T)$

Language modeling with masked Transformer



Use the chain rule of probability:

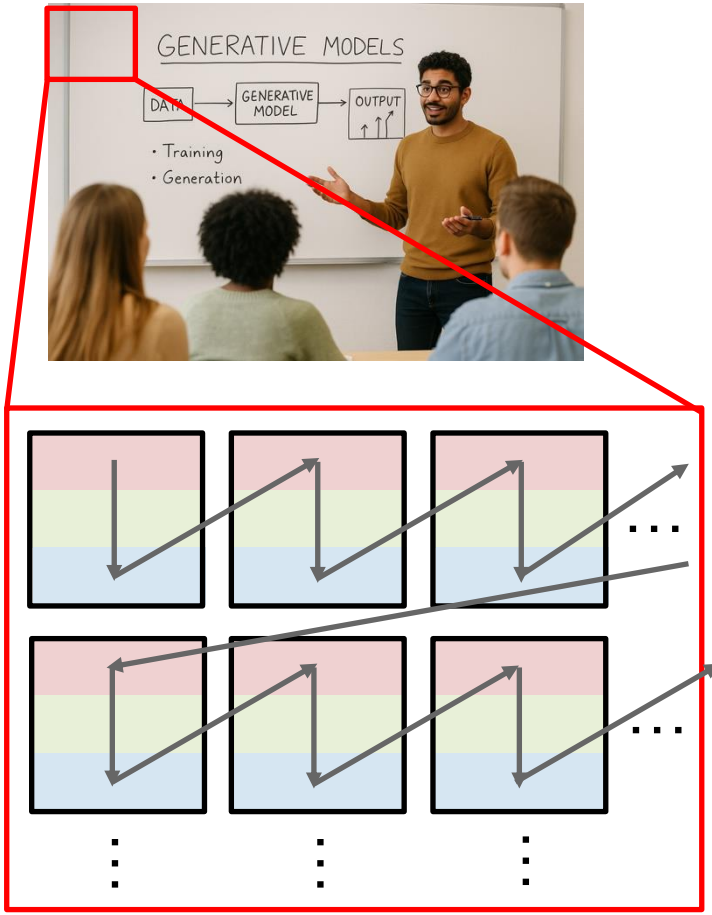
$$p(x) = p(x_1, x_2, x_3, \ldots, x_T)$$
$$= p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \ldots$$
$$= \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$

Autoregressive Models of Images

Treat an image as a sequence of 8-bit subpixel values (scanline order)

Predict each subpixel as a classification among 256 values [0…255]

Model with an RNN or Transformer



Van den Oord et al, "Pixel Recurrent Neural Networks", ICML 2016
Van den Oord et al, "Conditional Image Generation with PixelCNN Decoders", NeurIPS 2016

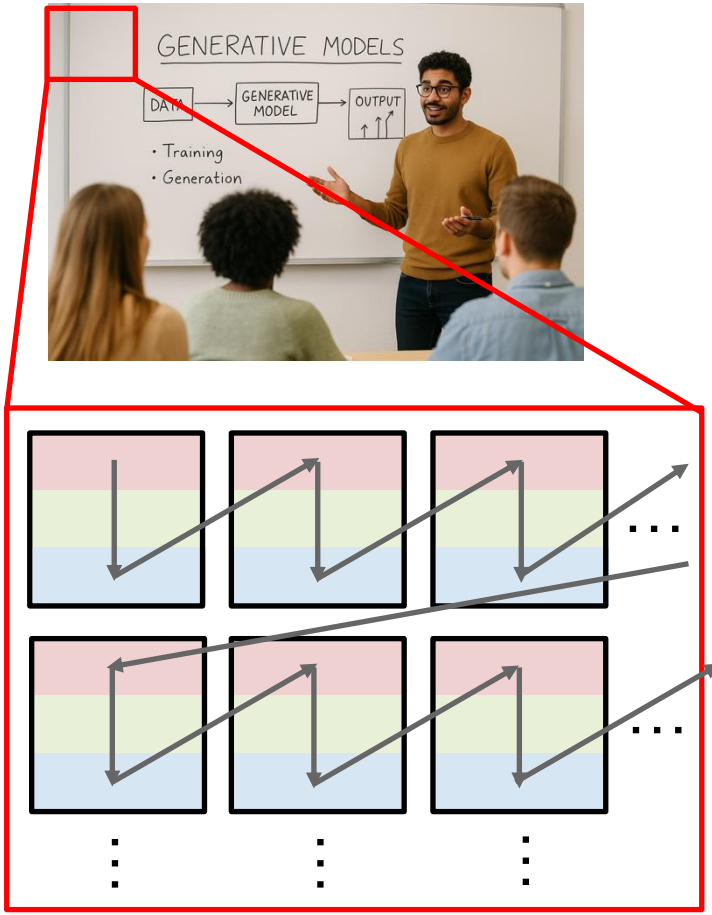
Autoregressive Models of Images

Treat an image as a sequence of 8-bit subpixel values (scanline order)

Predict each subpixel as a classification among 256 values [0…255]

Model with an RNN or Transformer

Problem: Too expensive. 1024x1024 image is a sequence of 3M subpixels



Van den Oord et al, "Pixel Recurrent Neural Networks", ICML 2016
Van den Oord et al, "Conditional Image Generation with PixelCNN Decoders", NeurIPS 2016

Autoregressive Models of Images

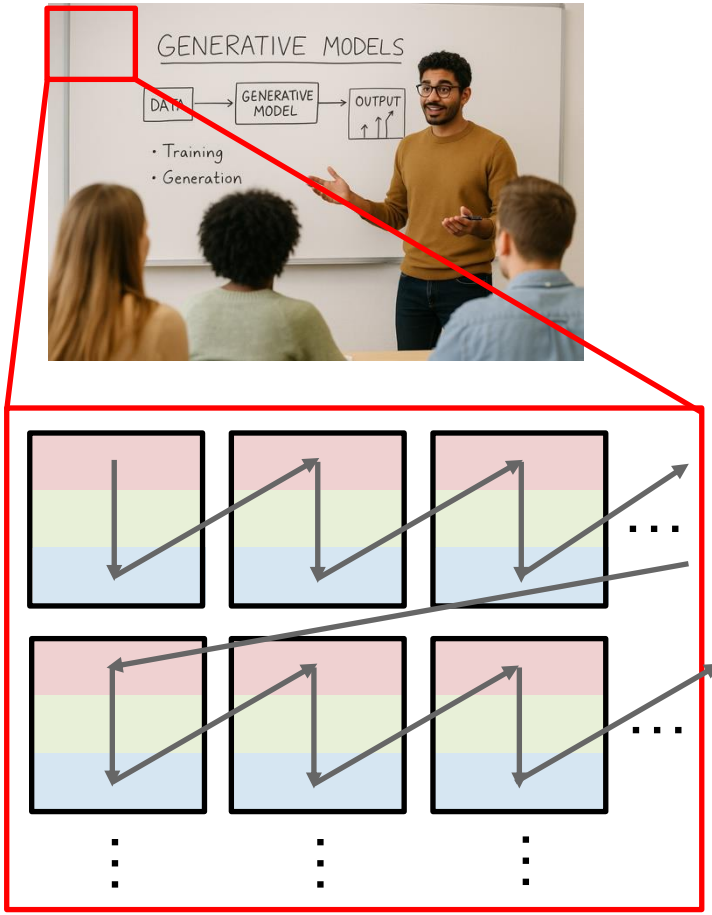
Treat an image as a sequence of 8-bit subpixel values (scanline order)

Predict each subpixel as a classification among 256 values [0…255]

Model with an RNN or Transformer

Problem: Too expensive. 1024x1024 image is a sequence of 3M subpixels

Solution (jumping ahead): Model as sequence of tiles, not sequence of subpixels



Variational Autoencoders (VAEs)

Variational Autoencoders

PixelRNN / PixelCNN explicitly parameterizes density function with a neural network, so we can train to maximize likelihood of training data:

$$\mathrm{p}_W(x) = \prod_{t=1}^{T} p_W(x_t \mid x_1, \ldots, x_{t-1})$$

Variational Autoencoders (VAE) define an **intractable density** that we cannot explicitly compute or optimize

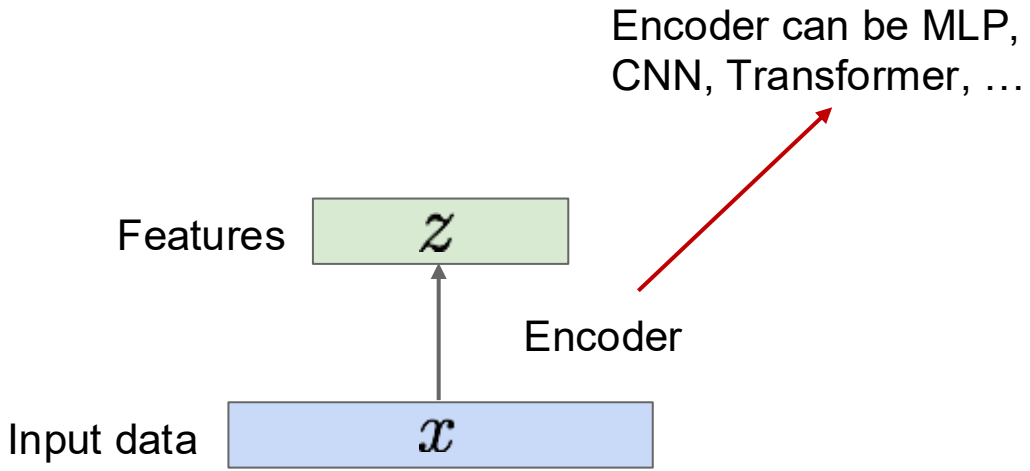
But we will be able to directly optimize a **lower bound** on the density

Variational Autoencoders (VAEs)

(Non-Variational) Autoencoders

Idea: Unsupervised method for learning to extract features z from inputs x, without labels

Features should extract useful information
(object identity, appearance, scene type, etc)
that can be used for downstream tasks

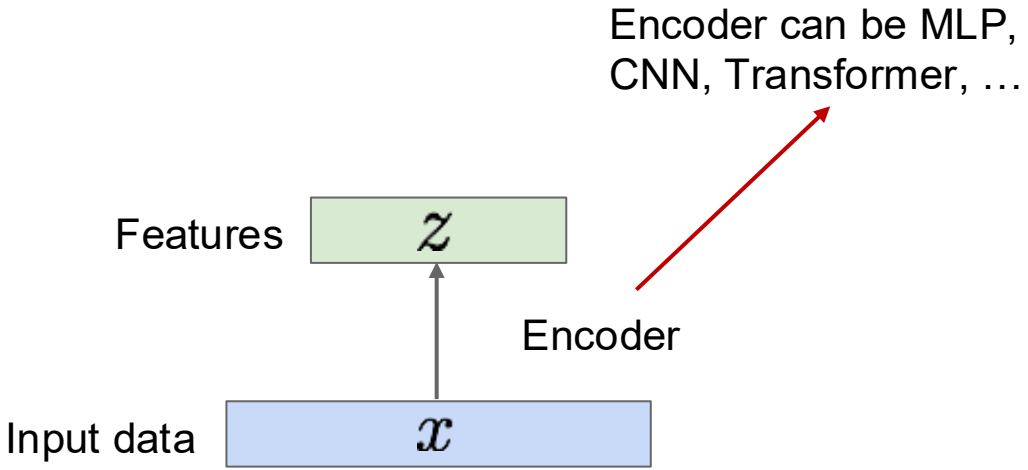


Input Data

(Non-Variational) Autoencoders

Problem: How can we learn without labels?

Features should extract useful information
(object identity, appearance, scene type, etc)
that can be used for downstream tasks



Input Data

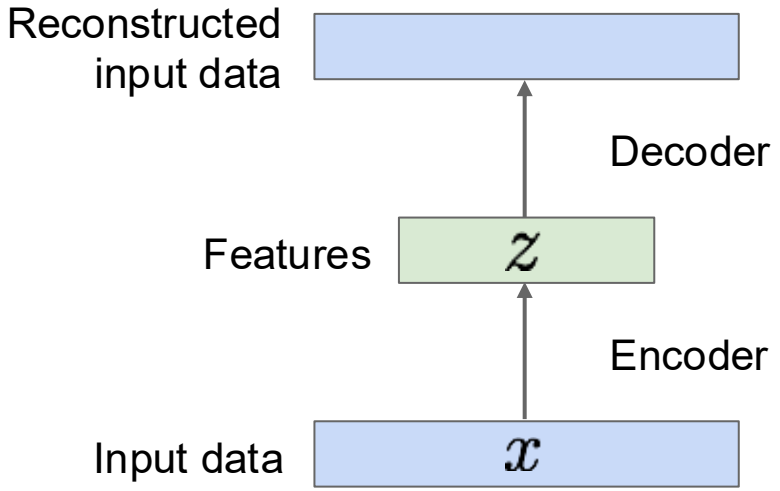
(Non-Variational) Autoencoders

"Autoencoding" = Encoding yourself

Problem: How can we learn without labels?

Solution: Reconstruct the input data with a decoder.

Decoder can be MLP, CNN, Transformer, …

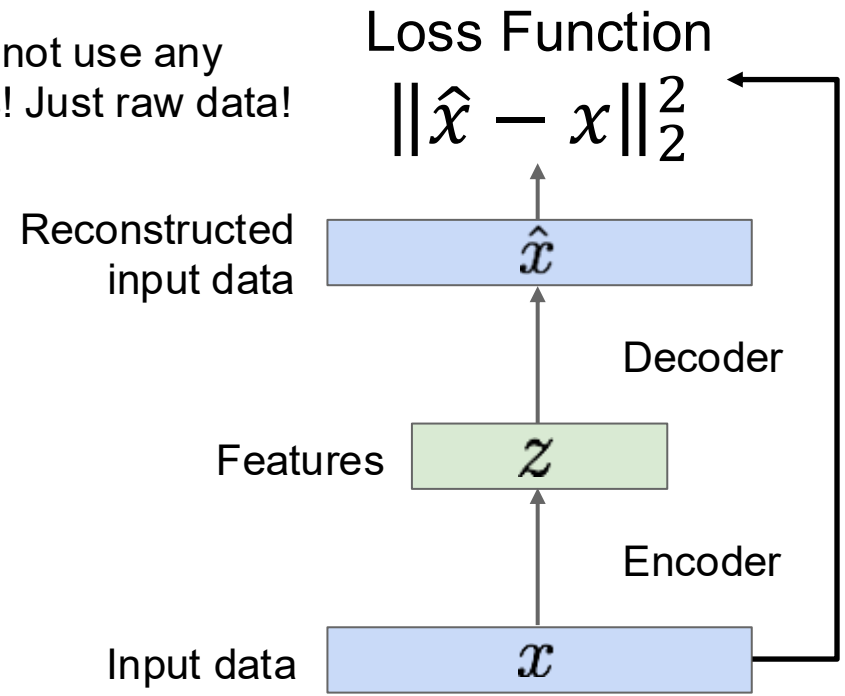


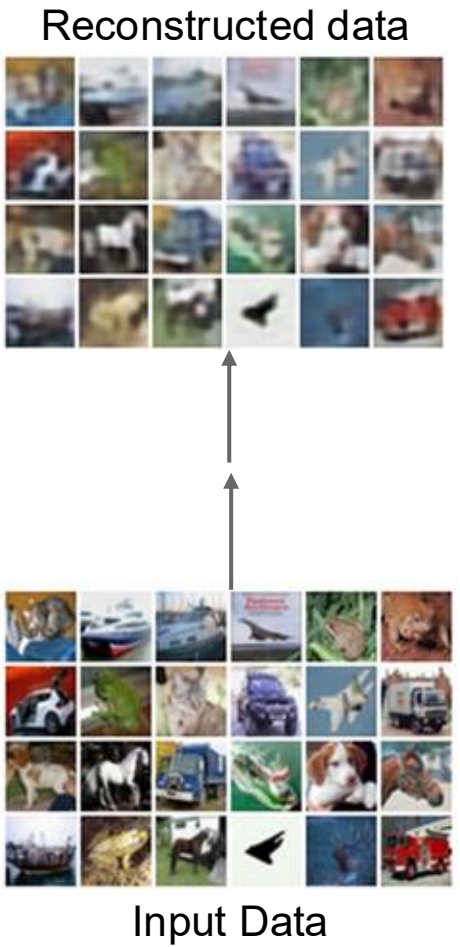
Input Data

(Non-Variational) Autoencoders

Loss: L2 distance between input and reconstructed data.

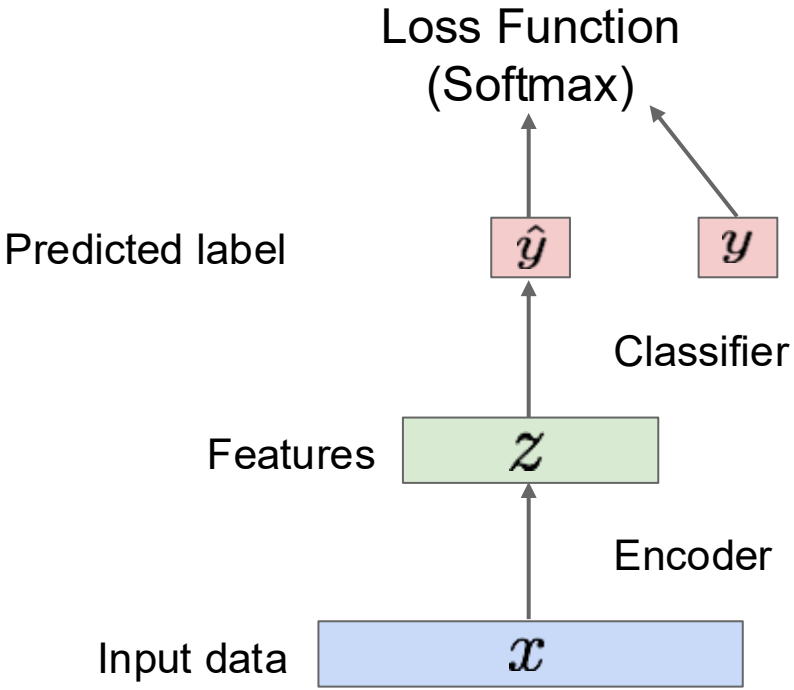
Does not use any labels! Just raw data!

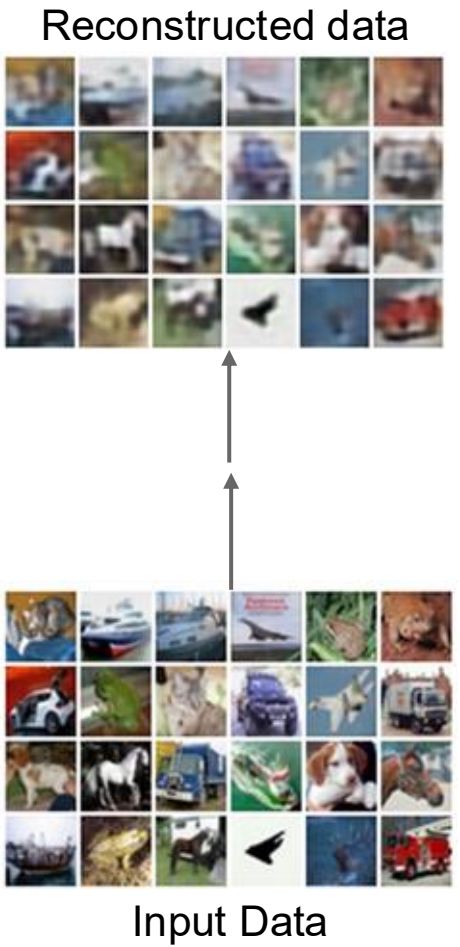




(Non-Variational) Autoencoders

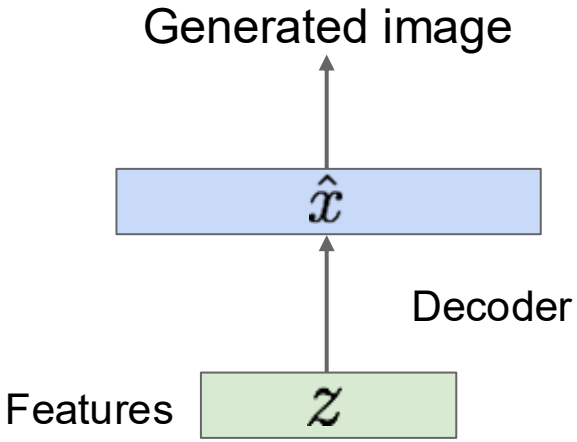
After training, can use encoder for downstream tasks





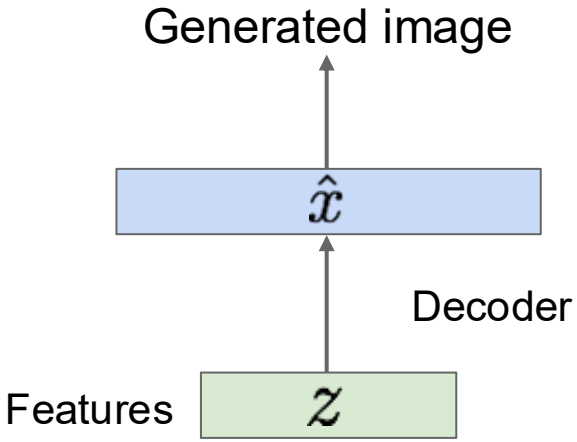
(Non-Variational) Autoencoders

If we could generate new z, could use the decoder to generate images



(Non-Variational) Autoencoders

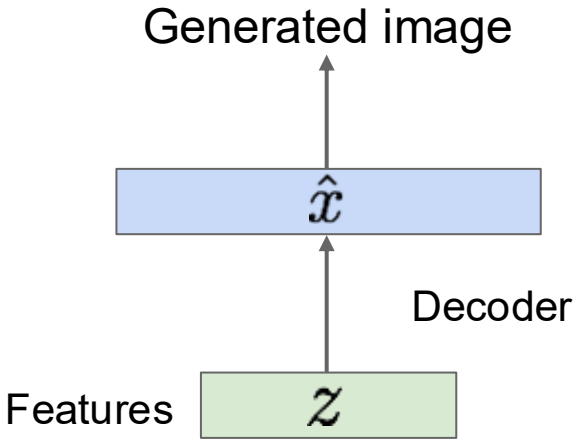
If we could generate new z, could use the decoder to generate images



Problem: Generating new z is not any easier than generating new x

(Non-Variational) Autoencoders

If we could generate new z, could use the decoder to generate images



Problem: Generating new z is not any easier than generating new x

Solution: What if we force all z to come from a known distribution?

Variational Autoencoders (VAEs)

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

Variational Autoencoders

Probabilistic spin on autoencoders:

1. Learn latent features z from raw data
2. Sample from the model to generate new data

Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features z from raw data
2. Sample from the model to generate new data

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

Intuition: x is an image, **z** is latent factors used to generate **x:** attributes, orientation, etc.

Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features z from raw data
2. Sample from the model to generate new data

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

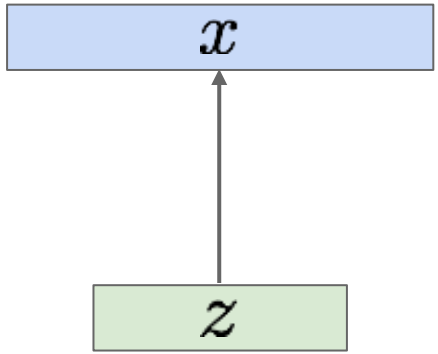
After training, sample new data like this:

Sample from conditional

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

Sample z from prior

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

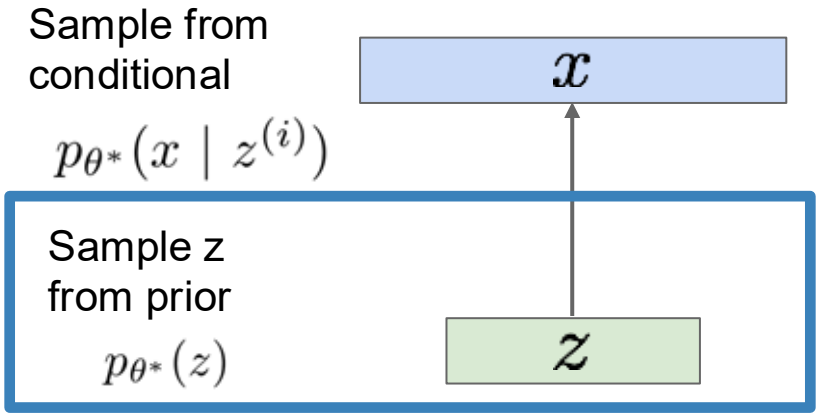


Intuition: x is an image, **z** is latent factors used to generate **x:** attributes, orientation, etc.

Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



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

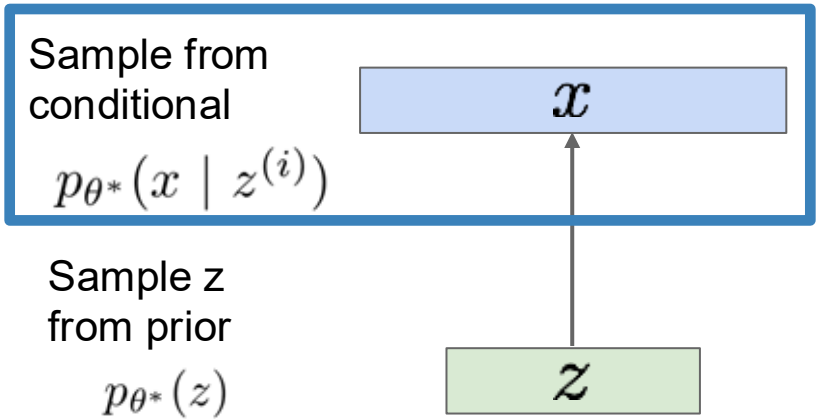
Intuition: x is an image, **z** is latent factors used to generate **x:** attributes, orientation, etc.

Assume simple prior p(z), e.g. Gaussian

Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



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

How can we train this?

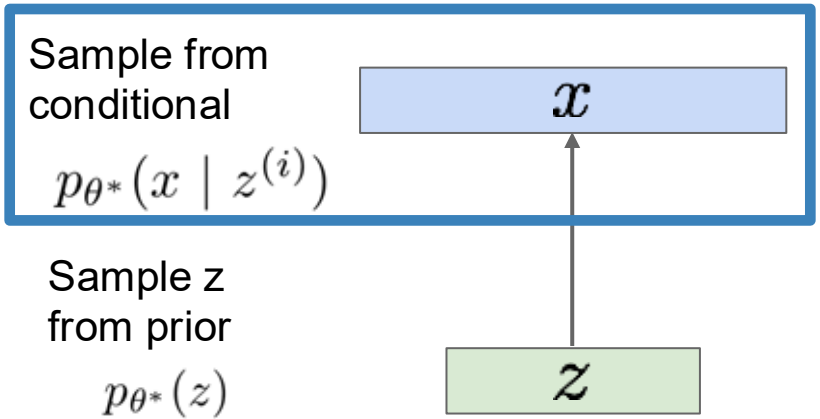
Basic idea: **maximum likelihood**

If we had a dataset of (x, z) then train a *conditional generative model* p(x | z)

Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



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

How can we train this?

Basic idea: **maximum likelihood**

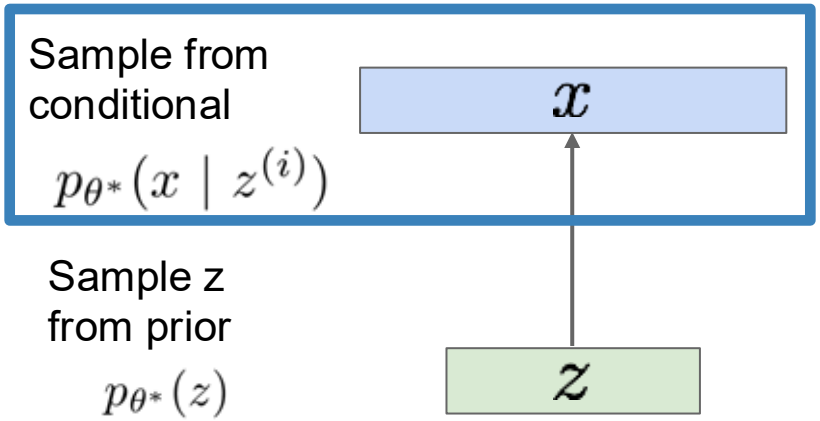
We don't observe z, so **marginalize**:

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

Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



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

How can we train this?

Basic idea: **maximum likelihood**

We don't observe z, so **marginalize**:

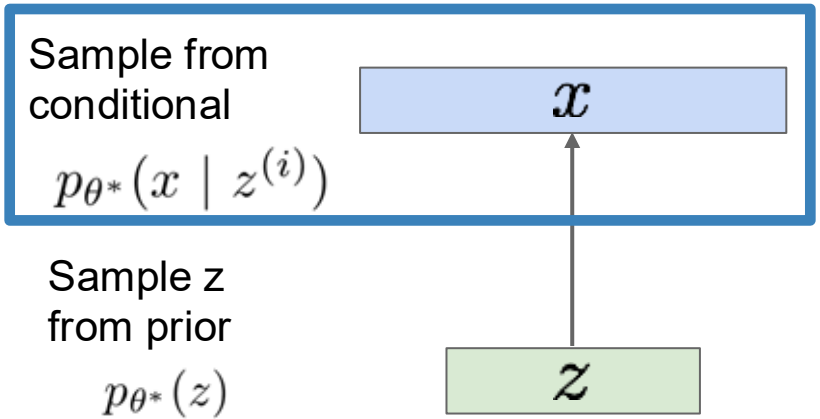
$$p_\theta(x) = \int p_\theta(x,z)dz = \int p_\theta(x|z)p_\theta(z)dz$$

Ok, we can compute this with the decoder

Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



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

How can we train this?

Basic idea: **maximum likelihood**

We don't observe z, so **marginalize**:

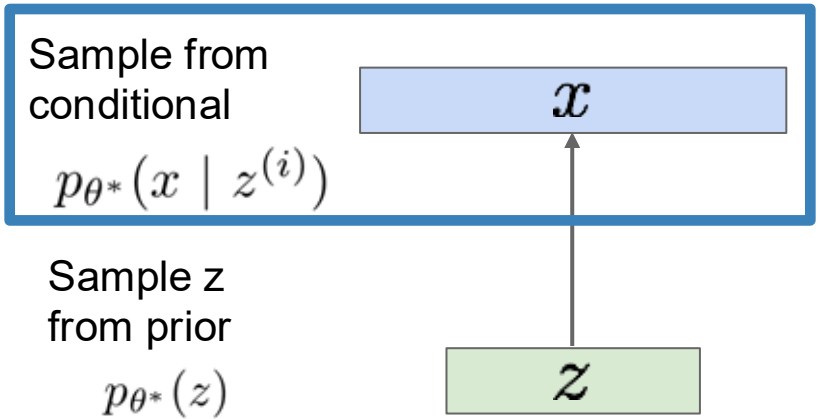
$$p_\theta(x) = \int p_\theta(x,z)dz = \int p_\theta(x|z)p_\theta(z)dz$$

Ok, we assumed Gaussian prior for z

Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



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

How can we train this?

Basic idea: **maximum likelihood**

We don't observe z, so **marginalize**:

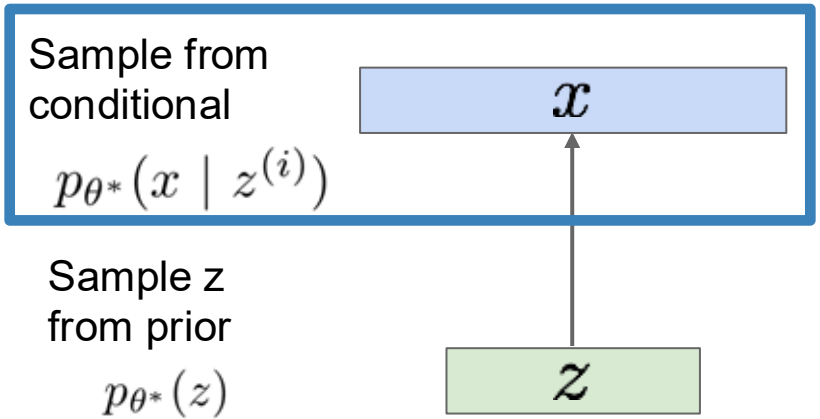
$$p_\theta(x) = \int p_\theta(x,z)dz = \int p_\theta(x|z)p_\theta(z)\boldsymbol{dz}$$

Problem, we can't integrate over all z

Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



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

How can we train this?

Basic idea: **maximum likelihood**

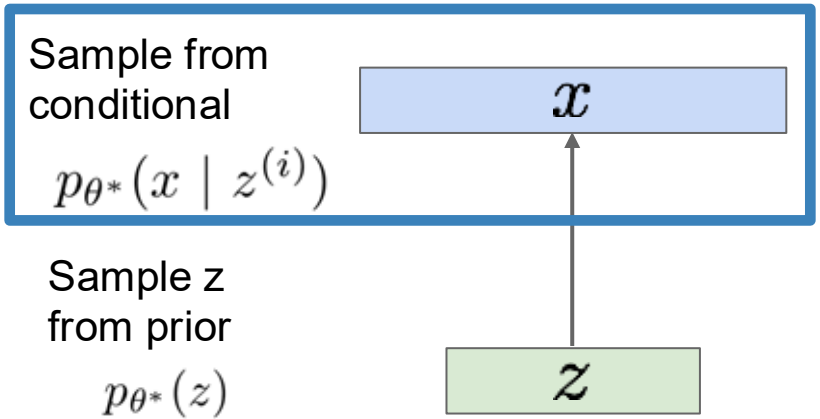
Another idea: Try Bayes' Rule:

$$p_\theta(x) = \frac{p_\theta(x \mid z)p_\theta(z)}{p_\theta(z \mid x)}$$

Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



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

How can we train this?

Basic idea: **maximum likelihood**

Another idea: Try Bayes' Rule:

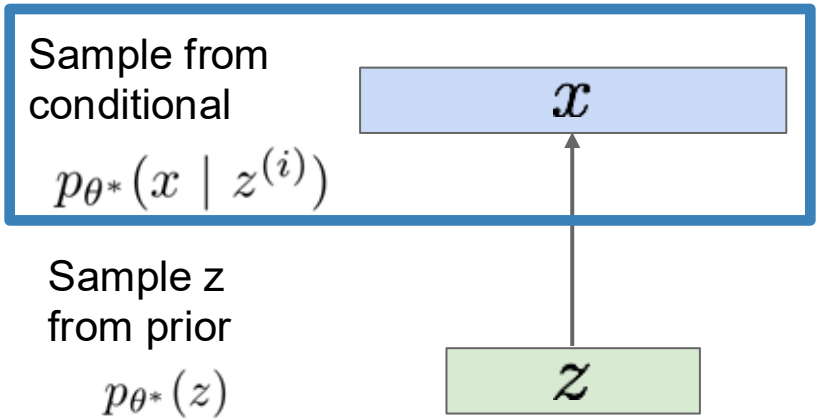
$$p_\theta(x) = \frac{p_\theta(x \mid z)p_\theta(z)}{p_\theta(z \mid x)}$$

Ok, we can compute this with the decoder

Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



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

How can we train this?

Basic idea: **maximum likelihood**

Another idea: Try Bayes' Rule:

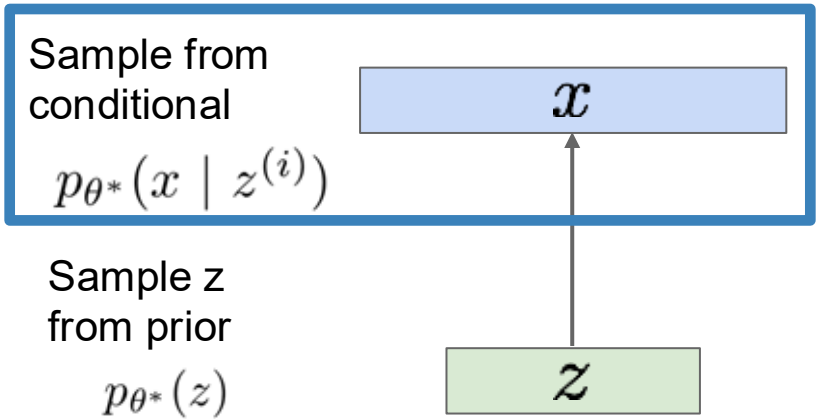
$$p_\theta(x) = \frac{p_\theta(x \mid z)p_\theta(z)}{p_\theta(z \mid x)}$$

Ok, we assumed Gaussian prior for z

Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



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

How can we train this?

Basic idea: **maximum likelihood**

Another idea: Try Bayes' Rule:

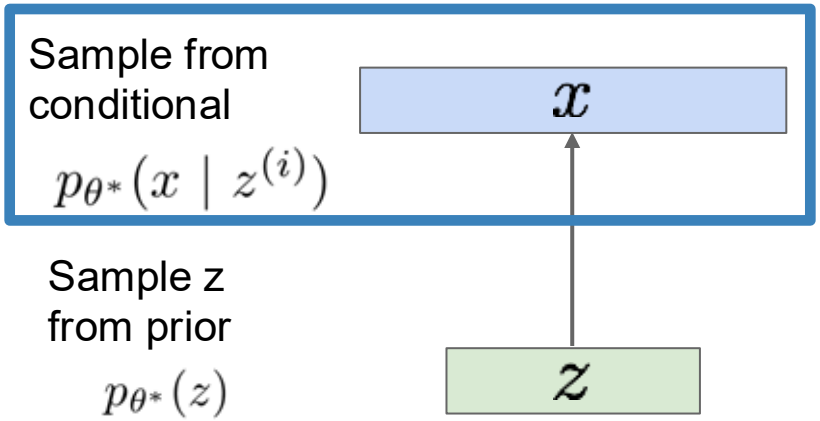
$$p_\theta(x) = \frac{p_\theta(x \mid z)p_\theta(z)}{p_\theta(z \mid x)}$$

Problem: no way to compute this

Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



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

How can we train this?

Basic idea: **maximum likelihood**

Another idea: Try Bayes' Rule:

$$p_\theta(x) = \frac{p_\theta(x \mid z)p_\theta(z)}{p_\theta(z \mid x)}$$

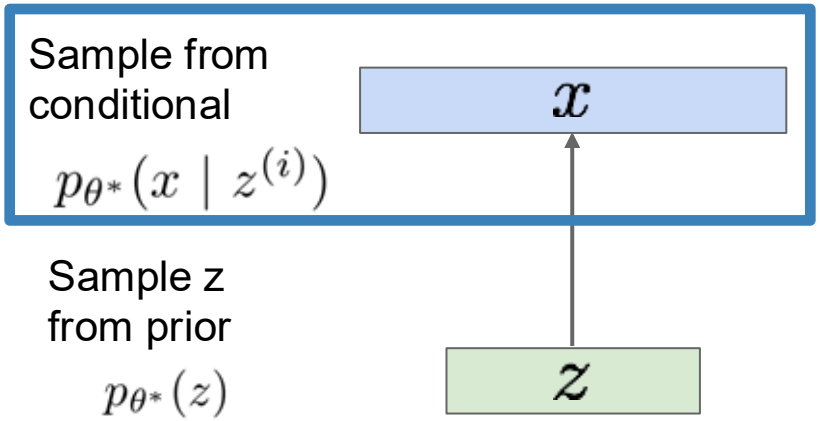
Problem: no way to compute this

Solution: Train another network that learns $q_\phi(z \mid x) \approx p_\theta(z \mid x)$

Variational Autoencoders

Probabilistic spin on autoencoders:
1. Learn latent features z from raw data
2. Sample from the model to generate new data

After training, sample new data like this:



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

How can we train this?

Basic idea: **maximum likelihood**

Another idea: Try Bayes' Rule:

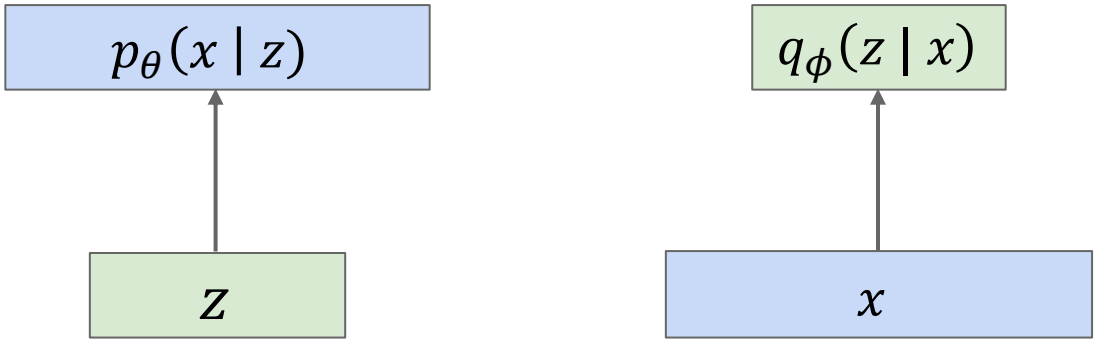
$$p_\theta(x) = \frac{p_\theta(x \mid z)p_\theta(z)}{p_\theta(z \mid x)} \approx \frac{p_\theta(x \mid z)p_\theta(z)}{q_\phi(z \mid x)}$$

Solution: Train another network that learns $q_\phi(z \mid x) \approx p_\theta(z \mid x)$

Variational Autoencoders

Decoder Network:
Input latent code z,
Output distribution over data x

Encoder Network:
Input data x,
Output distribution over latent codes z



If we can ensure that $q_\phi(z \mid x) \approx p_\theta(z \mid x)$,

then we can approximate

$$p_\theta(x) \approx \frac{p_\theta(x \mid z)p(z)}{q_\phi(z \mid x)}$$

Idea: Jointly train both encoder and decoder

Variational Autoencoders

Decoder Network:
Input latent code z,
Output distribution over data x

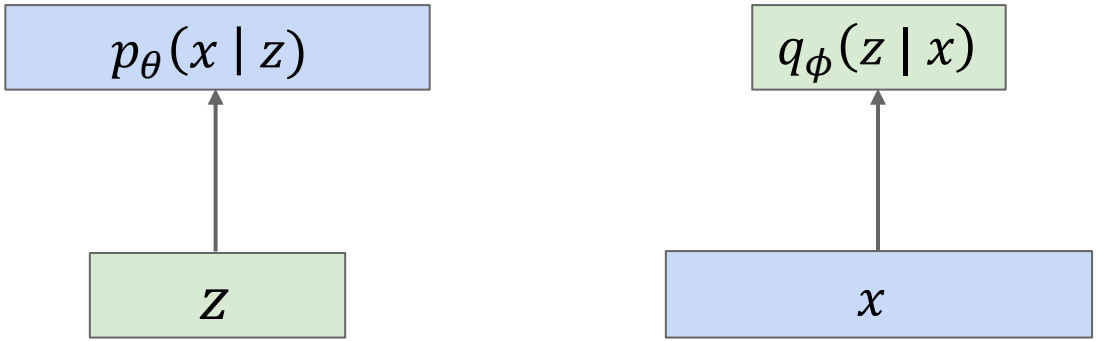
Encoder Network:
Input data x,
Output distribution over latent codes z

If we can ensure that $q_\phi(z \mid x) \approx p_\theta(z \mid x)$,

then we can approximate

$$p_\theta(x) \approx \frac{p_\theta(x \mid z)p(z)}{q_\phi(z \mid x)}$$

Idea: Jointly train both encoder and decoder



Aside: How to output probability distributions from neural networks?

Network outputs mean (and std) of a (diagonal) distribution

Variational Autoencoders

Decoder Network:
Input latent code z,
Output distribution over data x

$$p_\theta(x \mid z) = N(\mu_{x|z}, \sigma^2)$$

Encoder Network:
Input data x,
Output distribution
over latent codes z

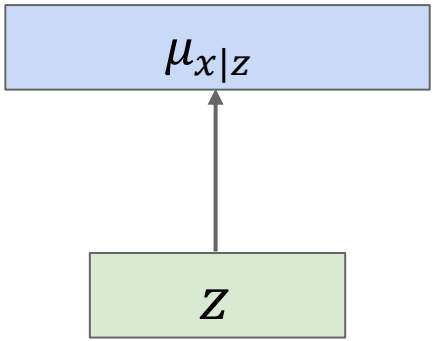
$$q_\phi(z \mid x) = N(\mu_{z|x}, \Sigma_{z|x})$$

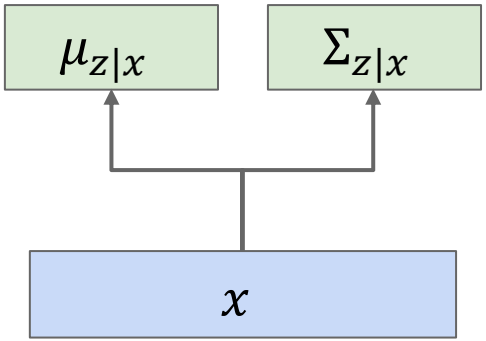
If we can ensure that $q_\phi(z \mid x) \approx p_\theta(z \mid x)$,

then we can approximate

$$p_\theta(x) \approx \frac{p_\theta(x \mid z)p(z)}{q_\phi(z \mid x)}$$

Idea: Jointly train both encoder and decoder





Aside: How to output probability distributions from neural networks?

Network outputs mean (and std) of a (diagonal) distribution

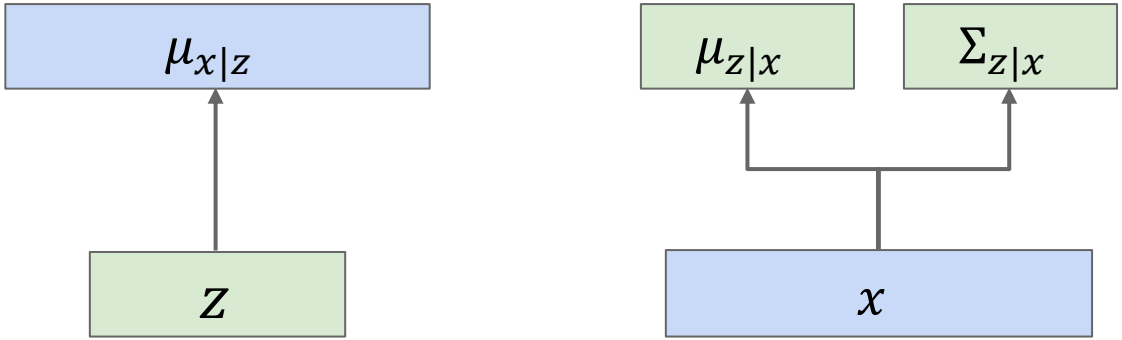
Variational Autoencoders

Decoder Network:
Input latent code z,
Output distribution over data x

Encoder Network:
Input data x,
Output distribution
over latent codes z

$$p_\theta(x \mid z) = N(\mu_{x|z}, \sigma^2) \qquad q_\phi(z \mid x) = N(\mu_{z|x}, \Sigma_{z|x})$$

$$\log p_\theta(x \mid z) = -\frac{1}{2\sigma^2}\|x - \mu\|_2^2 + C_2$$



If we can ensure that
$q_\phi(z \mid x) \approx p_\theta(z \mid x)$,

then we can approximate

$$p_\theta(x) \approx \frac{p_\theta(x \mid z)p(z)}{q_\phi(z \mid x)}$$

Idea: Jointly train both encoder and decoder

Maximizing $\log p_\theta(x \mid z)$ is equivalent to minimizing L2 distance between x and network output!

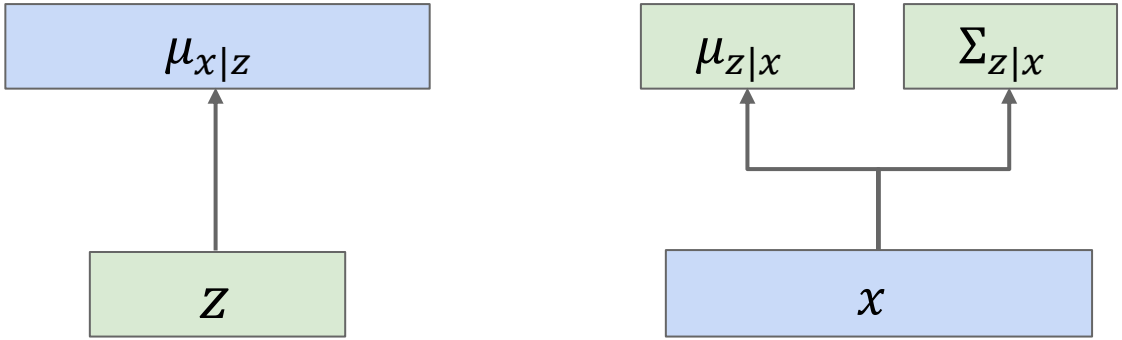
Variational Autoencoders

Decoder Network:
Input latent code z,
Output distribution over data x

Encoder Network:
Input data x,
Output distribution
over latent codes z

$$p_\theta(x \mid z) = N(\mu_{x|z}, \sigma^2) \qquad q_\phi(z \mid x) = N(\mu_{z|x}, \Sigma_{z|x})$$

$$\log p_\theta(x \mid z) = -\frac{1}{2\sigma^2}\|x - \mu\|_2^2 + C_2$$



If we can ensure that
$q_\phi(z \mid x) \approx p_\theta(z \mid x)$,

then we can approximate

$$p_\theta(x) \approx \frac{p_\theta(x \mid z)p(z)}{q_\phi(z \mid x)}$$

Idea: Jointly train both encoder and decoder

Q: What's our training objective?

Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)}$$

Bayes' Rule

Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

Multiply top and bottom by $q_\phi(z|x)$

Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}$$

Logarithms + rearranging

Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}$$

$$\log p_\theta(x) = E_{z \sim q_\phi(z|x)}[\log p_\theta(x)]$$

We can wrap in an expectation since it doesn't depend on z

Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

$$\log p_\theta(x) = E_{z\sim q_\phi(z|x)}[\log p_\theta(x)]$$

We can wrap in an expectation since it doesn't depend on z

Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

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

Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

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

Data reconstruction: x => encoder => decoder should reconstruct x

Can compute in closed form for Gaussians.

Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

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

Prior: Encoder output should match prior over z.
Can compute in closed for for Gaussians.

Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

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

Posterior Approximation: Encoder output $q_\phi(z|x)$ should match $p_\theta(z|x)$

We cannot compute this for Gaussians…

Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

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

Posterior Approximation: Decoder output $q_\phi(z|x)$ should match $p_\theta(z|x)$

KL is >= 0, so we can drop it to get a lower bound on likelihood

Variational Autoencoders (ELBO)

$$\log p_\theta(x) = \log \frac{p_\theta(x \mid z)p(z)}{p_\theta(z \mid x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z\left[\log \frac{q_\phi(z|x)}{p(z)}\right] + E_z\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right]$$

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

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

This is our VAE training objective

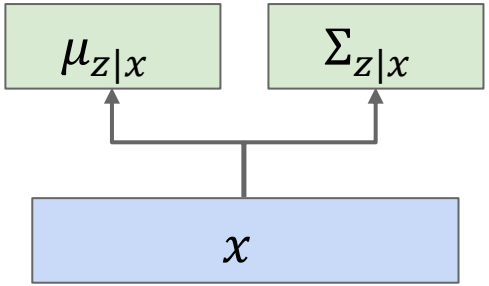
Variational Autoencoders

Jointly train **encoder** q and **decoder** p to maximize the **variational lower bound** on the data likelihood
Also called **Evidence Lower Bound** (**ELBo**)

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

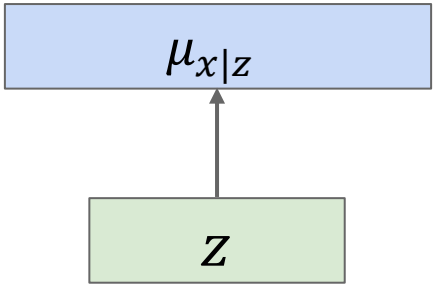
Encoder Network

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



Decoder Network

$$p_\theta(x \mid z) = N(\mu_{x|z}, \sigma^2)$$



Variational Autoencoders: Training

Train by maximizing the
variational lower bound

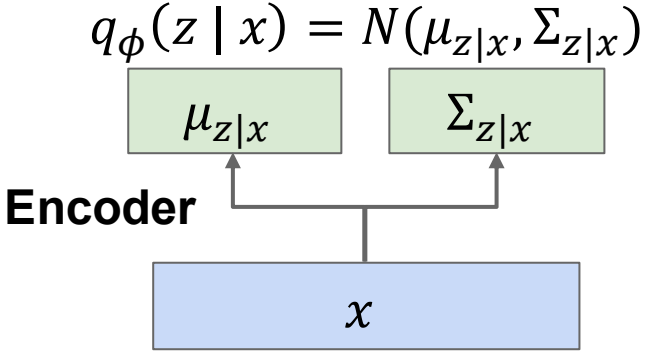
$$E_{z\sim q_\phi(z|x)}[log\, p_\theta(x|z)] - D_{KL}\left(q_\phi(z|x), p(z)\right)$$

Variational Autoencoders: Training

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 distribution over z

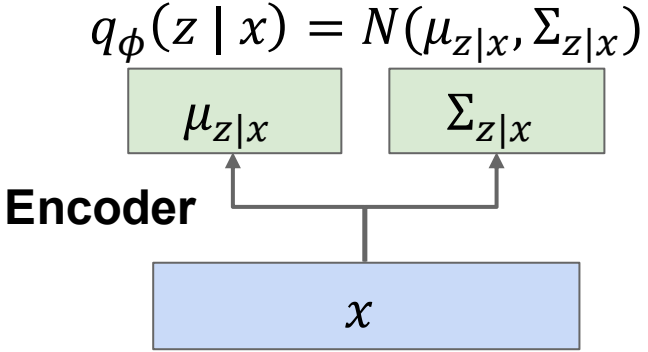


Variational Autoencoders: Training

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 distribution over z
2. Prior loss: Encoder output should be unit Gaussian (zero mean, unit variance)

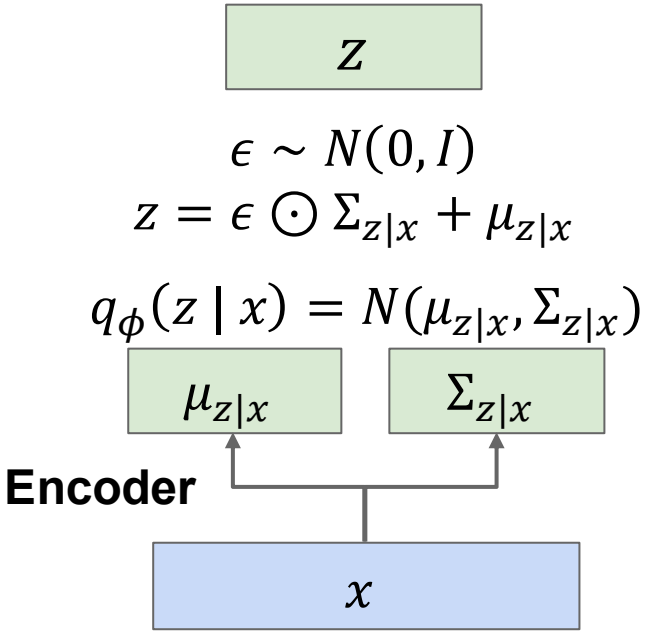


Variational Autoencoders: Training

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 distribution over z
2. Prior loss: Encoder output should be unit Gaussian (zero mean, unit variance)
3. Sample z from encoder output $q_\phi(z \mid x)$ (Reparameterization trick)

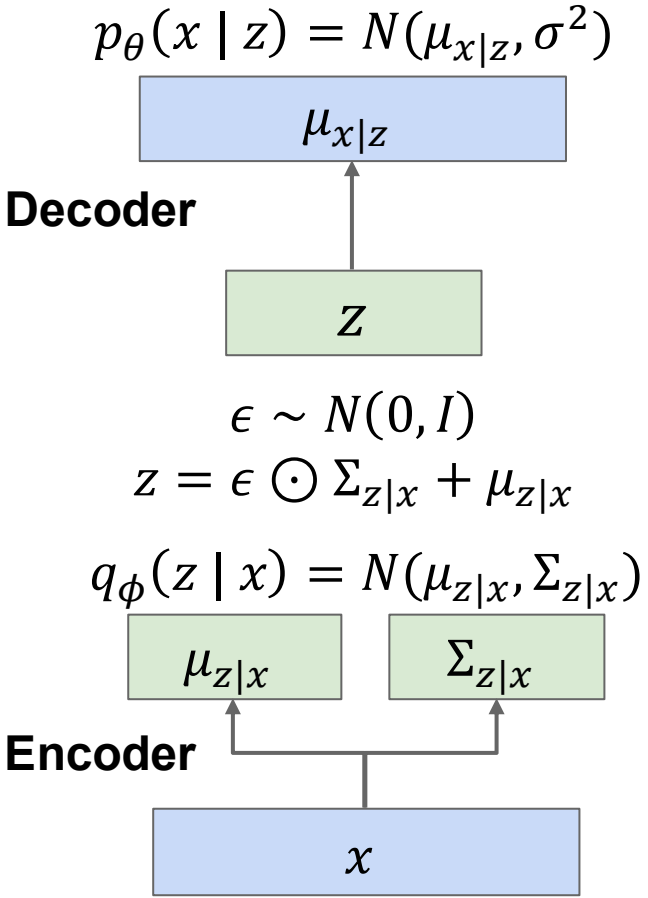


Variational Autoencoders: Training

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 distribution over z
2. Prior loss: Encoder output should be unit Gaussian (zero mean, unit variance)
3. Sample z from encoder output $q_\phi(z \mid x)$ (Reparameterization trick)
4. Run z through **decoder** to get predicted data mean

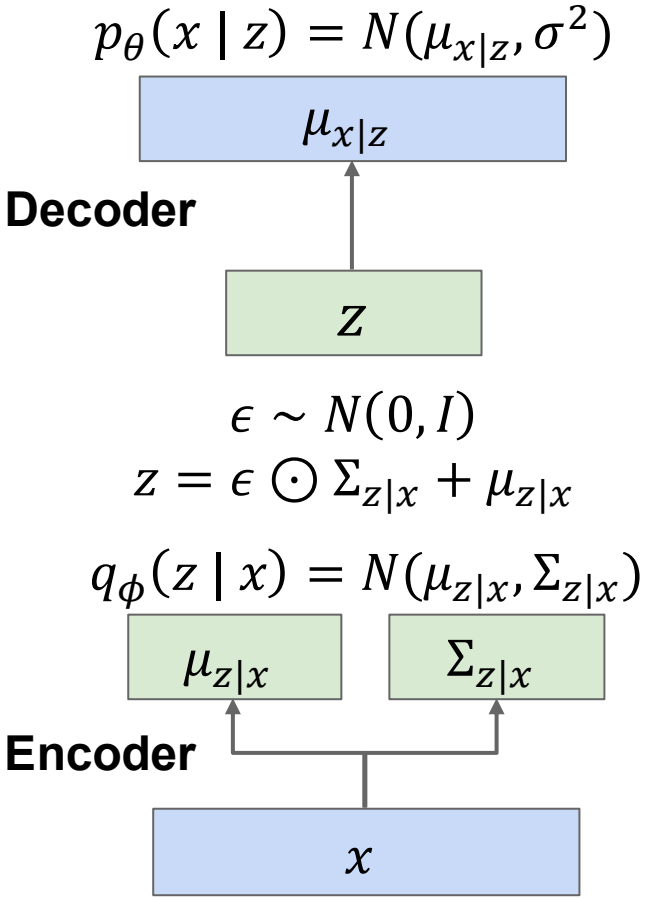


Variational Autoencoders: Training

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 distribution over z
2. Prior loss: Encoder output should be unit Gaussian (zero mean, unit variance)
3. Sample z from encoder output $q_\phi(z \mid x)$ (Reparameterization trick)
4. Run z through **decoder** to get predicted data mean
5. Reconstruction loss: predicted mean should match x in L2



Variational Autoencoders: Training

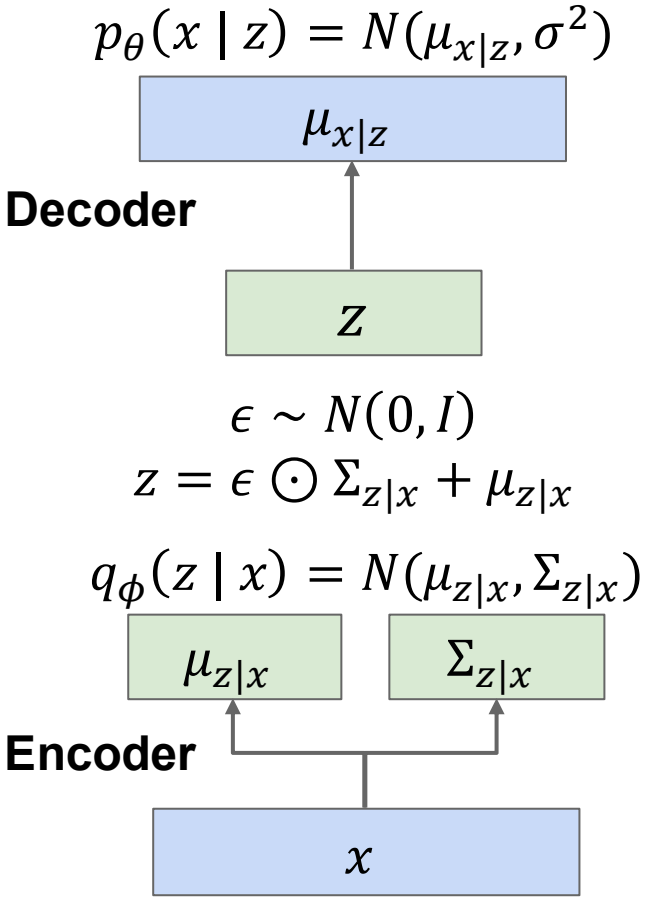
Train by maximizing the **variational lower bound**

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

The loss terms fight against each other!

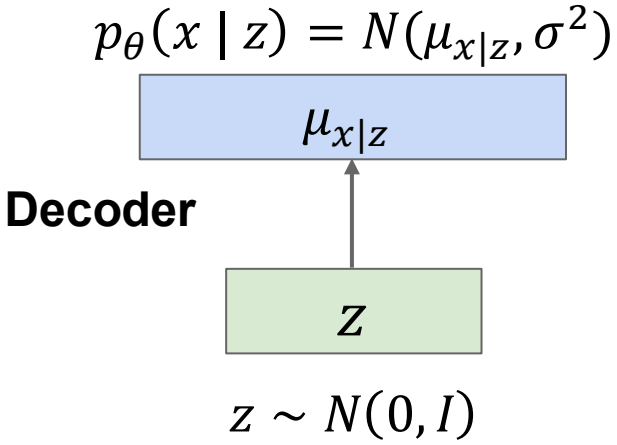
Reconstruction loss wants $\Sigma_{z|x} = 0$ and $\mu_{z|x}$ to be unique for each x, so decoder can deterministically reconstruct x

Prior loss wants $\Sigma_{z|x} = \mathbf{I}$ and $\mu_{z|x} = 0$ so encoder output is always a unit Gaussian



Variational Autoencoders: Sampling

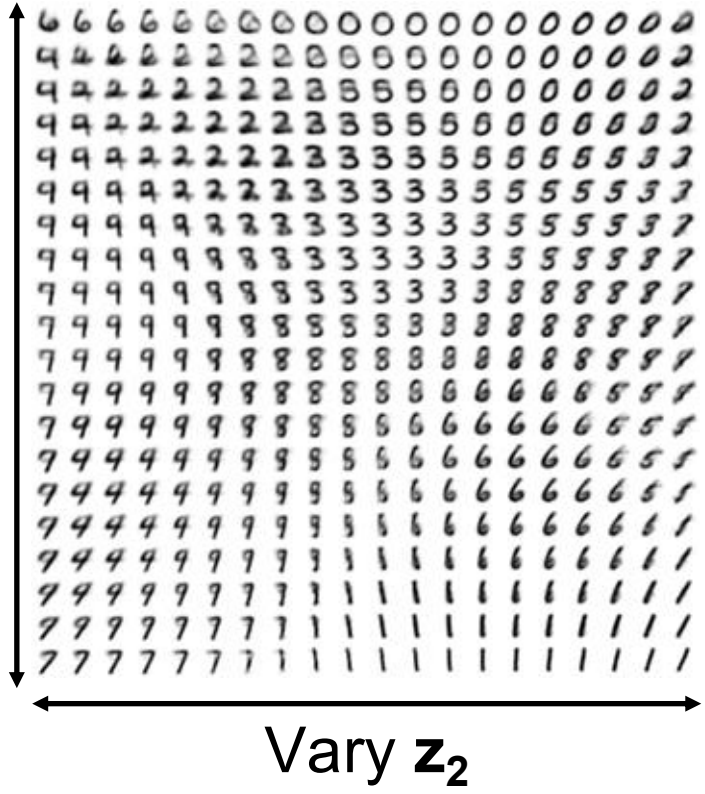
1. Sample z from the prior
2. Run through decoder to get an image



Variational Autoencoders: Disentangling

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

Recap: 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 hidden structure in data

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

Recap: 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

Density Function

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

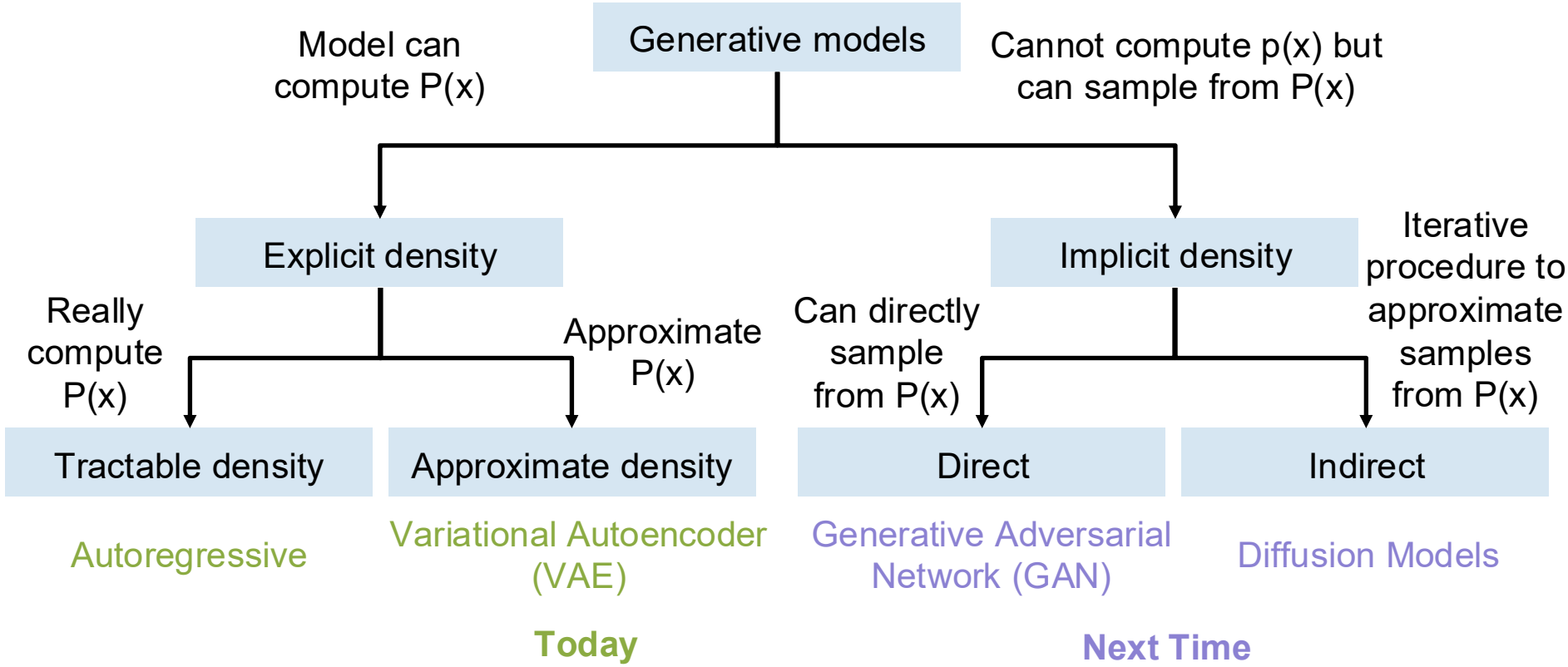
Density functions are **normalized:**

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

Different values of x **compete** for density

Recap: Generative Models

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



Next Time:
Generative Models (part 2)
Generative Adversarial Networks
Diffusion Models