Lecture 9: Generative Models

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.

Classification

Cat

This image is CC0 public domain

Supervised vs Unsupervised Learning

Supervised Learning

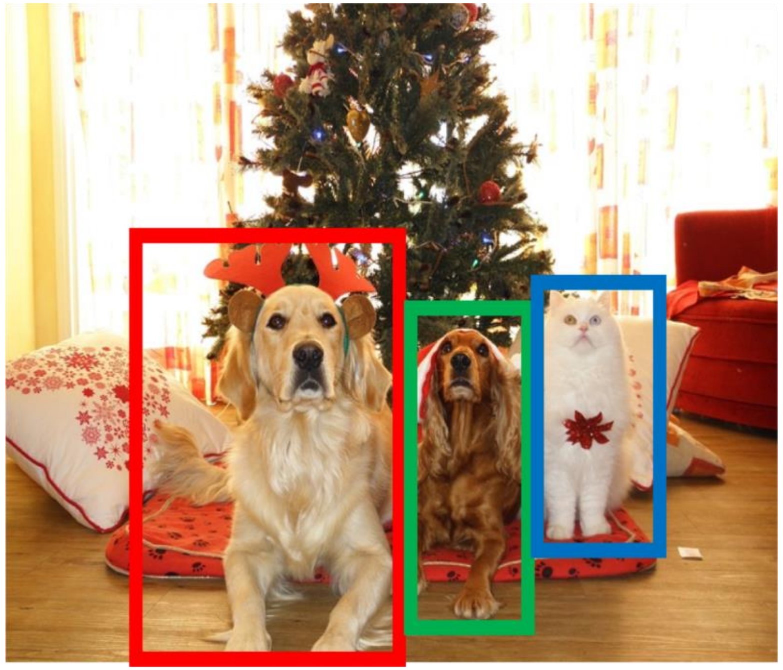
Data: (x, y)

x is data, y is label

Goal: Learn a *function* to map x -> y

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

Object Detection



DOG, DOG, CAT

Supervised vs Unsupervised Learning

Supervised Learning

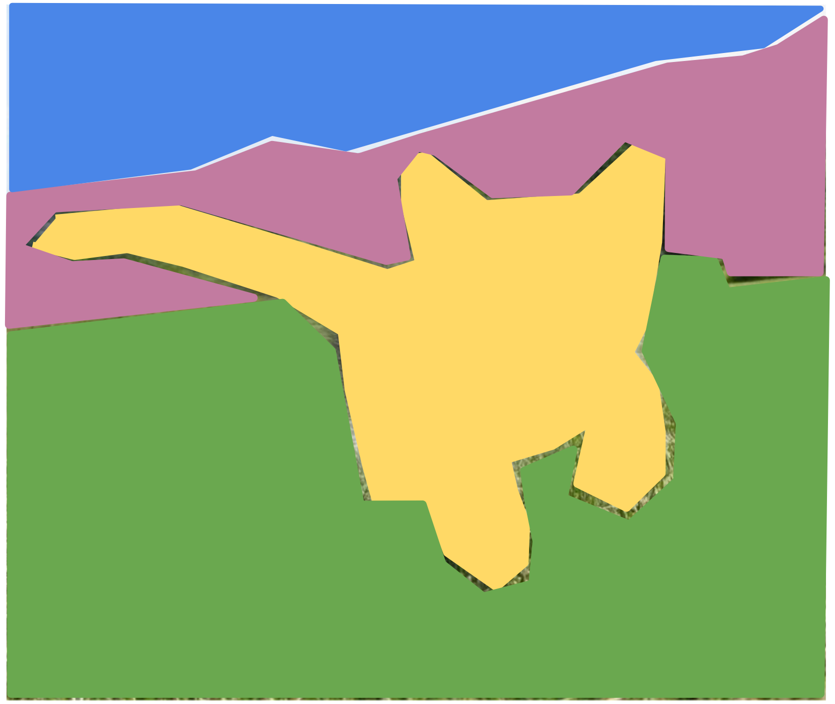
Data: (x, y)

x is data, y is label

Goal: Learn a *function* to map x -> y

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

Semantic Segmentation



GRASS, CAT, TREE, SKY

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.

Image captioning

A cat sitting on a suitcase on the floor

Caption generated using neuraltalk2
Image is CC0 Public domain.

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn a *function* to map x -> y

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

Unsupervised Learning

Data: x

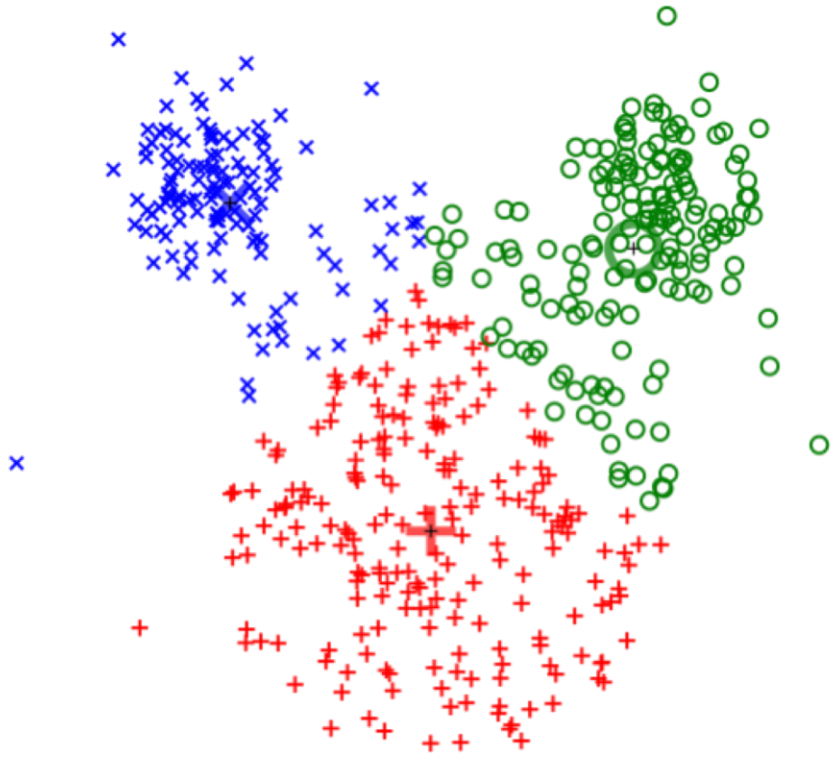
Just data, no labels!

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

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

Supervised vs Unsupervised Learning

Clustering
(e.g. K-Means)



Unsupervised Learning

Data: x

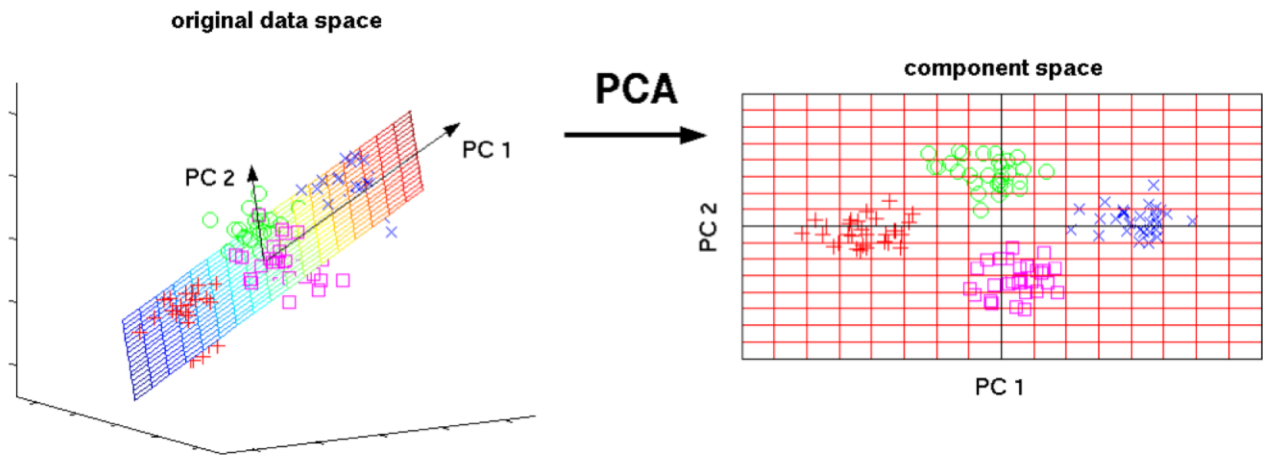
Just data, no labels!

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

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

Supervised vs Unsupervised Learning

Dimensionality Reduction (e.g. Principal Components Analysis)



3D → 2D

This image from Matthias Scholz is CC0 public domain

Unsupervised Learning

Data: x

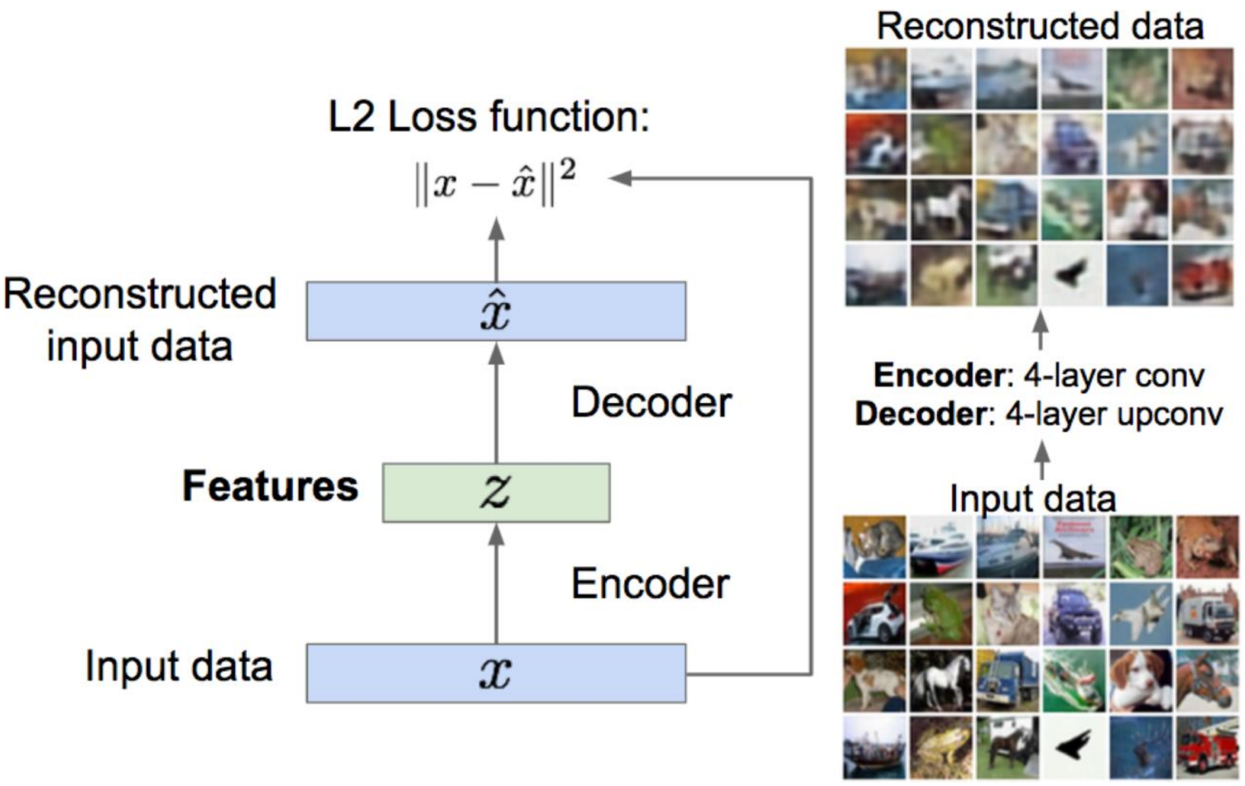
Just data, no labels!

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

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

Supervised vs Unsupervised Learning

Feature Learning
(e.g. autoencoders)



Unsupervised Learning

Data: x

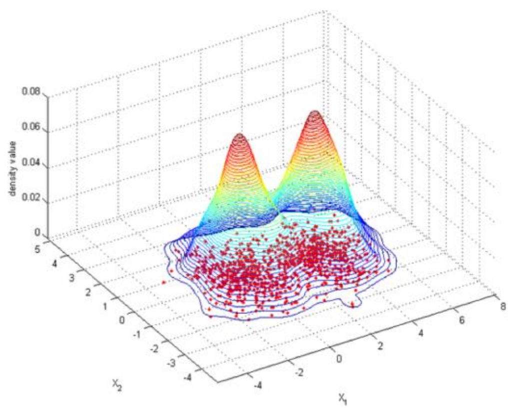
Just data, no labels!

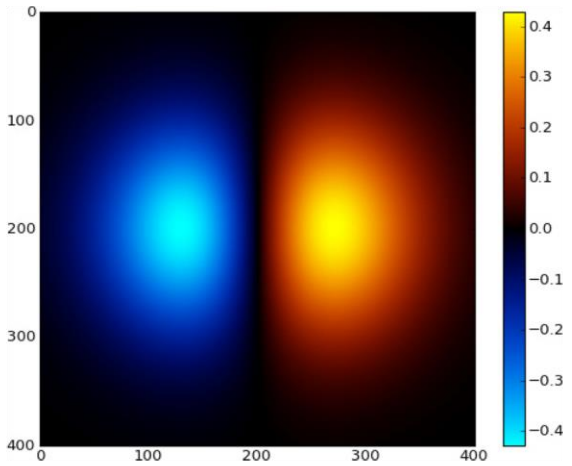
Goal: Learn some underlying hidden *structure* of the data

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

Supervised vs Unsupervised Learning

Density Estimation





Unsupervised Learning

Data: x

Just data, no labels!

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

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

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

x is data, y is label

Goal: Learn a *function* to map x -> y

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

Unsupervised Learning

Data: x

Just data, no labels!

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

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

Discriminative vs Generative Models

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

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

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

Data: x

Label: y

Cat

Discriminative vs Generative Models

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

Generative Model:
Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)

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**:

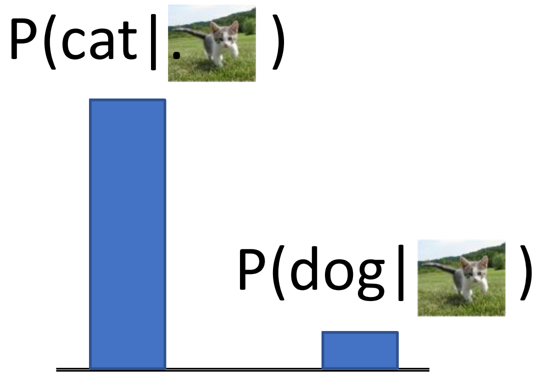
$$\int_X p(x)dx = 1$$

Different values of x **compete** for density

Discriminative vs Generative Models

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

Data: x



Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)

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

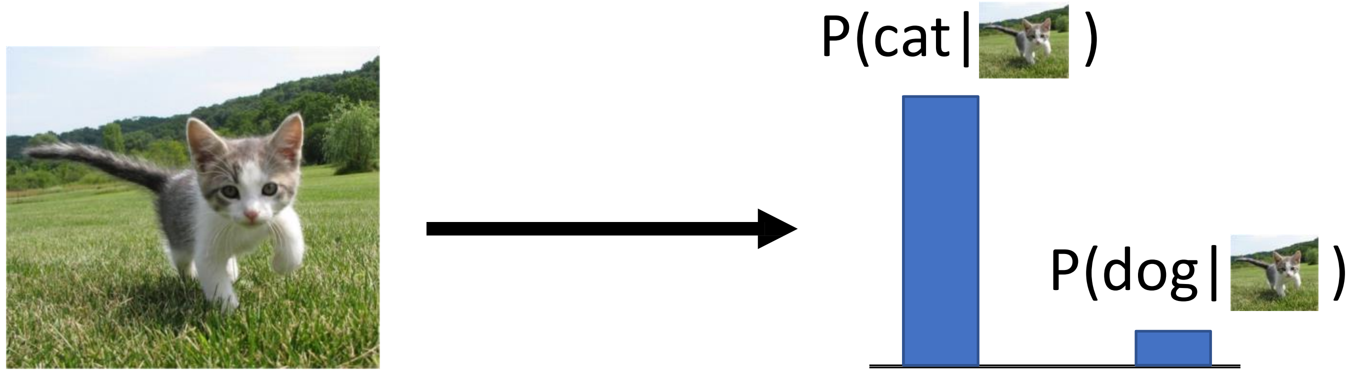
Density functions are **normalized**:

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

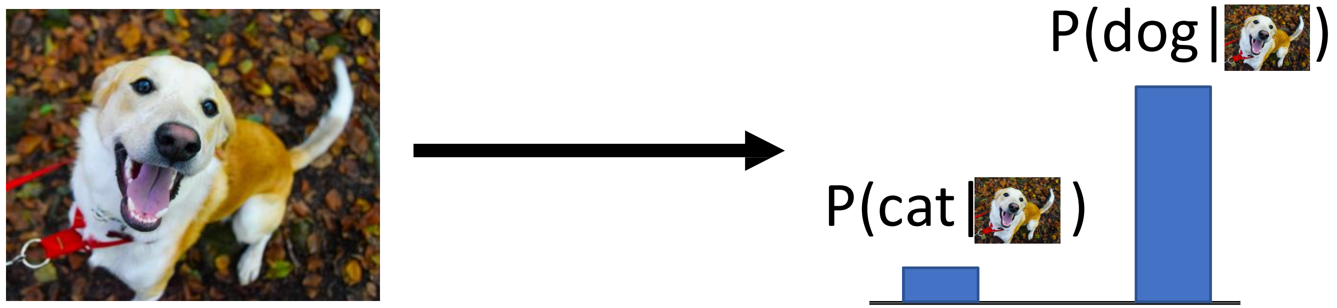
Different values of x **compete** for density

Discriminative vs Generative Models

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



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



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

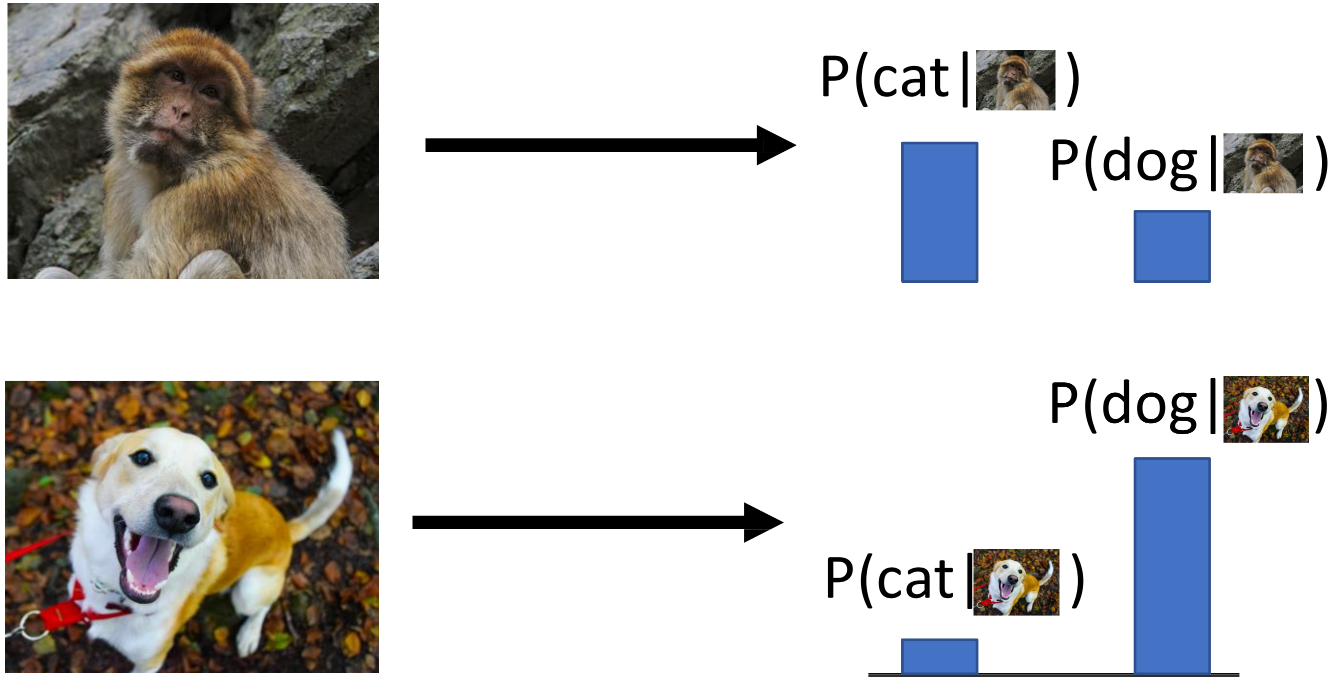
Discriminative model: the possible labels for each input "compete" for probability mass. But no competition between **images**

Discriminative vs Generative Models

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

Generative Model: Learn a probability distribution p(x)

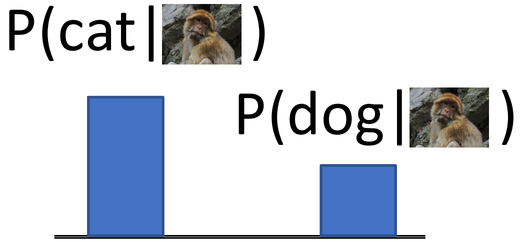
Conditional Generative Model: Learn p(x|y)



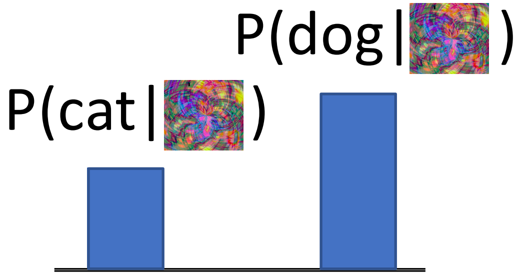
Discriminative model: No way for the model to handle unreasonable inputs; it must give label distributions for all images

Discriminative vs Generative Models

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



Generative Model: Learn a probability distribution p(x)

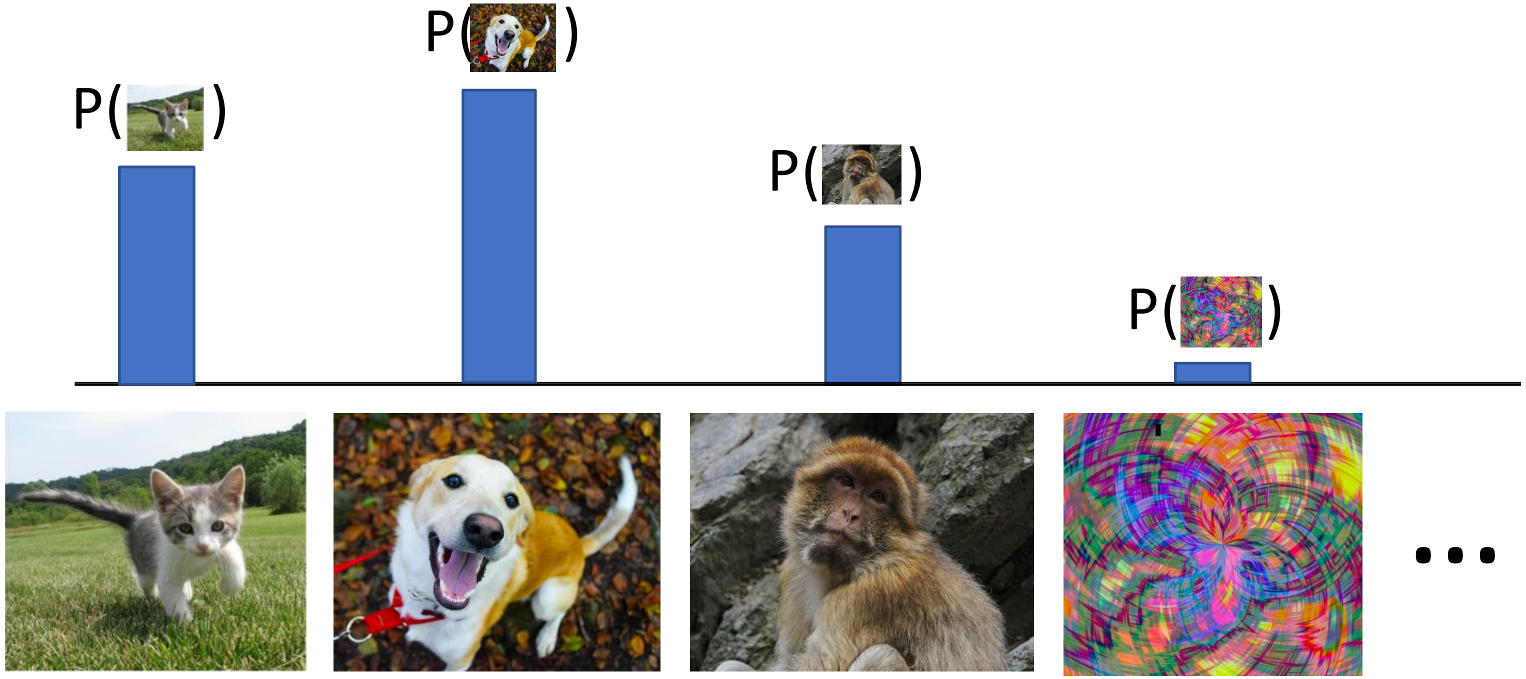


Conditional Generative Model: Learn p(x|y)

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

Discriminative vs Generative Models

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



Generative Model: Learn a probability distribution p(x)

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

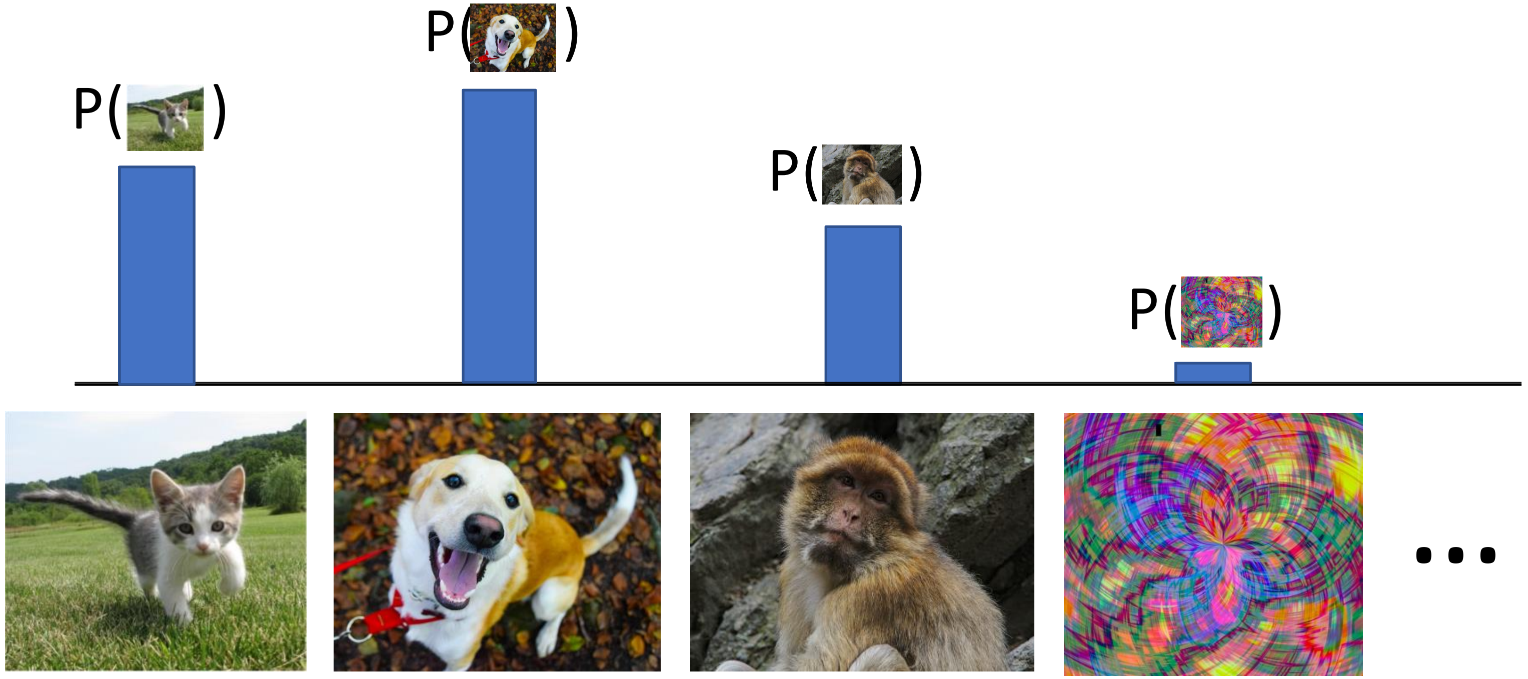
Conditional Generative Model: Learn p(x|y)

Discriminative vs Generative Models

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

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)



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

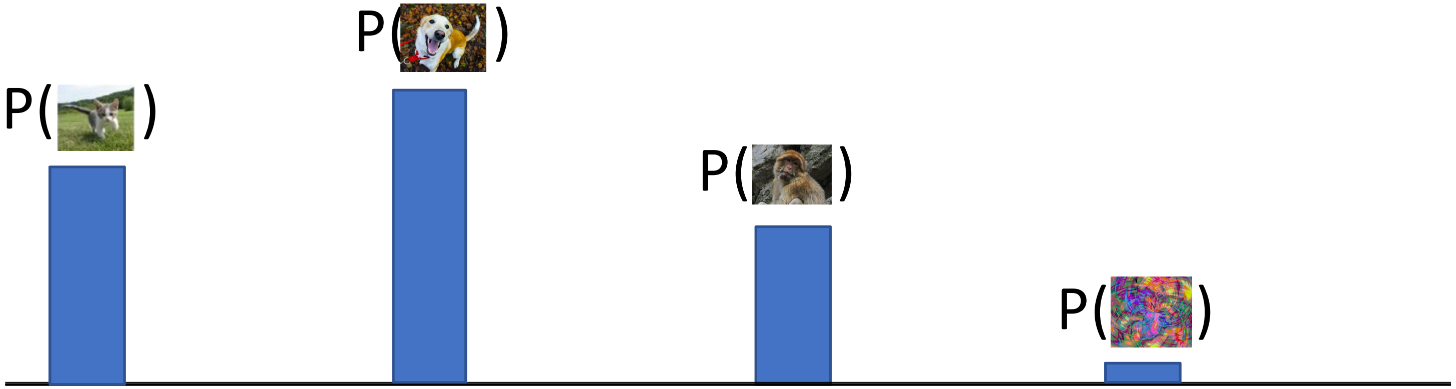
Requires deep image understanding! Is a dog more likely to sit or stand? How about 3-legged dog vs 3-armed monkey?

Discriminative vs Generative Models

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

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)



…

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

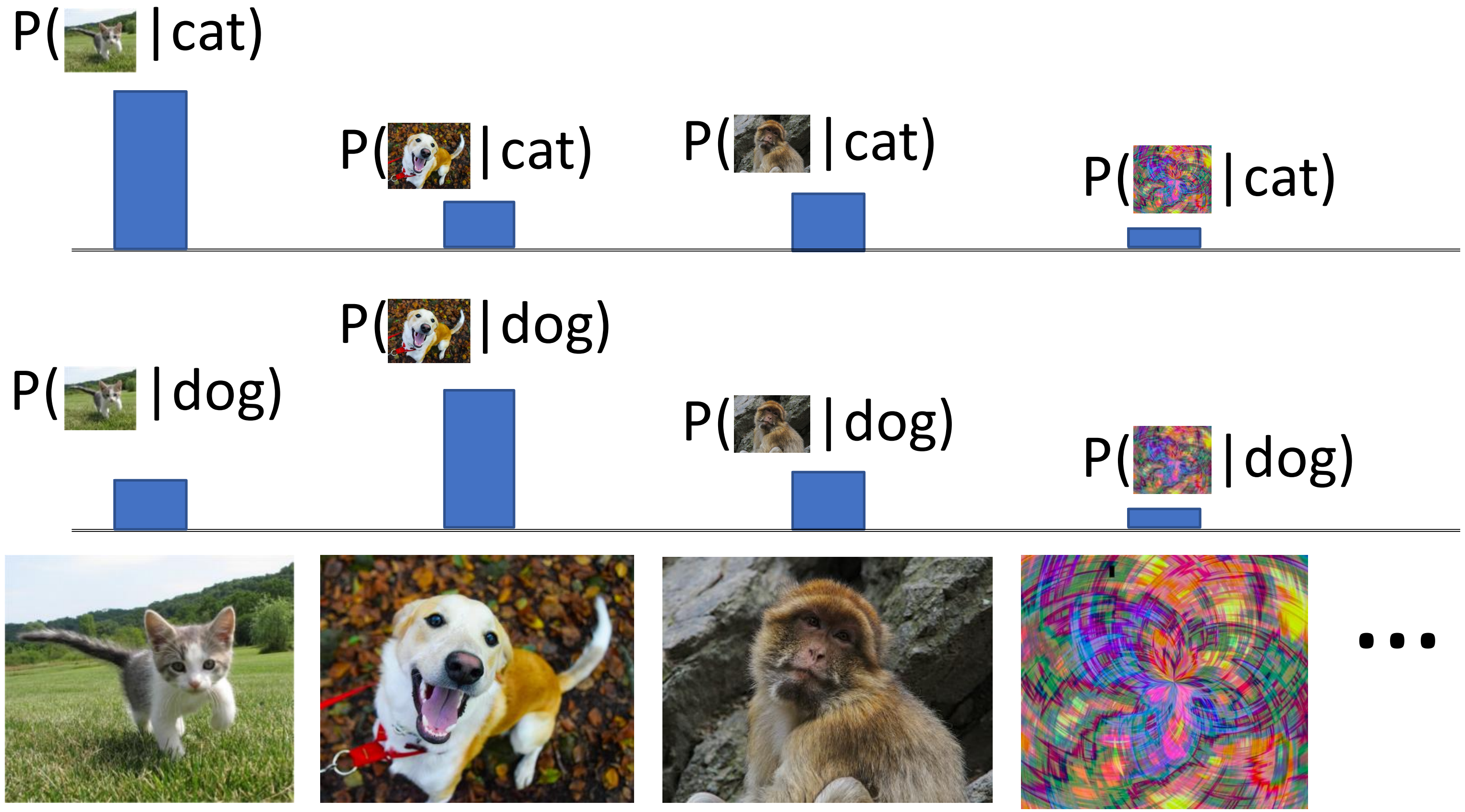
Model can "reject" unreasonable inputs by assigning them small values

Discriminative vs Generative Models

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

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)



Conditional Generative Model: Each possible label induces a competition among all images

Discriminative vs Generative Models

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

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)

Recall **Bayes' Rule:**

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

Discriminative vs Generative Models

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

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)

Recall **Bayes' Rule:**

$$\underbrace{\boxed{P(x \mid y)}}_{\text{Conditional Generative Model}} = \frac{\overbrace{\boxed{P(y \mid x)}}^{\text{Discriminative Model}}}{\underbrace{\boxed{P(y)}}_{\text{Prior over labels}}} \overbrace{\boxed{P(x)}}^{\text{(Unconditional) Generative Model}}$$

We can build a conditional generative model from other components!

What can we do with a discriminative model?

- **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 $p(x|y)$

What can we do with a generative model?

- **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 $p(x|y)$

What can we do with a generative model?

- **Discriminative Model:** Learn a probability distribution p(y|x) → Assign labels to data; Feature learning (supervised)

- **Generative Model**: Learn a probability distribution p(x) → Detect outliers; Feature learning (unsupervised); Sample to **generate** new data

- **Conditional Generative Model:** Learn p(x|y) → Assign labels, while rejecting outliers!; Generate new data conditioned on input labels

Taxonomy of Generative Models

Generative models

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

Taxonomy of Generative Models

Model can compute p(x)

Generative models

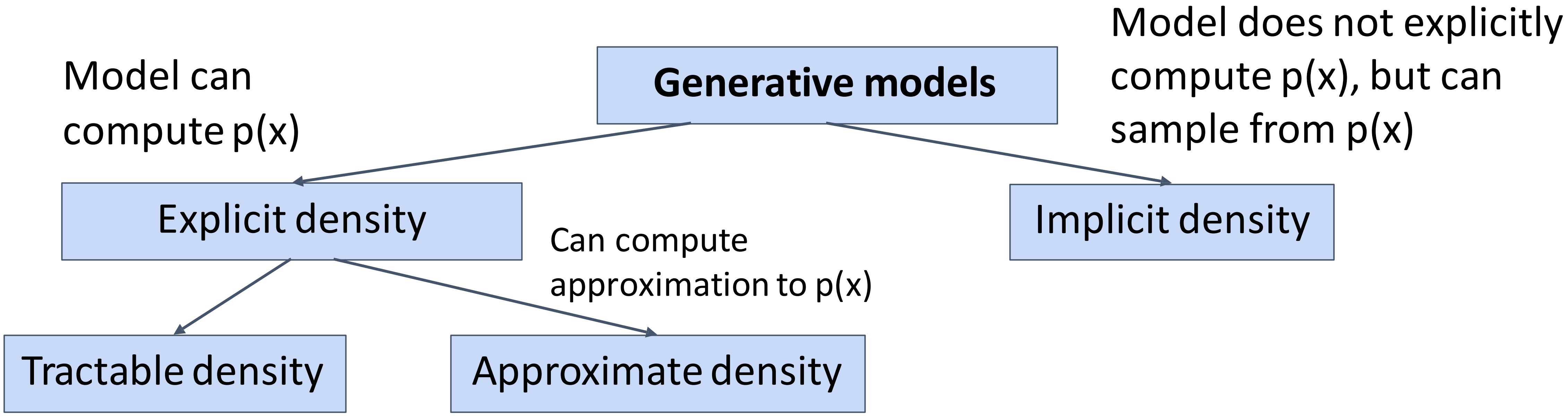
Model does not explicitly compute p(x), but can sample from p(x)

Explicit density

Implicit density

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

Taxonomy of Generative Models



Can compute p(x)

- Autoregressive
- NADE / MADE
- NICE / RealNVP
- Glow
- Ffjord

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

Taxonomy of Generative Models

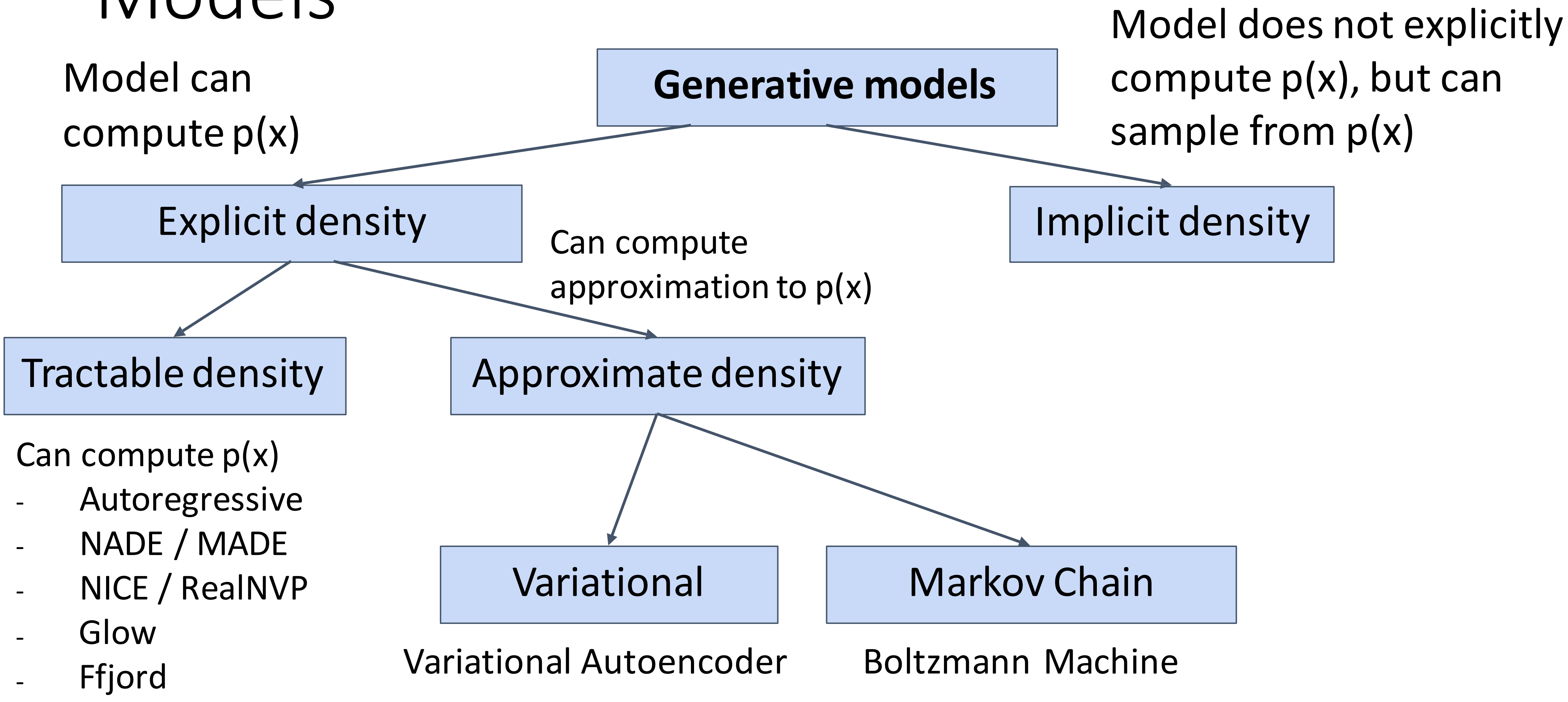


Figure adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Taxonomy of Generative Models

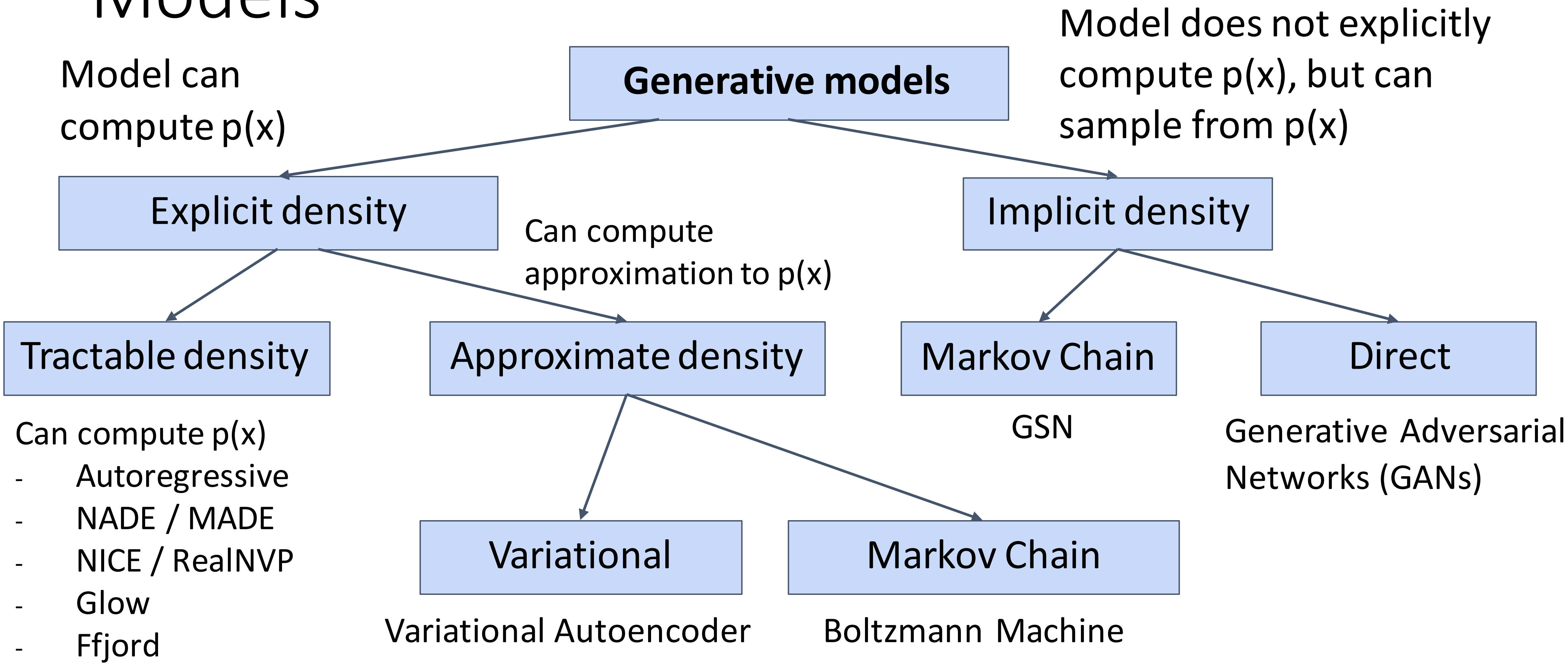


Figure adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Taxonomy of Generative Models

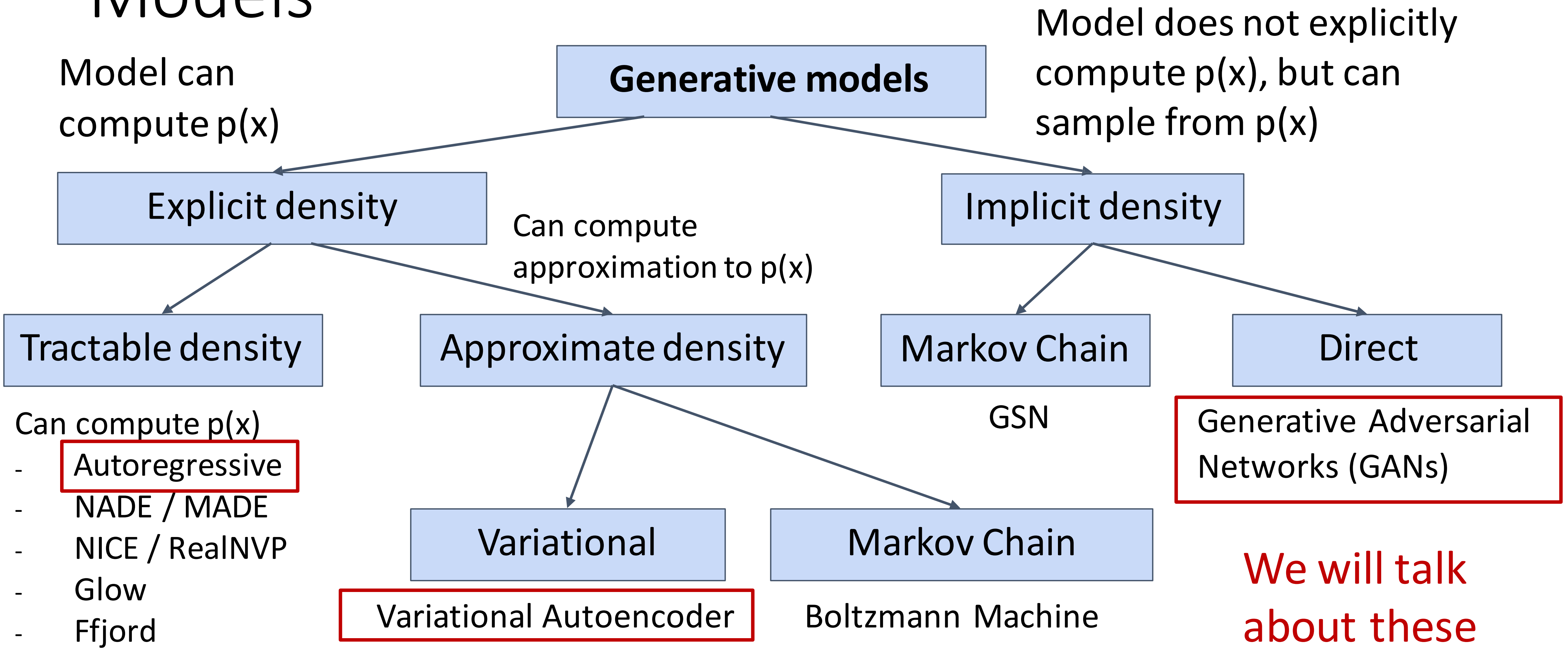


Figure adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Variational Autoencoders

Variational Autoencoders

Typical density models explicitly parameterize density function with a neural network, so we can train to maximize likelihood of training data:

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

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

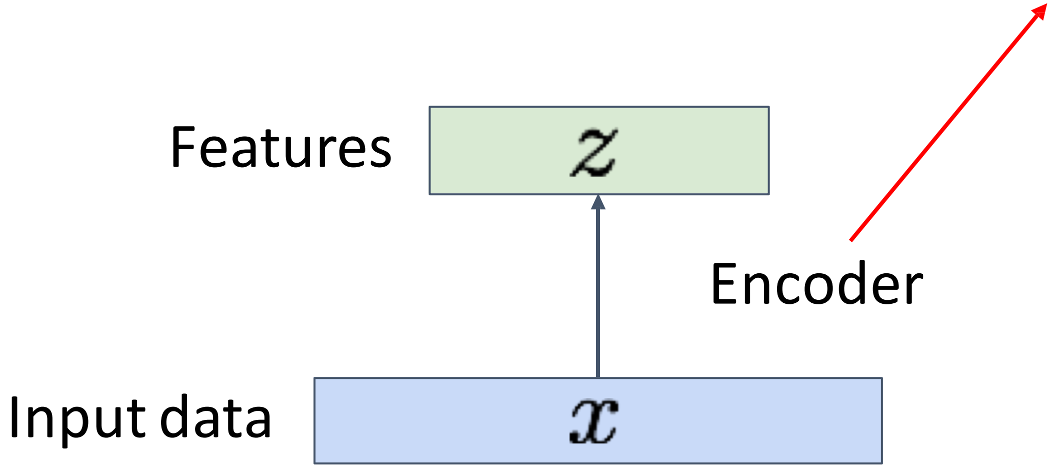
Variational Autoencoders

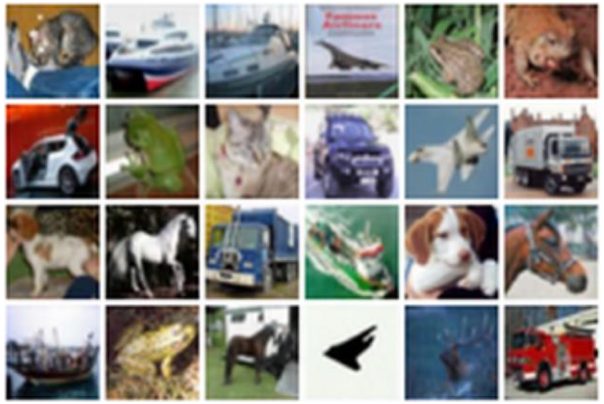
(Regular, non-variational) Autoencoders

Unsupervised method for learning feature vectors from raw data x, without any labels

Features should extract useful information (maybe object identities, properties, scene type, etc) that we can use for downstream tasks

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN





(Regular, non-variational) Autoencoders

Problem: How can we learn this feature transform from raw data?

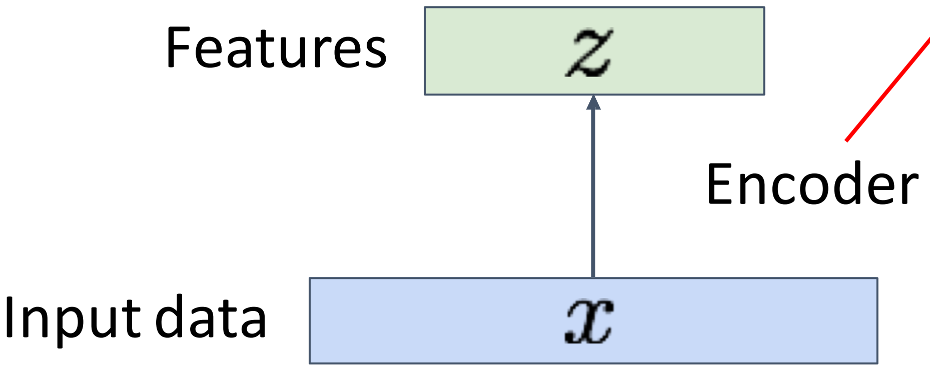
Features should extract useful information (maybe object identities, properties, scene type, etc) that we can use for downstream tasks

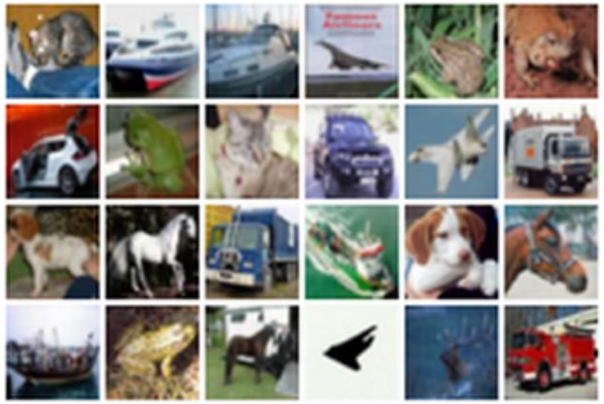
But we can't observe features!

Originally: Linear + nonlinearity (sigmoid)

Later: Deep, fully-connected

Later: ReLU CNN





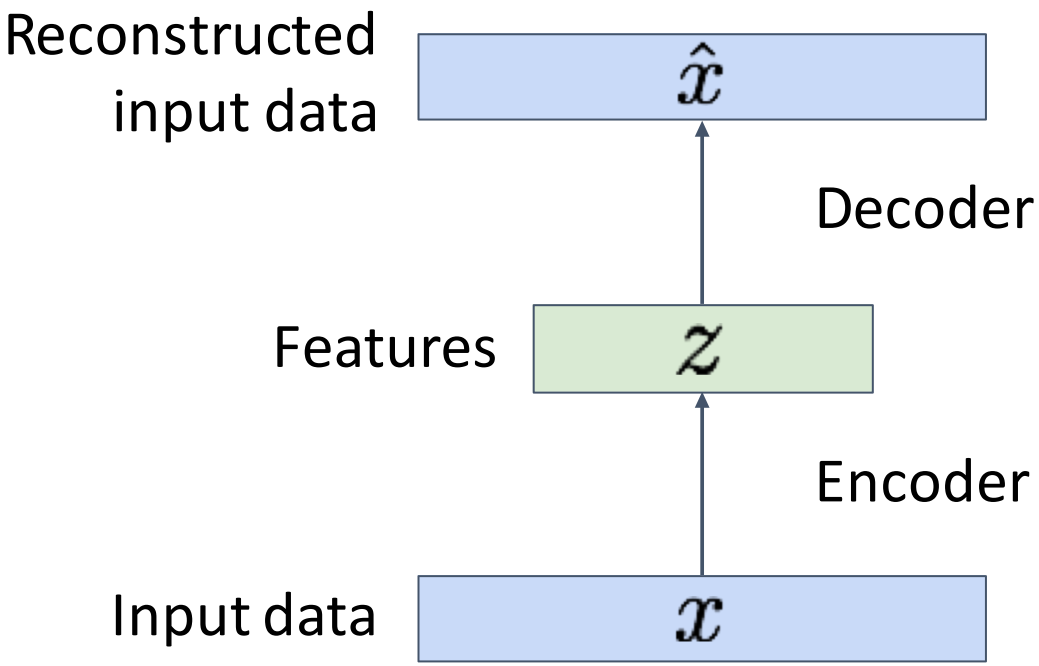
Input Data

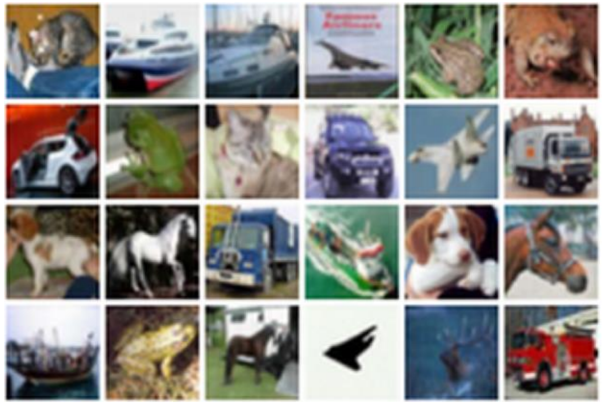
(Regular, non-variational) Autoencoders

- **Problem**: How can we learn this feature transform from raw data?

Idea: Use the features to <u>reconstruct</u> the input data with a **decoder**

"Autoencoding" = encoding itself



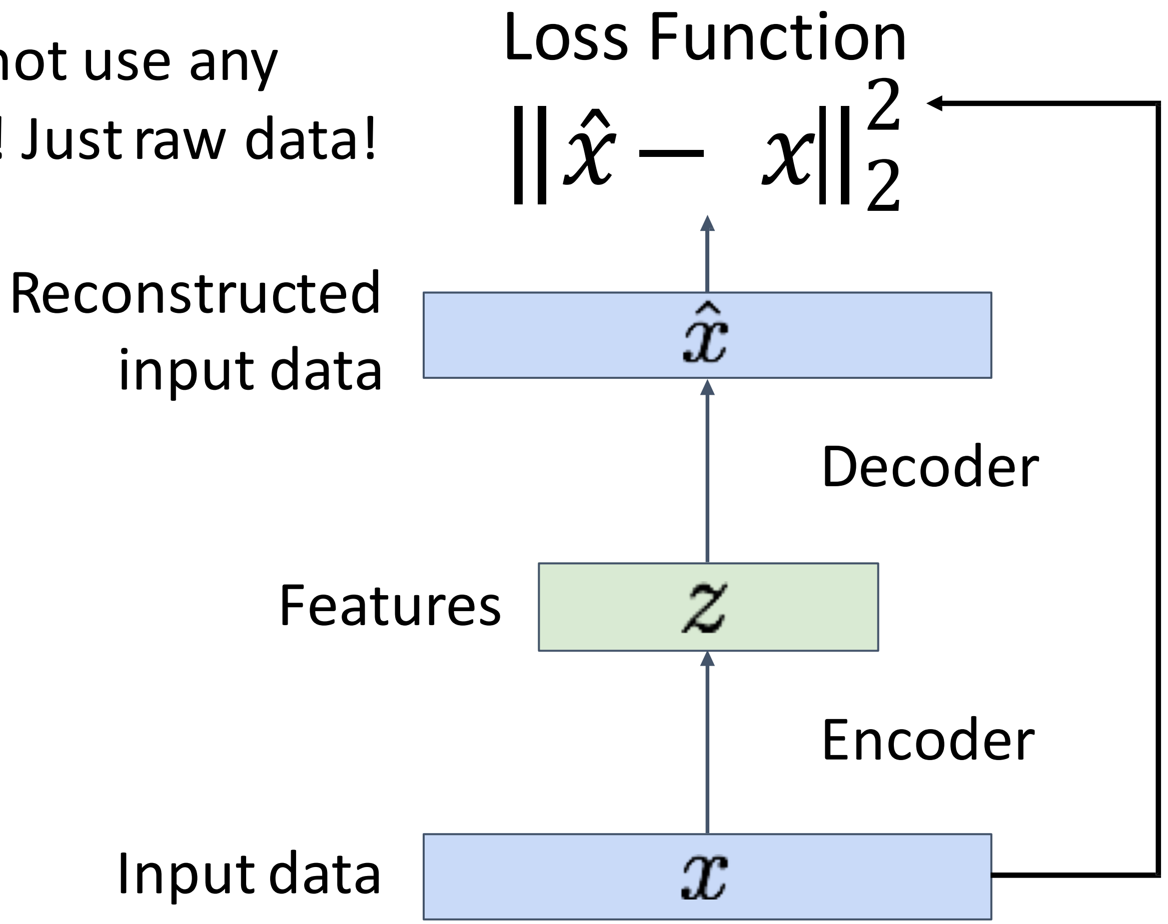


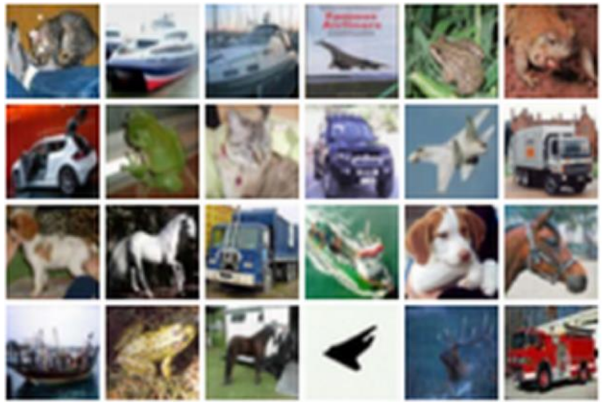
Input Data

(Regular, non-variational) Autoencoders

Loss: L2 distance between input and reconstructed data.

Does not use any labels! Just raw data!



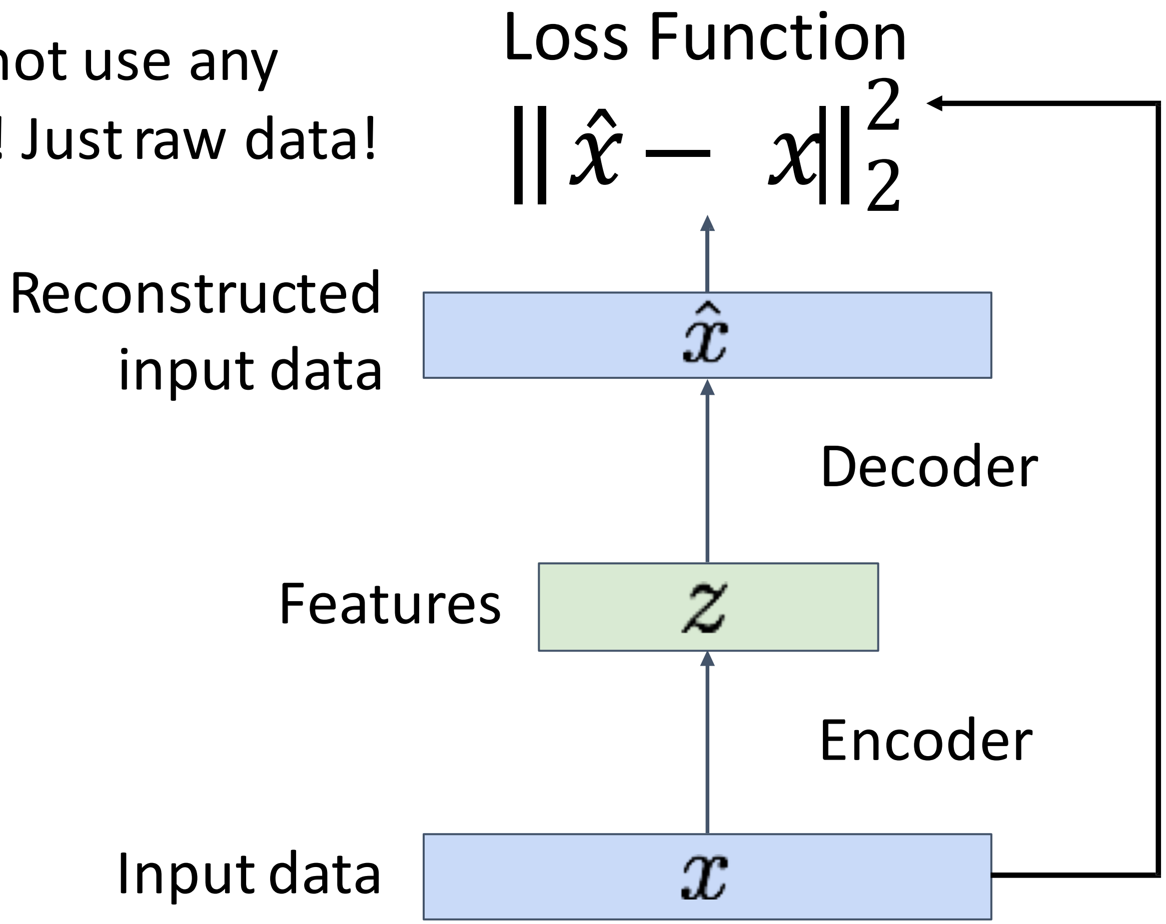


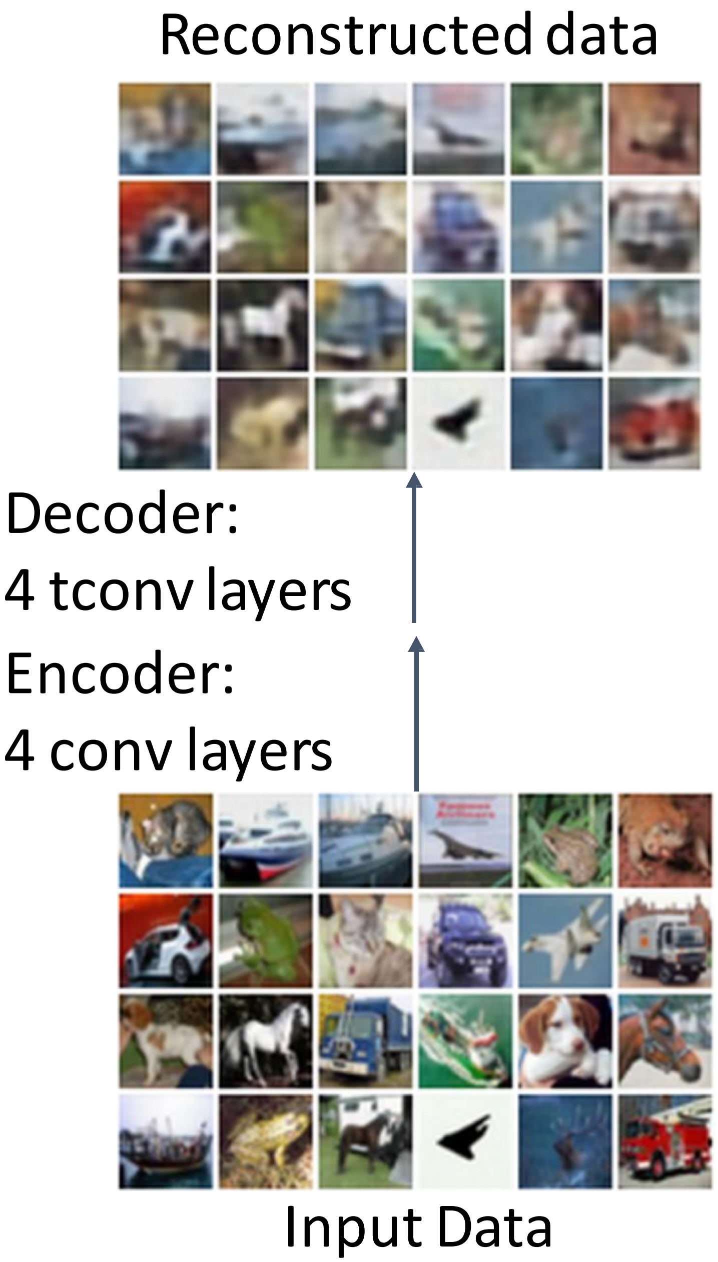
Input Data

(Regular, non-variational) Autoencoders

Loss: L2 distance between input and reconstructed data.

Does not use any labels! Just raw data!

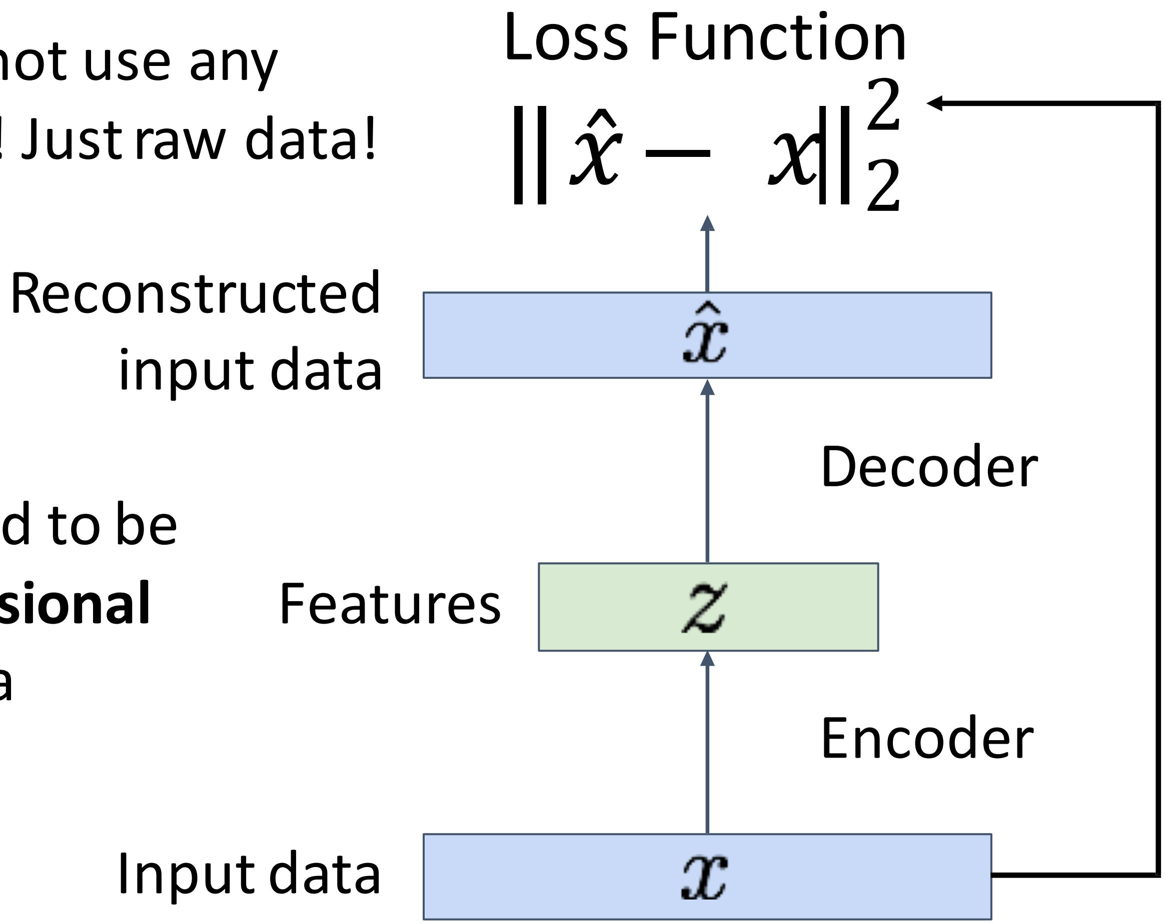




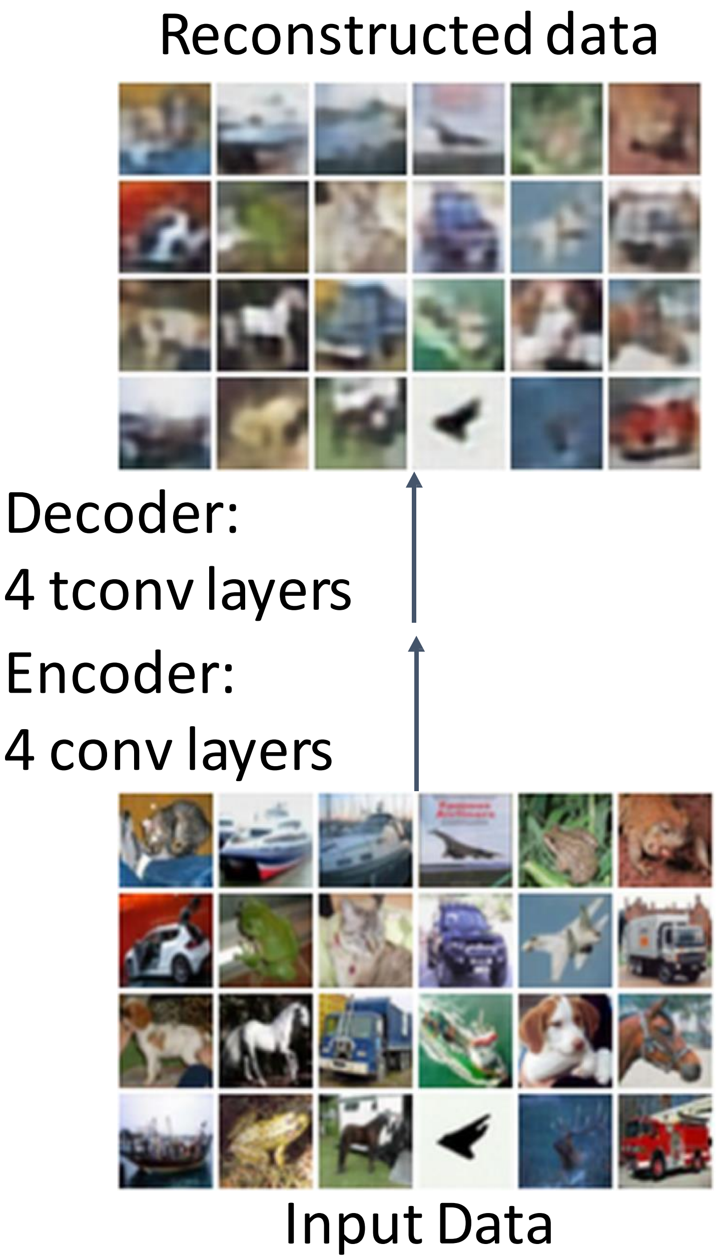
(Regular, non-variational) Autoencoders

Loss: L2 distance between input and reconstructed data.

Does not use any labels! Just raw data!

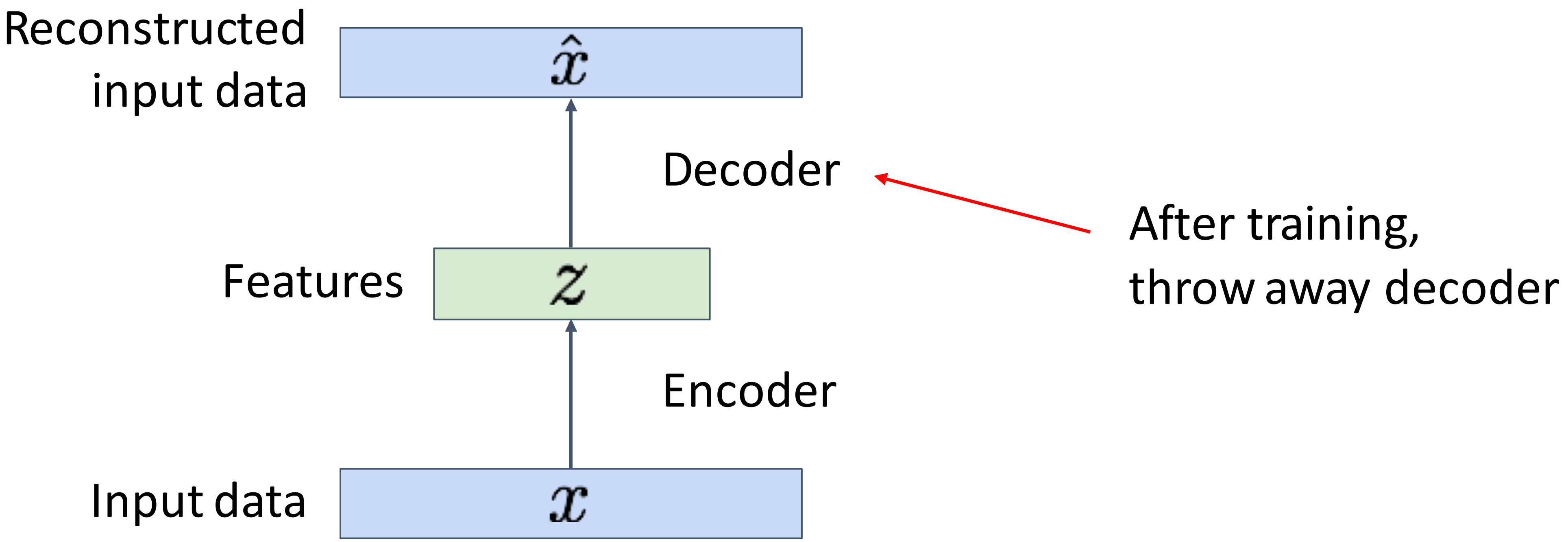


Features need to be **lower dimensional** than the data



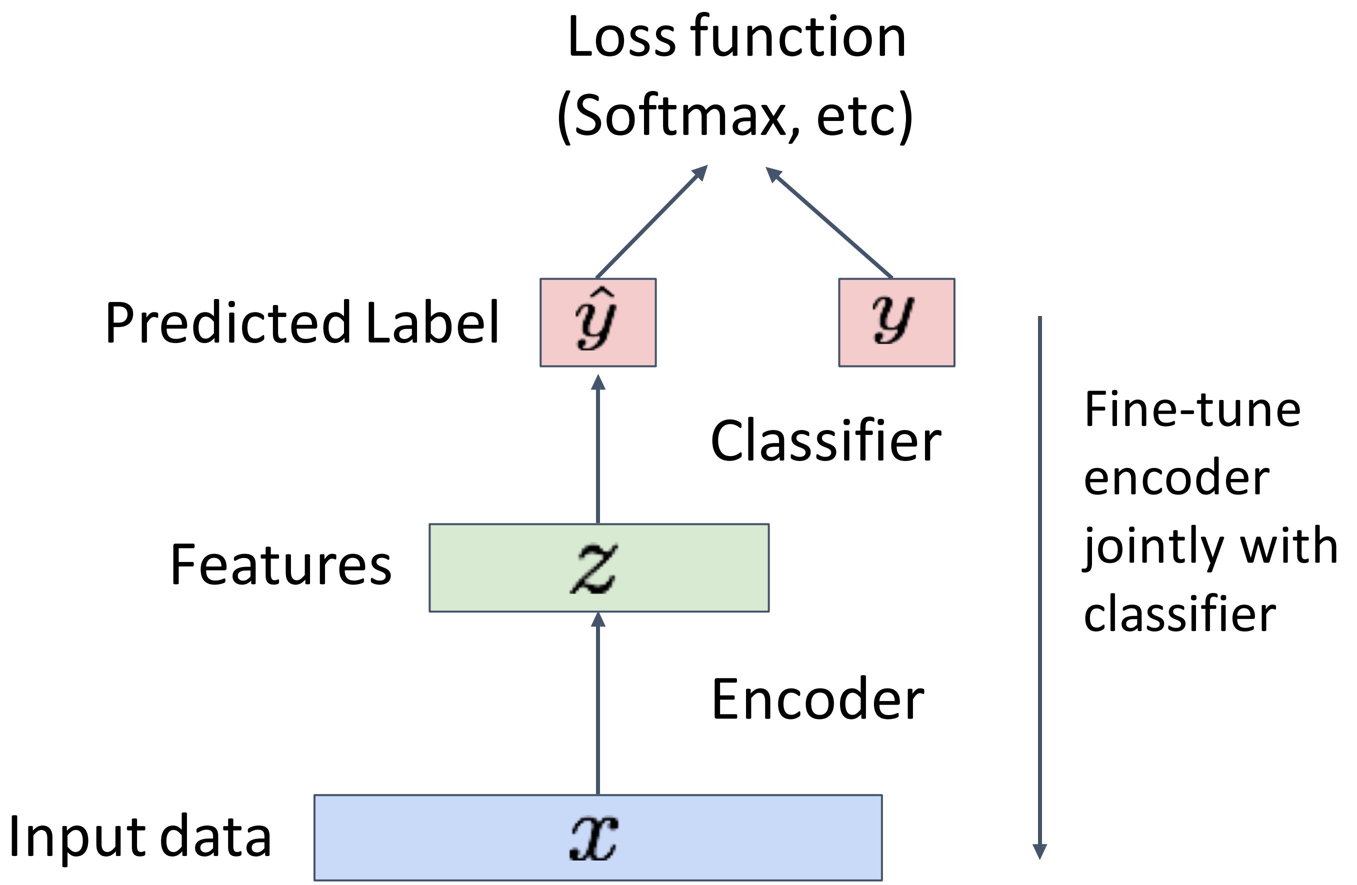
(Regular, non-variational) Autoencoders

After training, **throw away decoder** and use encoder for a downstream task



(Regular, non-variational) Autoencoders

After training, **throw away decoder** and use encoder for a downstream task



Encoder can be used to initialize a **supervised** model

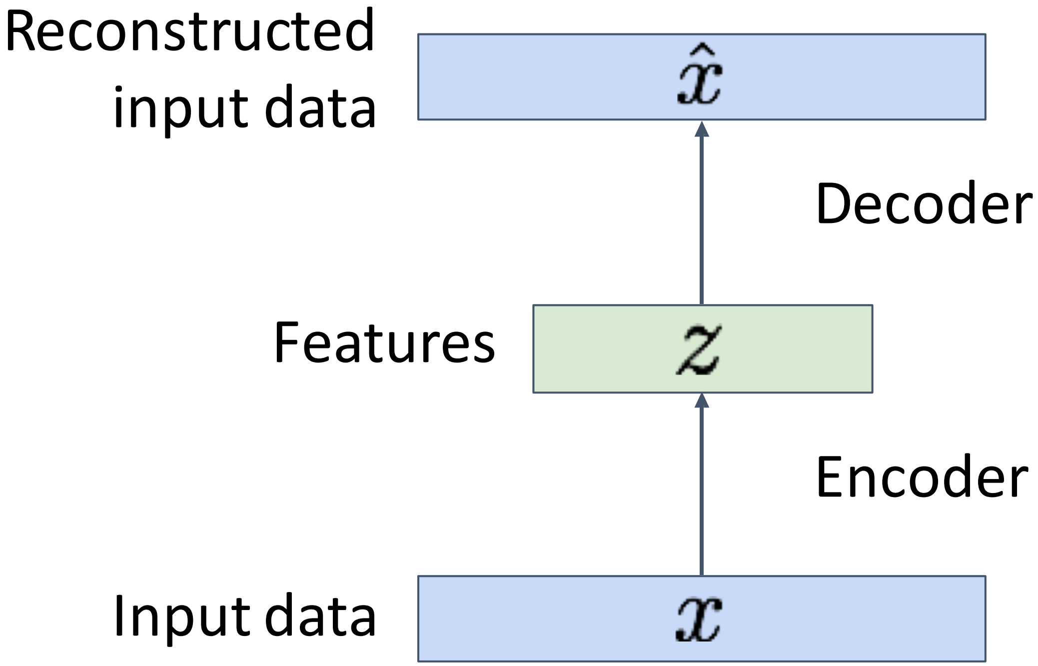
Train for final task (sometimes with small data)

(Regular, non-variational) Autoencoders

Autoencoders learn **latent features** for data without any labels!

Can use features to initialize a **supervised** model

Not probabilistic: No way to sample new data from learned model



Variational Autoencoders

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

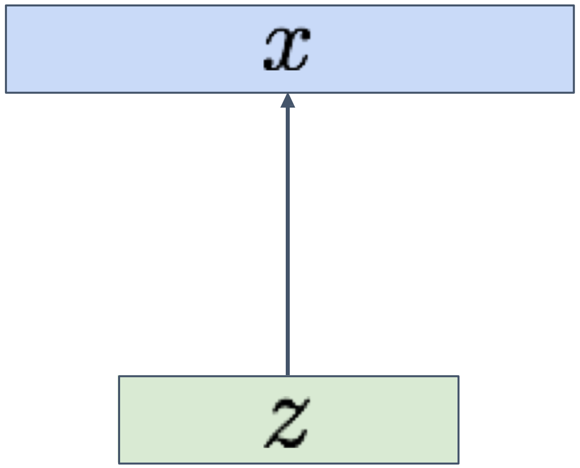
After training, sample new data like this:

Sample from conditional

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

Sample z from prior

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



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

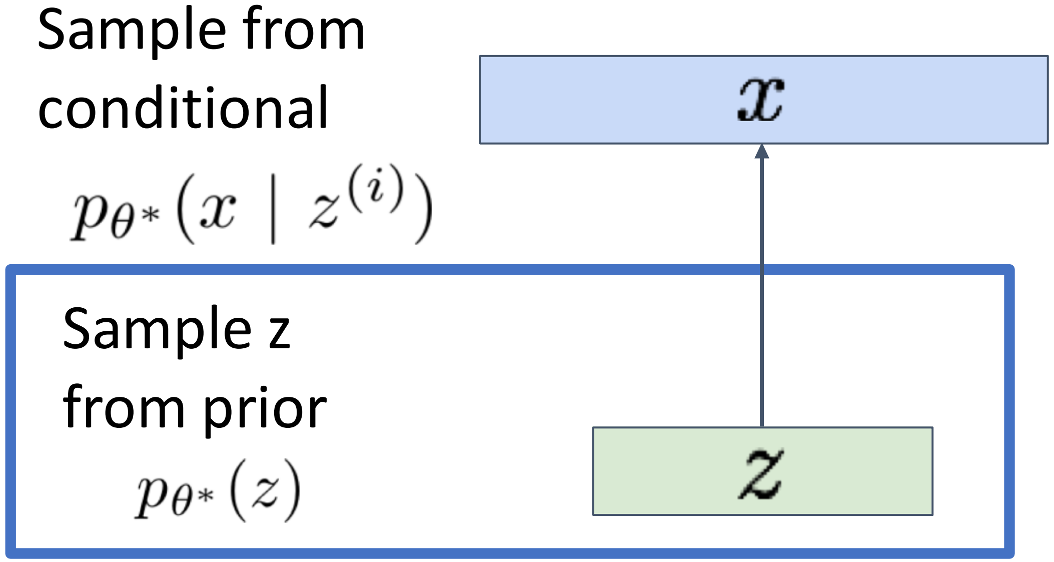
Intuition: x is an image, **z** is latent factors used to generate **x:** attributes, orientation, etc.

Variational Autoencoders

Probabilistic spin on autoencoders:

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

After training, sample new data like this:



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

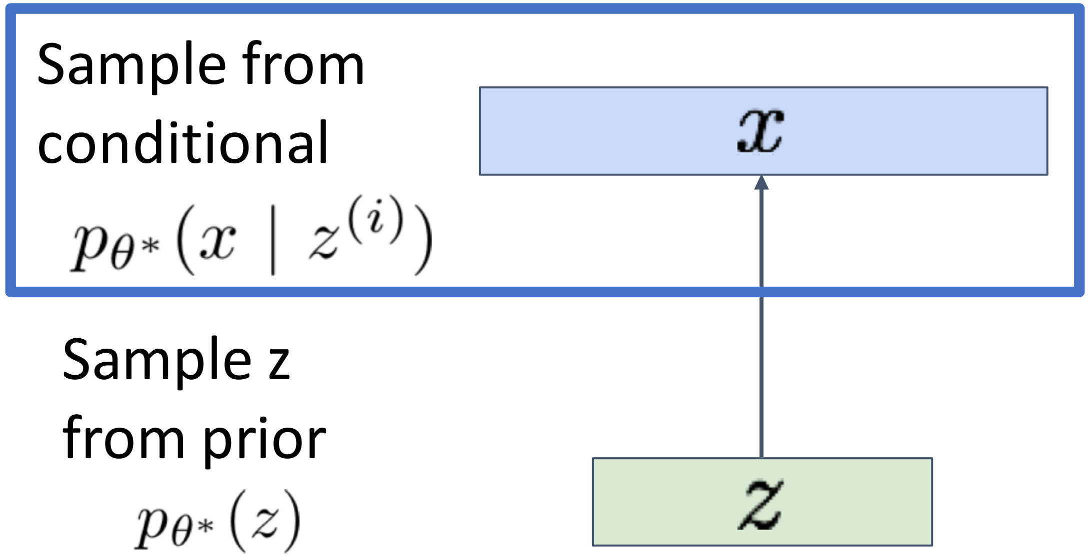
Intuition: x is an image, **z** is latent factors used to generate **x:** attributes, orientation, etc.

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**
- **Intuition: x** is an image, **z** is latent factors used to generate **x:** attributes, orientation, etc.
- Assume simple prior p(z), e.g. Gaussian
- Represent p(x|z) with a neural network (Similar to **decoder** from autencoder)

Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

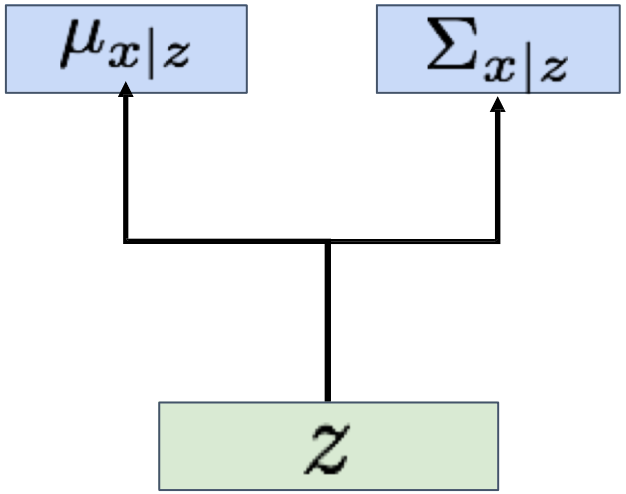
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample from conditional

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

Sample z from prior

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



- 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
- Represent p(x|z) with a neural network (Similar to **decoder** from autencoder)

Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

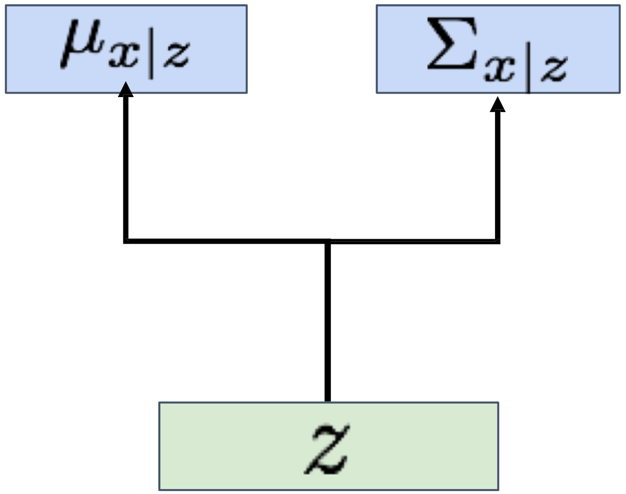
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample from conditional

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

Sample z from prior

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



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

How to train this model?

Basic idea: **maximize likelihood of data**

If we could observe the z for each x, then could train a *conditional generative model* p(x|z)

Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

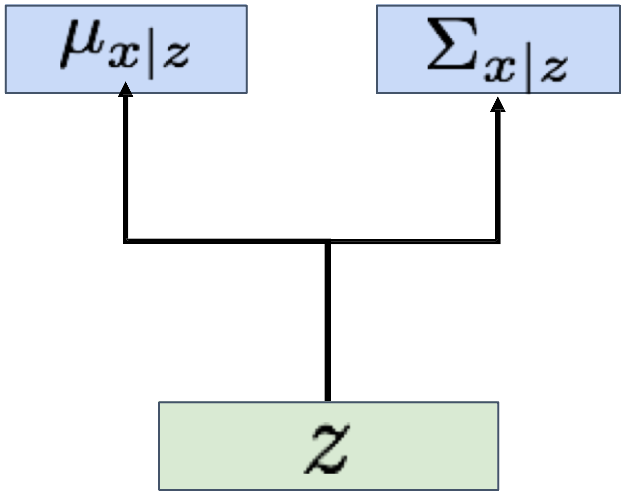
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample from conditional

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

Sample z from prior

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



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

How to train this model?

Basic idea: **maximize likelihood of data**

We don't observe z, so need to marginalize:

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

Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

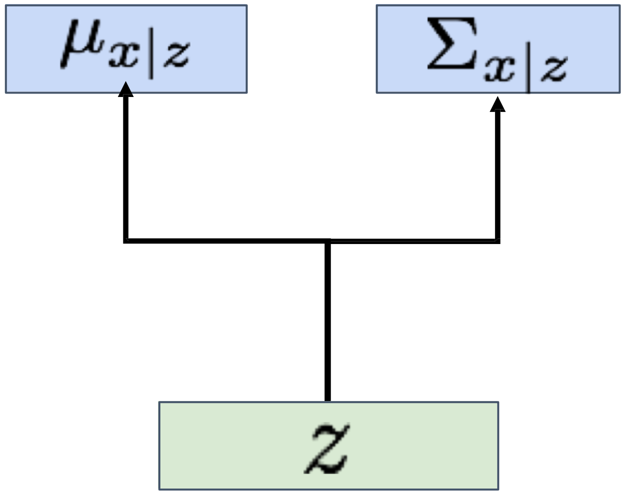
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

Sample from conditional

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

Sample z from prior

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



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

How to train this model?

Basic idea: **maximize likelihood of data**

We don't observe z, so need to marginalize:

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

Ok, can compute this with decoder network

Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

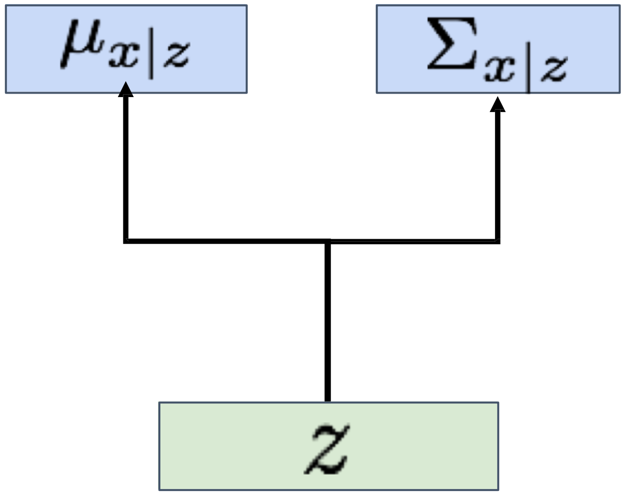
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample from conditional

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

Sample z from prior

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



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

How to train this model?

Basic idea: **maximize likelihood of data**

We don't observe z, so need to 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

Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

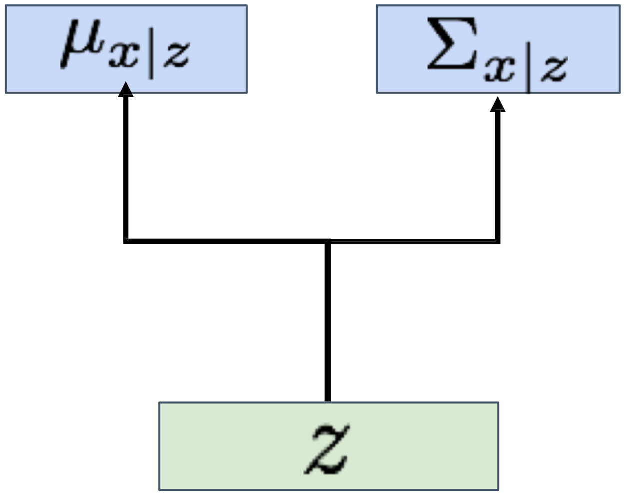
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

Sample from conditional

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

Sample z from prior

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



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

How to train this model?

Basic idea: **maximize likelihood of data**

We don't observe z, so need to marginalize:

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

Problem: Impossible to integrate over all z!

Variational Autoencoders

Recall $p(x, z) = p(x \mid z)p(z) = p(z \mid x)p(x)$

Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

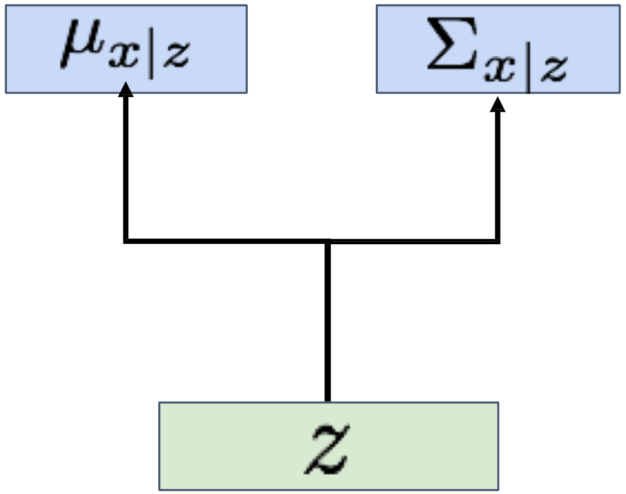
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample from conditional

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

Sample z from prior

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



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

How to train this model?

Basic idea: **maximize likelihood of data**

Another idea: Try Bayes' Rule:

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

Recall $p(x, z) = p(x \mid z)p(z) = p(z \mid x)p(x)$

Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

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

How to train this model?

Basic idea: **maximize likelihood of data**

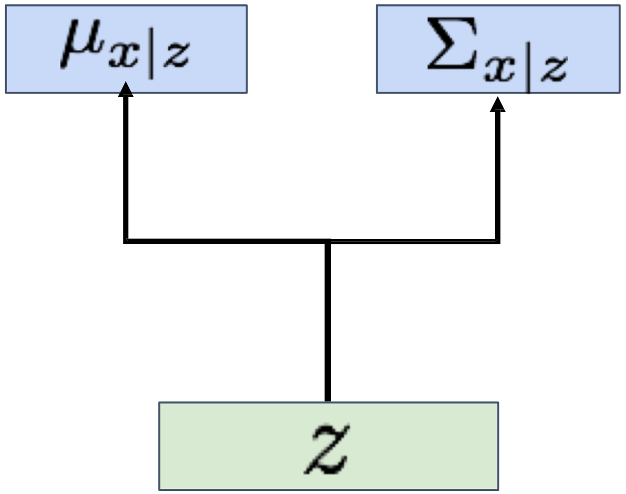
Another idea: Try Bayes' Rule:

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

Ok, compute with decoder network

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

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



Recall $p(x,z) = p(x \mid z)p(z) = p(z \mid x)p(x)$

Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

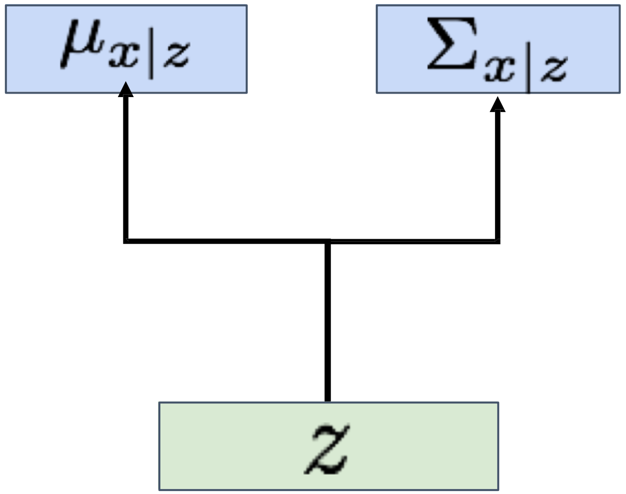
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample from conditional

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

Sample z from prior

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



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

How to train this model?

Basic idea: **maximize likelihood of data**

Another idea: Try Bayes' Rule:

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

Ok, we assumed Gaussian prior

Variational Autoencoders

Recall $p(x, z) = p(x \mid z)p(z) = p(z \mid x)p(x)$

Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

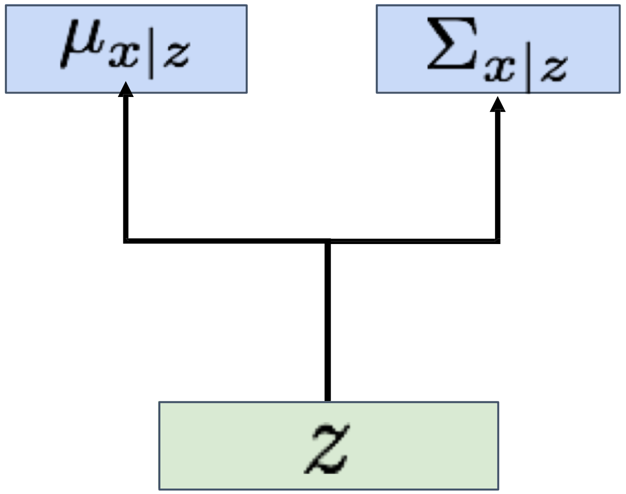
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample from conditional

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

Sample z from prior

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



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

How to train this model?

Basic idea: **maximize likelihood of data**

Another idea: Try Bayes' Rule:

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

Problem: No way to compute this!

Recall $p(x, z) = p(x \mid z)p(z) = p(z \mid x)p(x)$

Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

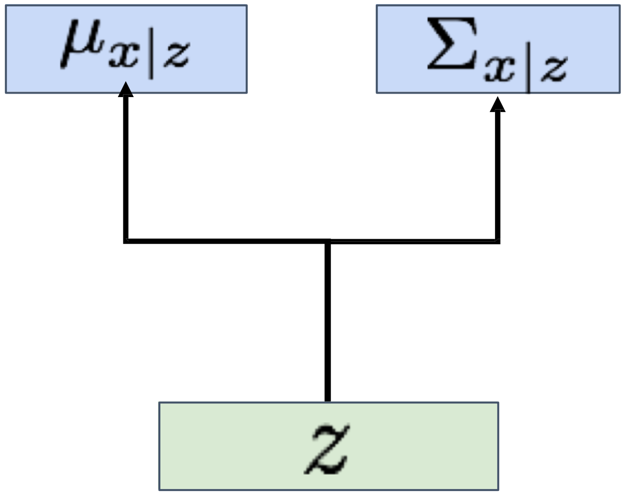
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

Sample from conditional

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

Sample z from prior

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



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

How to train this model?

Basic idea: **maximize likelihood of data**

Another idea: Try Bayes' Rule:

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

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

Recall $p(x, z) = p(x \mid z)p(z) = p(z \mid x)p(x)$

Variational Autoencoders

Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

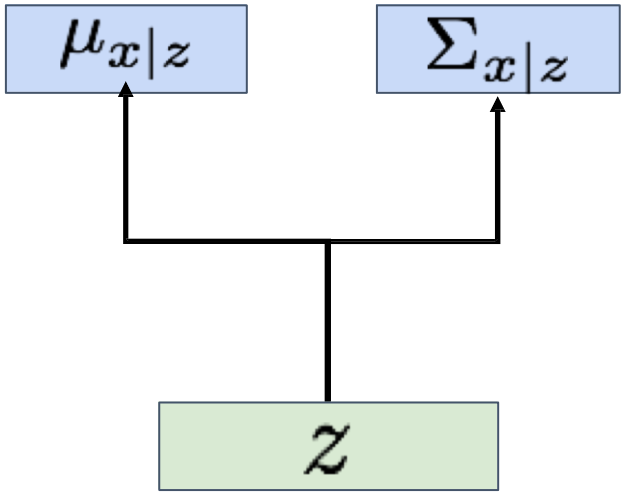
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\Sigma_{x|z}$

Sample from conditional

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

Sample z from prior

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



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

How to train this model?

Basic idea: **maximize likelihood of data**

Another idea: Try Bayes' Rule:

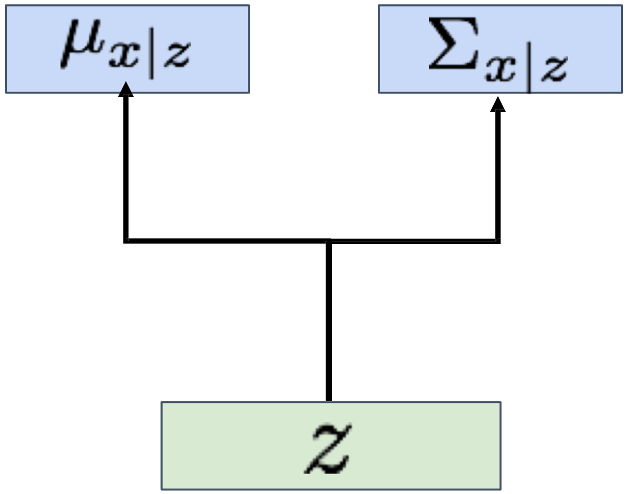
$$p_\theta(x) = \frac{p_\theta(x \mid z)p_\theta(z)}{p_\theta(z \mid x)} \approx \frac{p_\theta(x \mid z)p_\theta(z)}{\boxed{q_\phi(z \mid x)}}$$

Use **encoder** to compute $q_\phi(z \mid x) \approx p_\theta(z \mid x)$

Variational Autoencoders

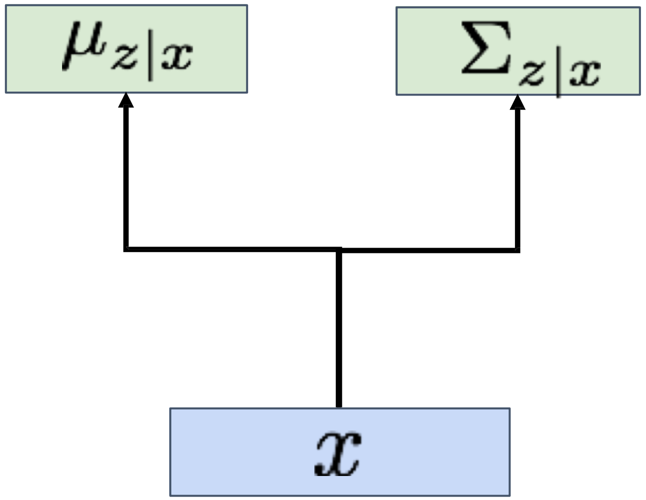
Decoder network inputs latent code z, gives distribution over data x

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



Encoder network inputs data x, gives distribution over latent codes z

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



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

Variational Autoencoders

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

Bayes' Rule

Variational Autoencoders

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

Multiply top and bottom by $q_{\Phi}(z|x)$

Variational Autoencoders

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

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

Split up using rules for logarithms

Variational Autoencoders

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

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

Split up using rules for logarithms

Variational Autoencoders

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

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

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

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

Variational Autoencoders

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

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

$$\log p_{\theta}(x) = E_{\mathrm{z} \sim \mathrm{q}_{\phi}(\mathrm{z}|\mathrm{x})}\,[\log p_{\theta}(x)]$$

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

Variational Autoencoders

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

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

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

Data reconstruction

Variational Autoencoders

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

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

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

KL divergence between prior, and samples from the encoder network

Variational Autoencoders

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

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

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

KL divergence between encoder and posterior of decoder

Variational Autoencoders

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

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

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

KL is >= 0, so dropping this term gives a **lower bound** on the data likelihood:

Variational Autoencoders

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

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

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

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

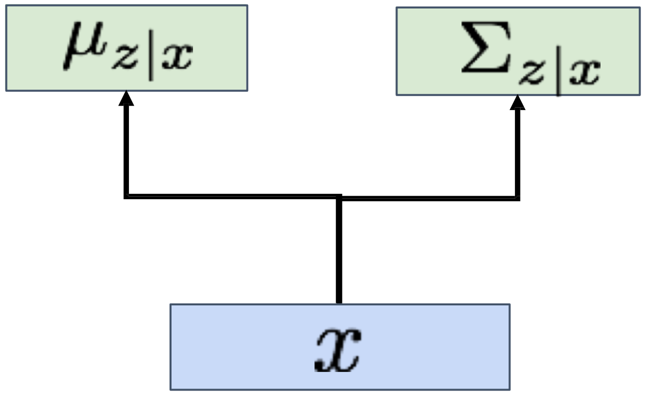
Variational Autoencoders

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

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

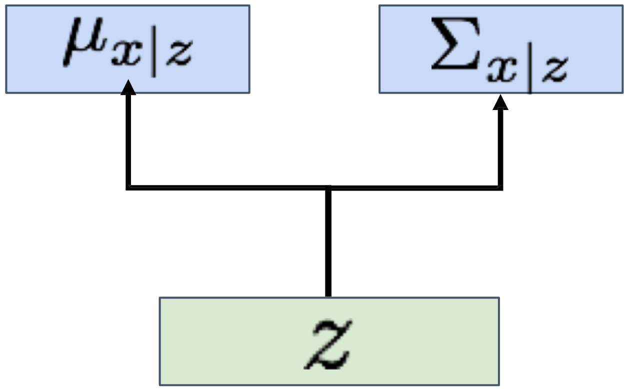
Encoder Network

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



Decoder Network

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



Generative Adversarial Nets

- Review of the GAN:

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

- A two-player game: G wants to fool D by creating an approximant distribution q close to the true distribution p_{data}, and D wants to be smart so that it can distinguish between p_{data} and q induced by G.