# CS 4644-DL / 7643-A: LECTURE 18
# DANFEI XU

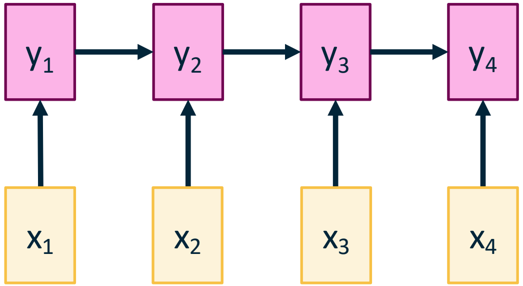
Generative Models:

PixelCNN / PixelRNN

Variational AutoEncoders (VAEs)

# Recap: Three Ways of Processing Sequences

## Recurrent Neural Network

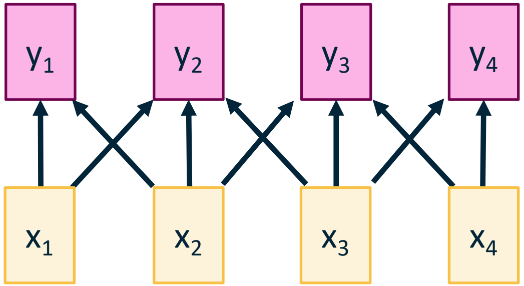



Works on **Ordered Sequences**

(+) Good at long sequences: After one RNN layer, $h_T$ "sees" the whole sequence

**(-) Not parallelizable: need to compute hidden states sequentially**

## 1D Convolution

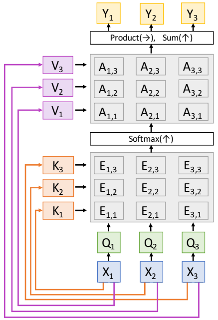



Works on **Multidimensional Grids**

**(-) Bad at long sequences: Need to stack many conv layers for outputs to "see" the whole sequence**

(+) Highly parallel: Each output can be computed in parallel

## Self-Attention




Works on **Sets of Vectors**

(+) Good at long sequences: after one self-attention layer, each output "sees" all inputs!

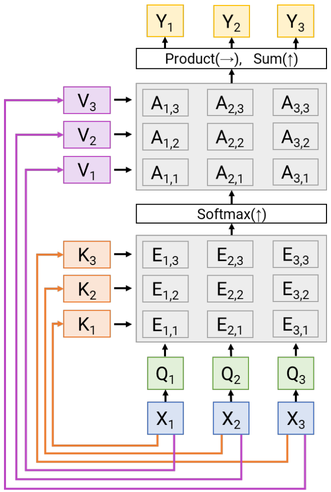
(+) Highly parallel: Each output can be computed in parallel

**(-) Very memory intensive**

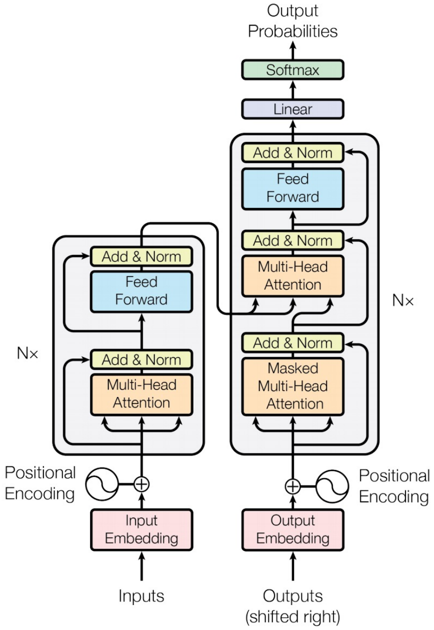
Slide credit: Justin Johnson

# Recap: Transformer

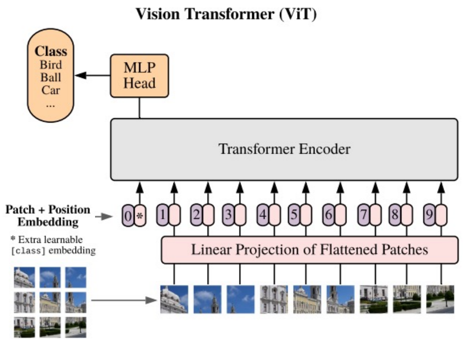
## Self-Attention




## Transformer Model




## Beyond Language



# 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.

# 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

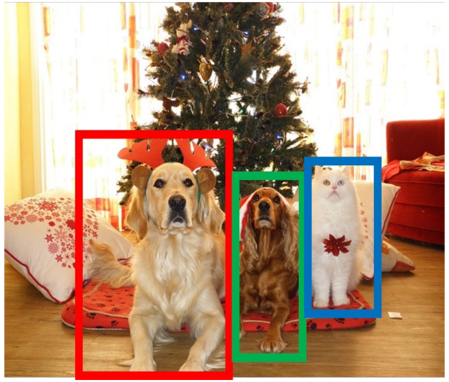
# Supervised vs Unsupervised Learning

**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

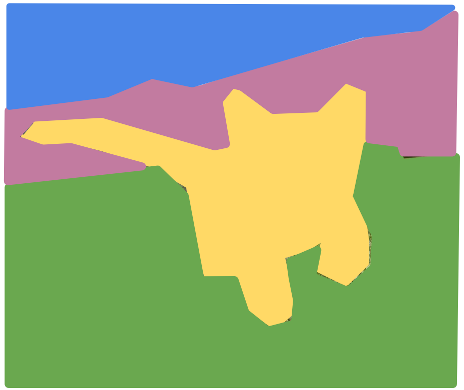
# Supervised vs Unsupervised Learning

**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.




Semantic Segmentation

# Supervised vs Unsupervised Learning

**Unsupervised Learning**

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

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

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

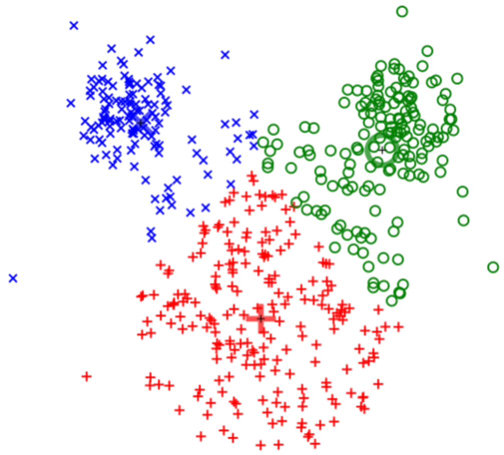
# Supervised vs Unsupervised Learning

**Unsupervised Learning**

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

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

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



K-means clustering

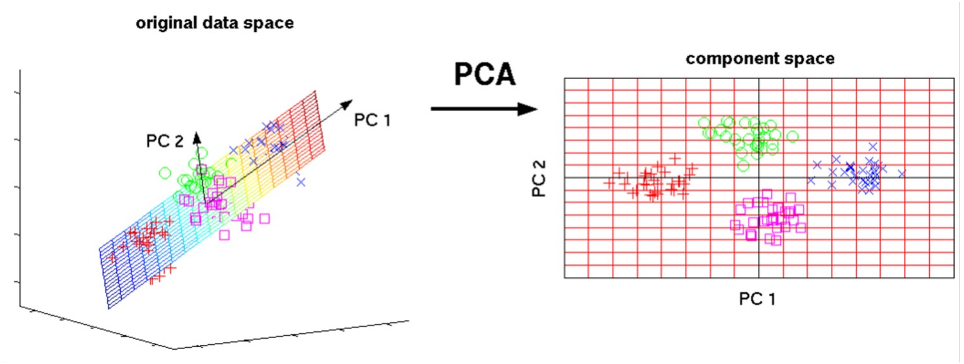
# Supervised vs Unsupervised Learning

## Unsupervised Learning

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

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

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




3-d → 2-d

Principal Component Analysis
(Dimensionality reduction)

# Supervised vs Unsupervised Learning

## Unsupervised Learning

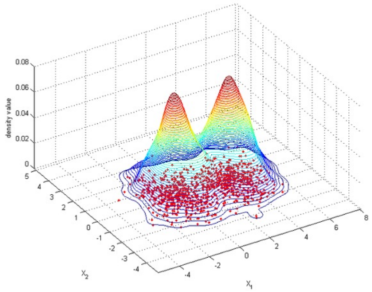
**Data**: x
Just data, no labels!

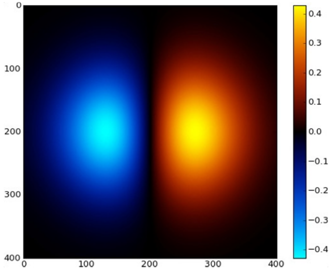
**Goal**: Learn some underlying hidden *structure* of the data

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



1-d density estimation






2-d density estimation

Modeling p(x)

# Supervised vs Unsupervised Learning

**Supervised Learning**

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

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

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

**Unsupervised Learning**

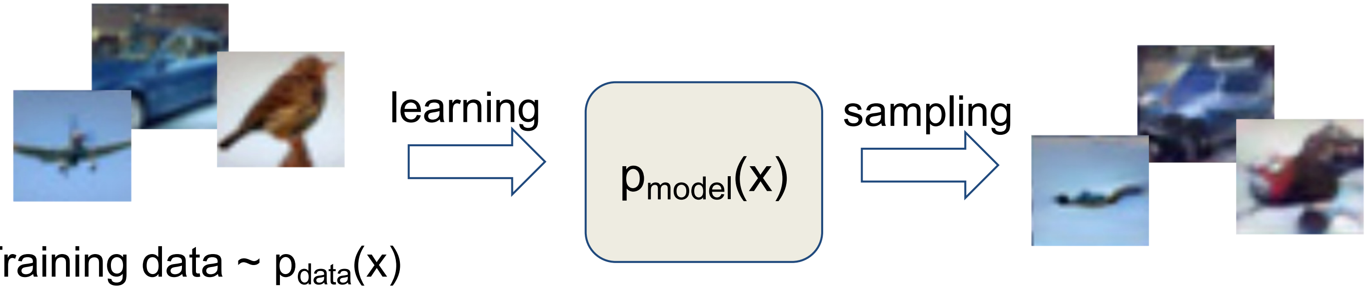
**Data**: x
Just data, no labels!

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

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

# Generative Modeling

Given training data, generate new samples from same distribution

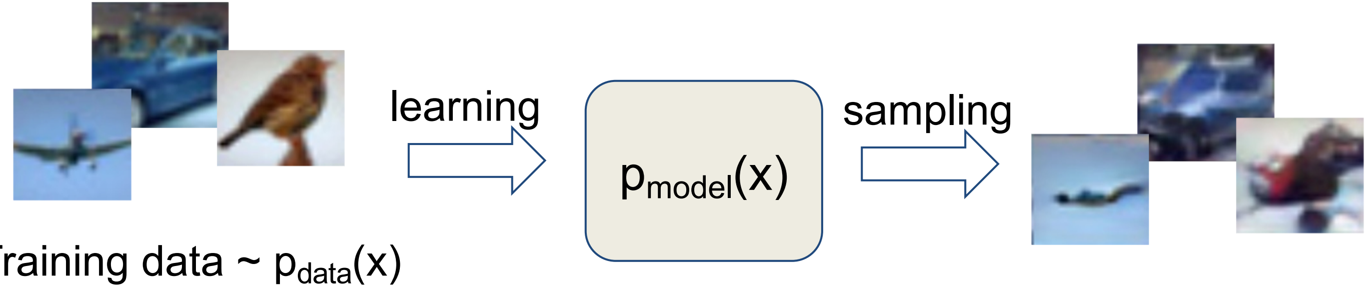



Objectives:

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

# Generative Modeling

Given training data, generate new samples from same distribution




Formulate as density estimation problems**:**

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

# Why Generative Models?



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

# Taxonomy of Generative Models

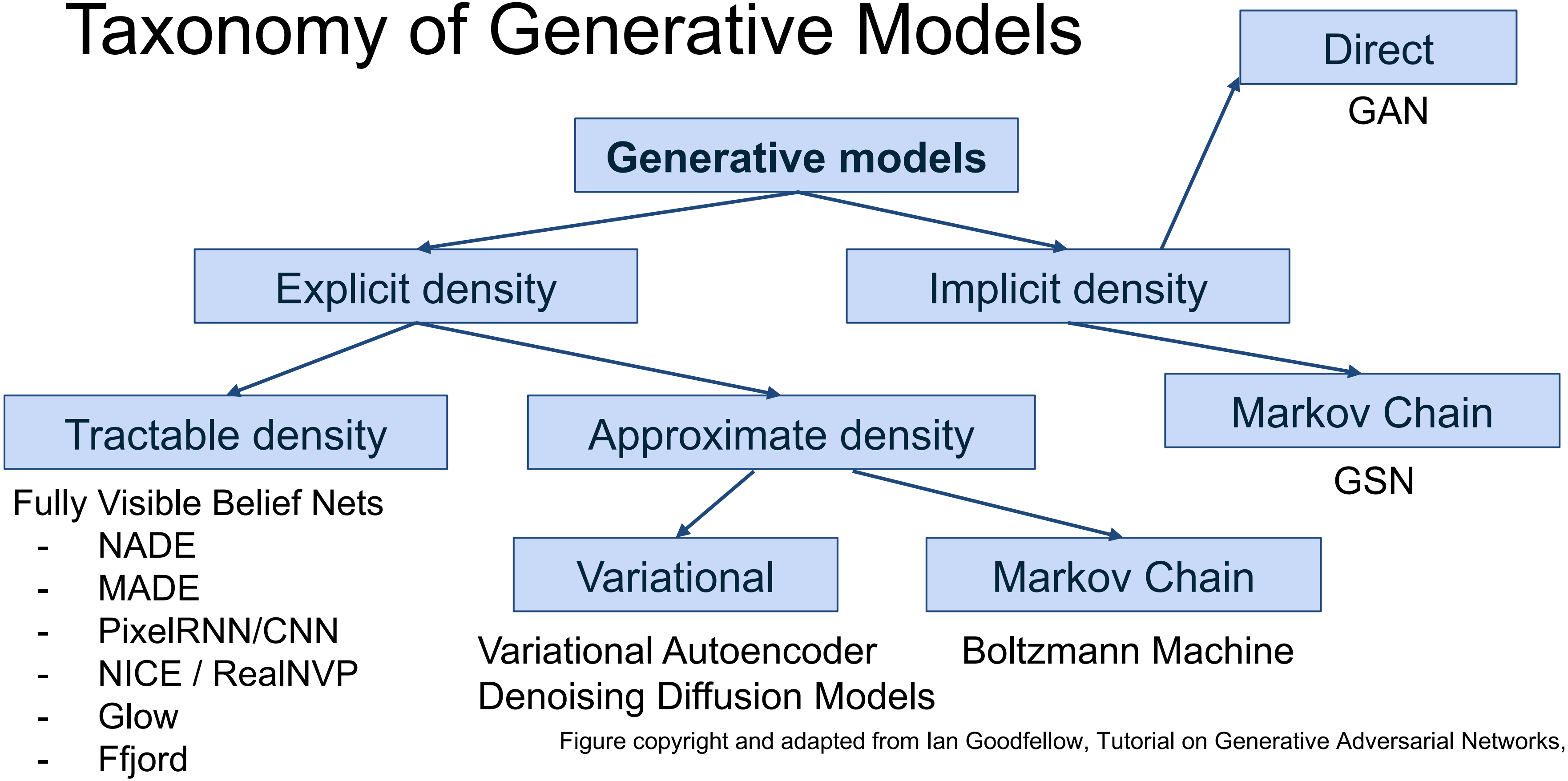



Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

# Taxonomy of Generative Models

Today and the next lecture: discuss 4 most popular types of generative models

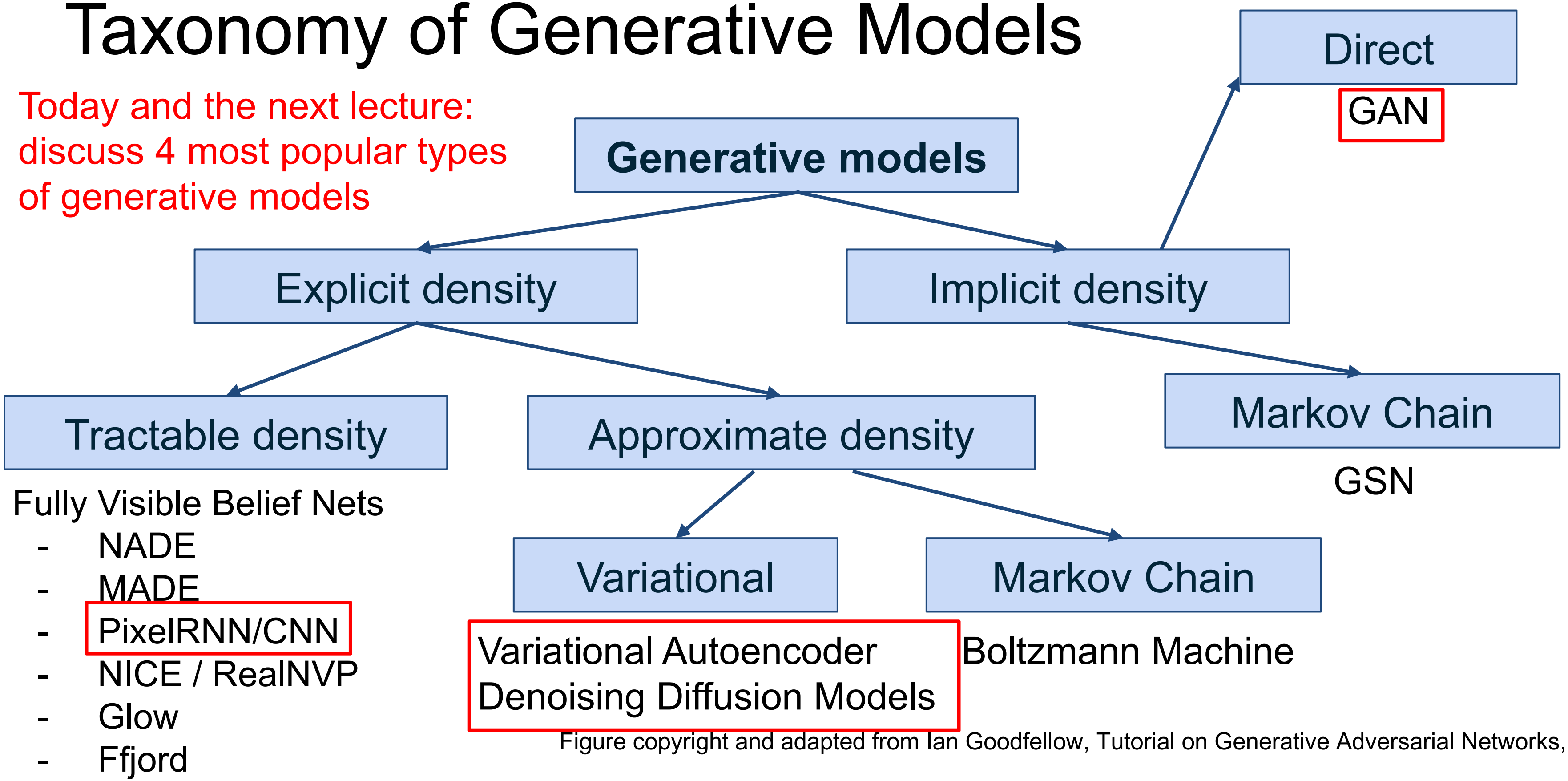



Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

# PixelRNN and PixelCNN

(A very brief overview)

# Fully visible belief network (FVBN)

Explicit density model

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

Likelihood of image x

Joint likelihood of each pixel in the image

# Fully visible belief network (FVBN)

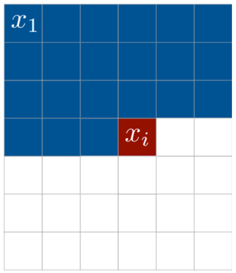
Explicit density model

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

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

↑ Likelihood of image x

↑ Probability of i'th pixel value given all previous pixels



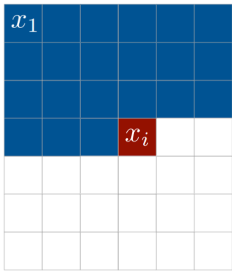

Then maximize likelihood of training data

# Fully visible belief network (FVBN)

Explicit density model

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

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



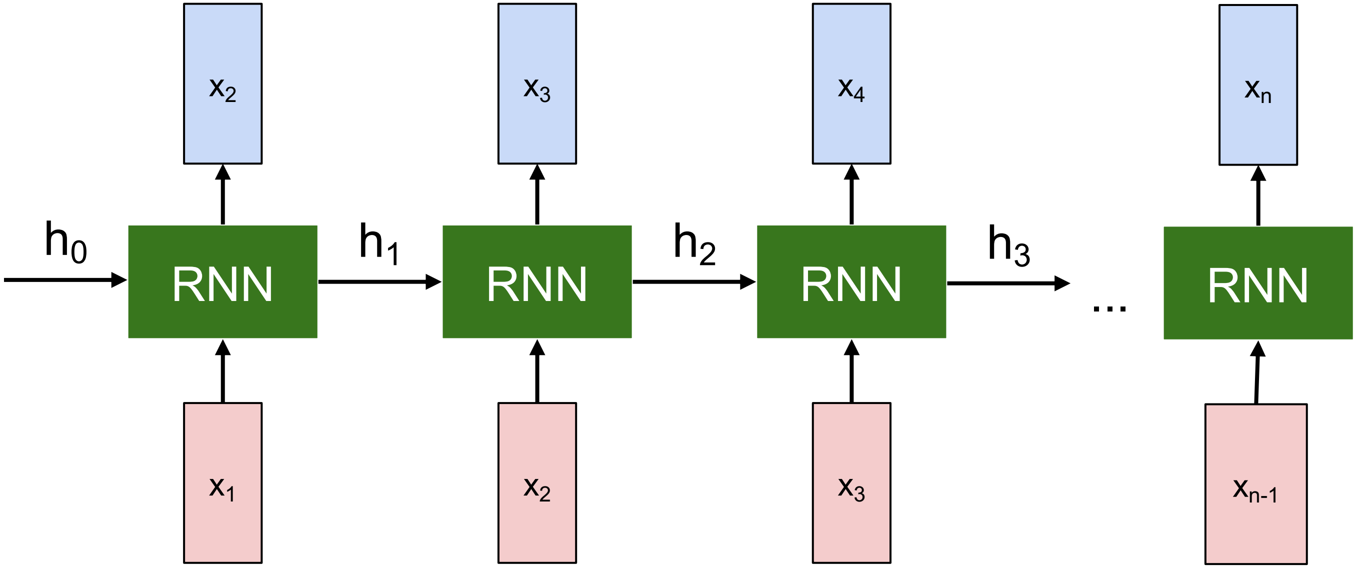

Likelihood of image x

Probability of i'th pixel value given all previous pixels

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

Then maximize likelihood of training data

# Recurrent Neural Network

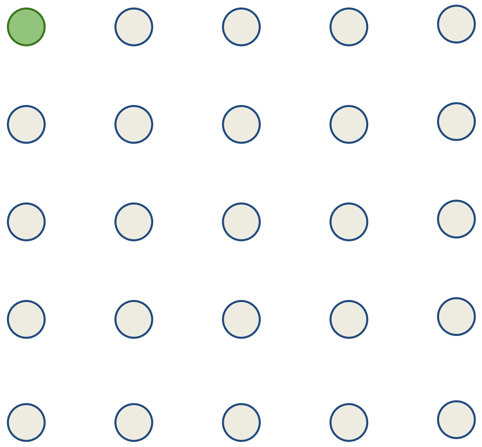



$$p(x_i|x_1, ..., x_{i-1})$$

# PixelRNN *[van der Oord et al. 2016]*

Generate image pixels starting from corner

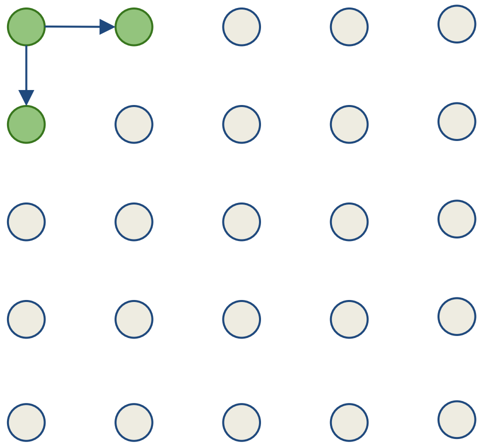
Dependency on previous pixels modeled using an RNN (LSTM)



# PixelRNN *[van der Oord et al. 2016]*

Generate image pixels starting from corner

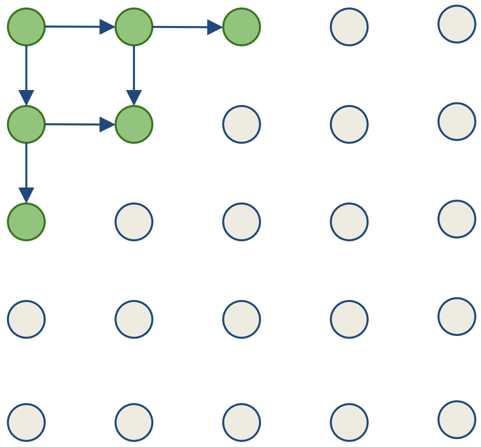
Dependency on previous pixels modeled using an RNN (LSTM)



# PixelRNN *[van der Oord et al. 2016]*

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

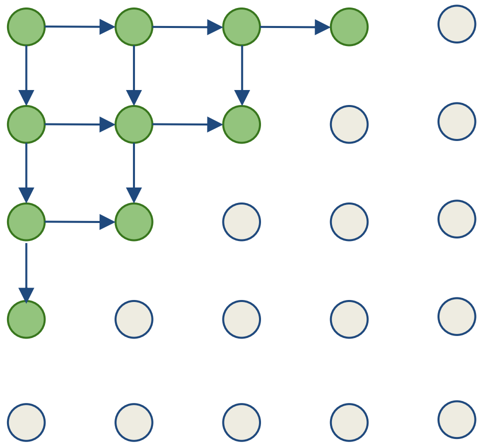


# PixelRNN *[van der Oord et al. 2016]*

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

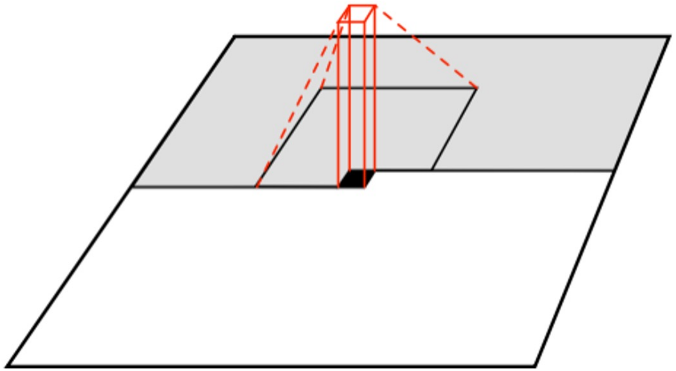
Drawback: sequential generation is slow in both training and inference!



# PixelCNN *[van der Oord et al. 2016]*

Still generate image pixels starting from corner

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



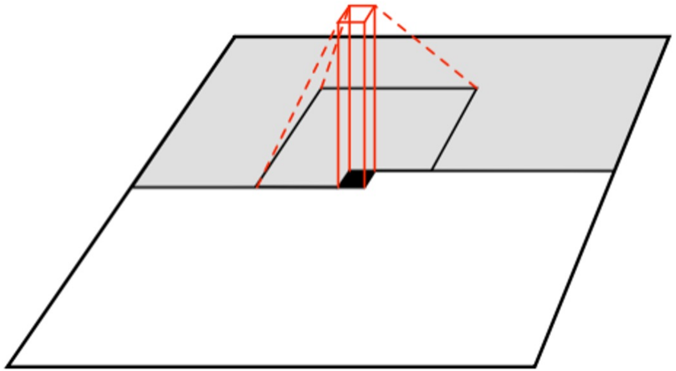
# PixelCNN *[van der Oord et al. 2016]*

Still generate image pixels starting from corner

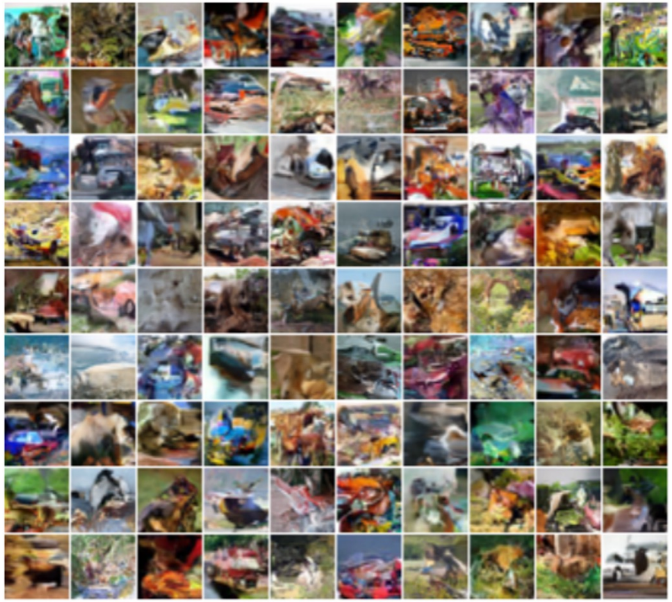
Dependency on previous pixels now modeled using a CNN over context region (masked convolution)

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

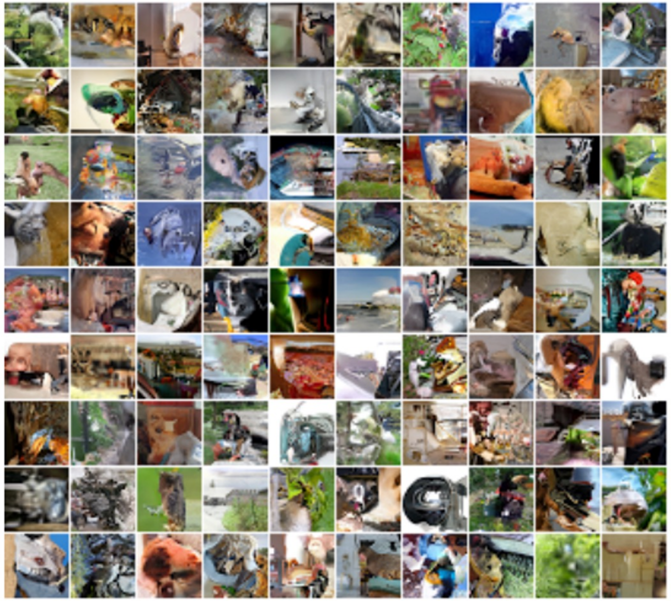
Generation is still slow:
For a 32x32 image, we need to do forward passes of the network 1024 times for a single image



# Generation Samples



32x32 CIFAR-10



32x32 ImageNet

# PixelRNN and PixelCNN

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

Con:
- Sequential generation => slow

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

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

# Taxonomy of Generative Models

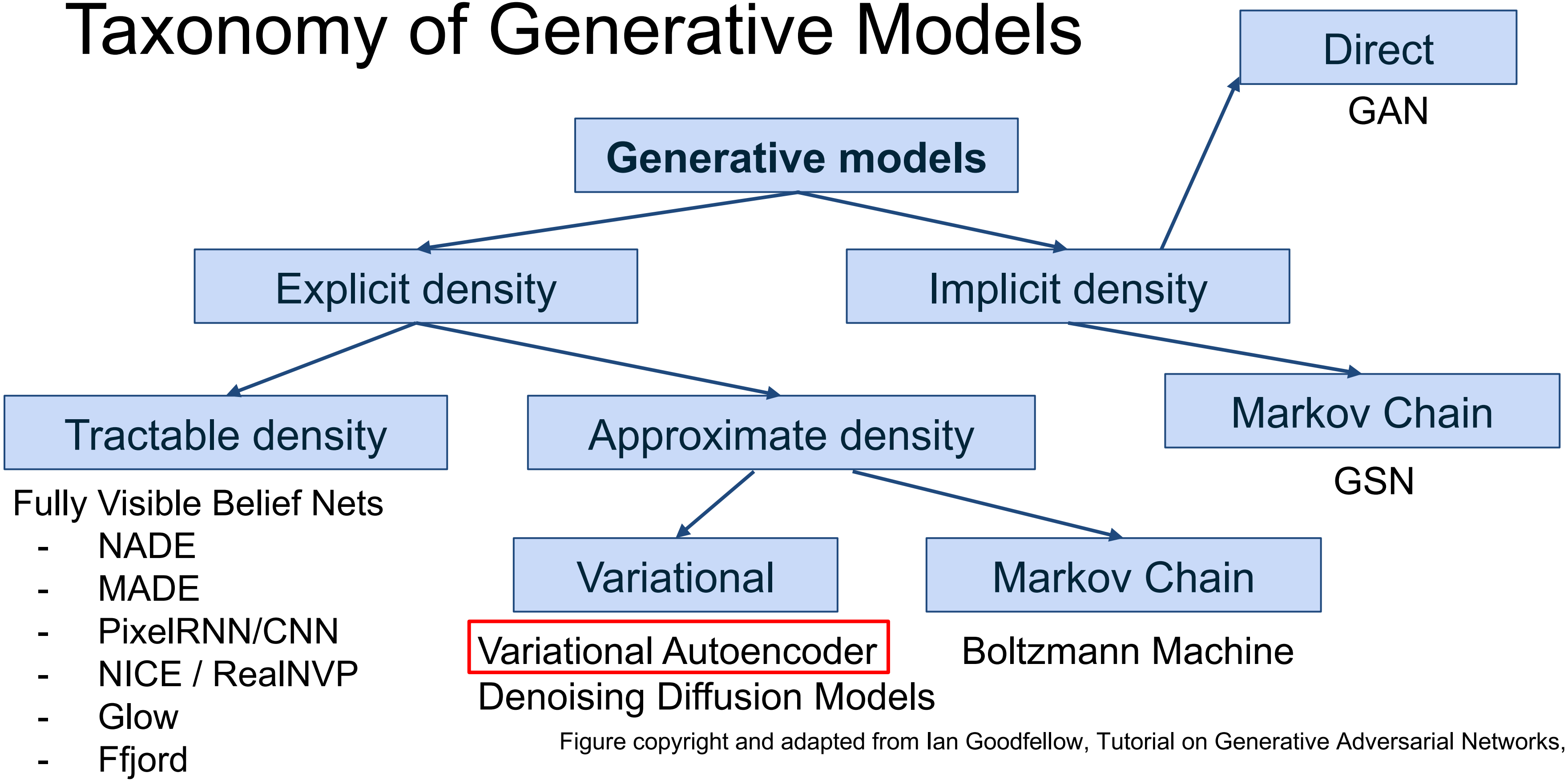



Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

# Variational Autoencoders (VAE)

# So far...

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

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

# So far...

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

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

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

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

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

# So far...

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

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

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

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

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

Cannot optimize (maximum likelihood estimation) directly, derive and optimize lower bound on likelihood instead

# So far...

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

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

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

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

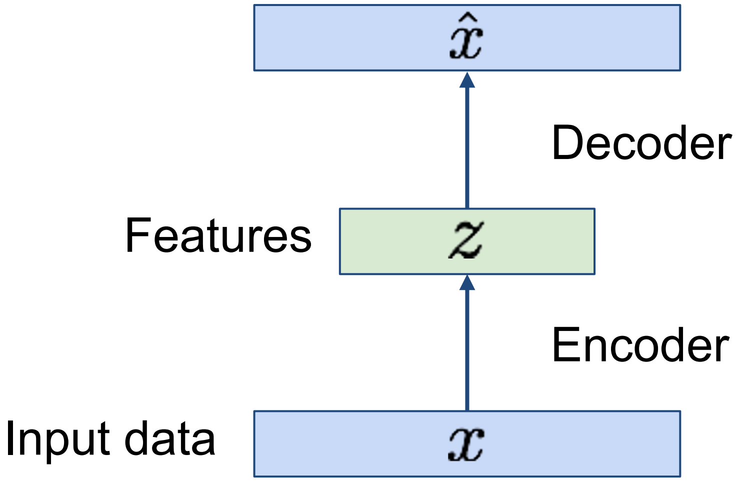
No dependencies among pixels, can generate all pixels at the same time!

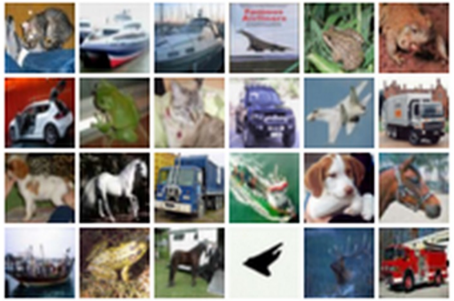
Cannot optimize (maximum likelihood estimation) directly, derive and optimize lower bound on likelihood instead

Why latent z?

# Some background first: Autoencoders

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



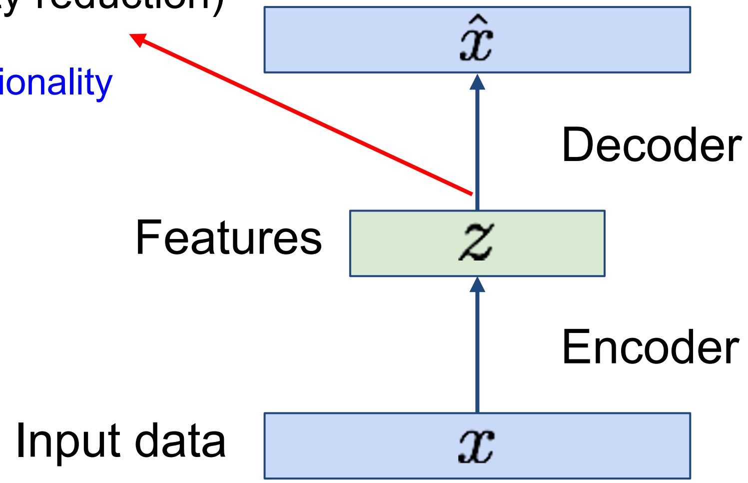



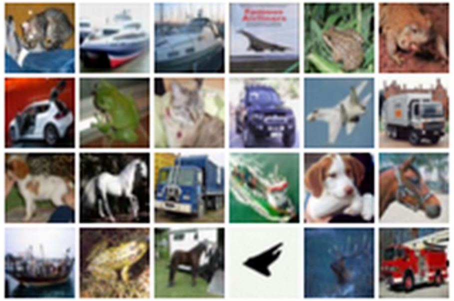
# Some background first: Autoencoders

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

**z** usually smaller than **x** (dimensionality reduction)

Q: Why dimensionality reduction?



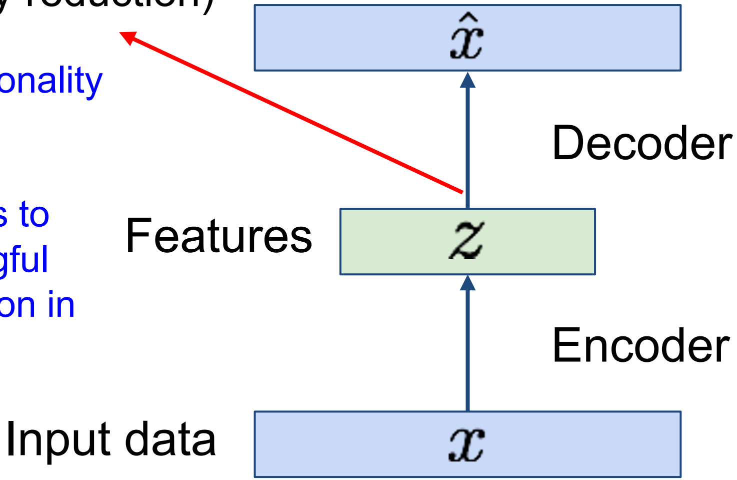
# Some background first: Autoencoders

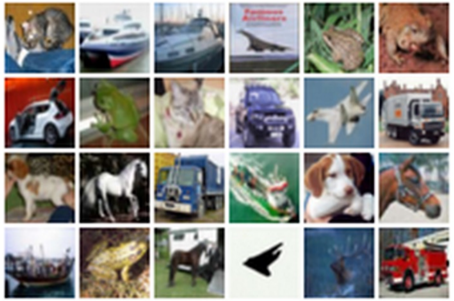
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

**z** usually smaller than **x**
(dimensionality reduction)

Q: Why dimensionality reduction?

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

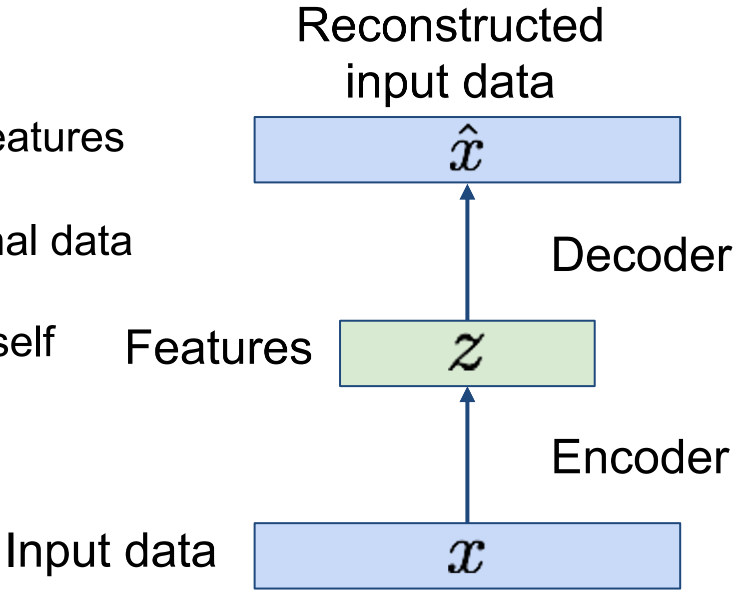


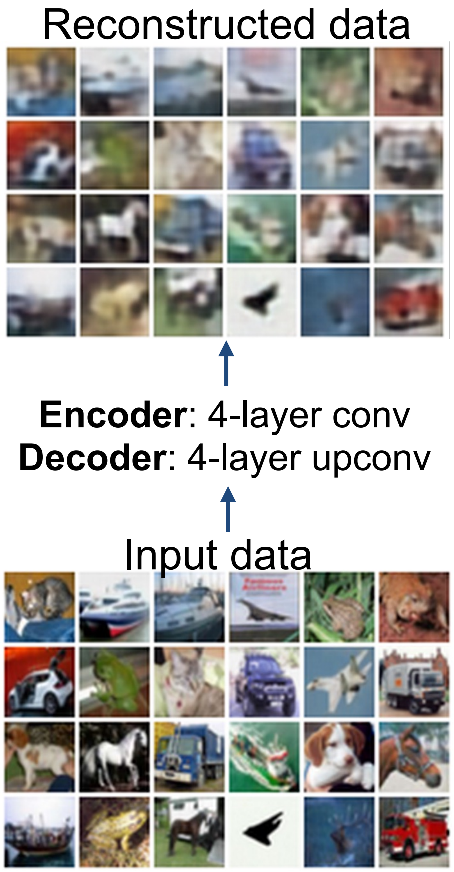



# Some background first: Autoencoders

How to learn this feature representation?

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

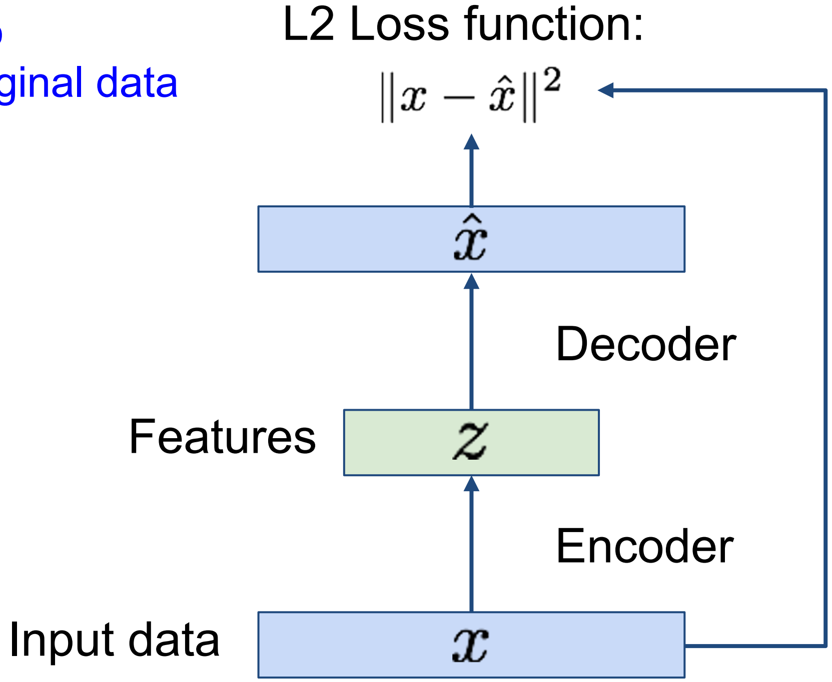


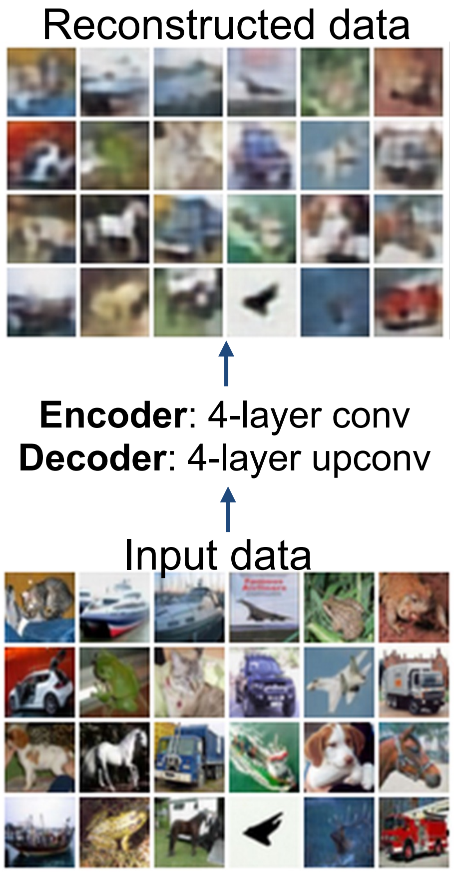



# Some background first: Autoencoders

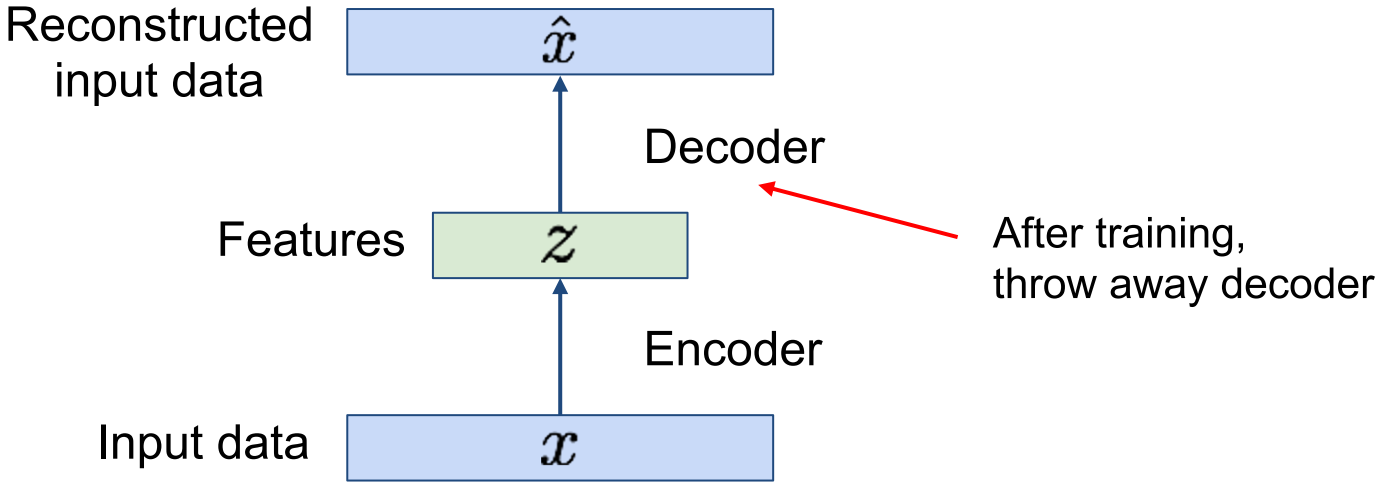
Train such that features can be used to reconstruct original data

Doesn't use labels!



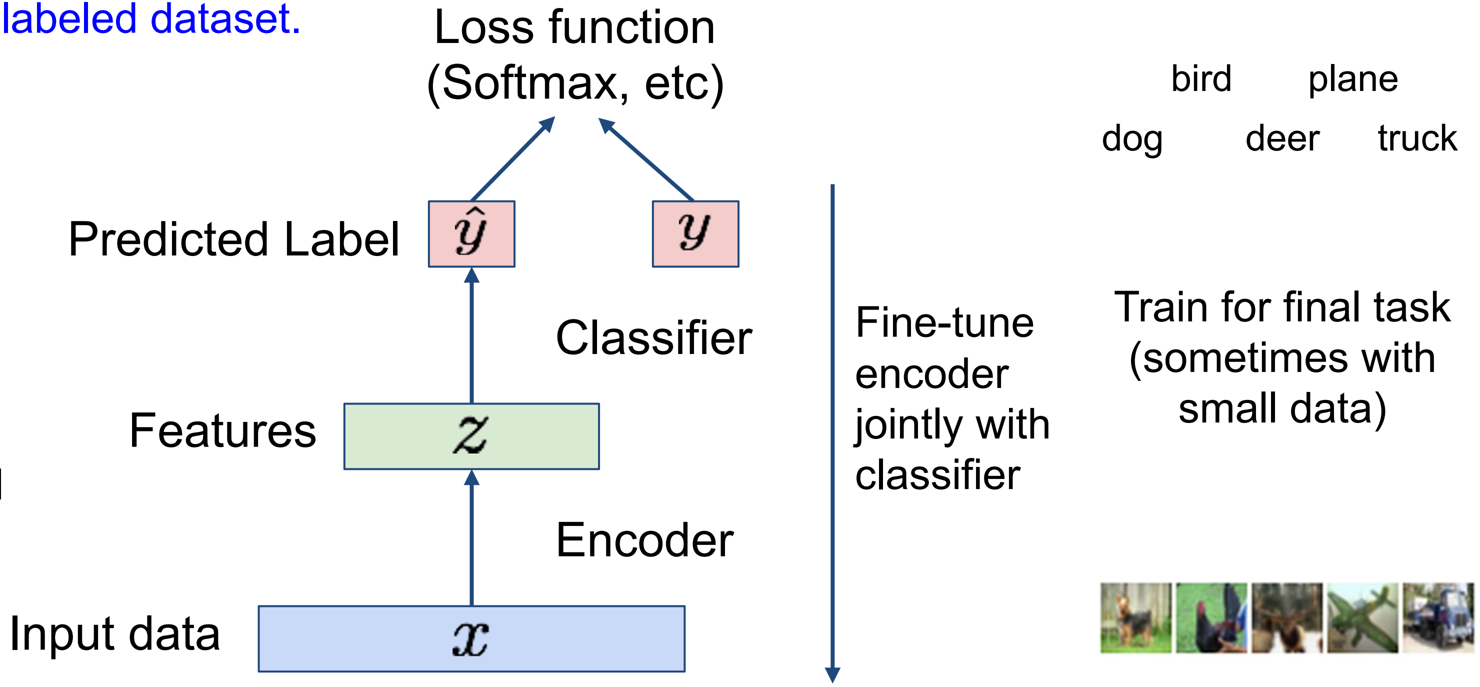



# Some background first: Autoencoders



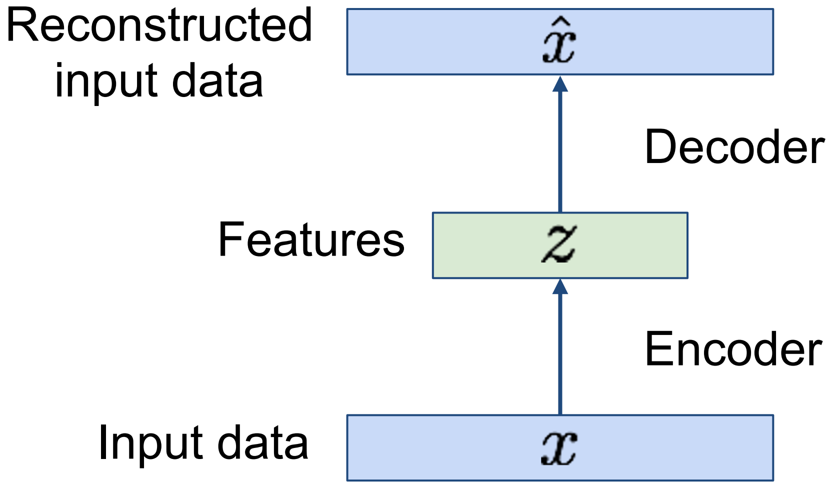
# Some background first: Autoencoders

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




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

# Some background first: Autoencoders




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

Features capture factors of variation in training data.

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

How do we make autoencoder a **generative model**?

# Variational Autoencoders

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

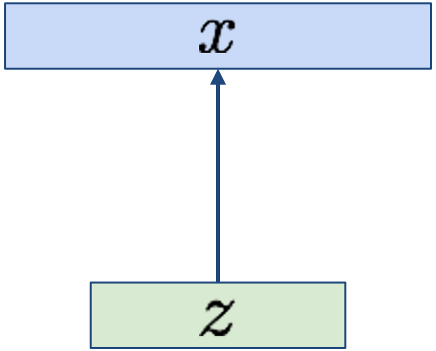
# Variational Autoencoders

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

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

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

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




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

# Variational Autoencoders

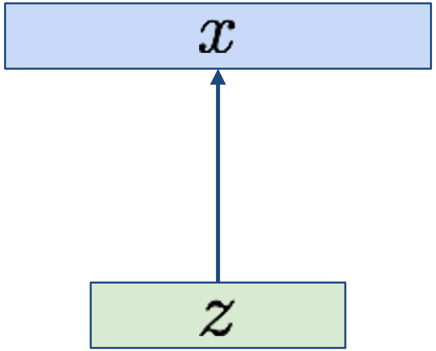
Probabilistic spin on autoencoders - will let us sample from the model to generate data!

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

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

Sample from true conditional

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




Sample from true prior

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

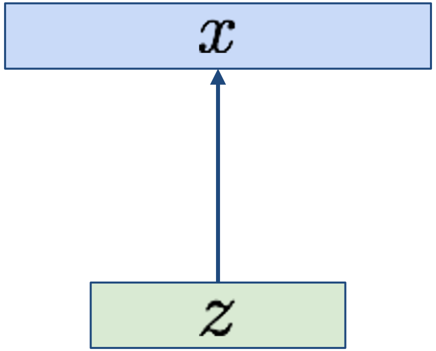
Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

# Variational Autoencoders

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

Sample from true conditional

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



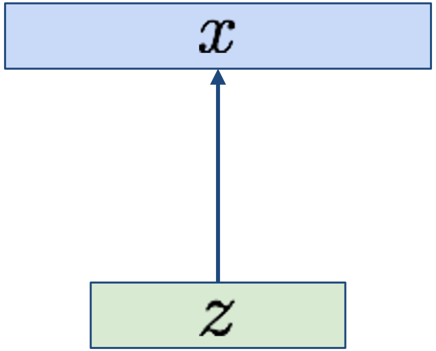

Sample from true prior

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

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

# Variational Autoencoders

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




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

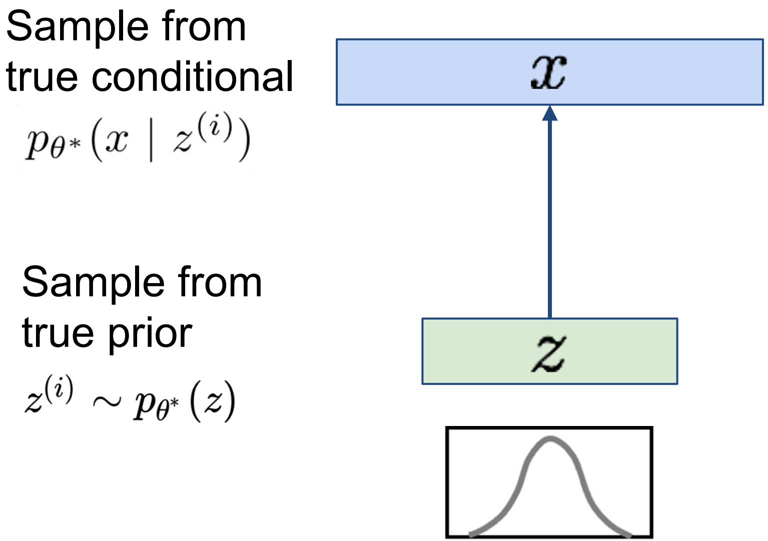
We want to estimate the true parameters $\theta^*$ of this generative model given training data x.

How should we represent this model?

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

# Variational Autoencoders

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




How should we represent this model?

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

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

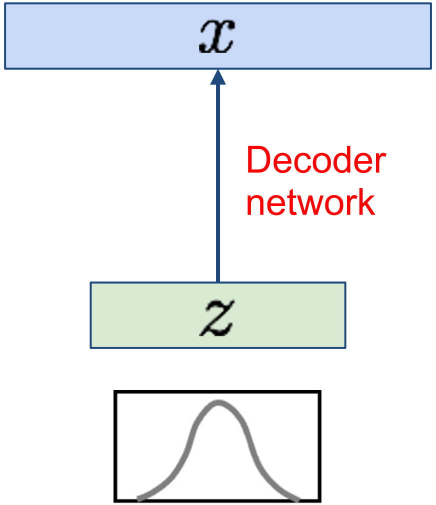
# Variational Autoencoders

Sample from true conditional

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

Sample from true prior

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




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

How should we represent this model?

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

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

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

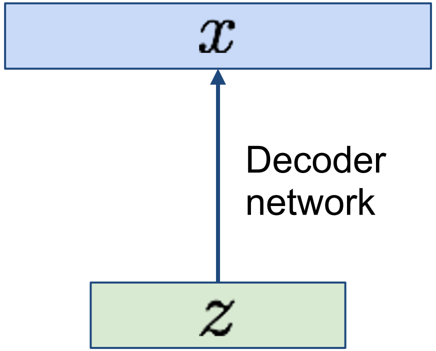
# Variational Autoencoders

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

How to train the model?

Sample from true conditional

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




Sample from true prior

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

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

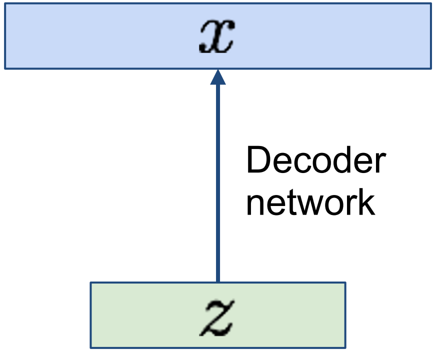
# Variational Autoencoders

Sample from true conditional

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

Sample from true prior

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




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

How to train the model?

Learn model parameters to maximize likelihood of training data

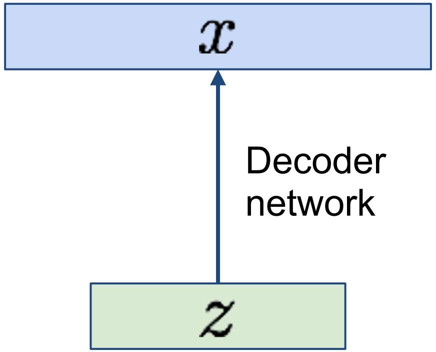
$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

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

# Variational Autoencoders

Sample from true conditional

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




Sample from true prior

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

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

How to train the model?

Learn model parameters to maximize likelihood of training data

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

Q: What is the problem with this?

Intractable!

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

# Variational Autoencoders: Intractability

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

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

# Variational Autoencoders: Intractability

✔

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

↑ Simple Gaussian prior

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

# Variational Autoencoders: Intractability

✓ ✓

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

Decoder neural network

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

# Variational Autoencoders: Intractability

😢 ✓ ✓

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

Intractable to compute p(x|z) for every z!

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

# Variational Autoencoders: Intractability

😢 ✔ ✔

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

Intractable to compute p(x|z) for every z!

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

Monte Carlo estimation is too high variance

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

# Variational Autoencoders: Intractability

😢 ✔ ✔

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

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

Intractable data likelihood

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

# Variational Autoencoders: Intractability

😢 ✔ ✔

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

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

Can we derive a *tractable approximate* of the data likelihood?

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

# Variational Autoencoders

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

# Variational Autoencoders

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

Let's assume we can sample from some approximate posterior for now …

# Variational Autoencoders

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

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

# Variational Autoencoders

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

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

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

# Variational Autoencoders

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule})\\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad (\text{Multiply by constant})\\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Logarithms})
\end{aligned}
$$

# Variational Autoencoders

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad (\text{Bayes' Rule}) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad (\text{Multiply by constant}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad (\text{Logarithms}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))
\end{aligned}
$$

Recall: $D_{KL}(q||p) = \mathbf{E}_q[\log \frac{q}{p}]$

# Variational Autoencoders

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z \sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z) p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule}) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z) p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad (\text{Multiply by constant}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Logarithms}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))
\end{aligned}
$$

We use a neural network encoder to approximate the posterior distribution, i.e., what is the distribution of z given an input $x^{(i)}$. Assume a Gaussian distribution.

# Variational Autoencoders

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z \sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule}) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad (\text{Multiply by constant}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Logarithms}) \\
&= \boxed{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))} + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))
\end{aligned}$$

Decoder network gives $p_\theta(x|z)$, can compute the expectation by sampling from the learned posterior. (need some trick to differentiate through sampling).

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

# Variational Autoencoders

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \qquad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \qquad (\text{Bayes' Rule}) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \qquad (\text{Multiply by constant}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \qquad (\text{Logarithms}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))
\end{aligned}
$$

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :( But we know KL divergence always >= 0.

# Variational Autoencoders

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule}) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad (\text{Multiply by constant}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Logarithms}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))
\end{aligned}
$$

We want to maximize the data likelihood

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term :( But we know KL divergence always >= 0.

# Variational Autoencoders

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Bayes' Rule}) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] \quad (\text{Multiply by constant}) \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] \quad (\text{Logarithms}) \\
&= \underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\geq 0}
\end{aligned}
$$

We want to maximize the data likelihood

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

# Variational Autoencoders

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

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

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

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

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

**Decoder:** reconstruct the input data

**Encoder:** make approximate posterior distribution close to prior

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

# Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

# Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

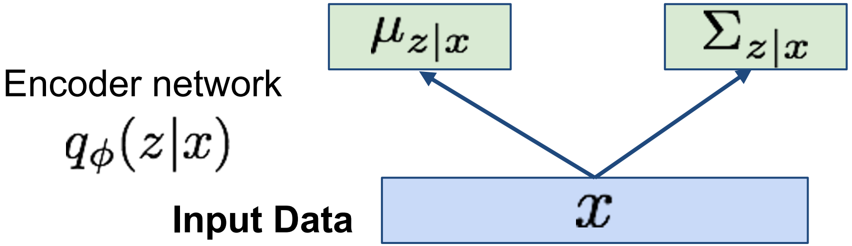
$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \boxed{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

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

# Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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



# Variational Autoencoders

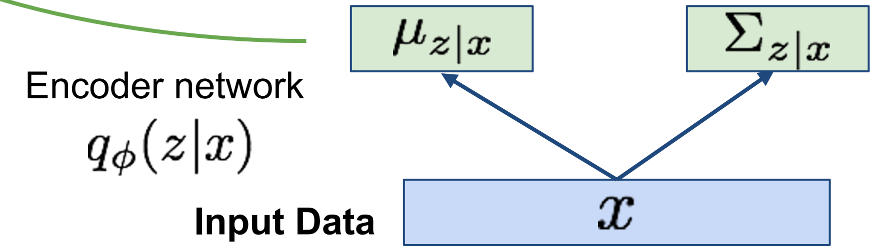
Putting it all together: maximizing the likelihood lower bound

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

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

Have analytical solution

Make approximate posterior distribution close to prior

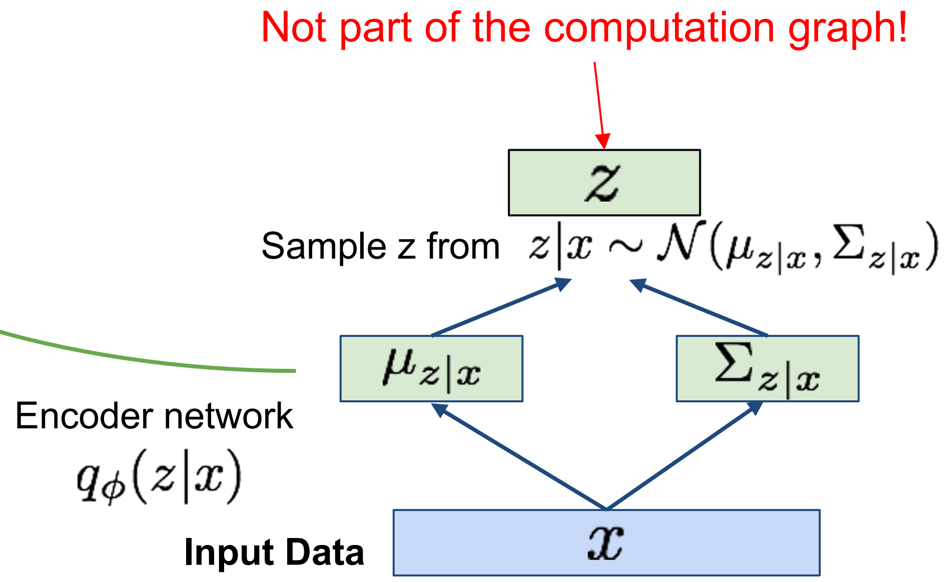


# Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior



# Variational Autoencoders

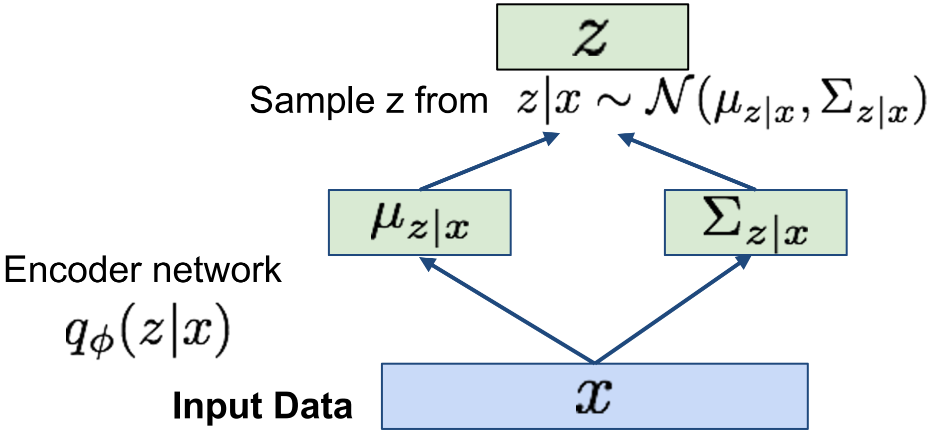
Putting it all together: maximizing the likelihood lower bound

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

Reparameterization trick to make sampling differentiable:

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

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



# Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

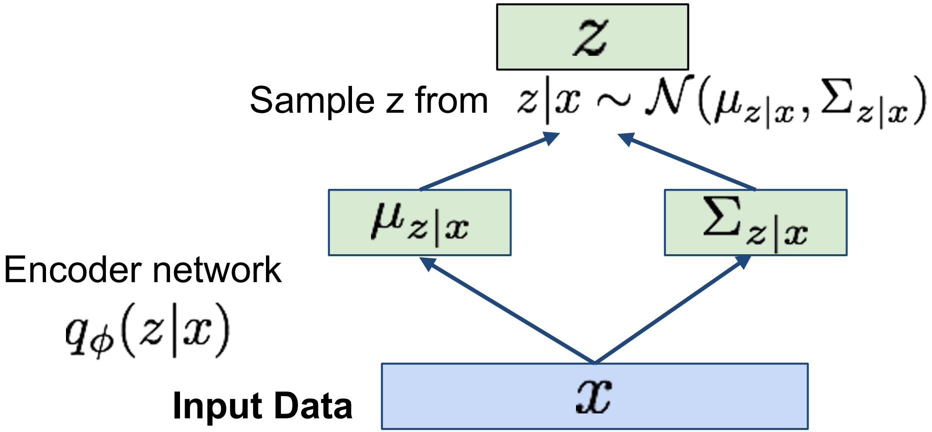
Reparameterization trick to make sampling differentiable:

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

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

Input to the graph

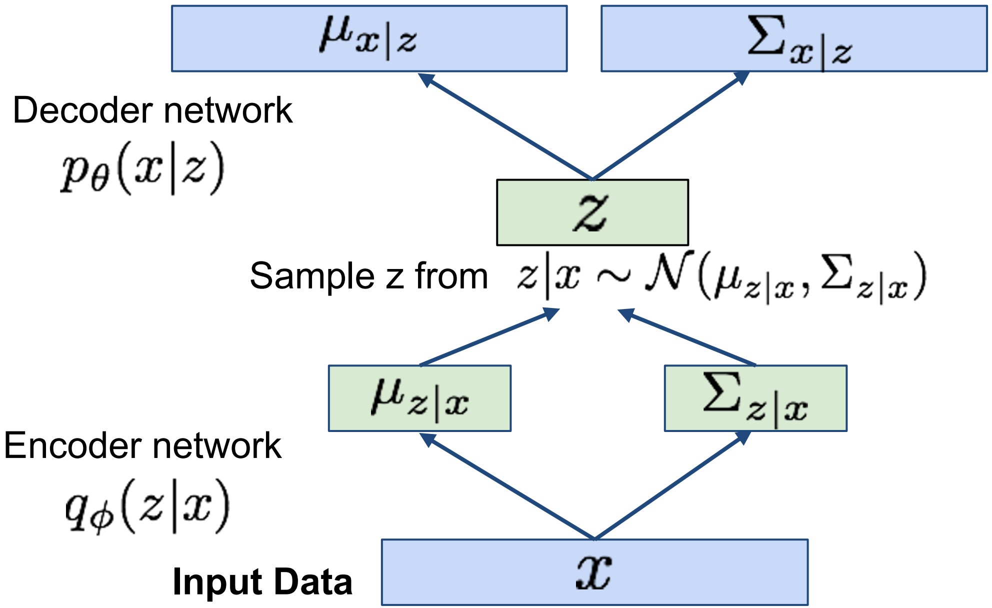
Part of computation graph



# Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

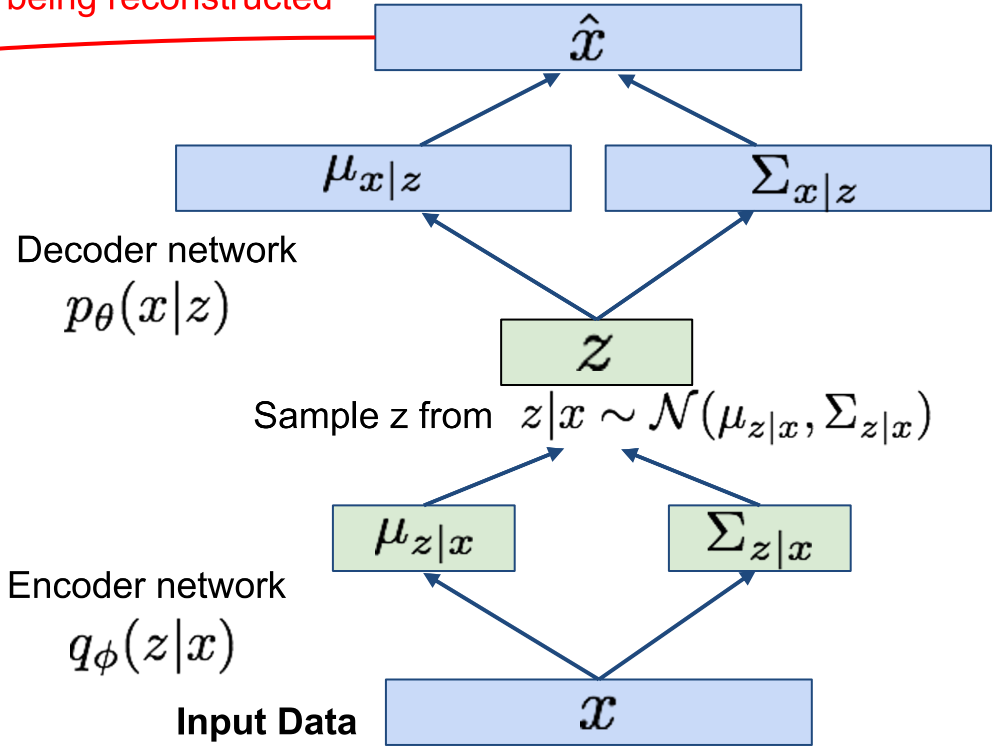


# Variational Autoencoders

Maximize likelihood of original input being reconstructed

Putting it all together: maximizing the likelihood lower bound

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

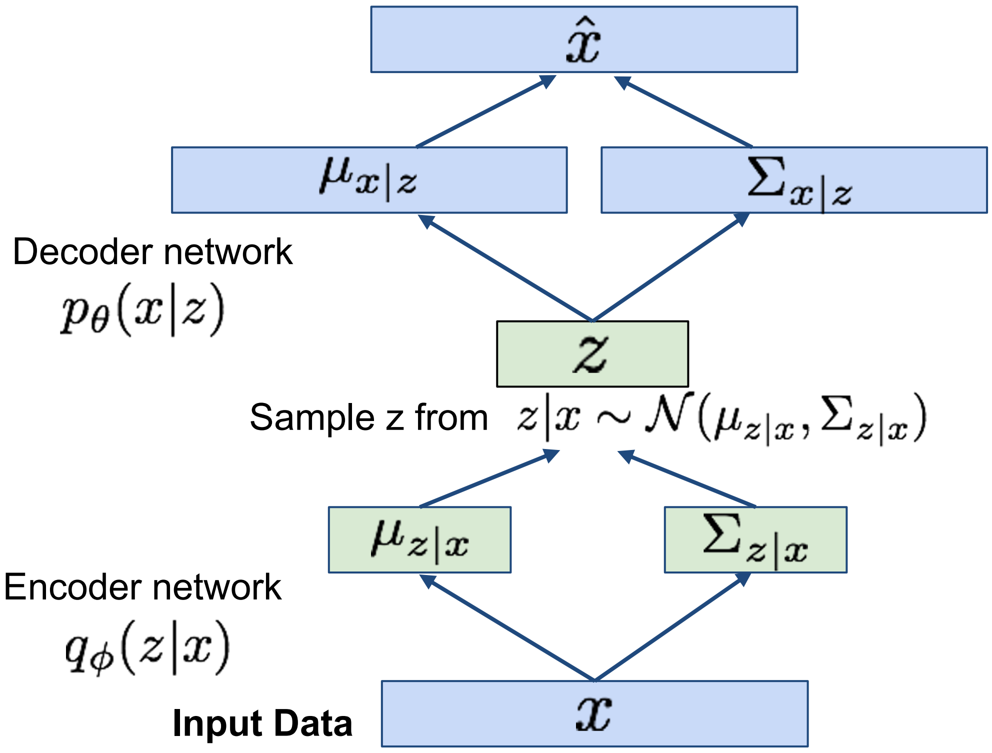


# Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

Hyperparameter to weigh the strength of the prior matching objective

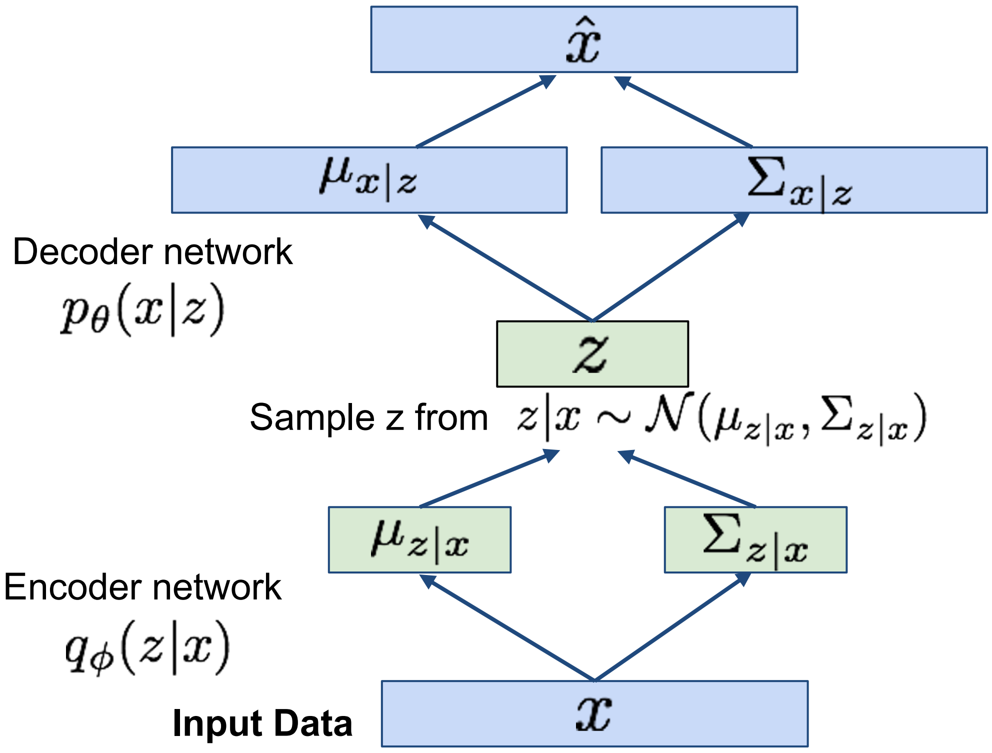


# Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

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

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



# Variational Autoencoders: Generating Data!

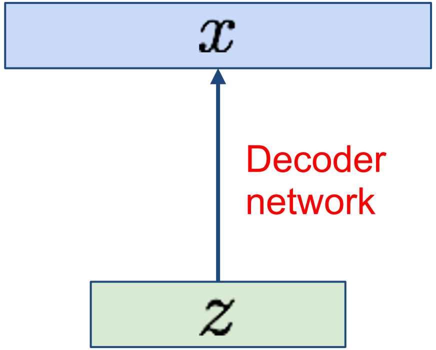
Our assumption about data generation process

Sample from true conditional

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

Sample from true prior

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




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

# Variational Autoencoders: Generating Data!

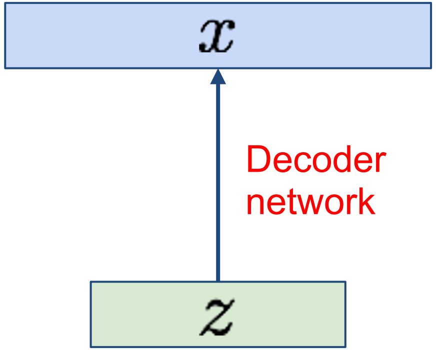
Our assumption about data generation process

Sample from true conditional

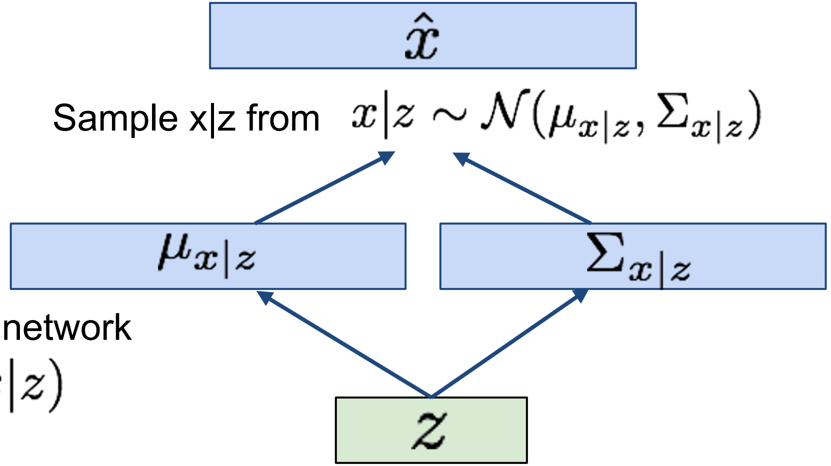
$p_{\theta^*}(x \mid z^{(i)})$

Sample from true prior

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




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




Sample x|z from $x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$

Decoder network

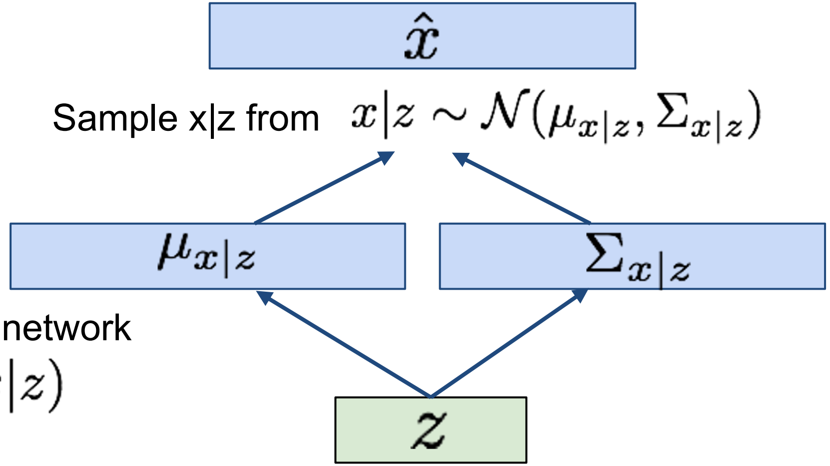
$p_\theta(x|z)$

Sample z from $z \sim \mathcal{N}(0, I)$

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

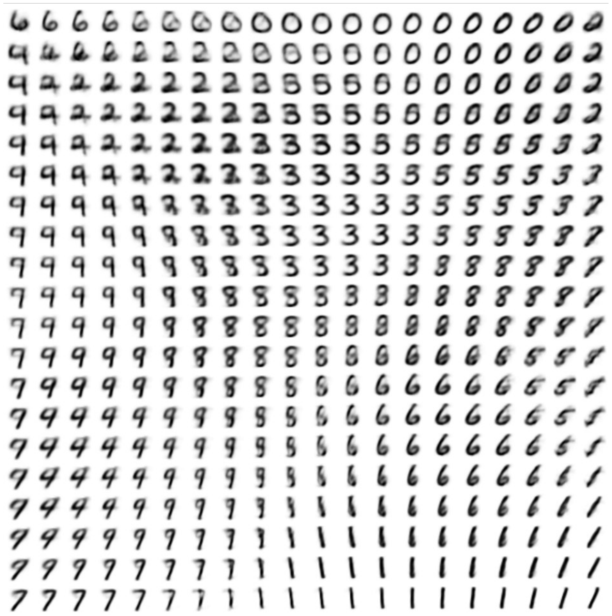
# Variational Autoencoders: Generating Data!

Use decoder network. Now sample z from prior!




Decoder network
$p_\theta(x|z)$

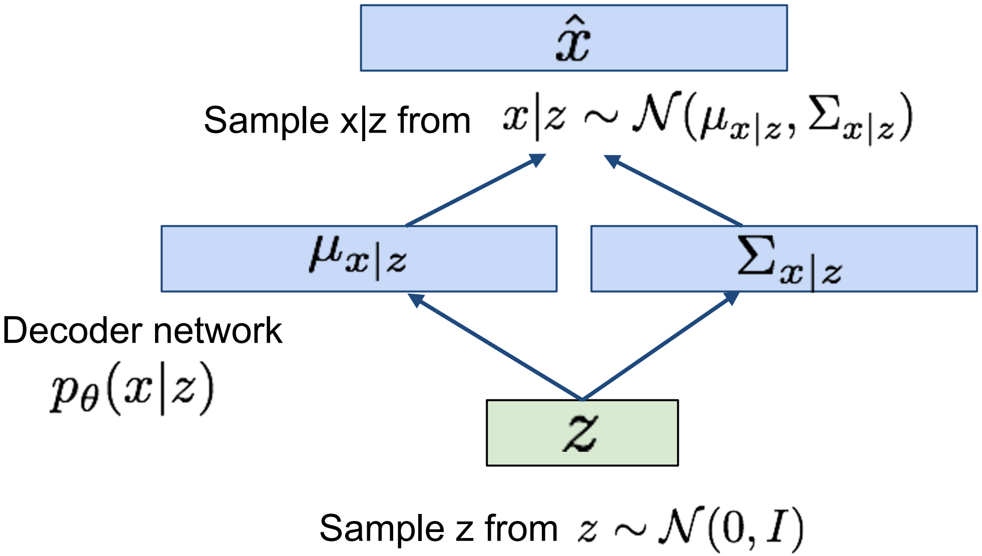
Sample z from $z \sim \mathcal{N}(0, I)$



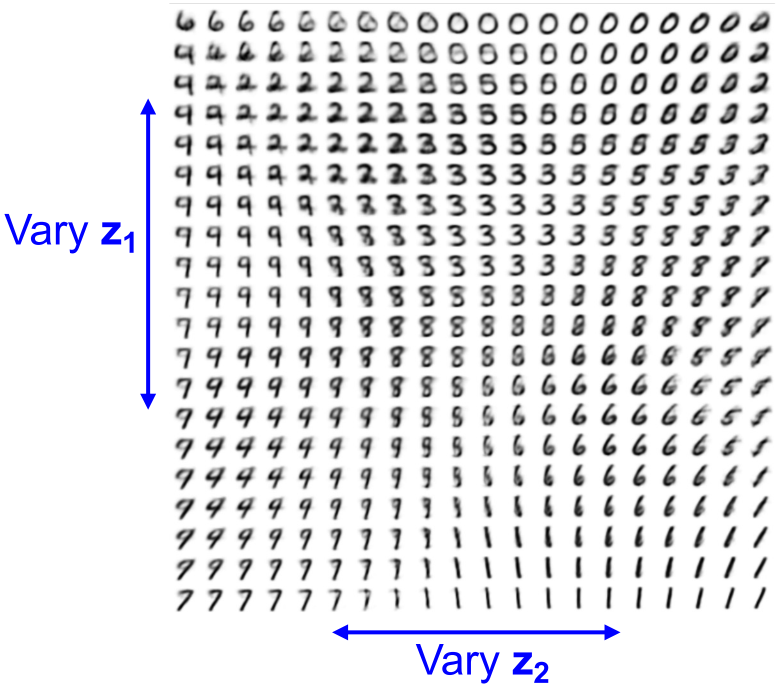
Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

# Variational Autoencoders: Generating Data!

Use decoder network. Now sample z from prior!




Data manifold for 2-d **z**




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

# Variational Autoencoders: Generating Data!

Diagonal prior on **z** => independent latent variables

Different dimensions of **z** encode interpretable factors of variation

Degree of smile

Vary $\mathbf{z}_1$

Vary $\mathbf{z}_2$

Head pose

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

# Variational Autoencoders: Generating Data!

Diagonal prior on **z** => independent latent variables

Different dimensions of **z** encode interpretable factors of variation

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


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

# Variational Autoencoders: Generating Data!

32x32 CIFAR-10

Labeled Faces in the Wild

# Variational Autoencoders

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

**Pros:**

- Principled approach to generative models
- Latent space z is interpretable and may be useful for other downstream tasks.

**Cons:**

- Samples are blurry
- KL weights are hard to tune
- Latent distributions are aggressive representation bottlenecks that may limit the expressiveness of the model.

**Active areas of research:**

- More flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian, e.g., Gaussian Mixture Models (GMMs), Categorical Distributions.
- Learning disentangled representations.

# Next Time: Denoising Diffusion and Generative Adversarial Networks