Lecture 4: Generative Models

Variational Autoencoders

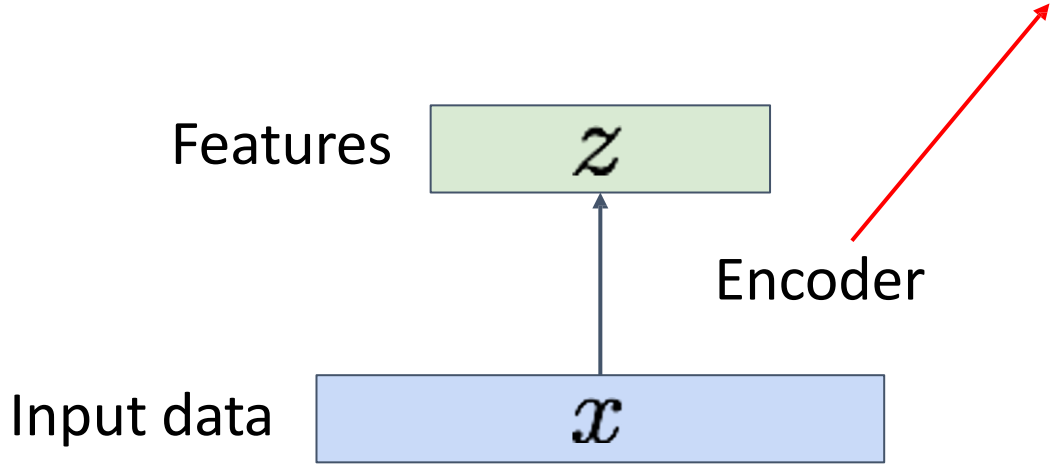
(Regular, non-variational) Autoencoders

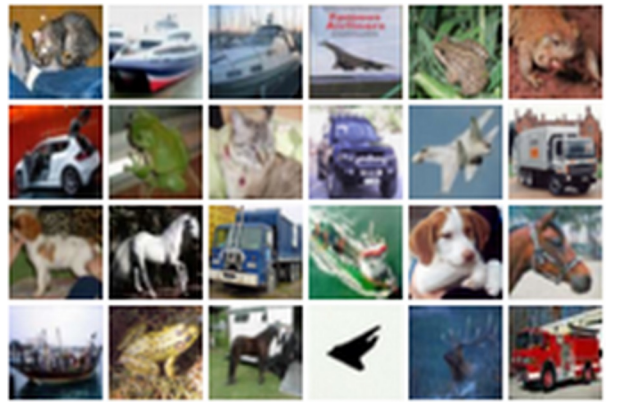
Unsupervised method for learning feature vectors from raw data x, without any labels

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

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





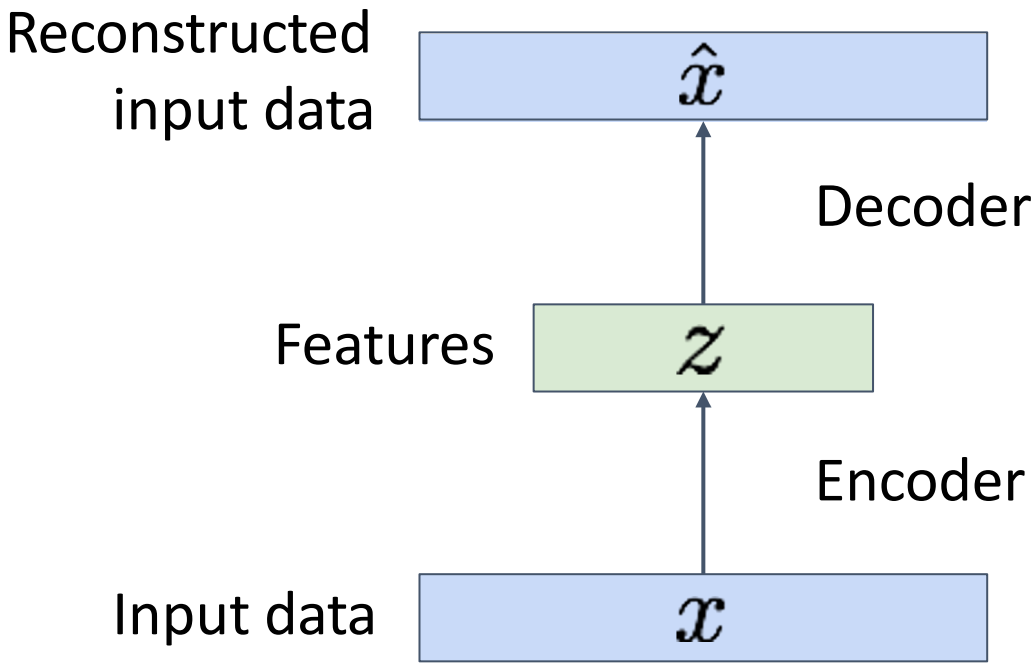
Input Data

(Regular, non-variational) Autoencoders

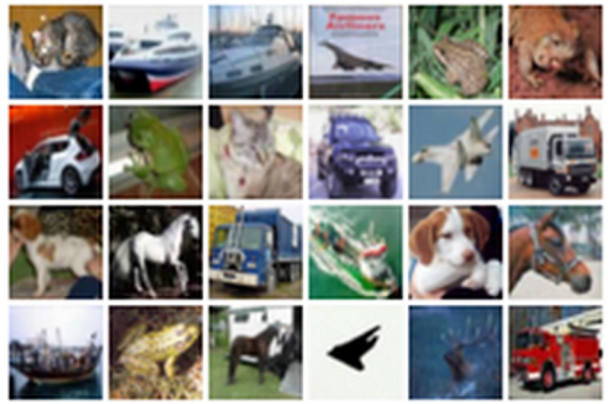
Problem: How can we learn this feature transform from raw data?

Idea: Use the features to reconstruct the input data with a **decoder**

"Autoencoding" = encoding itself



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



Input Data

(Regular, non-variational) Autoencoders

Loss: L2 distance between input and reconstructed data.

Does not use any labels! Just raw data!

Loss Function

$$\|x - \hat{x}\|_2^2$$

Reconstructed input data

$\hat{x}$

Decoder

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

Features

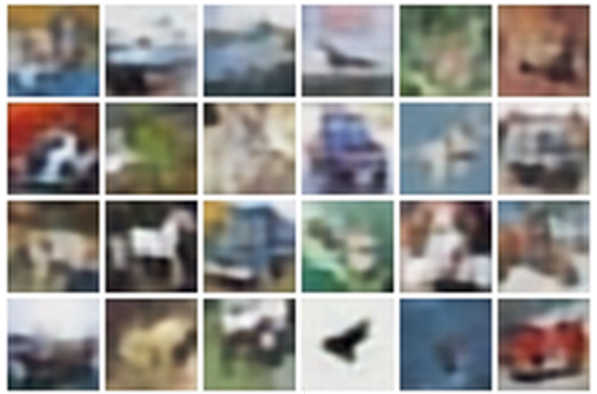
z

Encoder

Input data

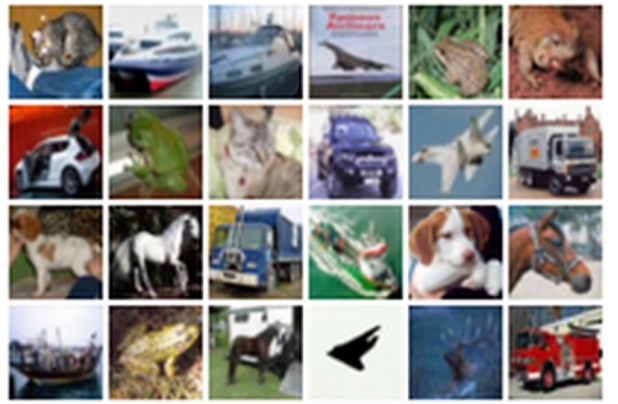
x

Reconstructed data



Decoder:
4 tconv layers

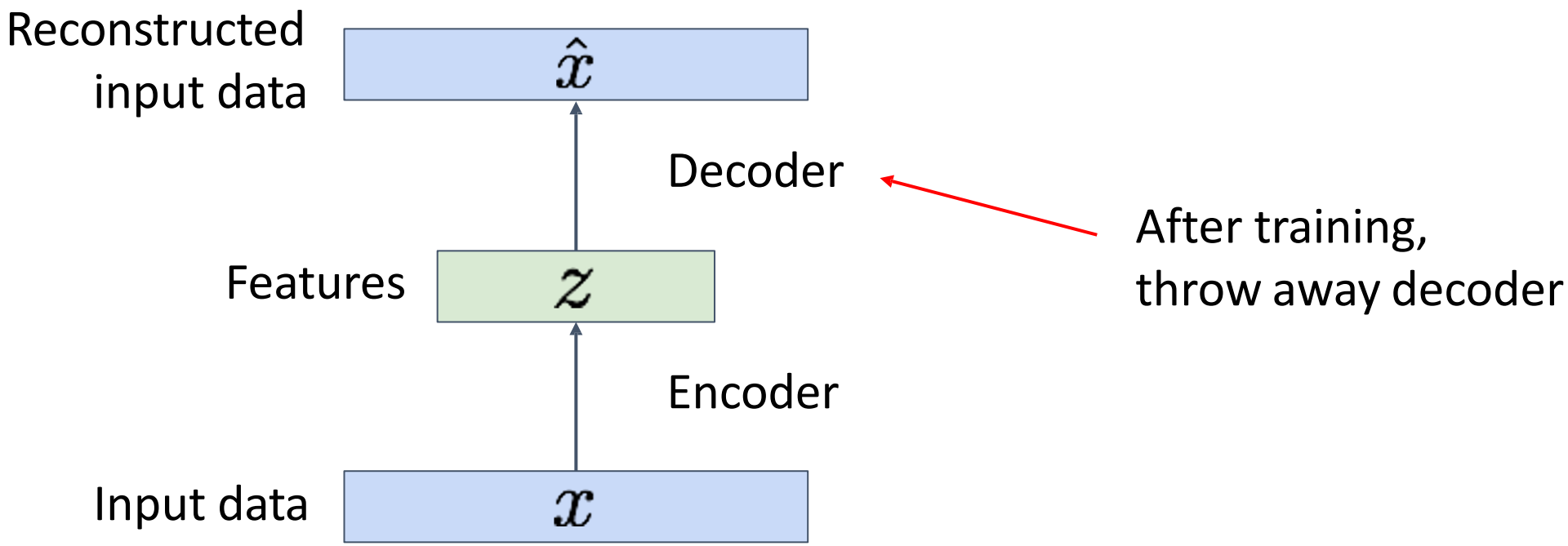
Encoder:
4 conv layers



Input Data

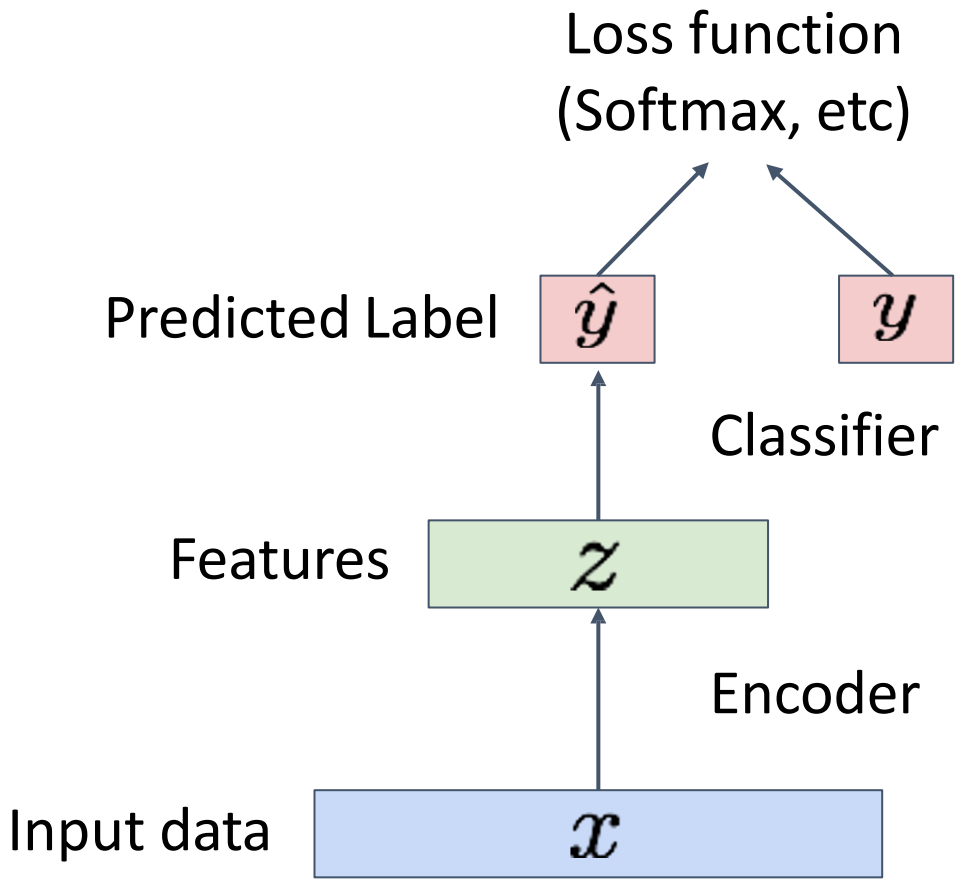
(Regular, non-variational) Autoencoders

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



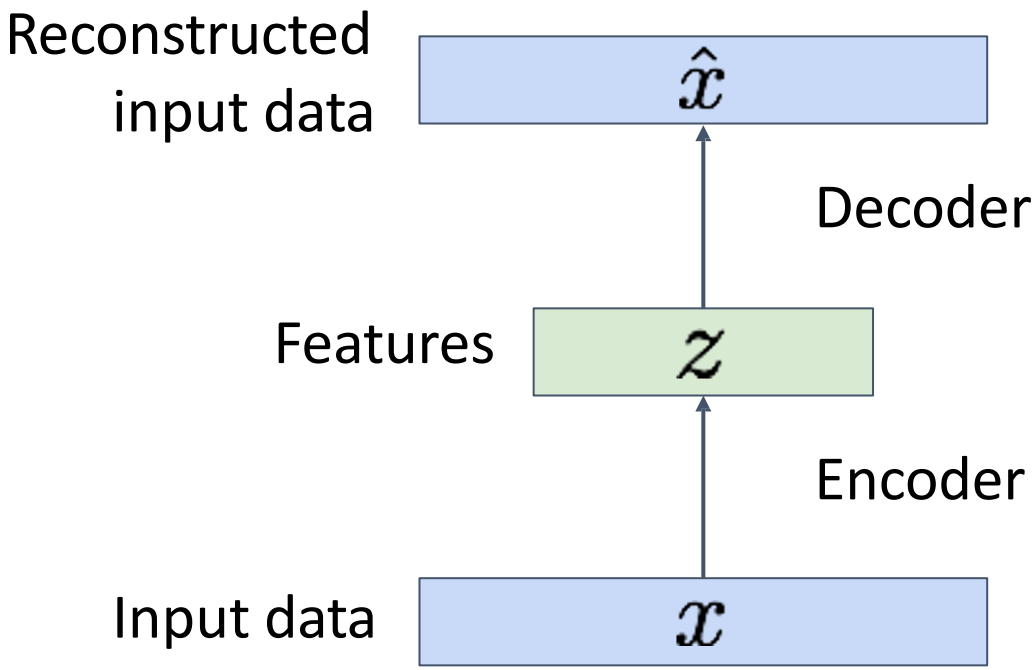
Fine-tune encoder jointly with classifier

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

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

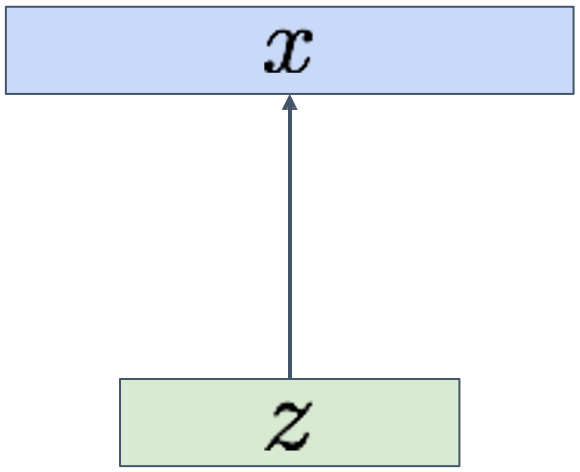
Assume training data $\{x^{(i)}\}_{i=1}^{N}$ is generated from unobserved (latent) representation **z**

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

After training, sample new data like this:

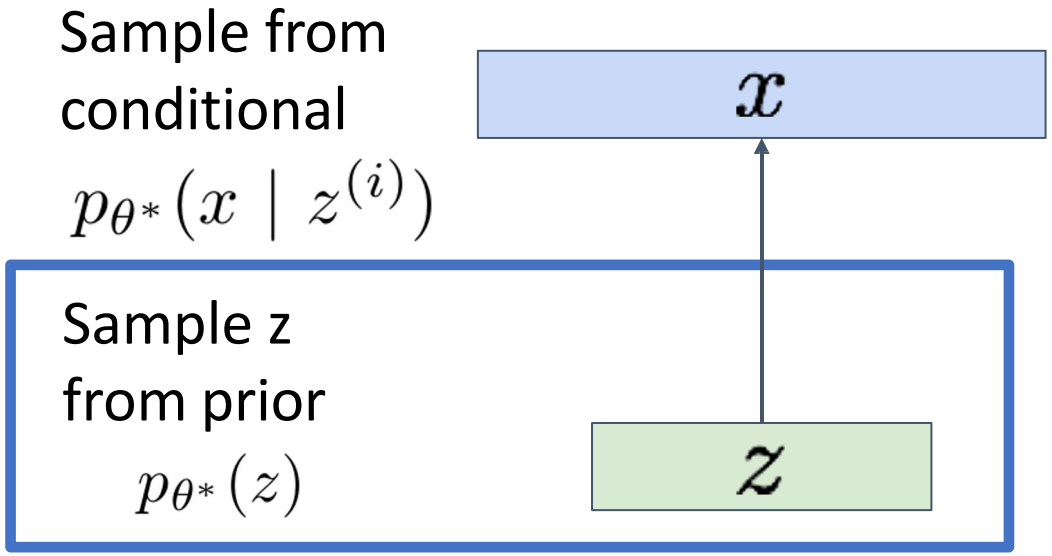
Sample from conditional $p_{\theta^*}(x \mid z^{(i)})$

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



Variational Autoencoders

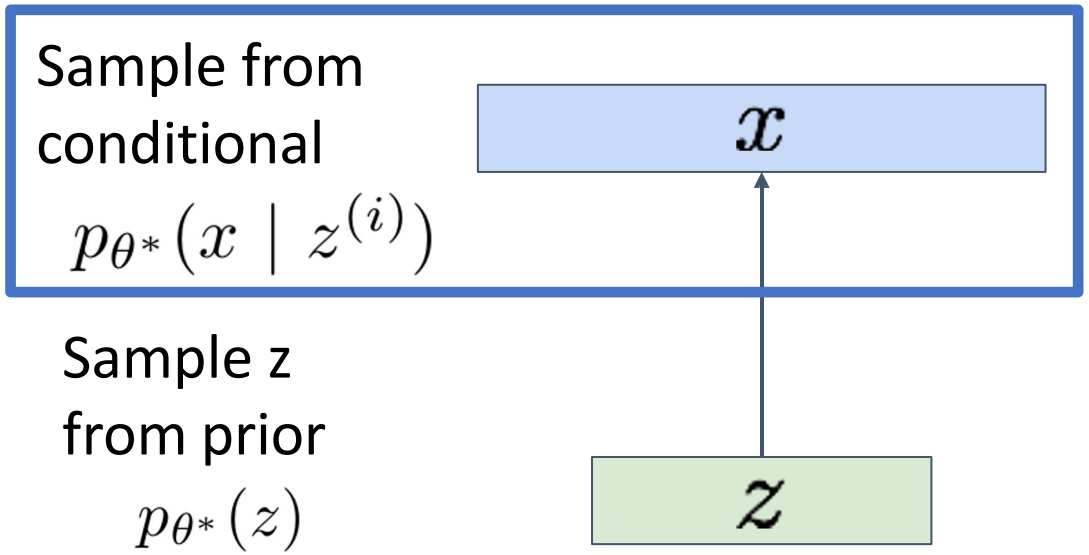
After training, sample new data like this:



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

Variational Autoencoders

After training, sample new data like this:



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

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

Variational Autoencoders

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

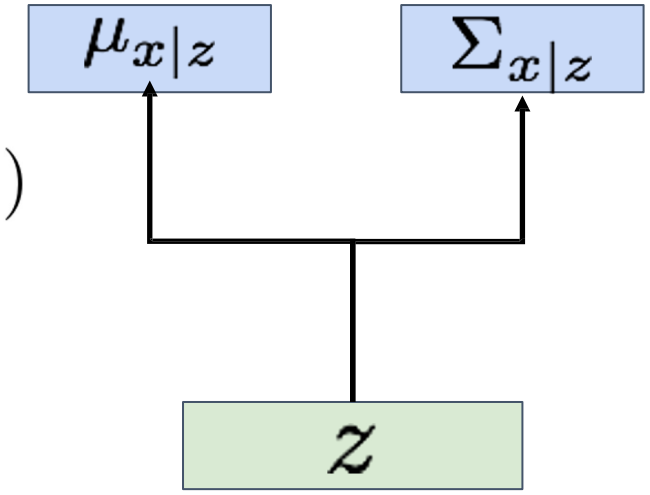
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

Sample from conditional

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

Sample z from prior

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



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)

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

Ok, we assumed Gaussian prior for z

Problem: Impossible to integrate over all z!

Variational Autoencoders

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

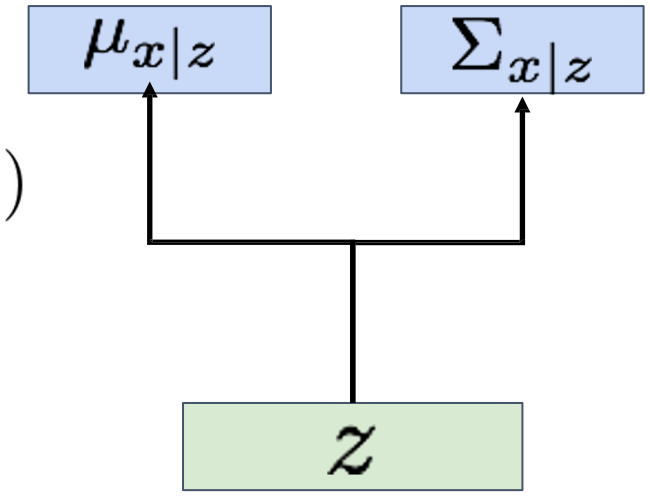
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

Sample from conditional

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

Sample z from prior

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



How to train this model?
Basic idea: **maximize likelihood of data**

Another idea: Try Bayes Rule

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

Ok, compute with decoder network

Ok, we assumed Gaussian prior for z

Problem: No way to compute this!

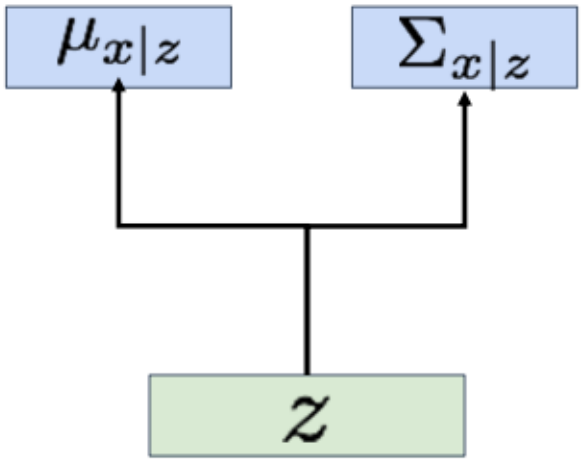
Solution: Train another network **(encoder)** that learns

$$q_\phi(z \mid x) \approx p_\theta(z \mid x)$$

Variational Autoencoders

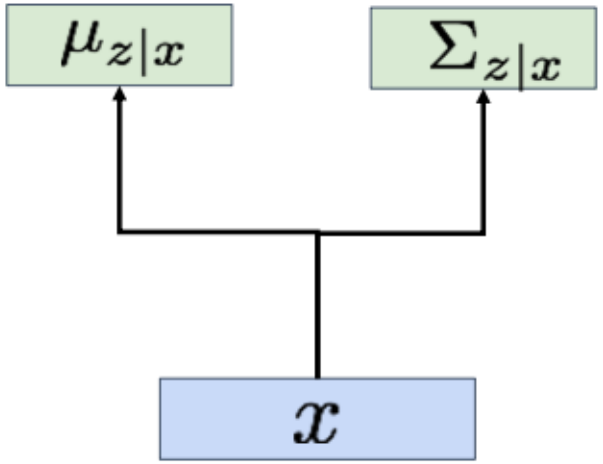
Decoder network inputs latent code z, gives distribution over data x

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



Encoder network inputs data x, gives 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

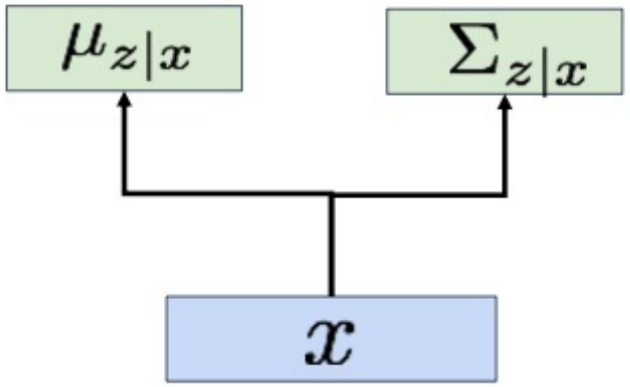
Variational Autoencoders

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

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

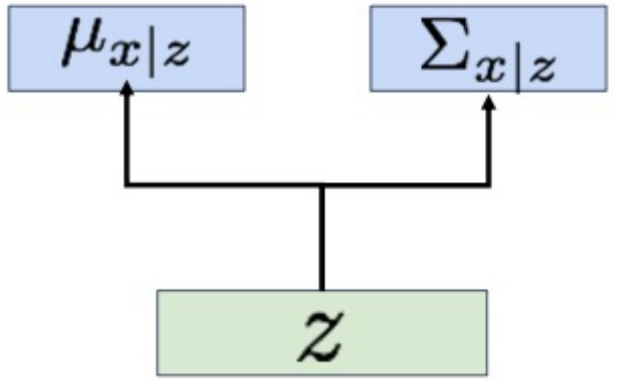
Encoder Network

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



Decoder Network

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



Skipping the proof of how the lower bound is computed.

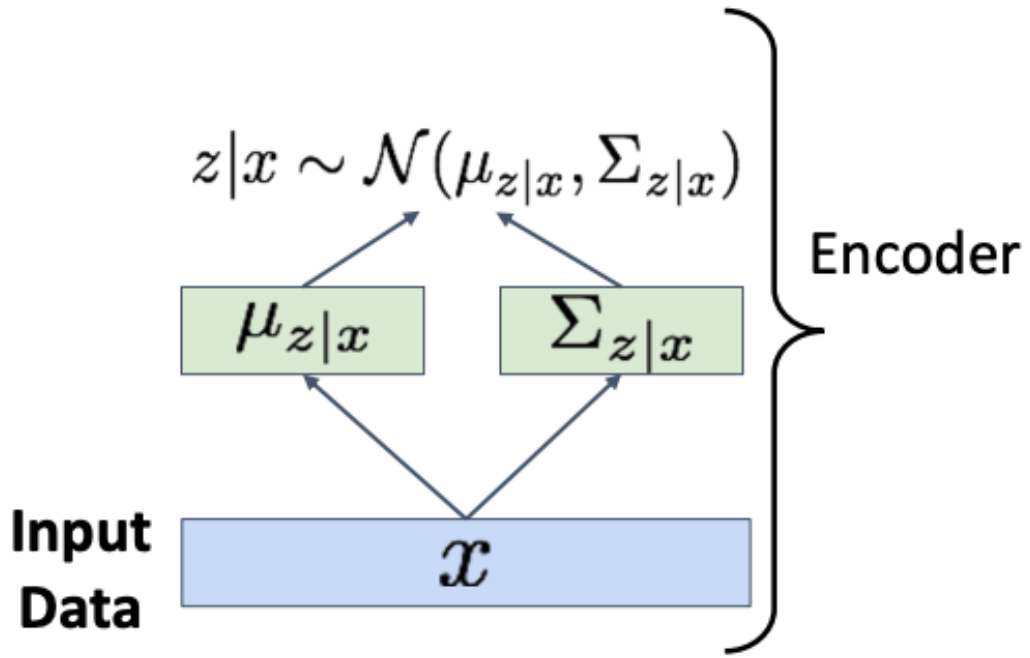
reading 1, reading 2

Variational Autoencoders

Train by maximizing the
variational lower bound

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

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

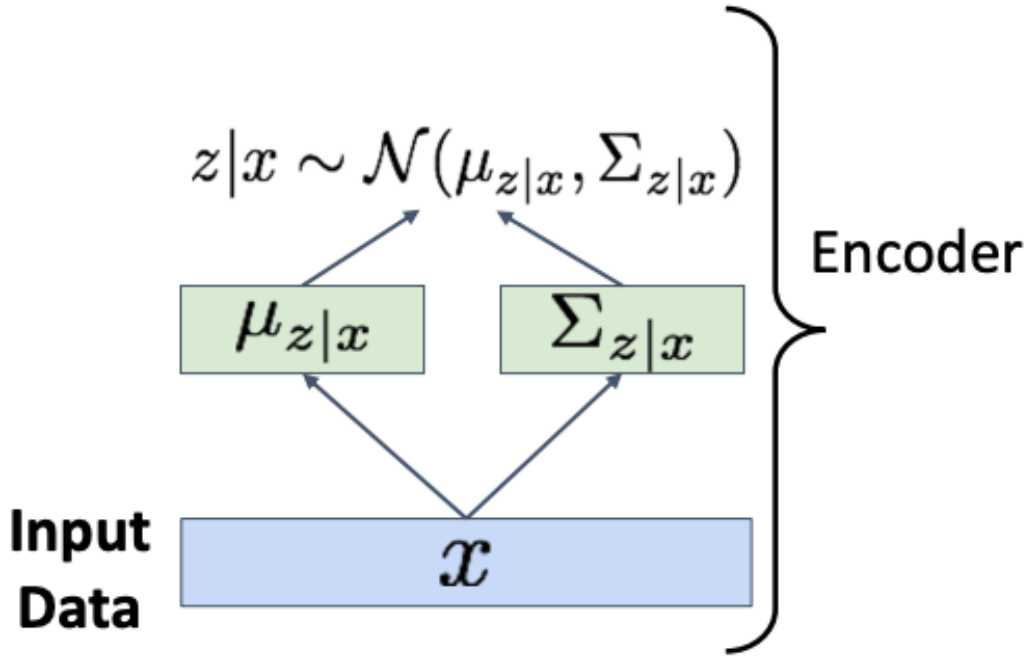


Variational Autoencoders

Train by maximizing the
variational lower bound

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

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



Variational Autoencoders

Train by maximizing the **variational lower bound**

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

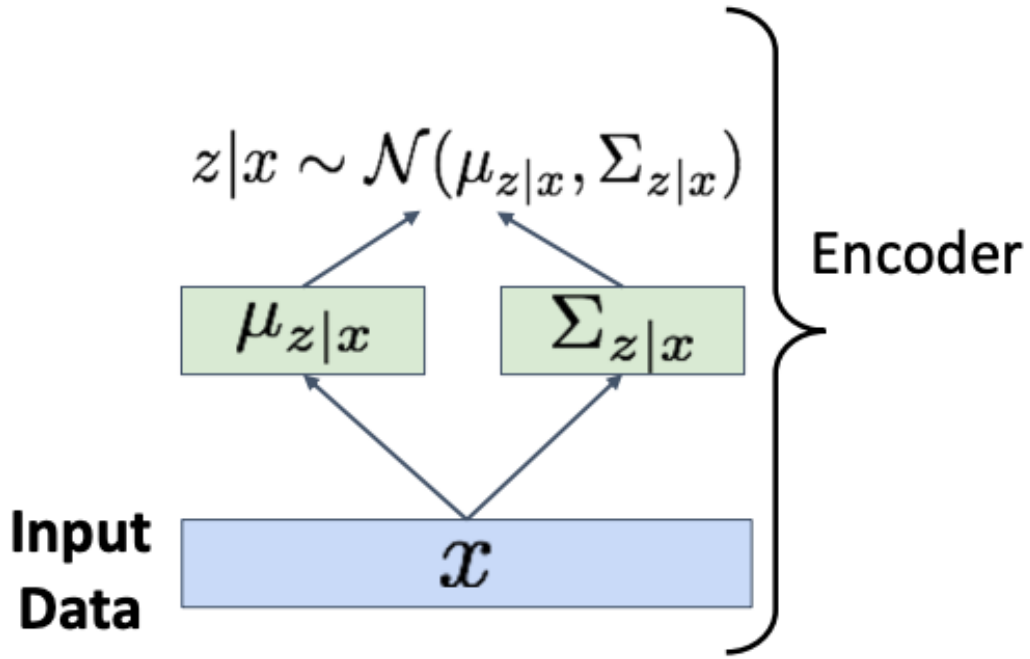
Closed form solution when q_ϕ is diagonal Gaussian and p is unit Gaussian! (Assume z has dimension J)

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

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

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

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

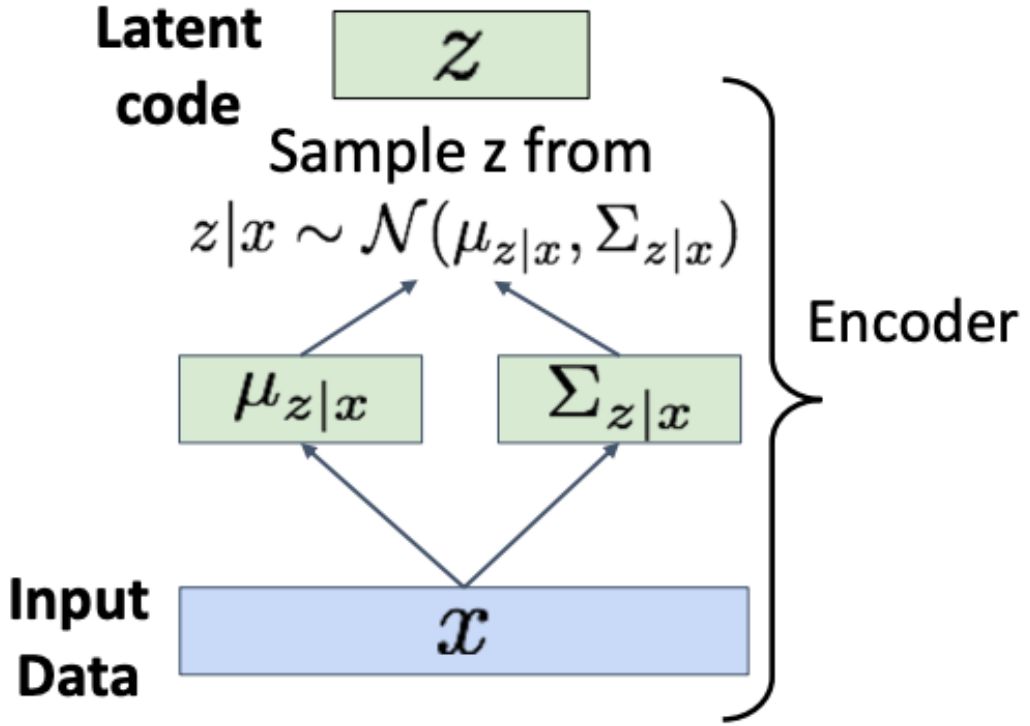


Variational Autoencoders

Train by maximizing the
variational lower bound

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

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

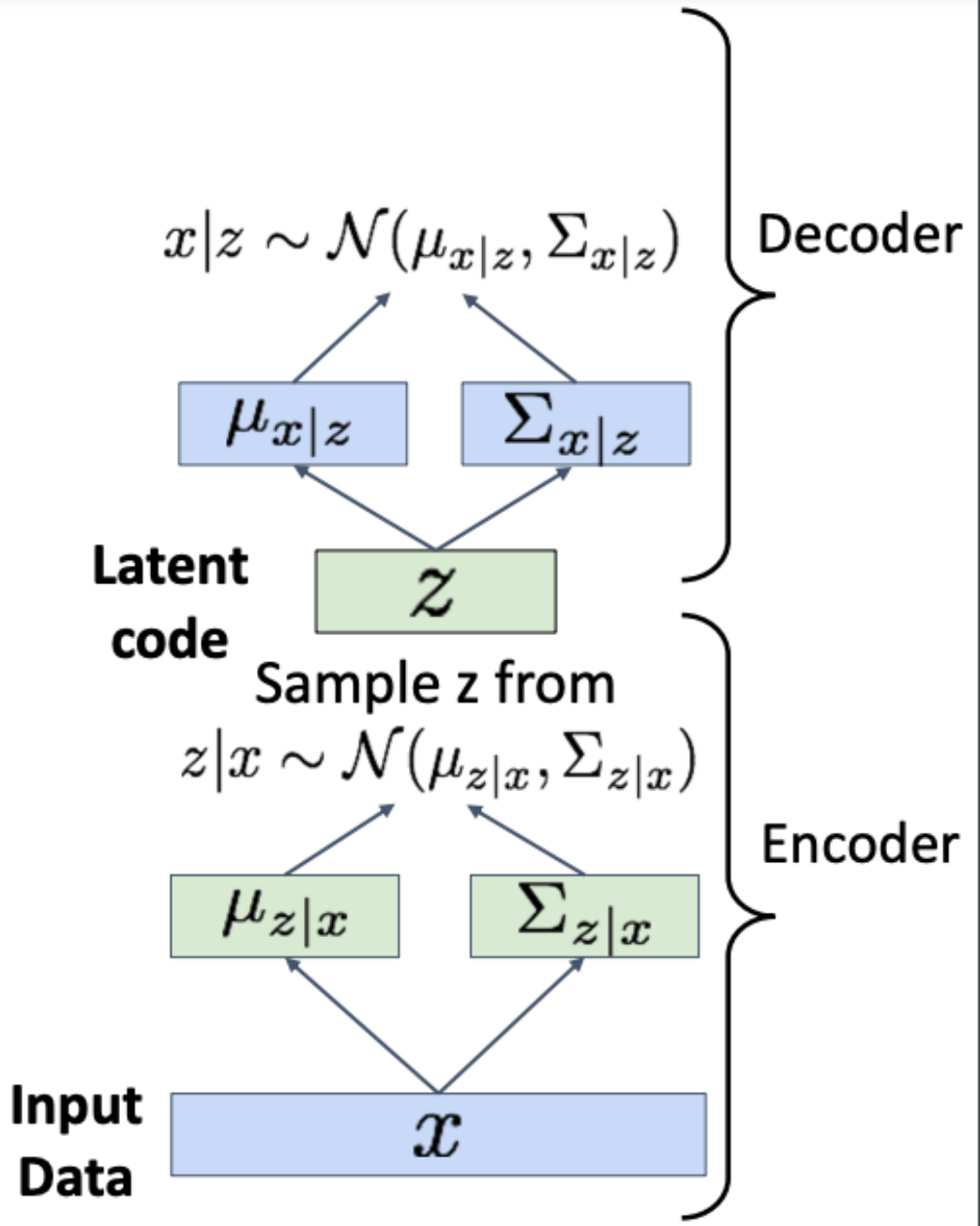


Variational Autoencoders

Train by maximizing the **variational lower bound**

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

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

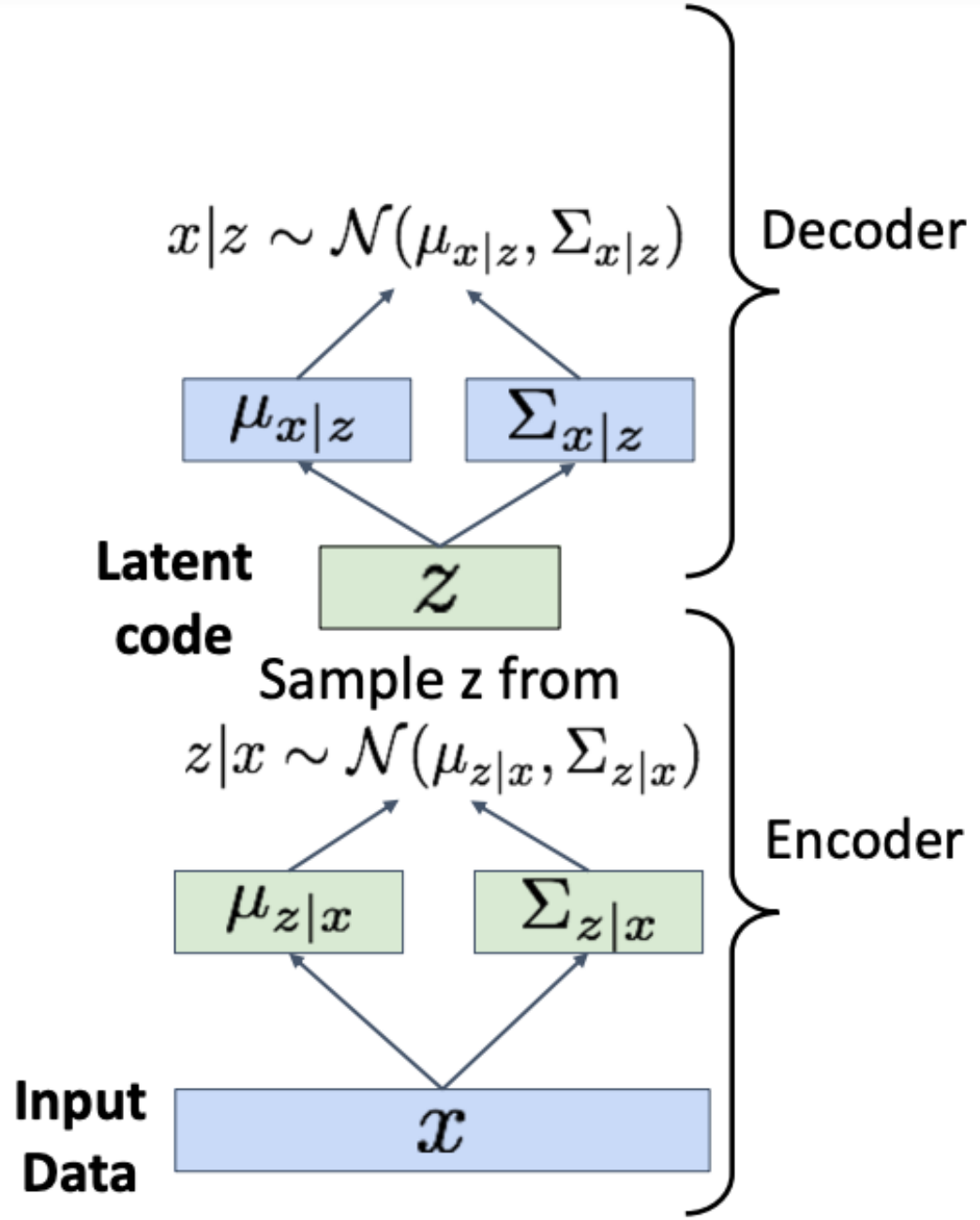


Variational Autoencoders

Train by maximizing the **variational lower bound**

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

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

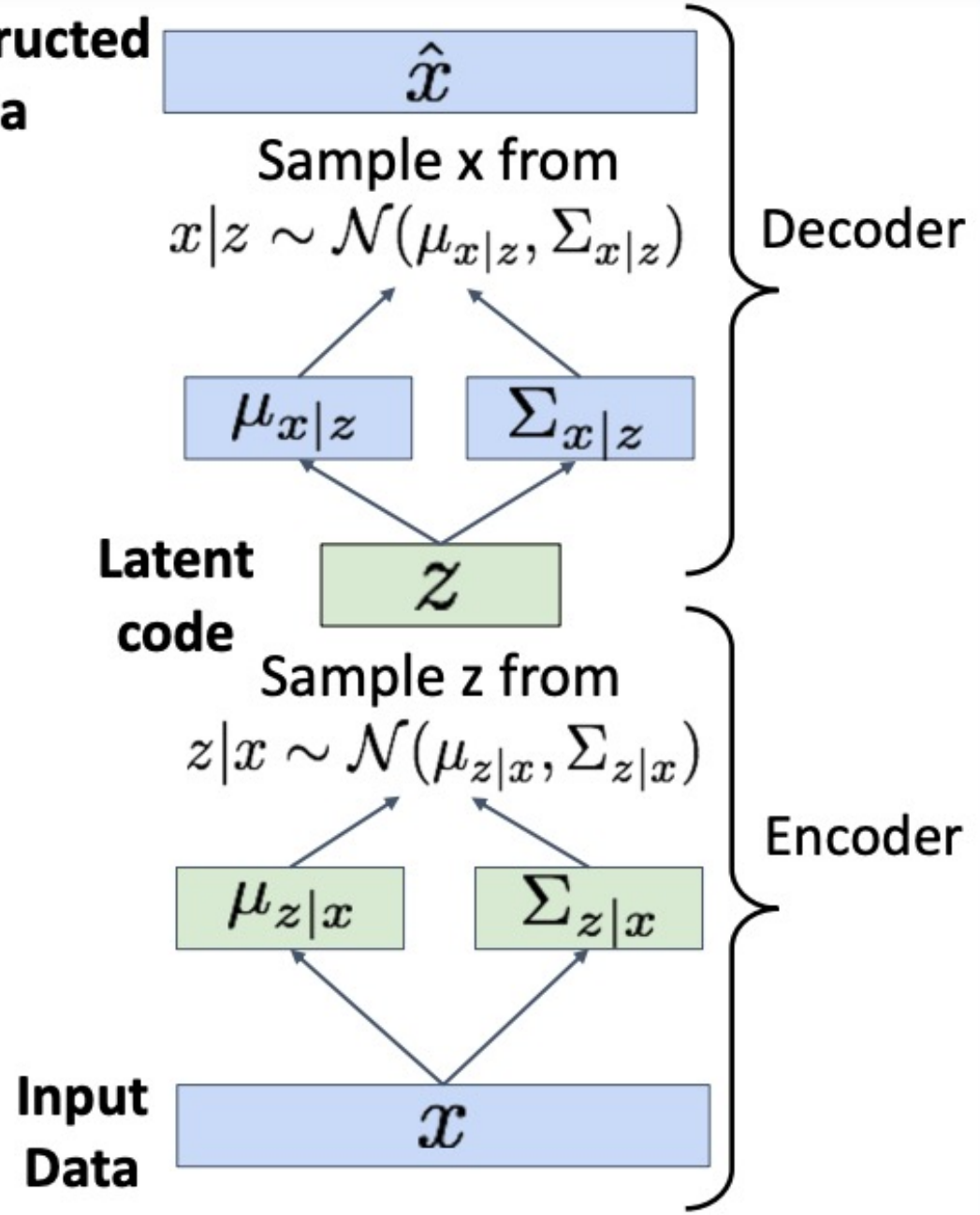


Variational Autoencoders

Train by maximizing the **variational lower bound**

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

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



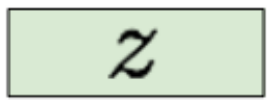
We minimize the reconstruction loss and make the latent space Z as gaussian as possible.

Variational Autoencoders: Generating Data

After training we can generate new data!

1. Sample z from prior p(z)

Latent code

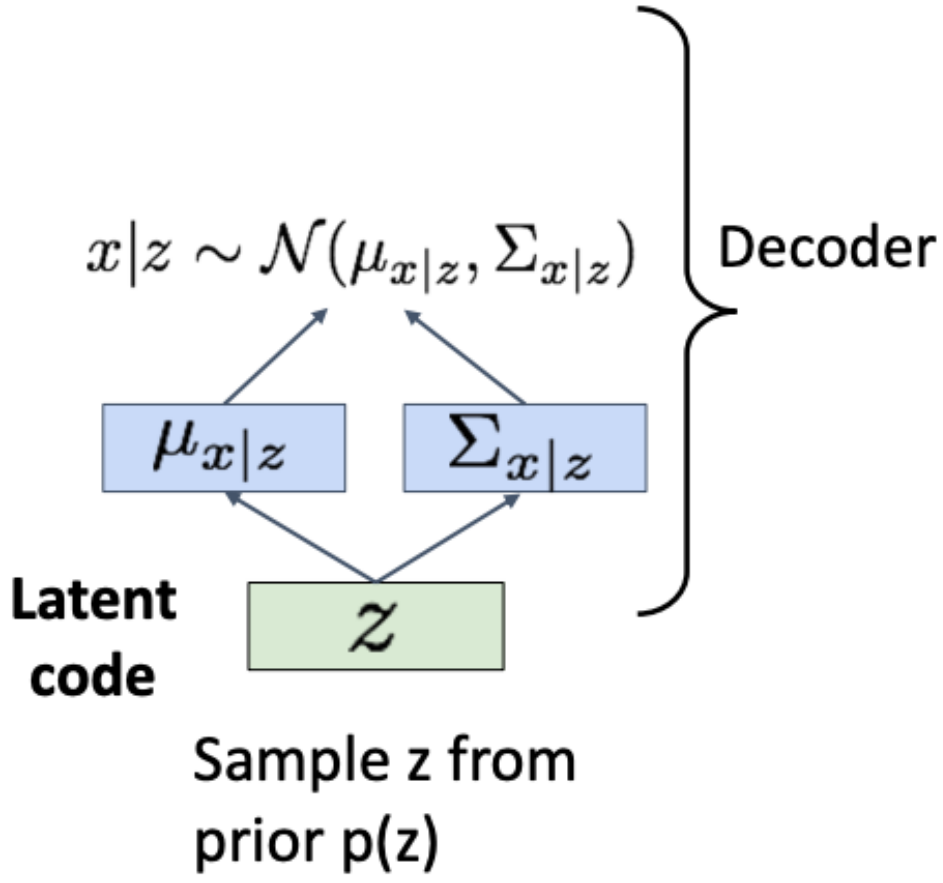


Sample z from prior p(z)

Variational Autoencoders: Generating Data

After training we can generate new data!

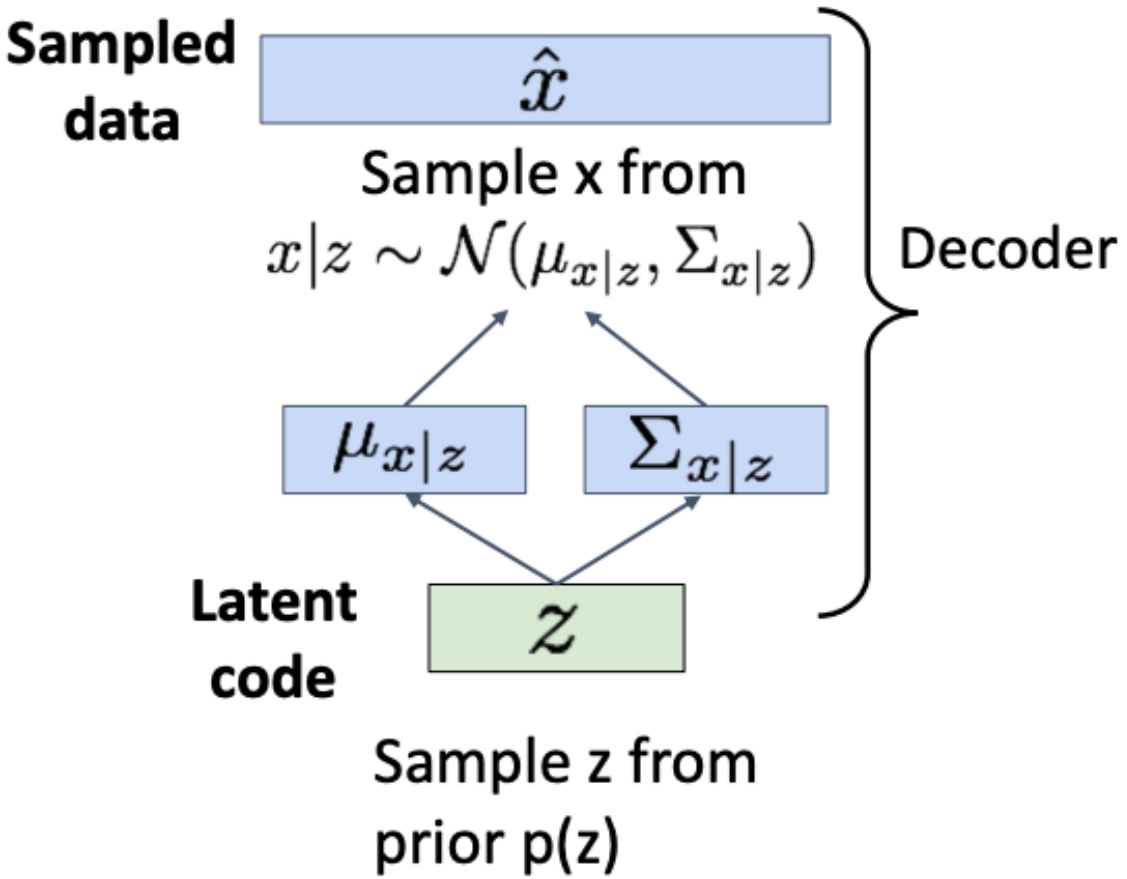
1. Sample z from prior p(z)
2. Run sampled z through decoder to get distribution over data x



Variational Autoencoders: Generating Data

After training we can generate new data!

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



Variational Autoencoders: Generating Data

32x32 CIFAR-10

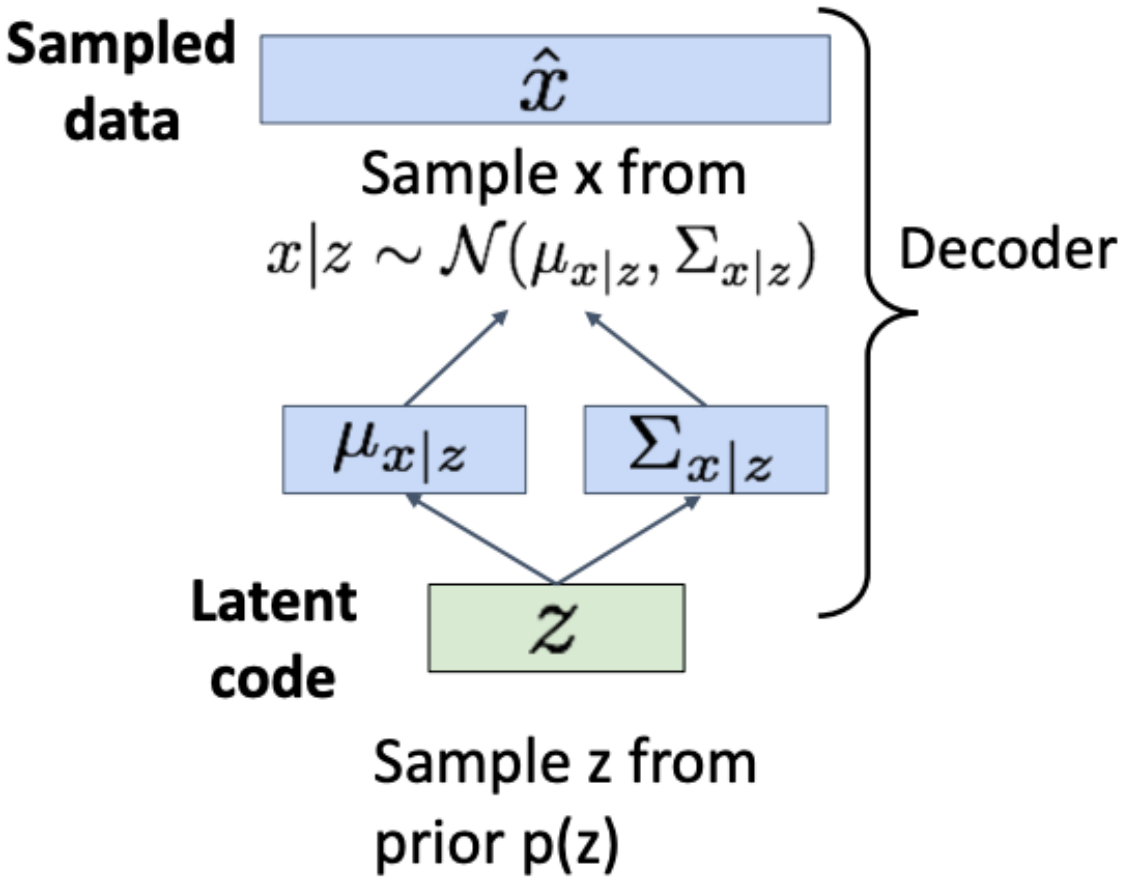
Labeled Faces in the Wild

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

Variational Autoencoders: Generating Data

After training we can generate new data!

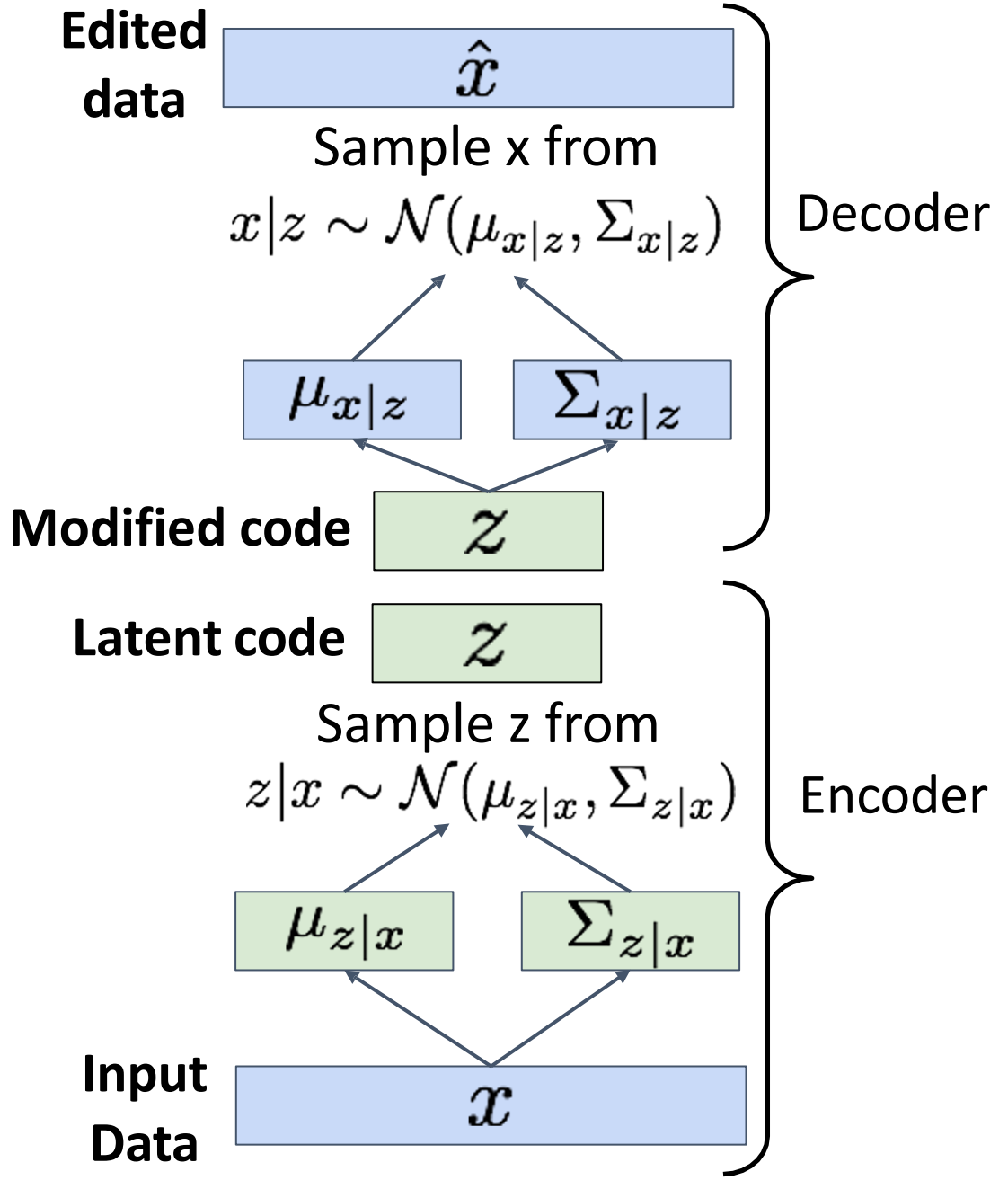
1. Sample z from prior p(z)
2. Run sampled z through decoder to get distribution over data x
3. Sample from distribution in (2) to generate data



Variational Autoencoders

After training we can **edit images**

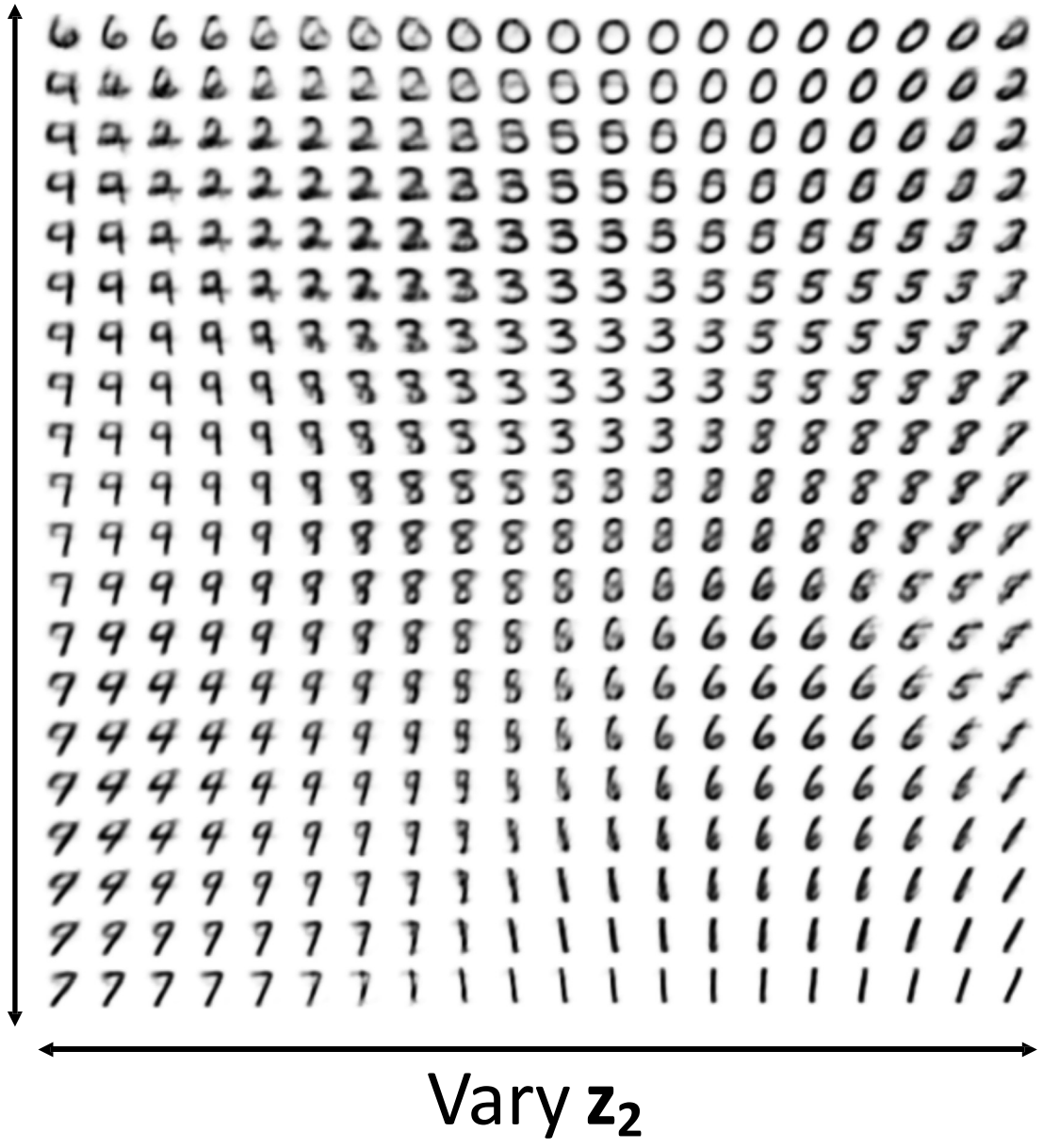
1. Run input data through **encoder** to get a distribution over latent codes
2. Sample code z from encoder output
3. Modify some dimensions of sampled code
4. Run modified z through **decoder** to get a distribution over data samples
5. Sample new data from (4)



Variational Autoencoders

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

“Disentangling factors of variation”



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

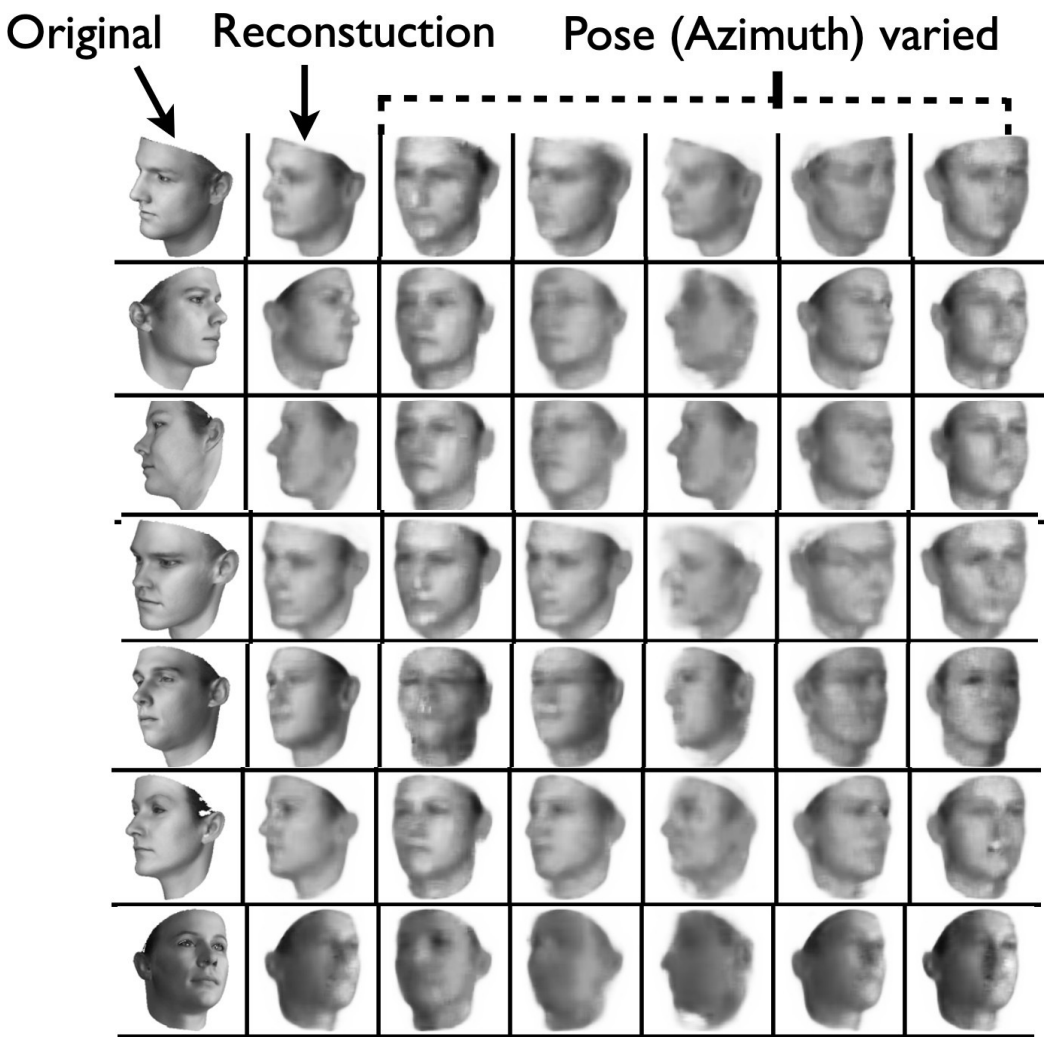
Variational Autoencoders

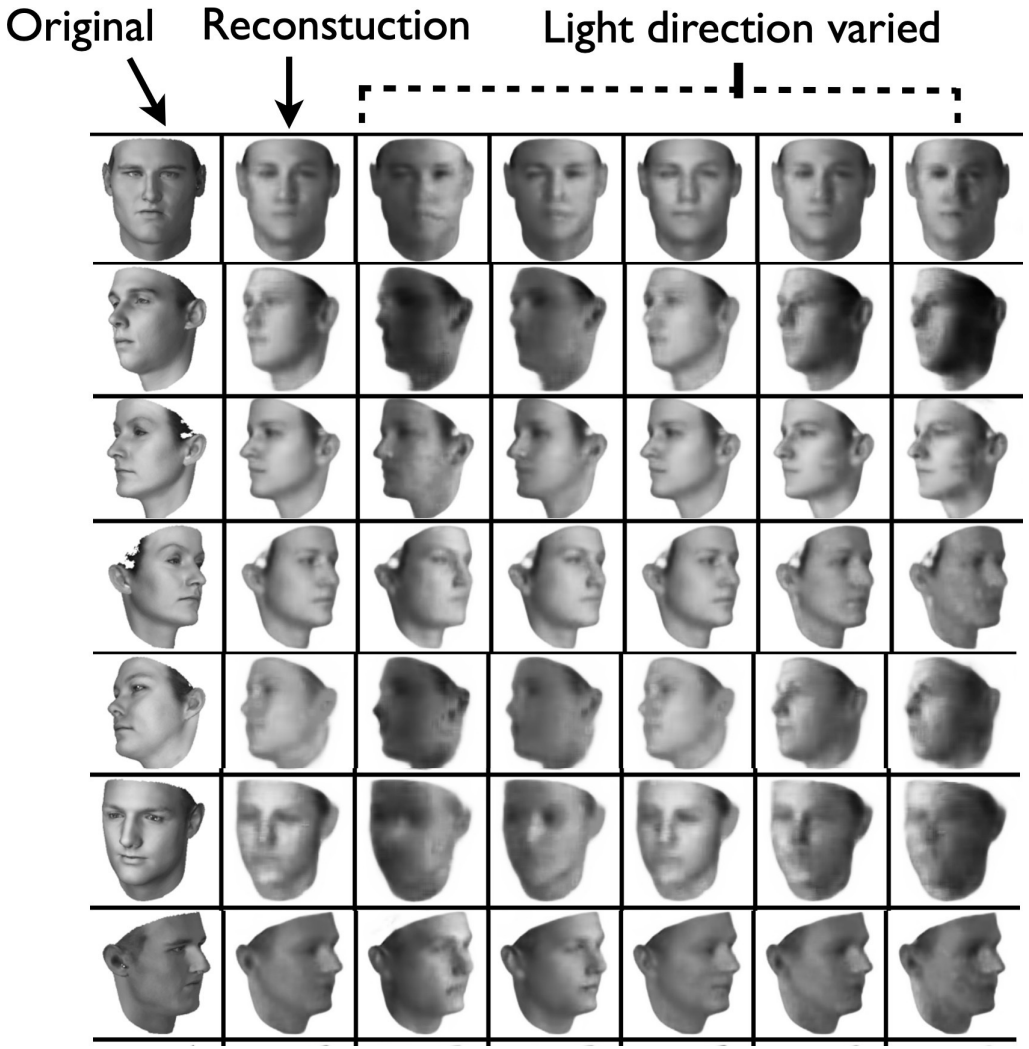
The diagonal prior on p(z) causes dimensions of z to be independent

“Disentangling factors of variation”

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

Variational Autoencoders: Image Editing





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

Variational Autoencoder: Summary

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

Pros:

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

Cons:

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

Active areas of research:

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

So far: Two types of generative models

Autoregressive models

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

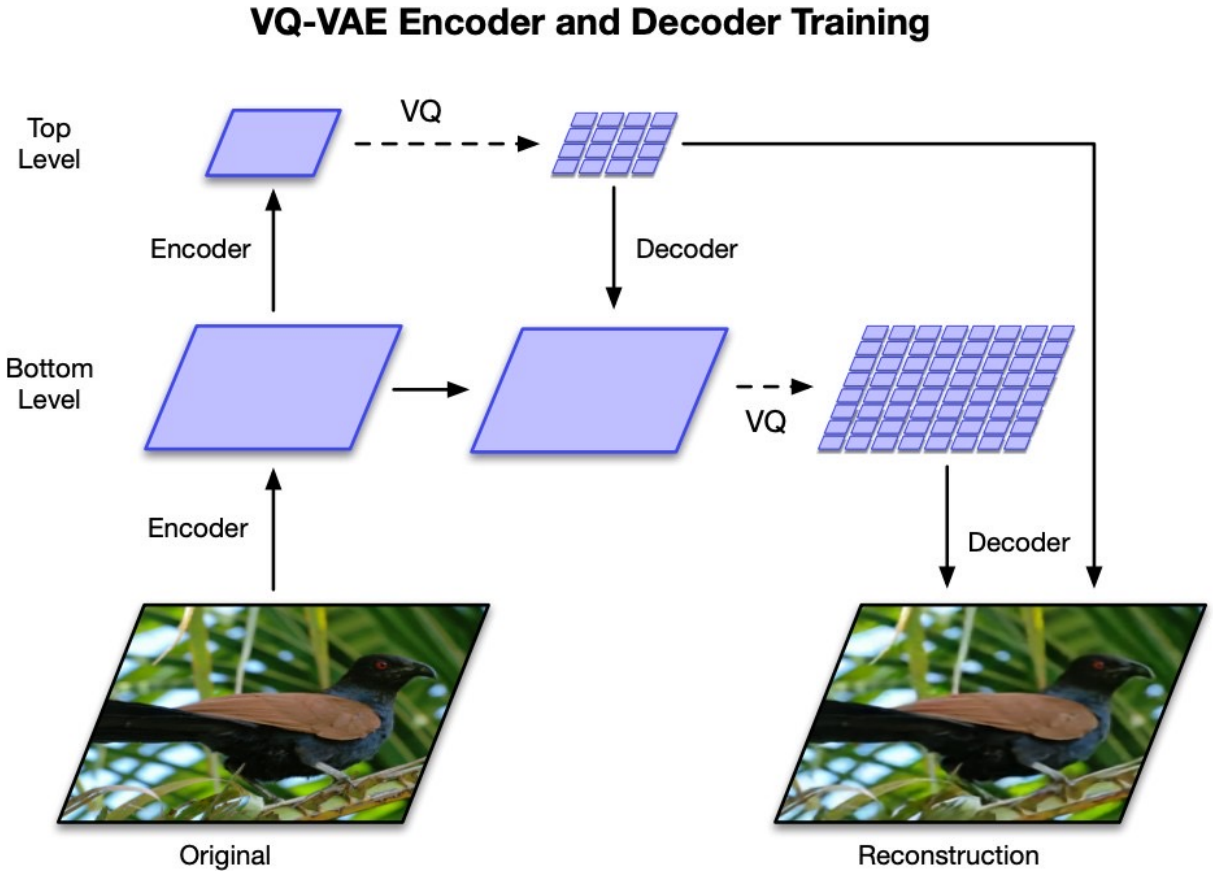
Variational models

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

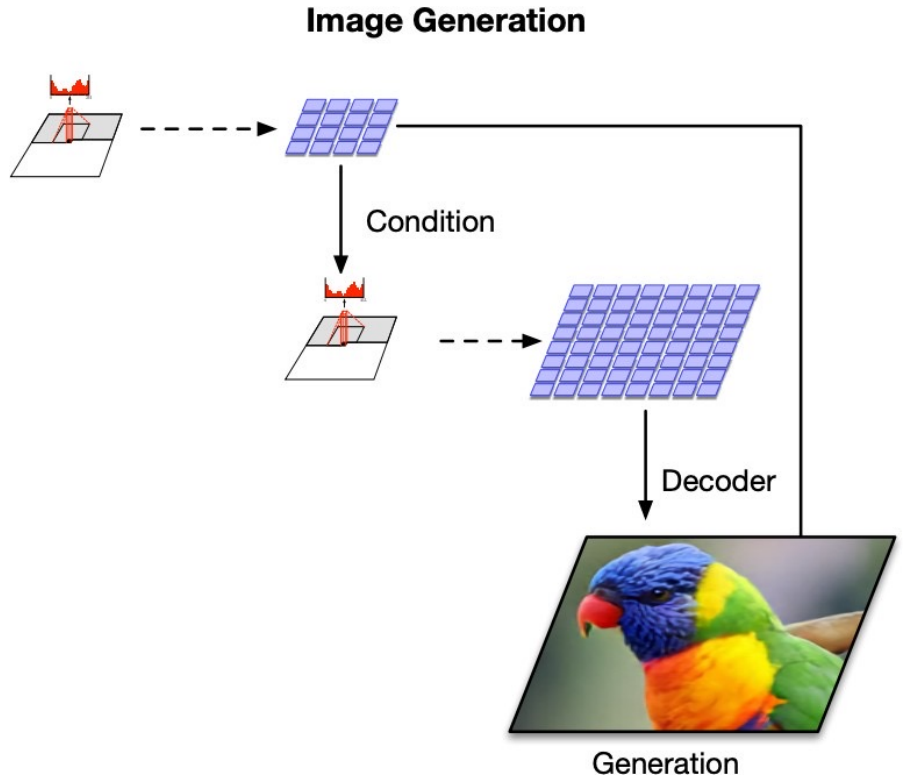
Can we combine them and get the best of both worlds?

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

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



Use a multiscale PixelCNN to sample in latent code space



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

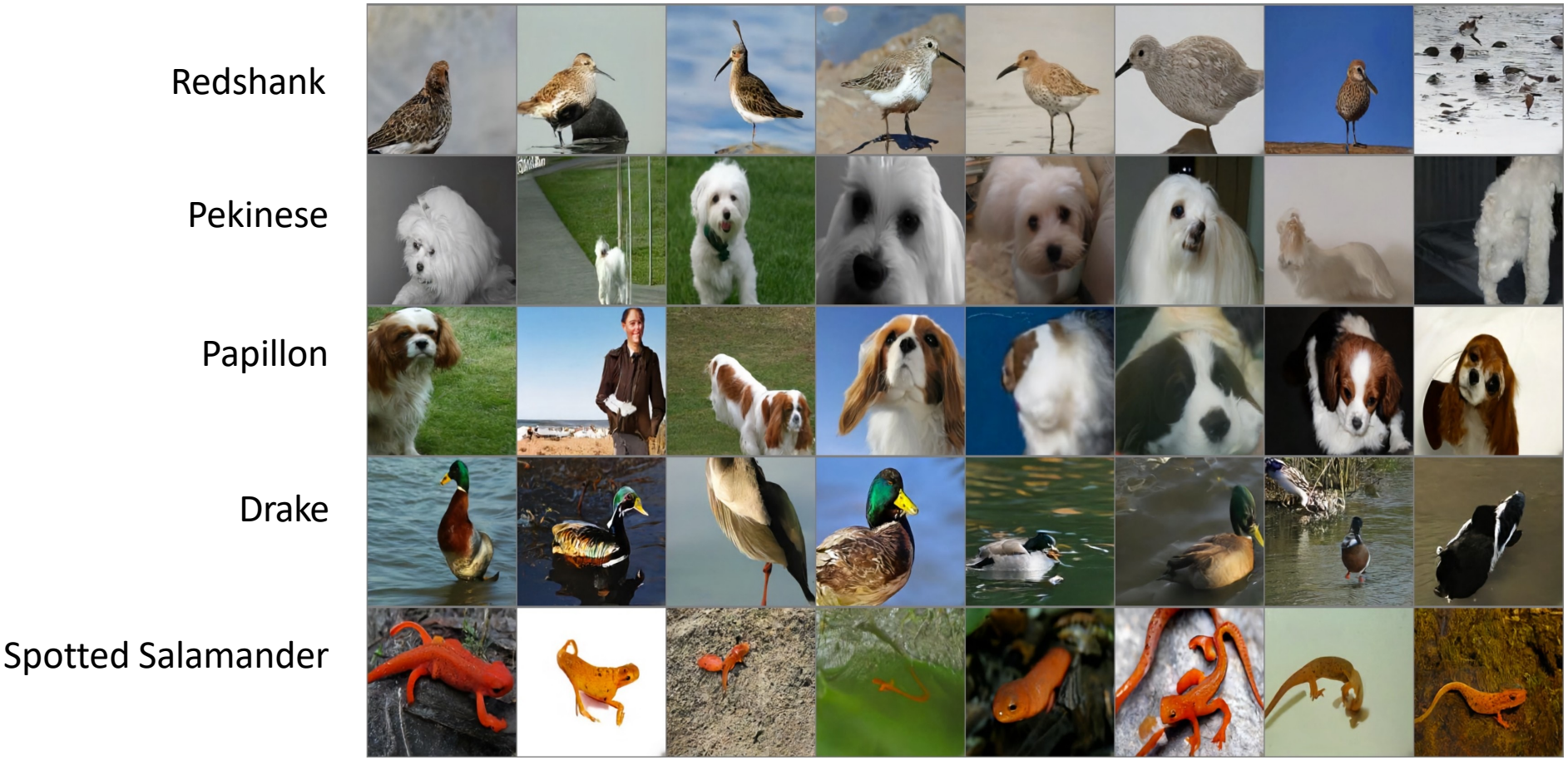
Combining VAE + Autoregressive: VQ-VAE2

256 x 256 class-conditional samples, trained on ImageNet

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

Combining VAE + Autoregressive: VQ-VAE2

256 x 256 class-conditional samples, trained on ImageNet



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

Combining VAE + Autoregressive: VQ-VAE2

1024 x 1024 generated faces, trained on FFHQ

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

Combining VAE + Autoregressive: VQ-VAE2

1024 x 1024 generated faces, trained on FFHQ

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

Next few lectures: Generative models for direct image based rendering.

Current Image

Latent Space of Generative models

Vision

Encoder

3D Intrinsic Components

Edit Components

Graphics

Decoder

New Image under different conditions

Change:

- Viewpoint
- Lighting
- Reflectance
- Background
- Attributes
- Many others…

Implicit: Use a Neural Network (Conditional Generative networks)
Often, end-to-end.

Taxonomy of Generative Models

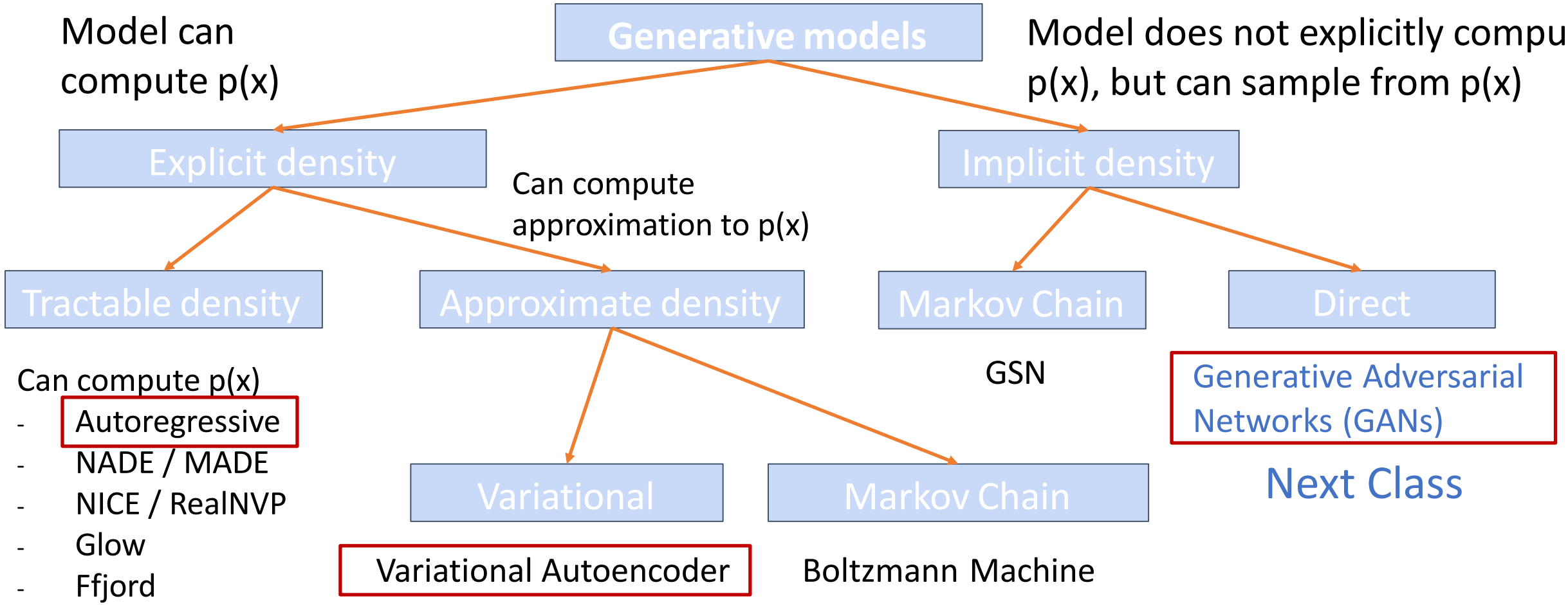


Figure adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Slide Credits

- EECS 6322 Deep Learning for Computer Vision, Kosta Derpanis (York University)
- EECS 498 Deep Learning for Computer Vision, Justin Johnson (U. Michigan)
- Many amazing research papers!