AI6126 Advanced Computer Vision

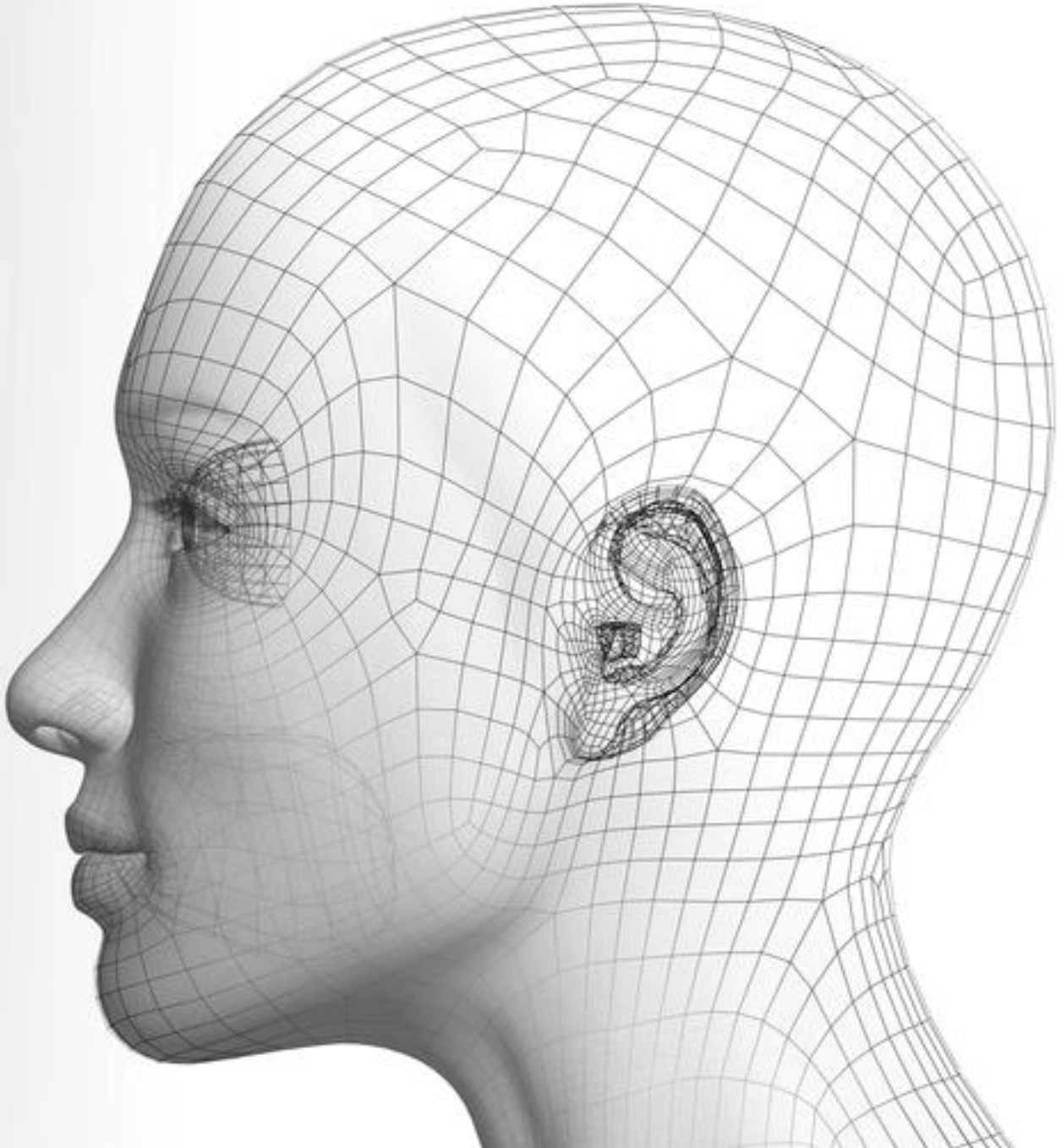
Last update: 22 February 2022

Autoencoder

Ziwei Liu

刘子纬

https://liuziwei7.github.io/



Slide Credits

- Justin Johnson, EECS 498/598
- Eurographics 2018 Tutorial, Deep Learning for Graphics
- Paper Authors

Outline

Sup. vs Unsup. Learning, Discriminative vs Generative Models

Autoencoder (AE):

- Avoid Shortcut Solution
- AE as Generative Model

Variational Autoencoder (VAE):

- Reparameterization Trick
- VAE as Generative Model

Open Problems

Supervised vs Unsupervised Learning

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

Supervised vs Unsupervised Learning

Supervised Learning

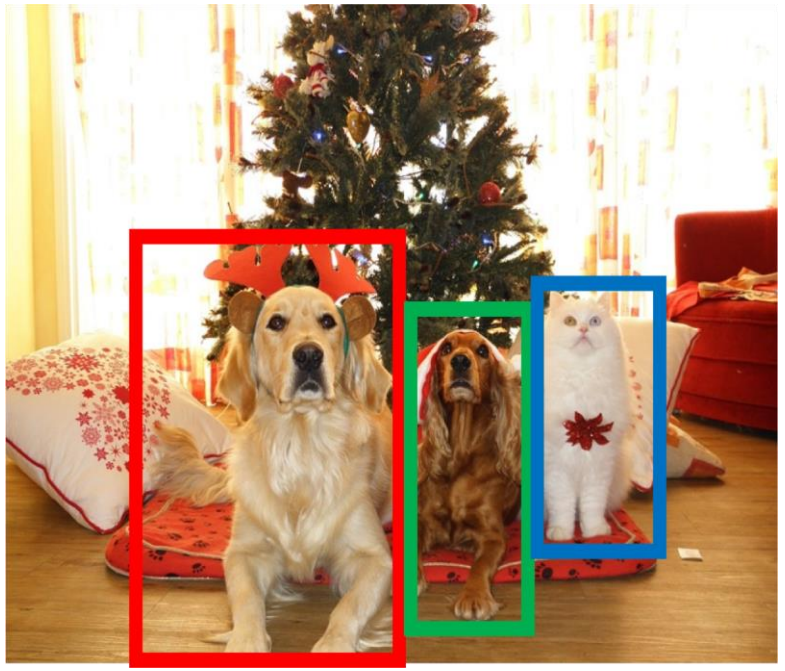
Data: (x, y)

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

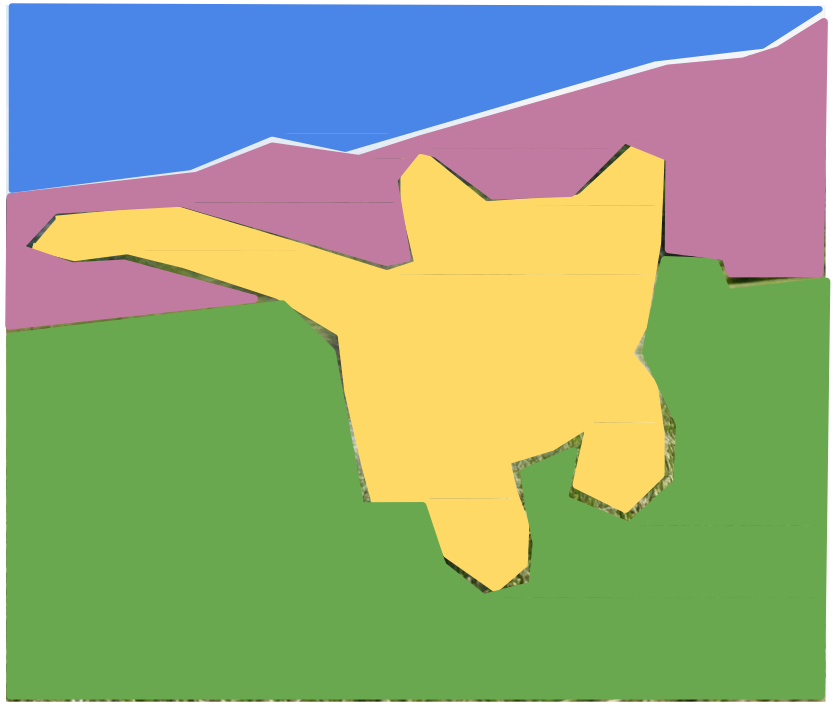
Data: (x, y)

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

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

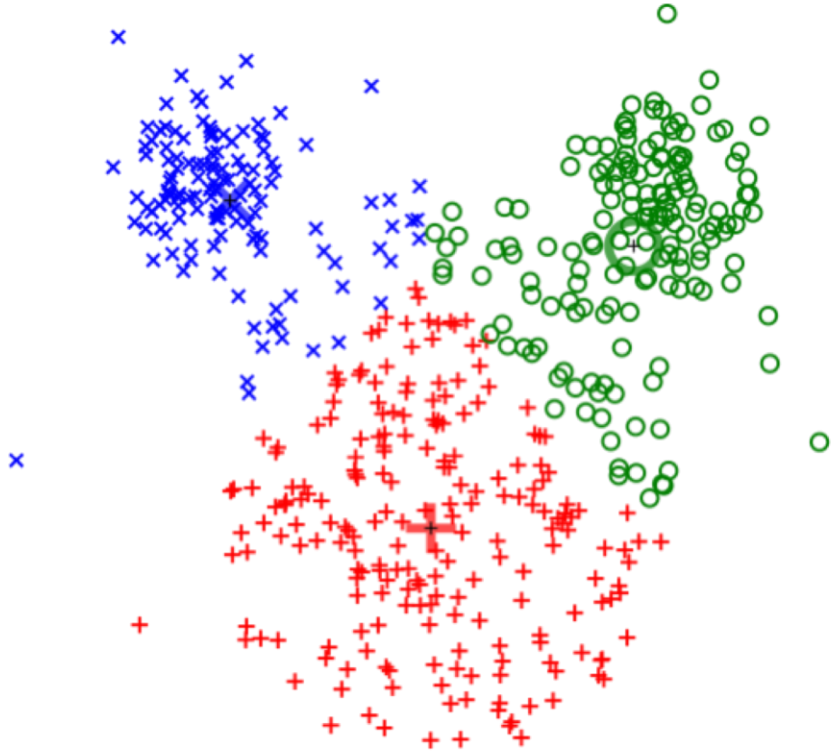
Just data, no labels!

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

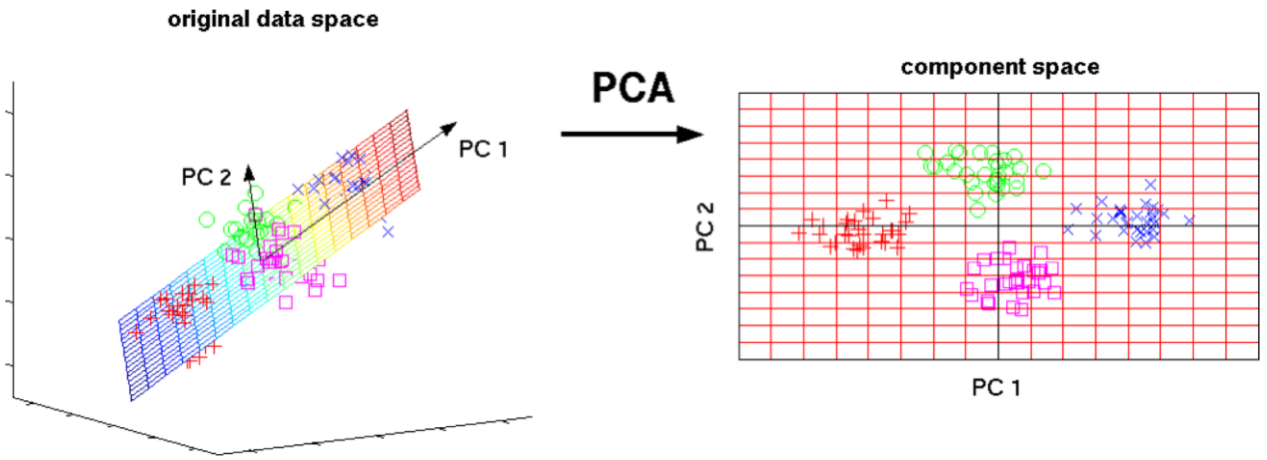
Just data, no labels!

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

Unsupervised Learning

Data: x

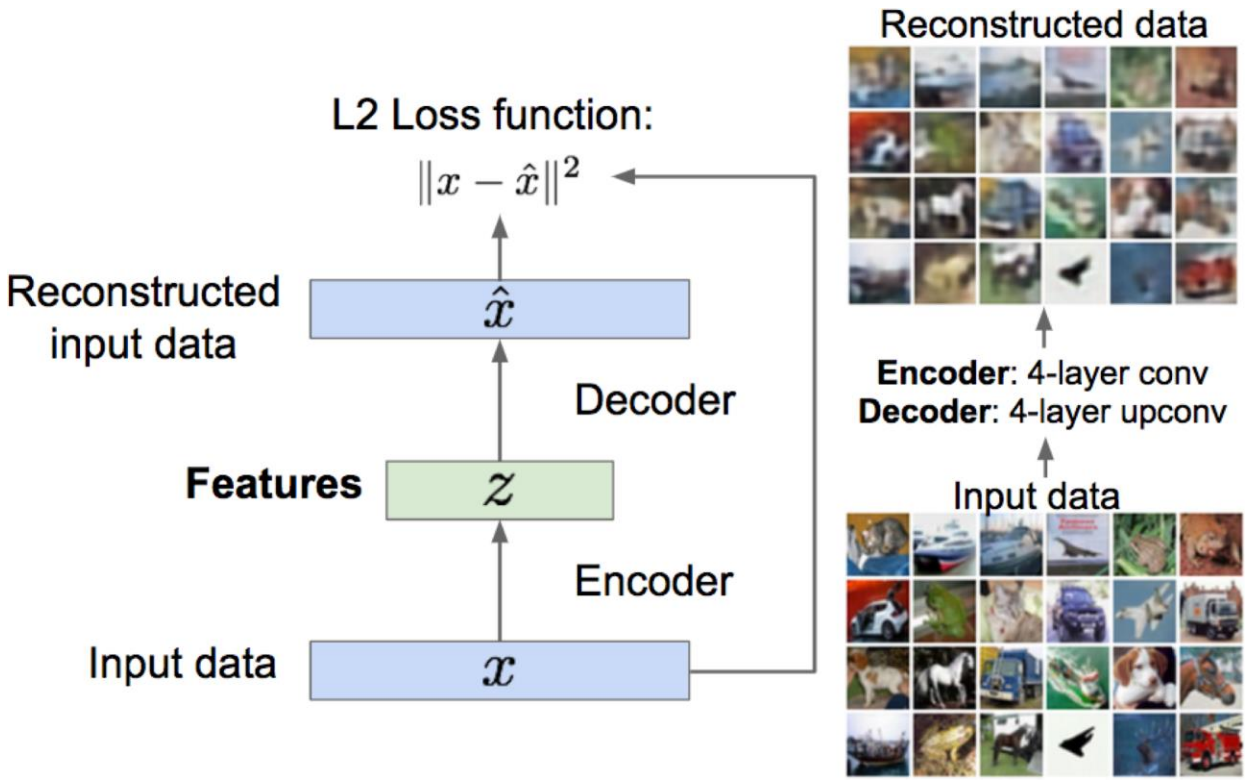
Just data, no labels!

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

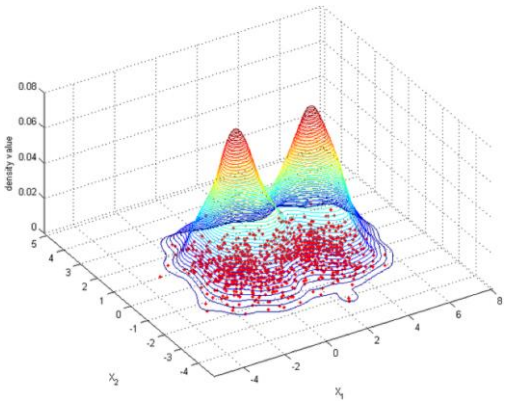
Just data, no labels!

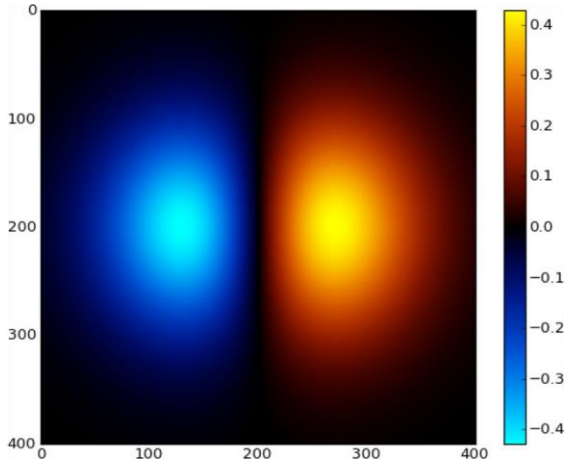
Goal: Learn some underlying hidden *structure* of the data

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

Supervised vs Unsupervised Learning

Density Estimation





Images left and right are 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

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 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**:

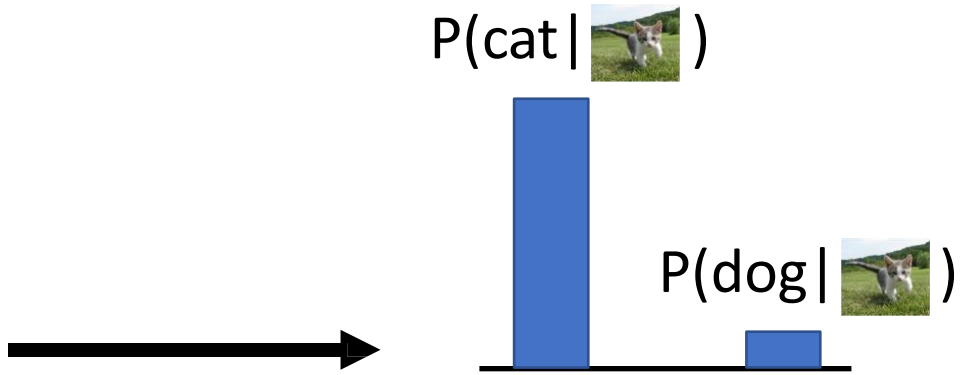
$$\int p(\mathrm{x})dx = 1$$

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)

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

Density functions are **normalized**:

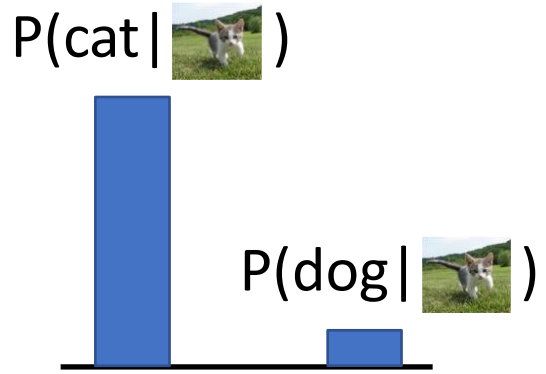
$$\int p(\mathrm{x})dx = 1$$

Conditional Generative Model: Learn p(x|y)

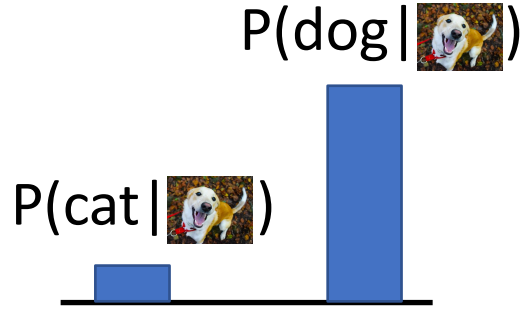
Different values of x **compete** for density

Discriminative vs Generative Models

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



Generative Model: Learn a probability distribution p(x)

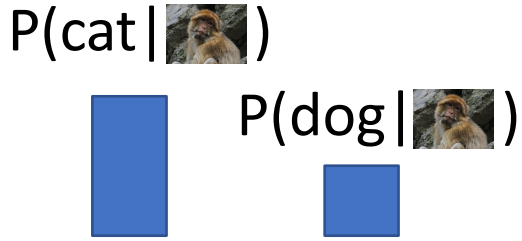


Conditional Generative Model: Learn p(x|y)

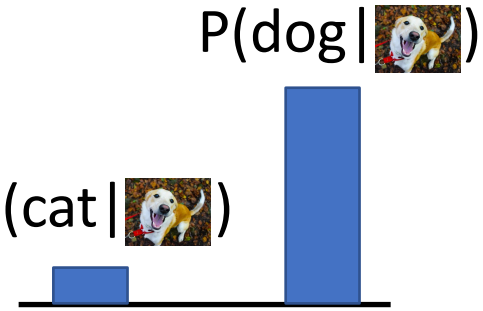
Discriminative model: the possible labels for each input ”compete” for probability mass. But no competition between **images**

Discriminative vs Generative Models

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



Generative Model: Learn a probability distribution p(x)

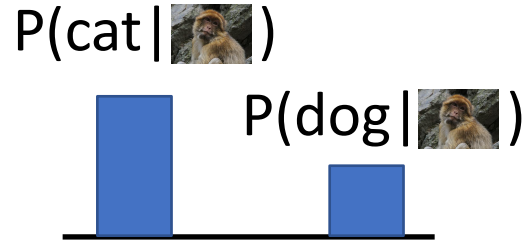


Conditional Generative Model: Learn p(x|y)

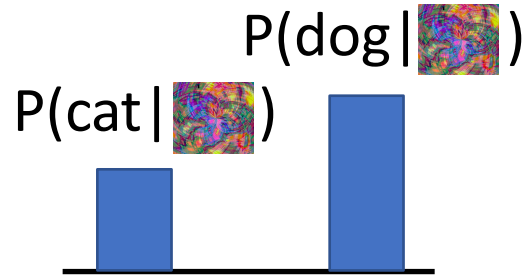
Discriminative model: No way for the model to handle unreasonable inputs; it must give label distributions for all images

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)

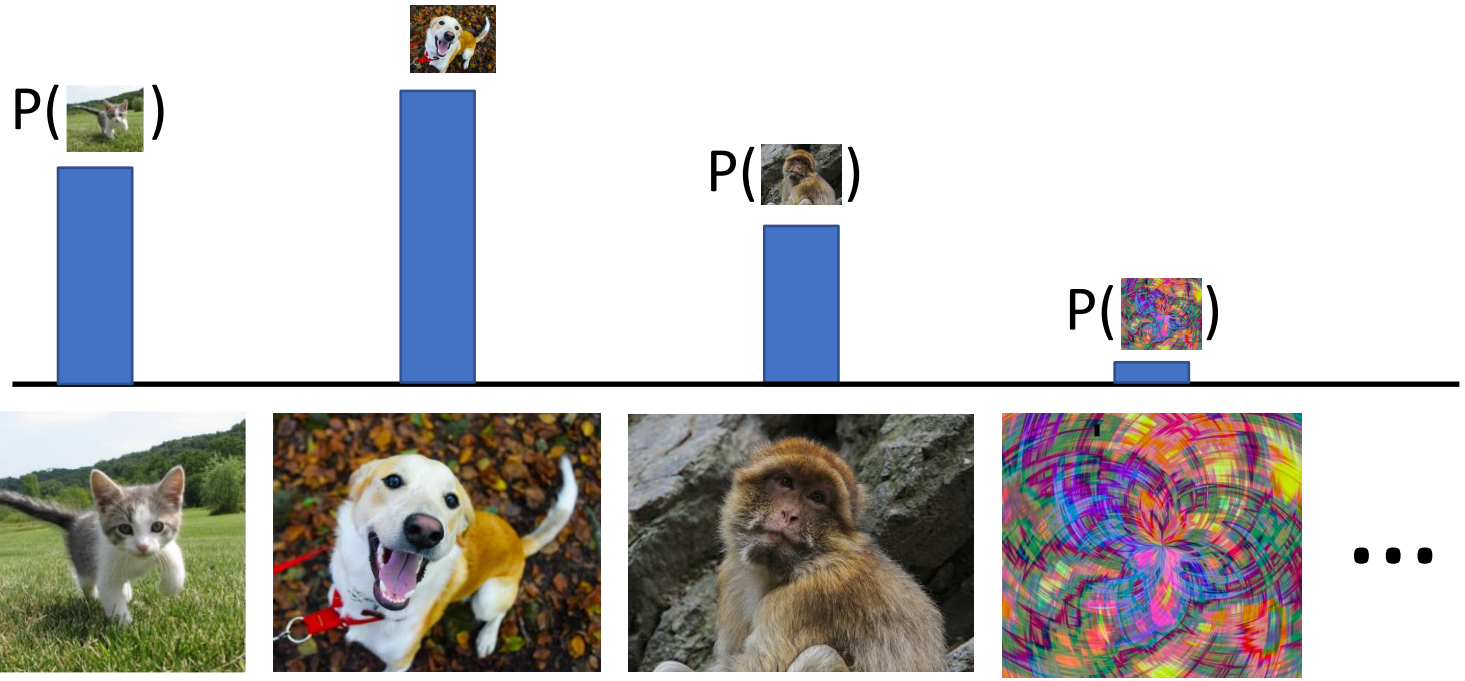
Discriminative model: No way for the model to handle unreasonable inputs; it must give label distributions for all images

Discriminative vs Generative Models

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

Generative Model: Learn a probability distribution p(x)

Conditional Generative Model: Learn p(x|y)



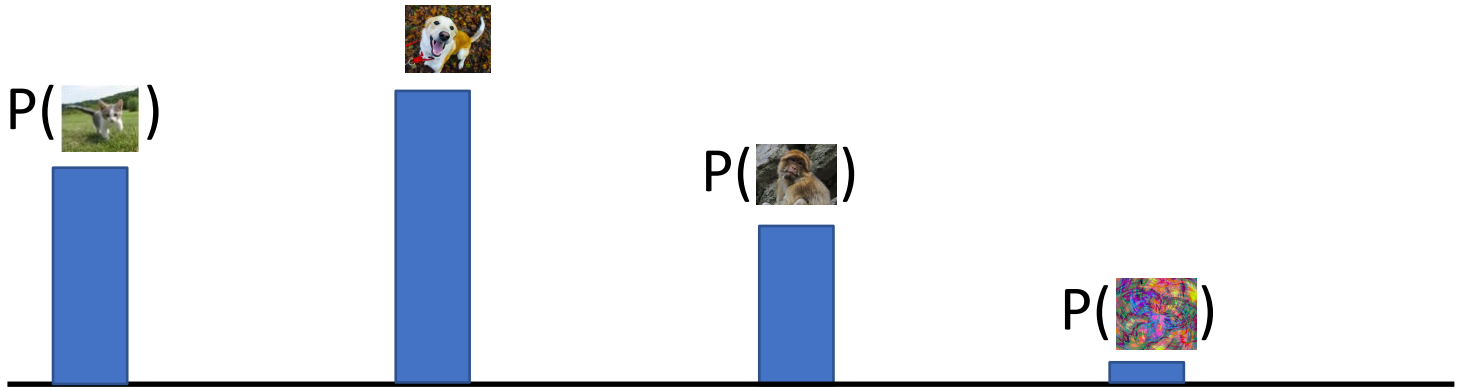
Generative model: All possible images compete with each other for probability mass

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

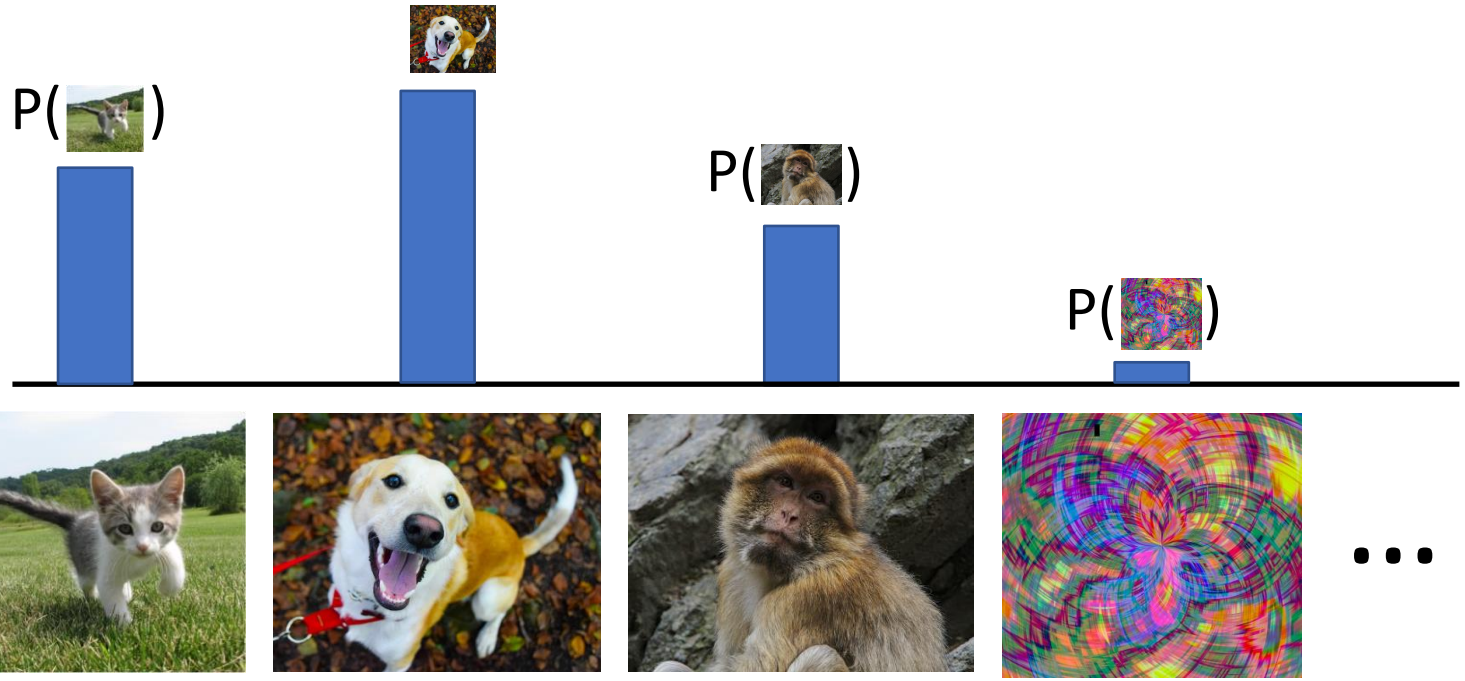
Requires deep image understanding! Is a dog more likely to sit or stand? How about 3-legged dog vs 3-armed monkey?

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

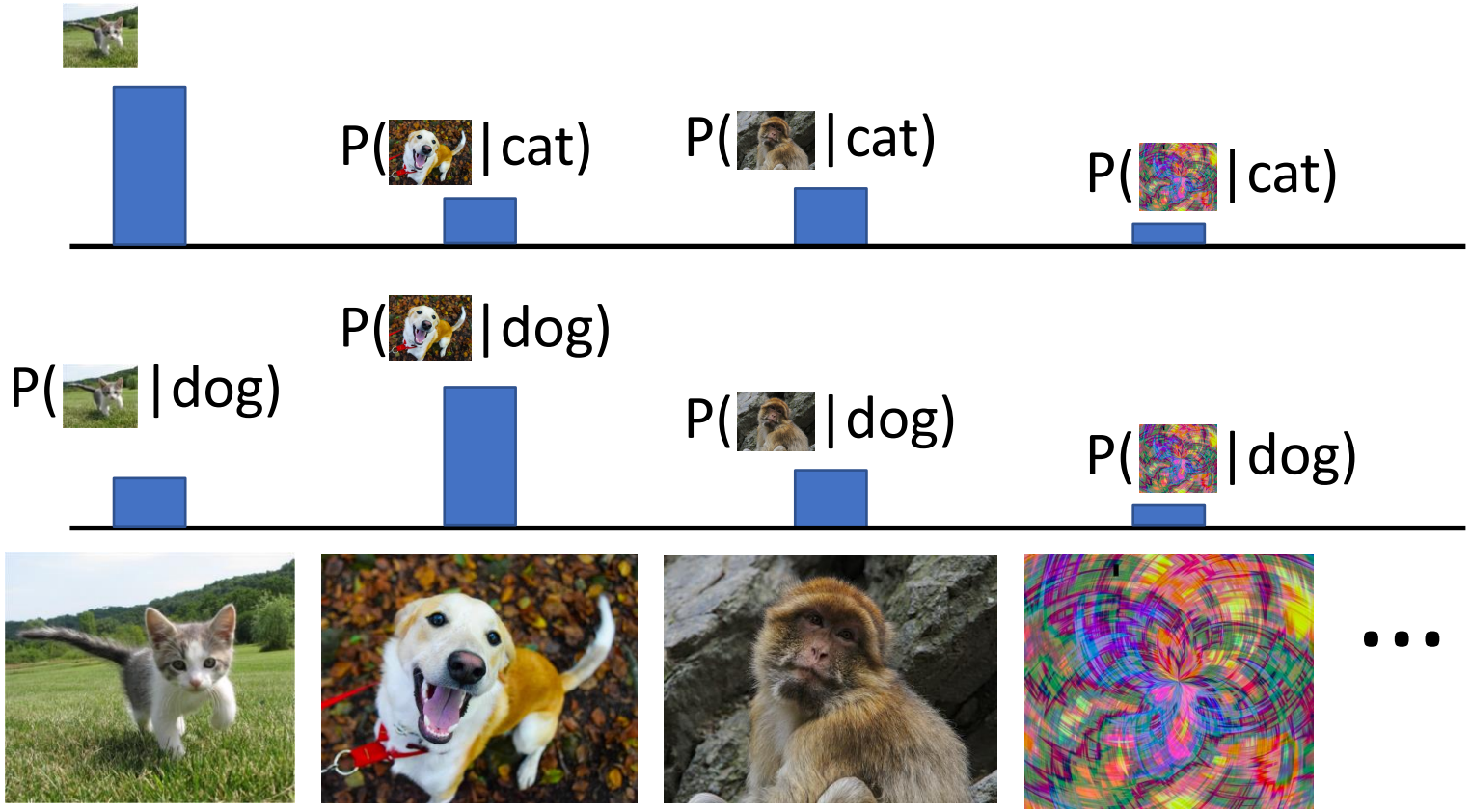
Model can “reject” unreasonable inputs by assigning them small values

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:**

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

Discriminative Model

(Unconditional) Generative Model

Conditional Generative Model

Prior over labels

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 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)

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

Conditional Generative Model: Learn p(x|y)

What can we do with a discriminative 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

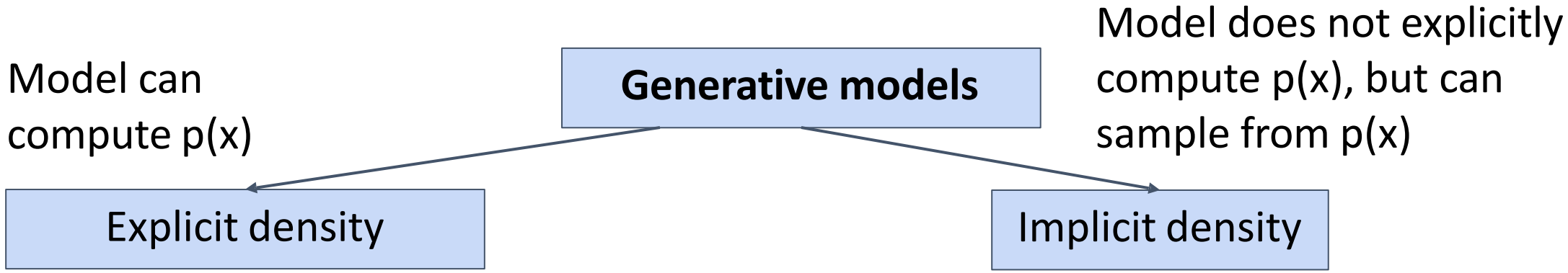
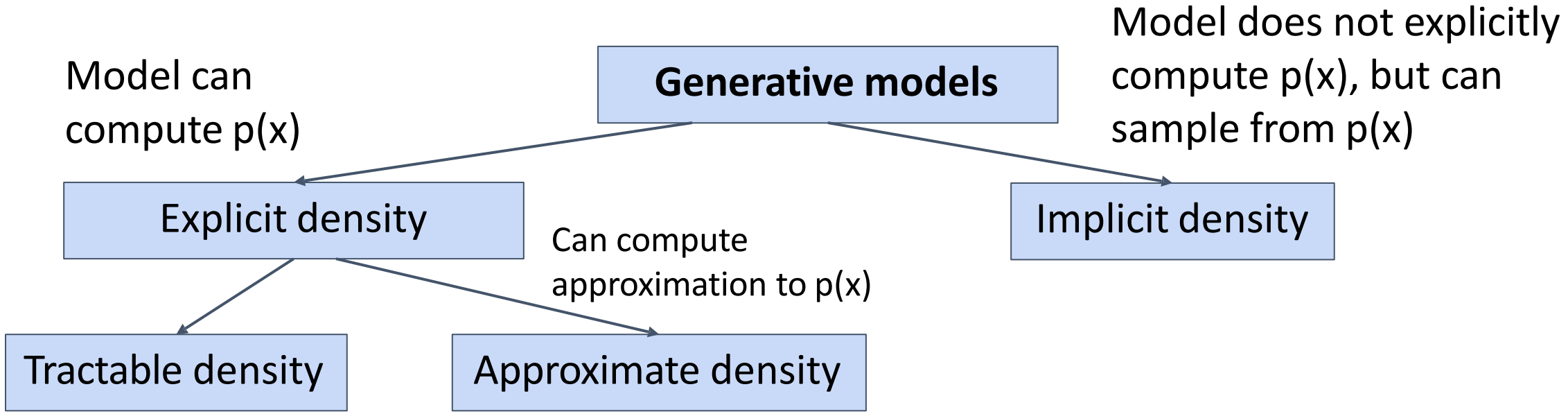


Figure adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

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

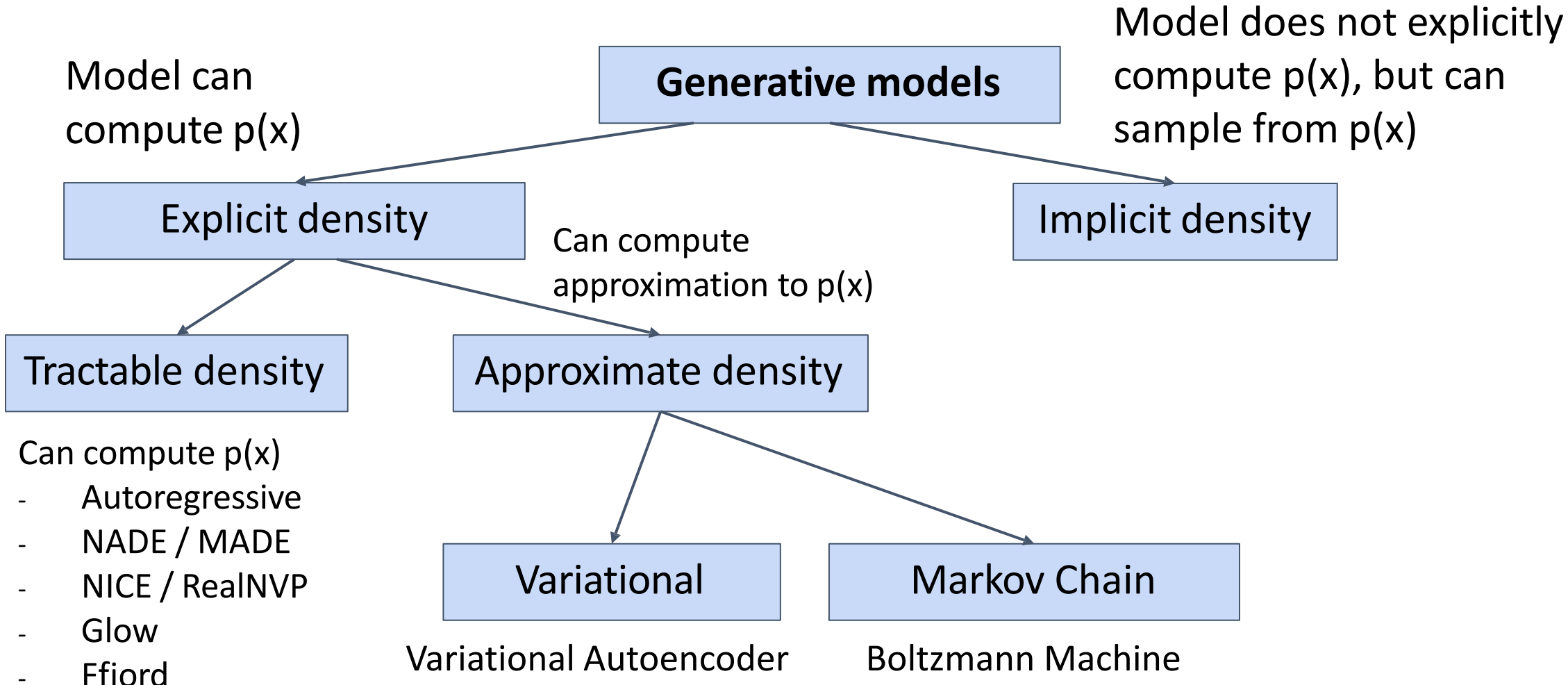


Figure adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Taxonomy of Generative Models

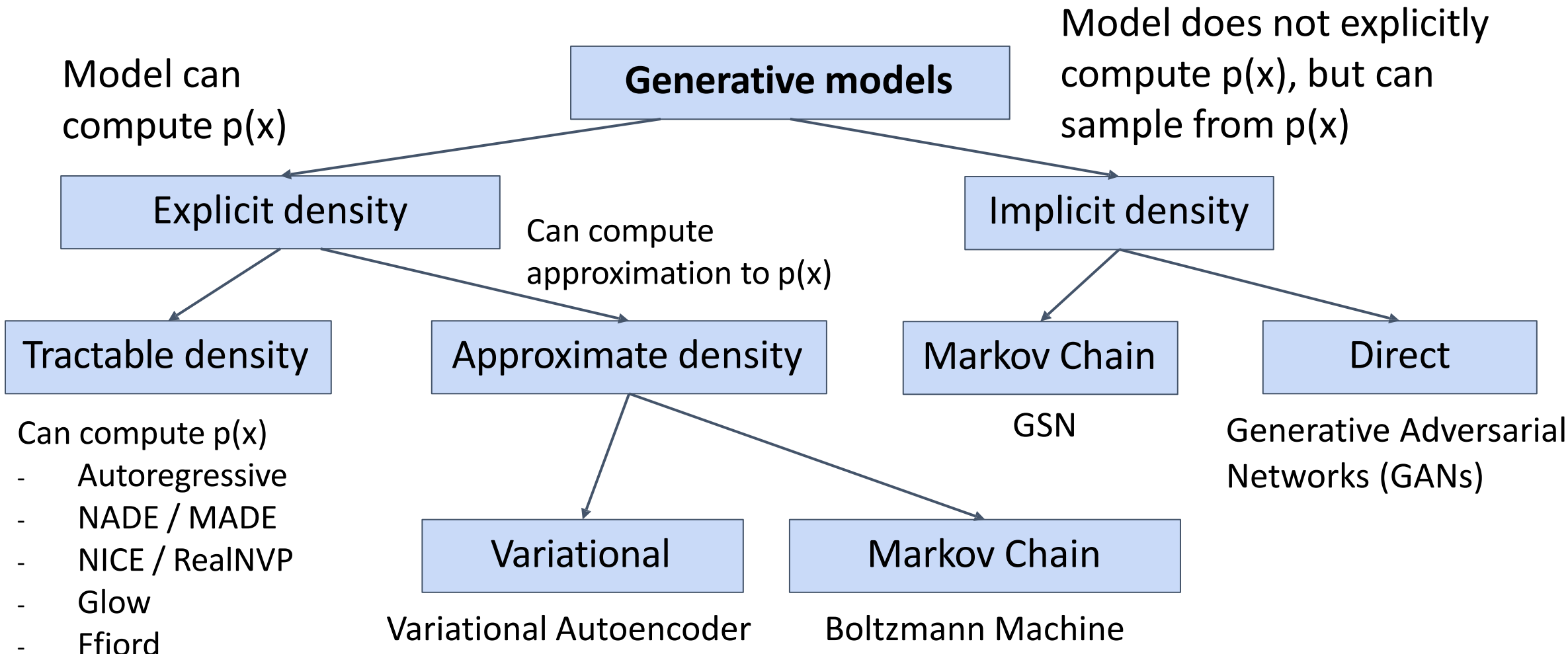


Figure adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Taxonomy of Generative Models

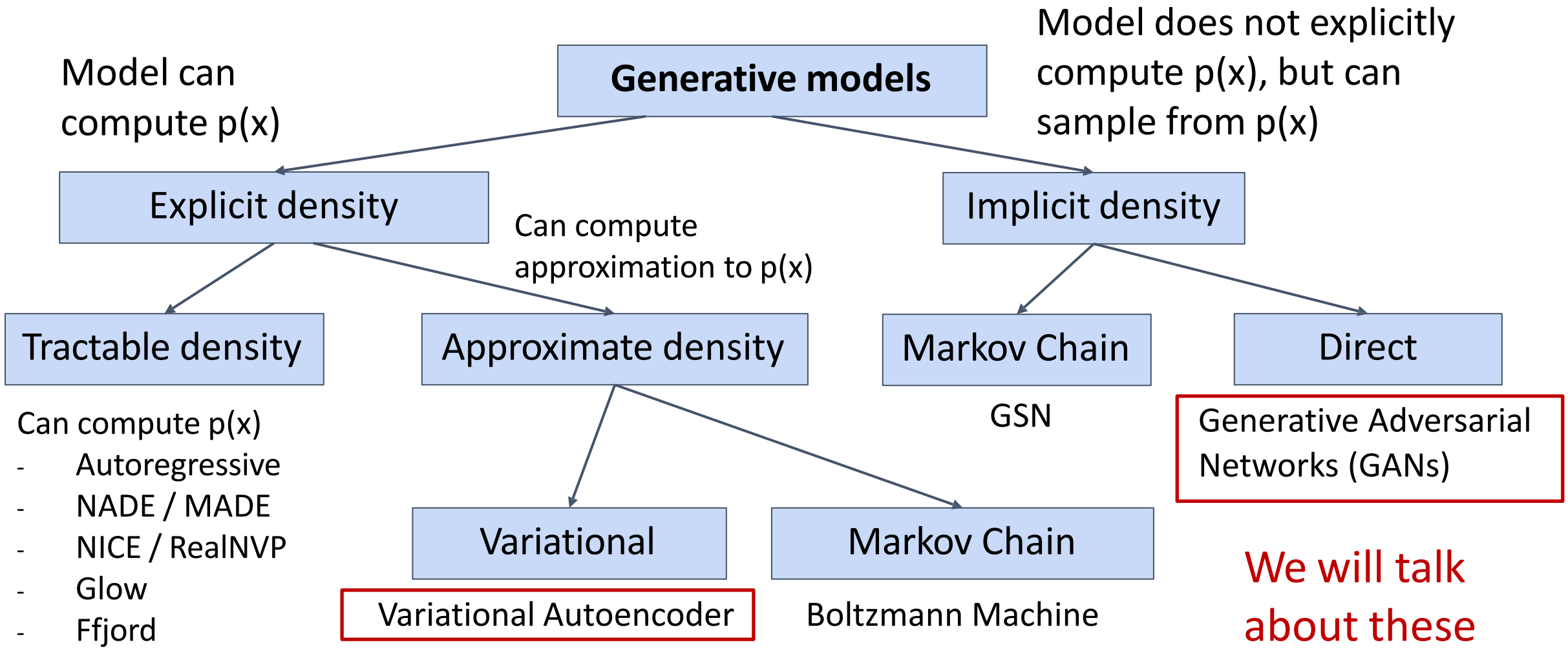


Figure adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Variational Autoencoders

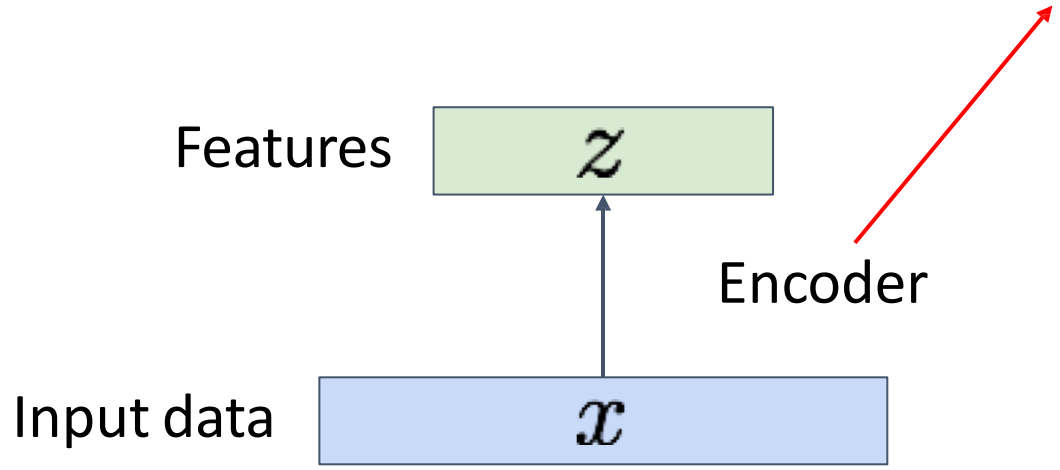
Variational Autoencoders

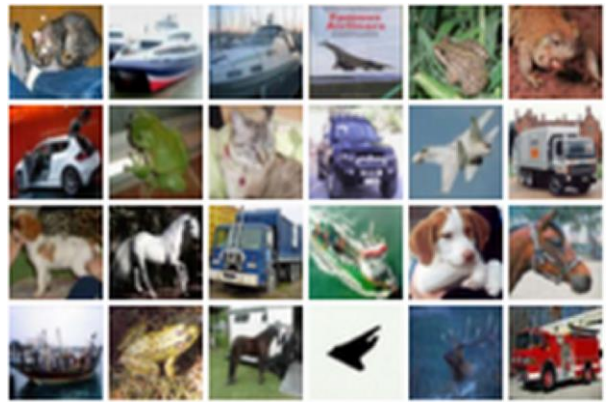
Autoencoders

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





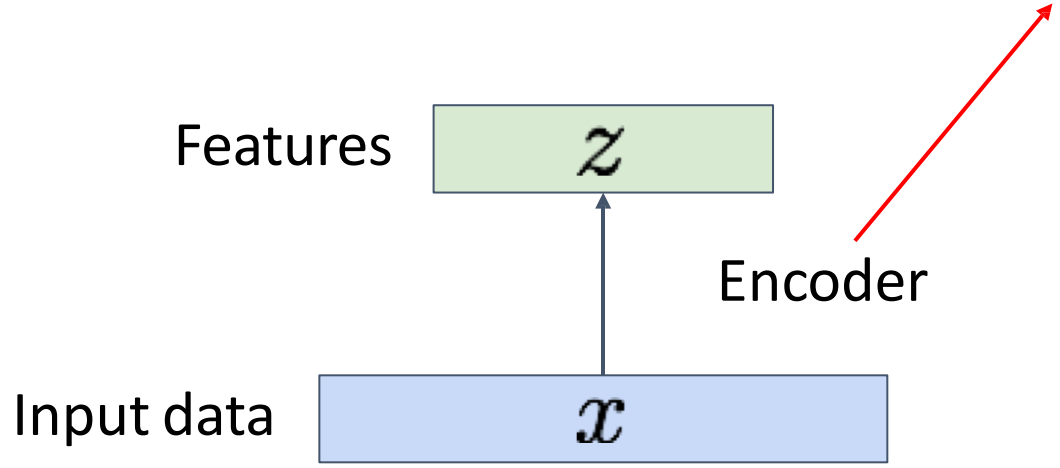
Input Data

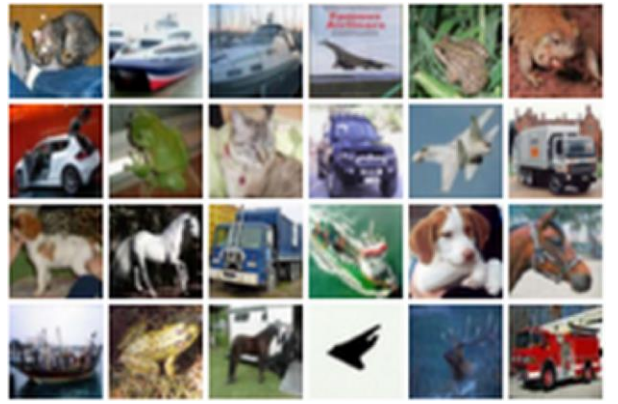
Autoencoders

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





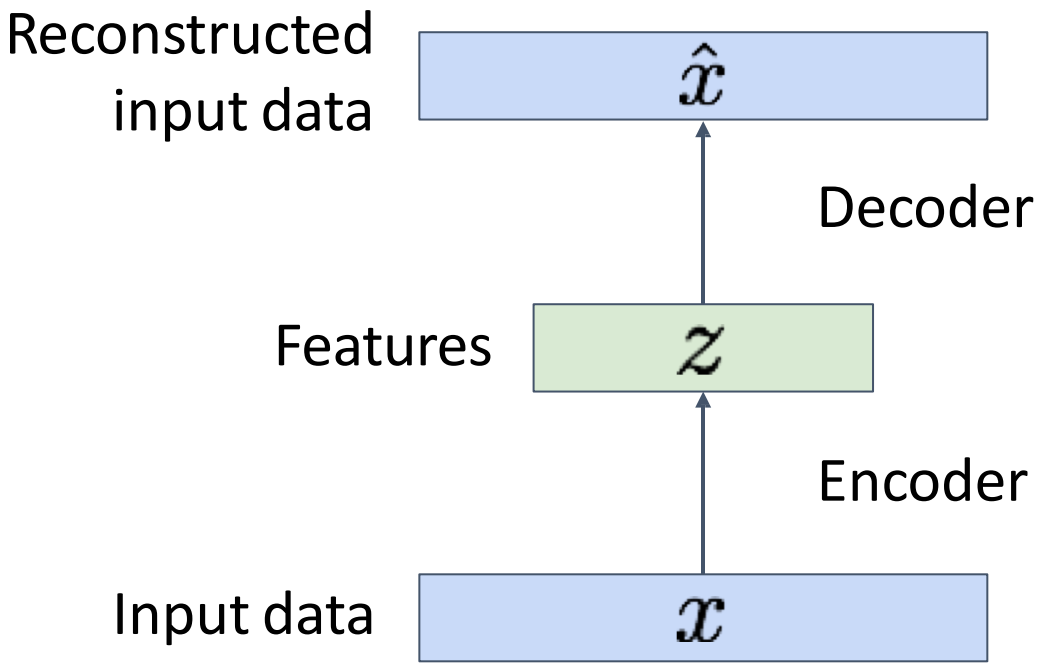
Input Data

Autoencoders

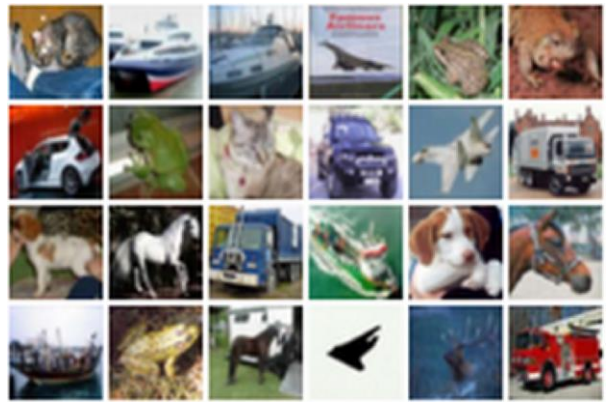
- **Problem**: How can we learn this feature transform from raw data?

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

"Autoencoding" = encoding itself



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

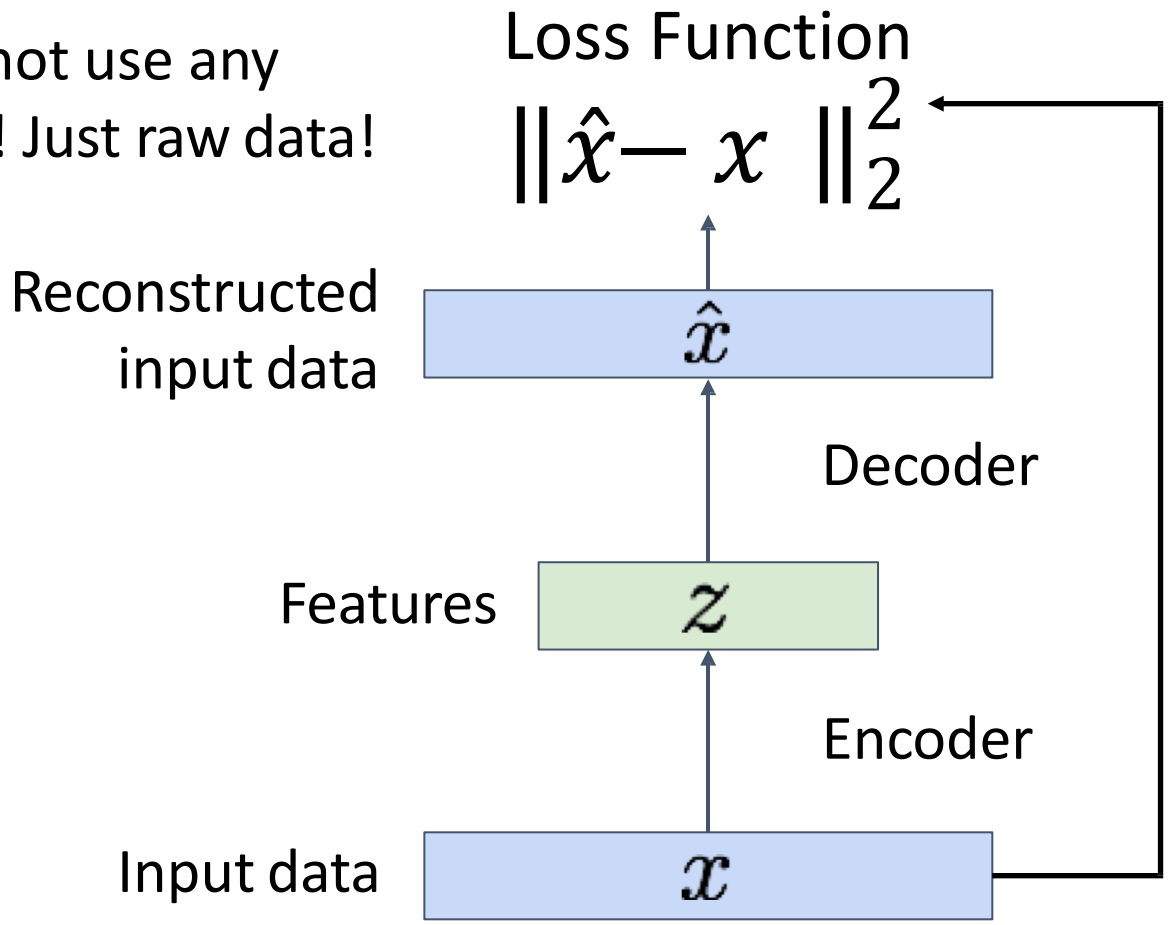


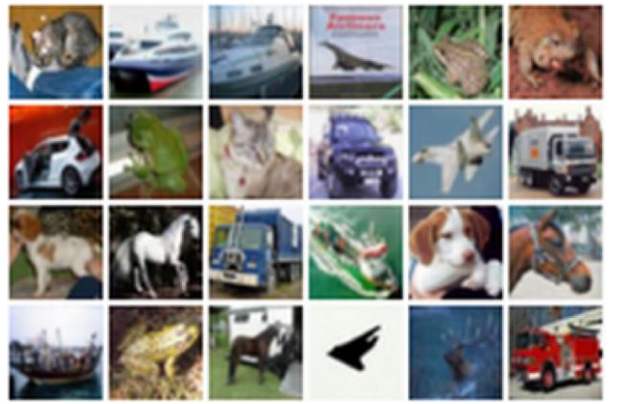
Input Data

Autoencoders

Loss: L2 distance between input and reconstructed data.

Does not use any labels! Just raw data!



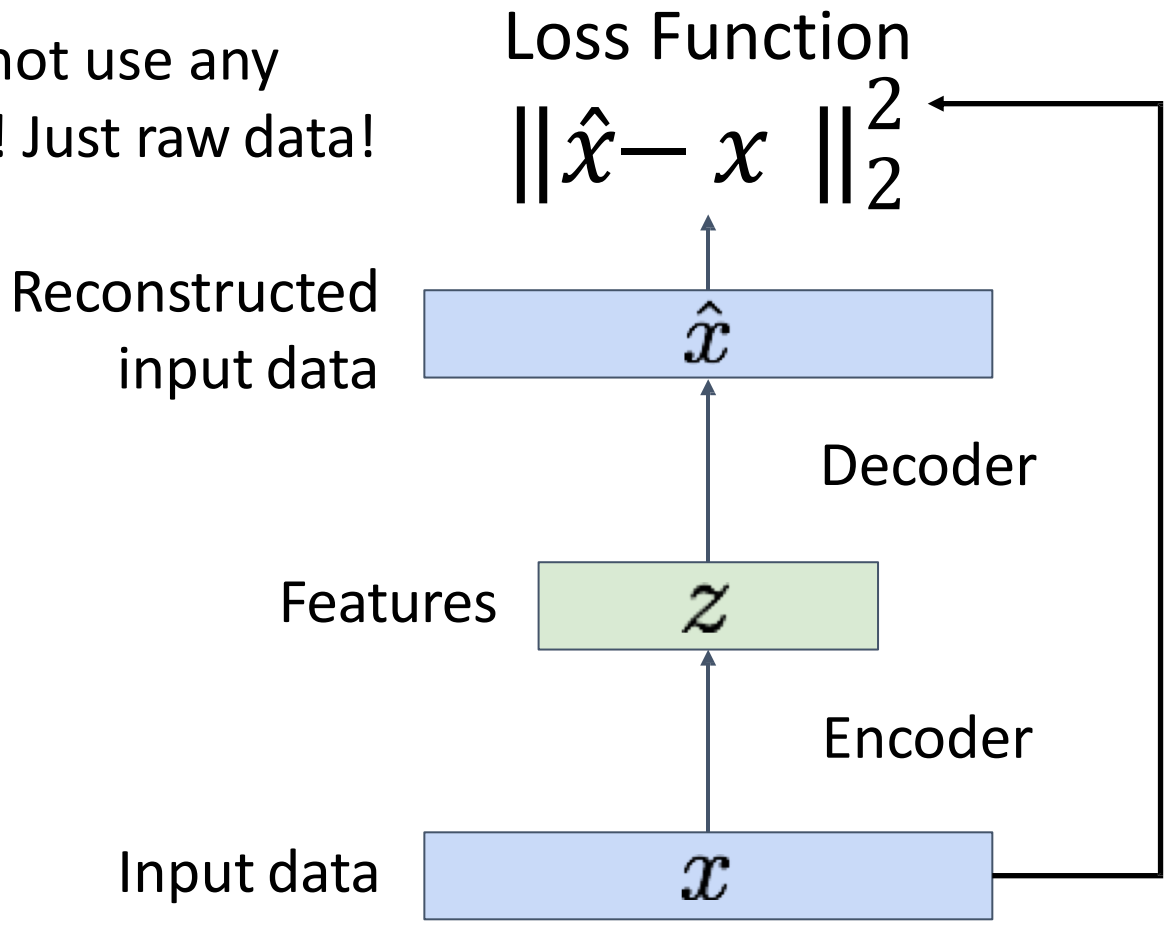


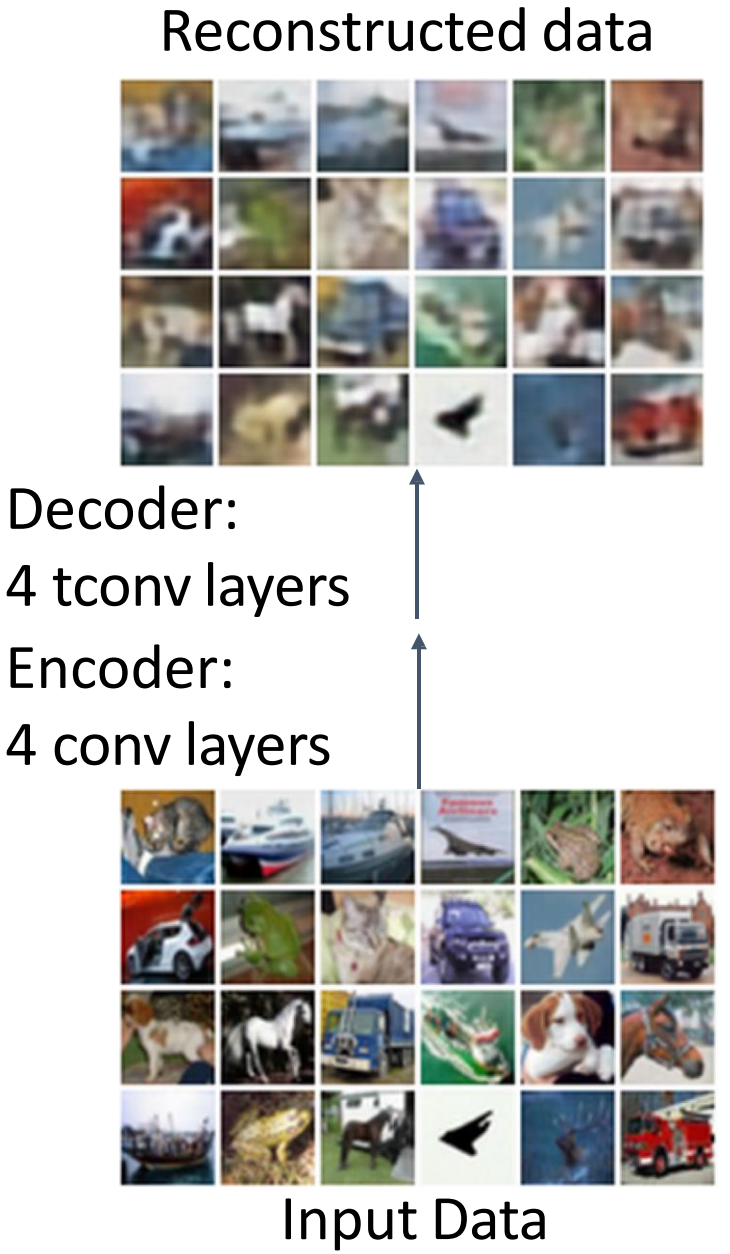
Input Data

Autoencoders

Loss: L2 distance between input and reconstructed data.

Does not use any labels! Just raw data!





Autoencoders

Loss: L2 distance between input and reconstructed data.

Does not use any labels! Just raw data!

Loss Function

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

Reconstructed input data

$\hat{x}$

Decoder

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

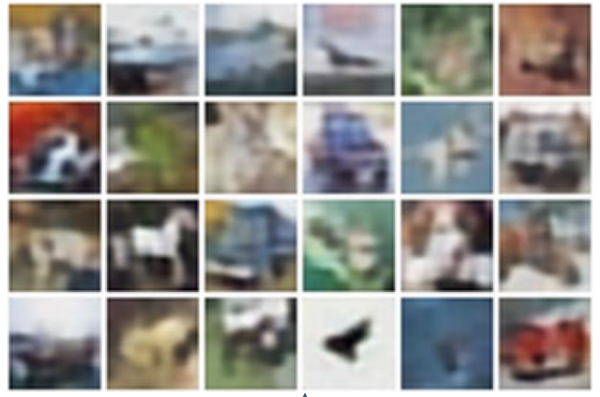
Features

z

Encoder

Input data

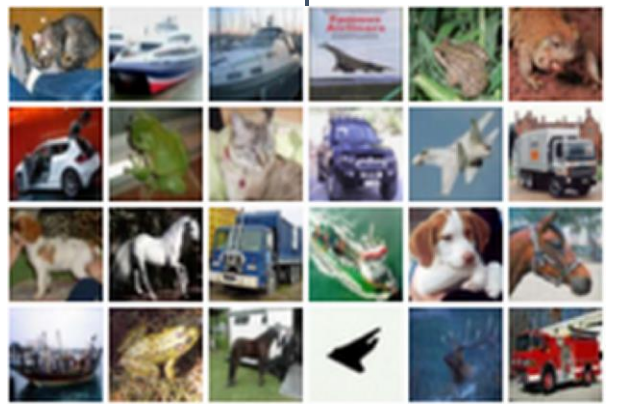
x



Reconstructed data

Decoder:
4 tconv layers

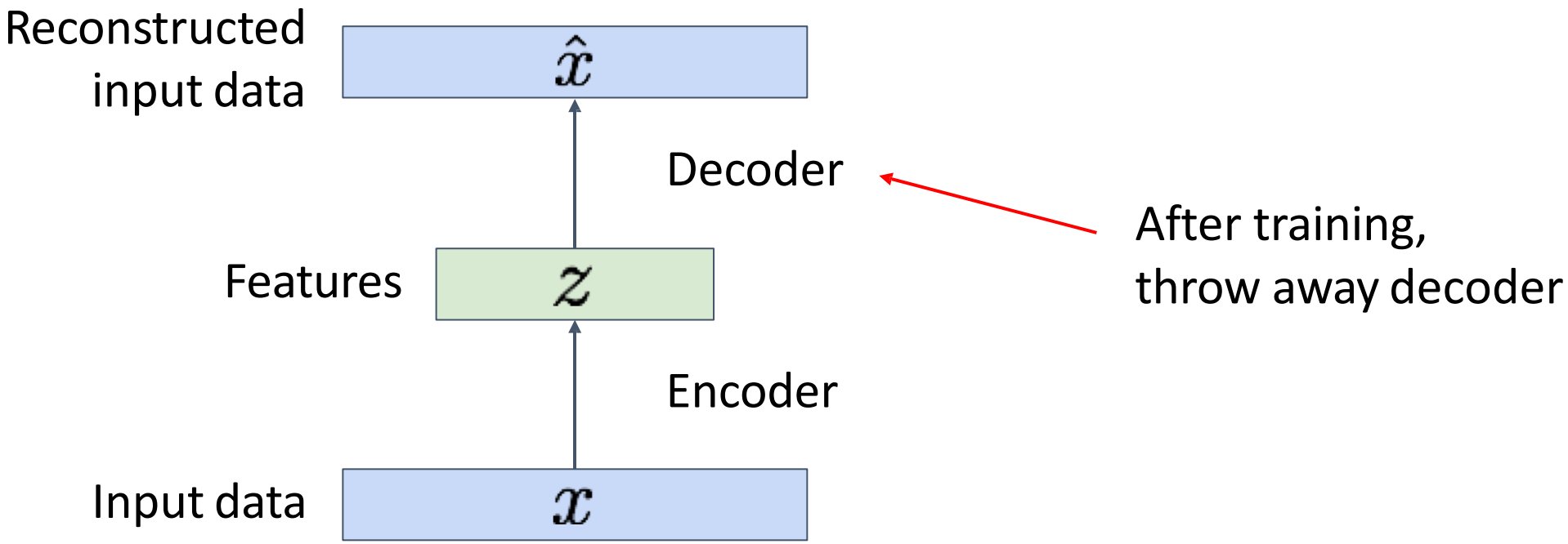
Encoder:
4 conv layers



Input Data

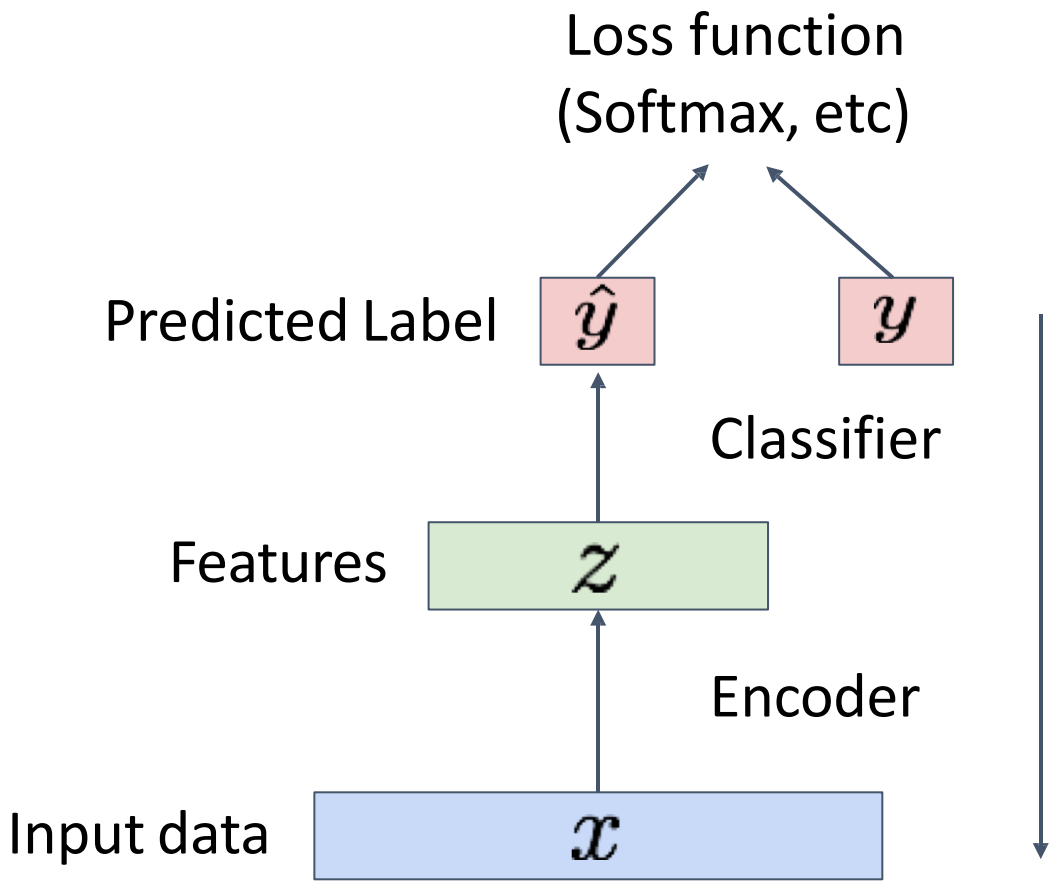
Autoencoders

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



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)

Avoid Shortcut Solution

Autoencoders: Hidden Layer Dimensionality

Smaller than the input

—Will compress the data, reconstruction of the data far from the training distribution will be difficult

—Linear-linear encoder-decoder with Euclidian loss is actually equivalent to PCA (under certain data normalization)

*slide from Louis-Philippe Morency

Autoencoders: Hidden Layer Dimensionality

Smaller than the input

—Will compress the data, reconstruction of the data far from the training distribution will be difficult

—Linear-linear encoder-decoder with Euclidian loss is actually equivalent to PCA (under certain data normalization)

Larger than the input

—No compression needed

—Can trivially learn to just copy, no structure is learned (unless you regularize)

—Does not encourage learning of meaningful features (unless you regularize)

*slide from Louis-Philippe Morency

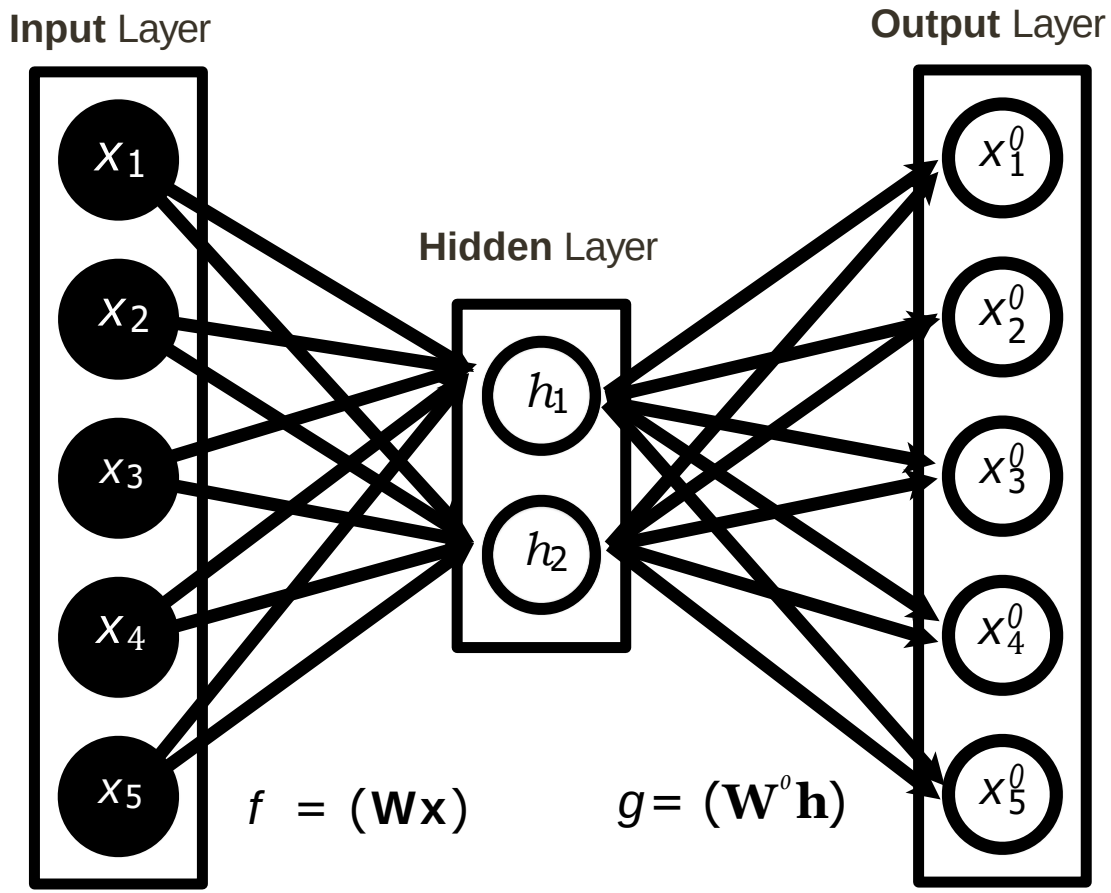
De-noising Autoencoder

Idea: add noise to input but learn to reconstruct the original

— Leads to better representations

— Prevents copying

Note: different noise is added during each epoch



*slide from Louis-Philippe Morency

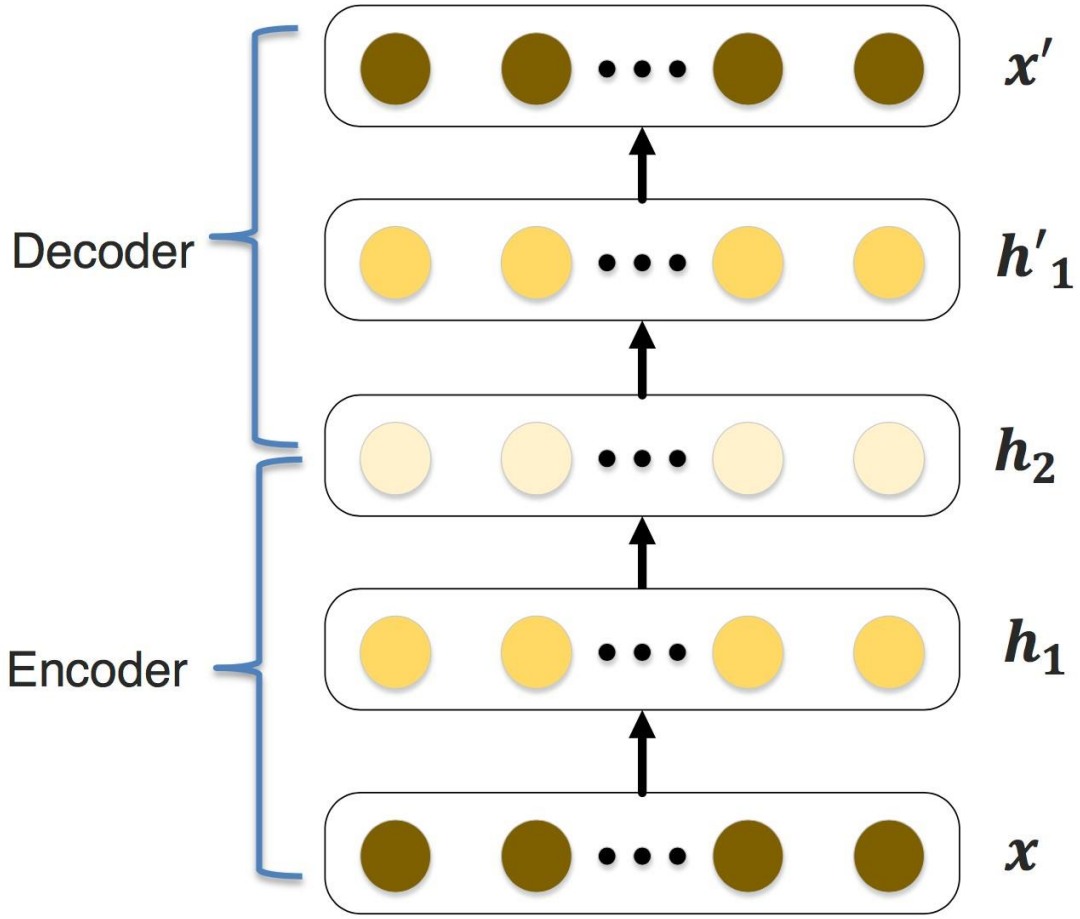
Stacked (deep) Autoencoders and Denoising Autoencoders

What **can we do** with them?

—Good for compression (better than PCA)

—Disregard the decoder and use the middle layer as a representation

—Fine-tune the autoencoder for a task

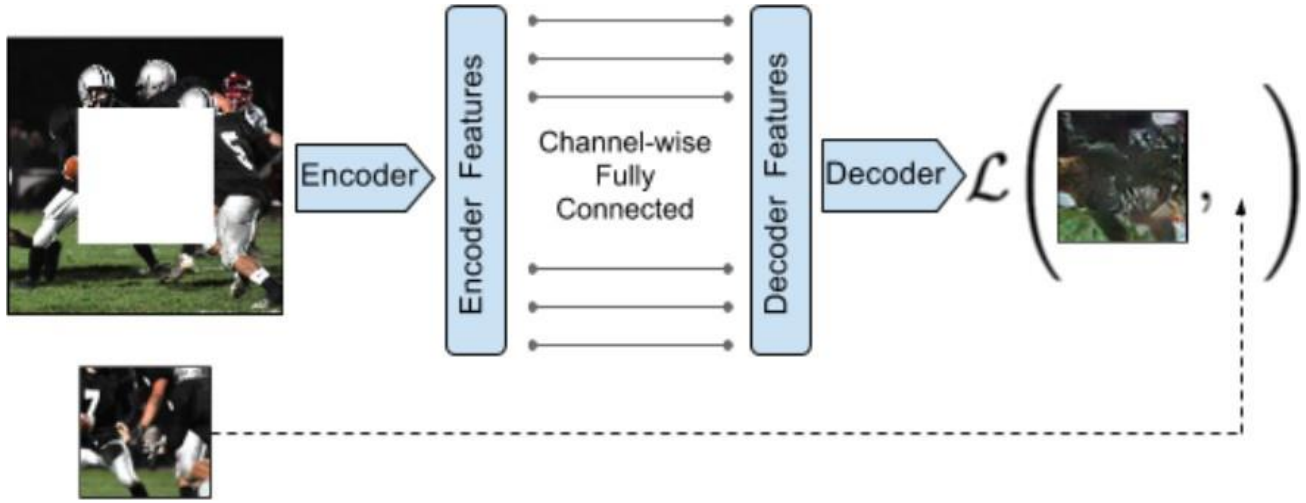


*slide from Louis-Philippe Morency

AE as Generative Model

Context Encoders

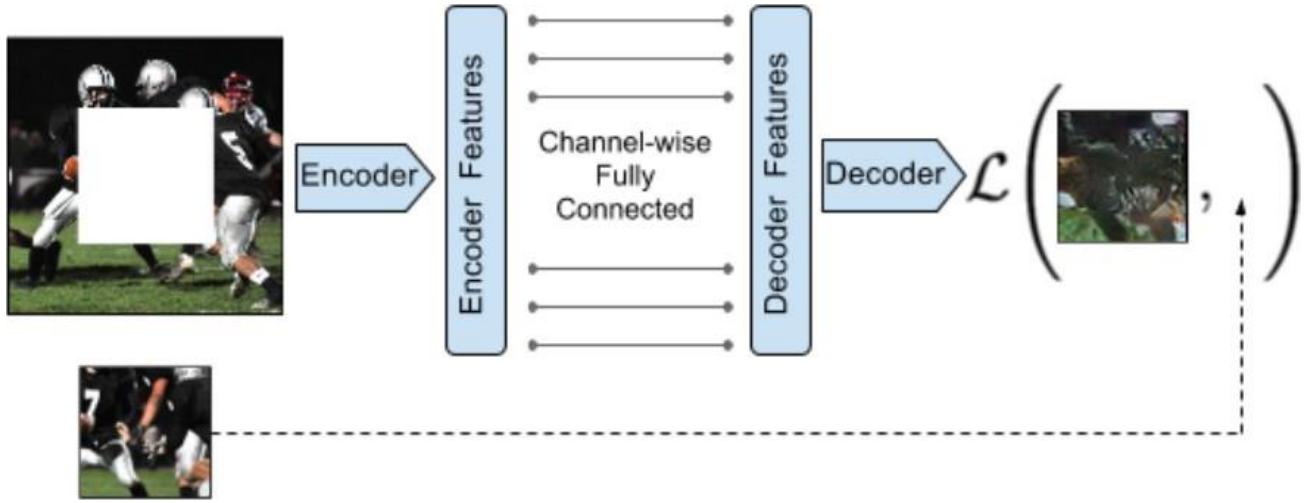
[Pathak et al., 2016]

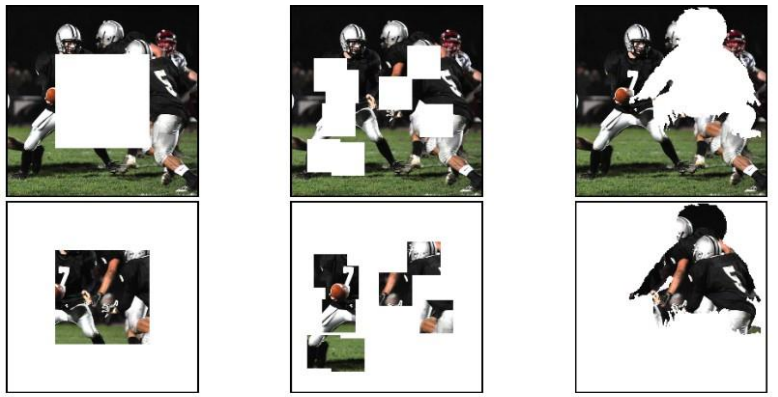


Pathak et al., Context Encoders: Feature Learning by Inpainting, CVPR 2016.

Context Encoders

[Pathak et al., 2016]

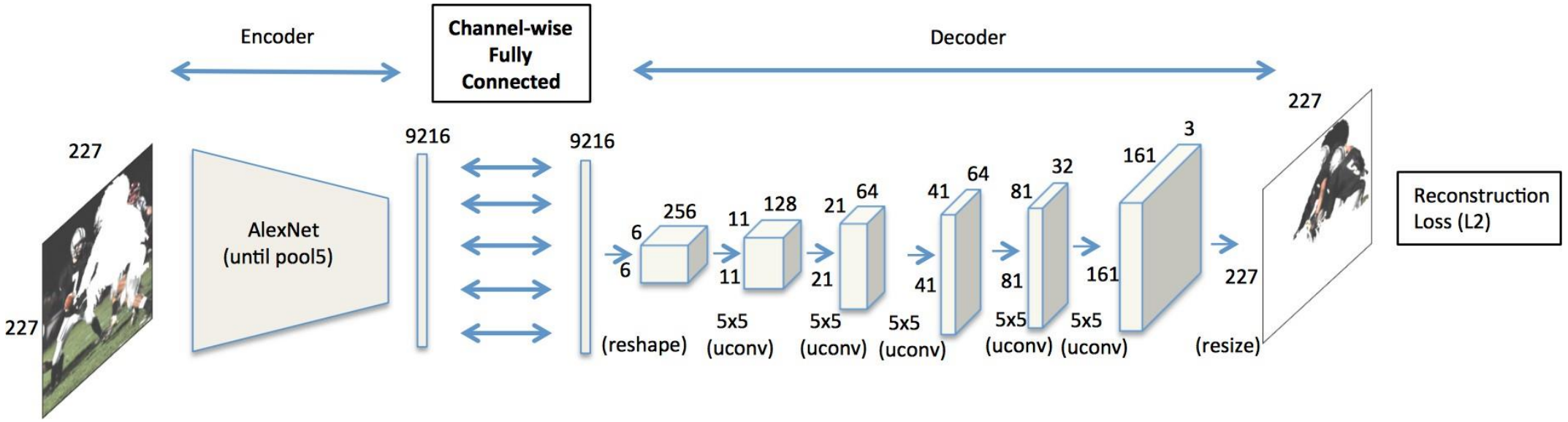




(a) Central region (b) Random block (c) Random region

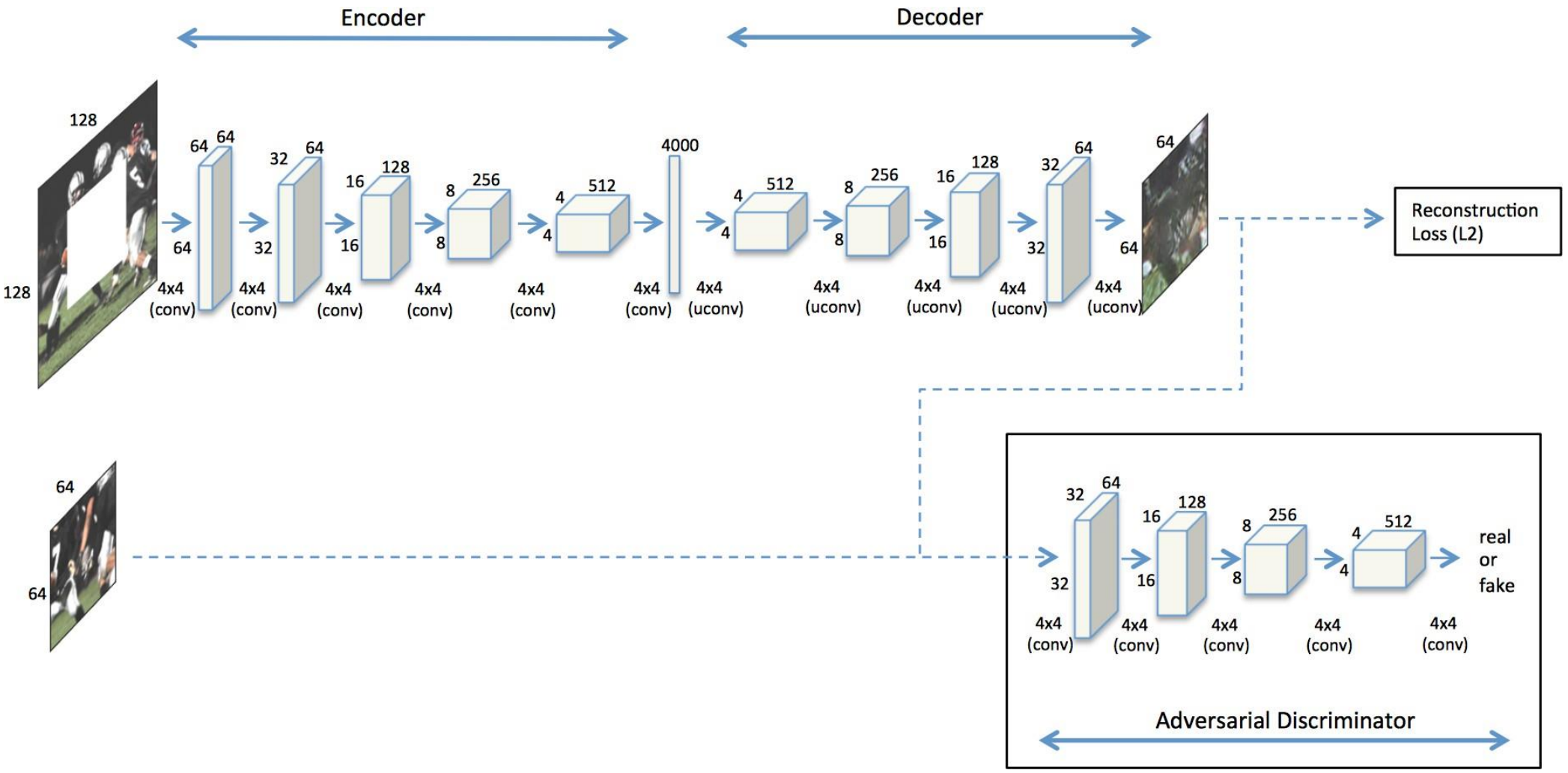
Context Encoders

[Pathak et al., 2016]



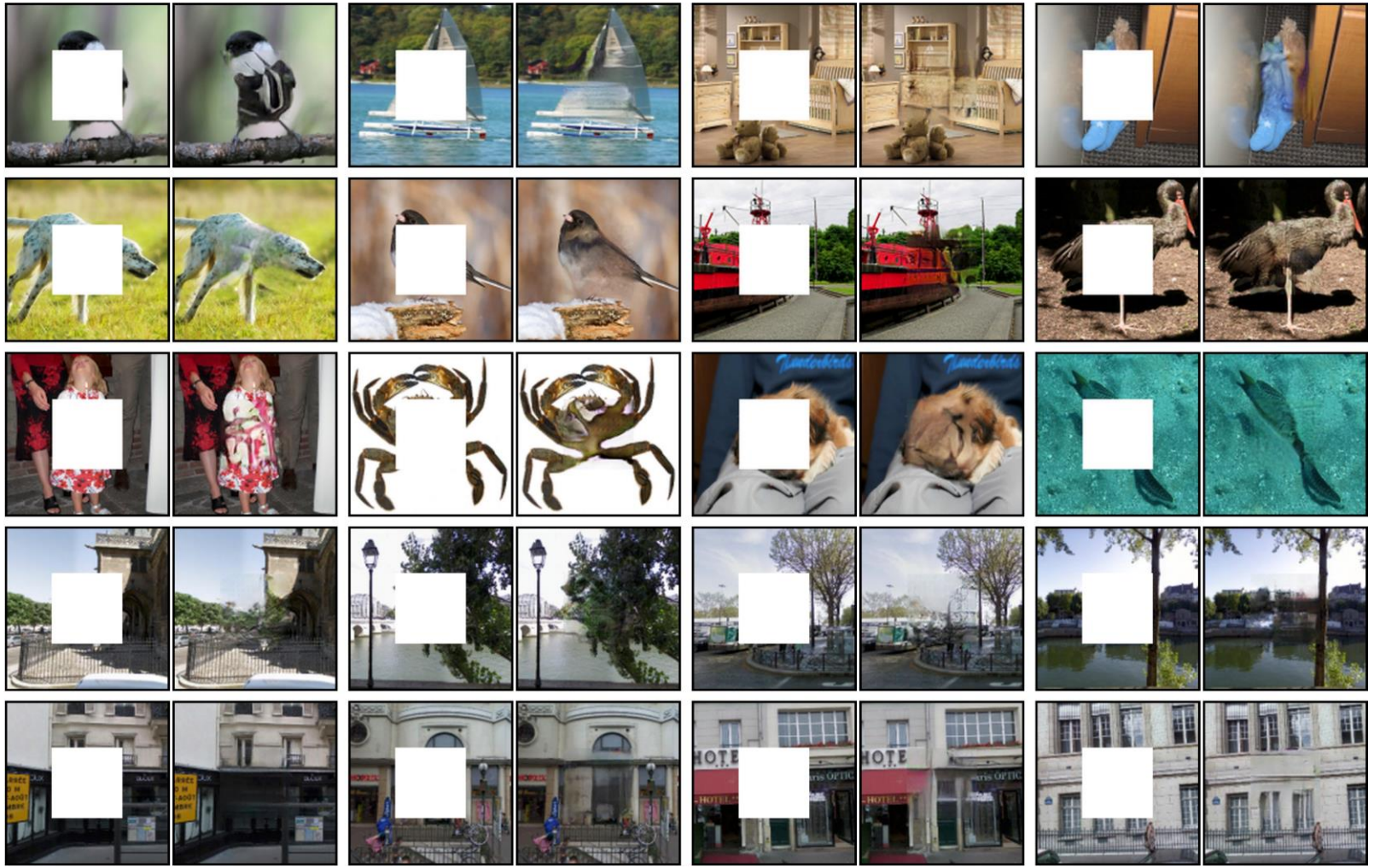
Context Encoders

[Pathak et al., 2016]



Context Encoders

[Pathak et al., 2016]



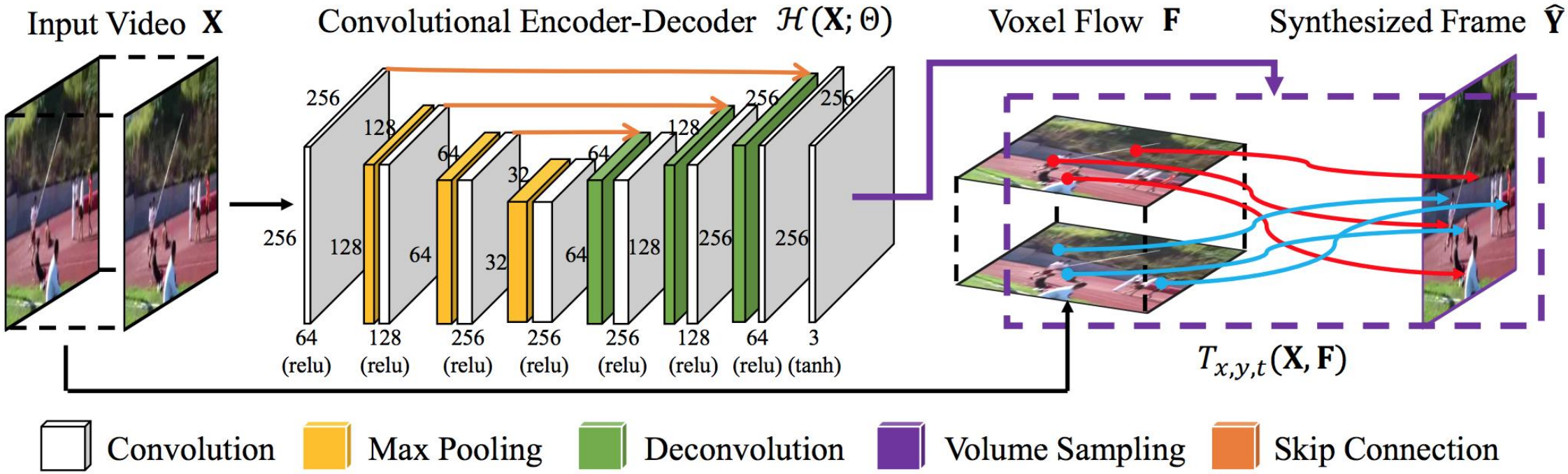
Context Encoders

[Pathak et al., 2016]

Pretraining Method	Supervision	Pretraining time	Classification	Detection	Segmentation
ImageNet [26]	1000 class labels	3 days	**78.2%**	**56.8%**	**48.0%**
Random Gaussian	initialization	< 1 minute	53.3%	43.4%	19.8%
Autoencoder	-	14 hours	53.8%	41.9%	25.2%
Agrawal *et al.* [1]	egomotion	10 hours	52.9%	41.8%	-
Doersch *et al.* [7]	context	4 weeks	55.3%	**46.6%**	-
Wang *et al.* [39]	motion	1 week	**58.4%**	44.0%	-
Ours	context	14 hours	56.5%	44.5%	**29.7%**

Temporal Context Encoders

[Liu et al., 2017]

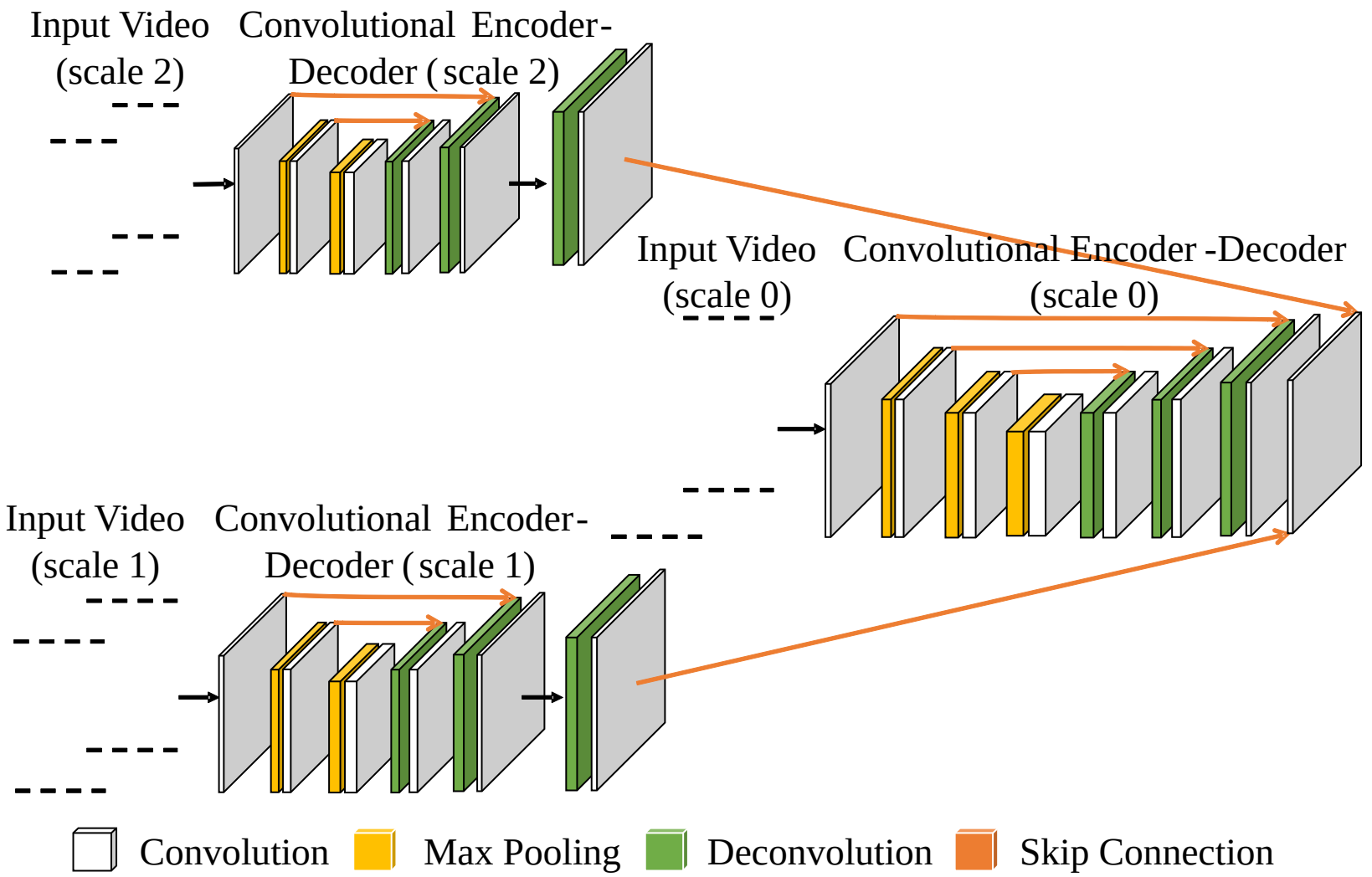


Liu et al., Video Frame Synthesis using Deep Voxel Flow, ICCV 2017.

Temporal Context Encoders

- Multi-Scale Pipeline

Handle large motion

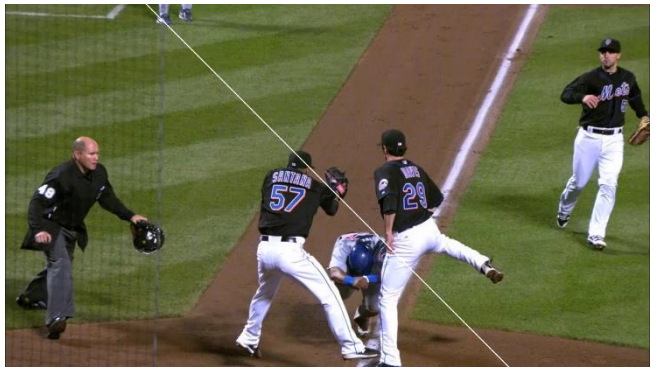


Slow-Motion Effect

- User Study

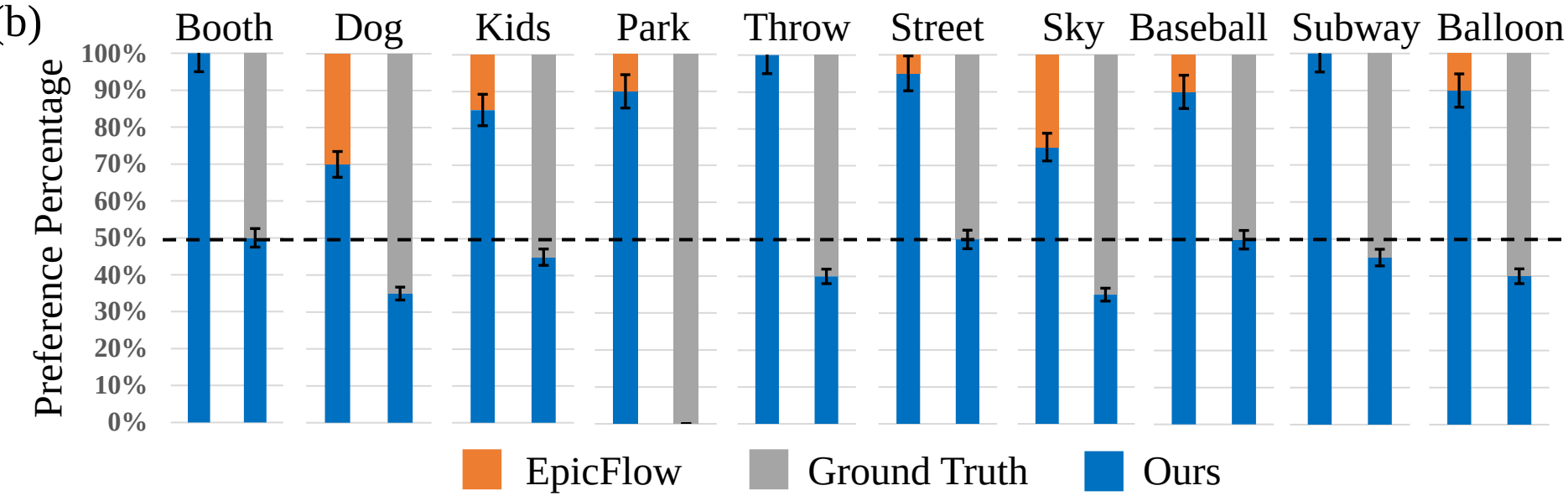
(a)

Diagonal-split Comparison



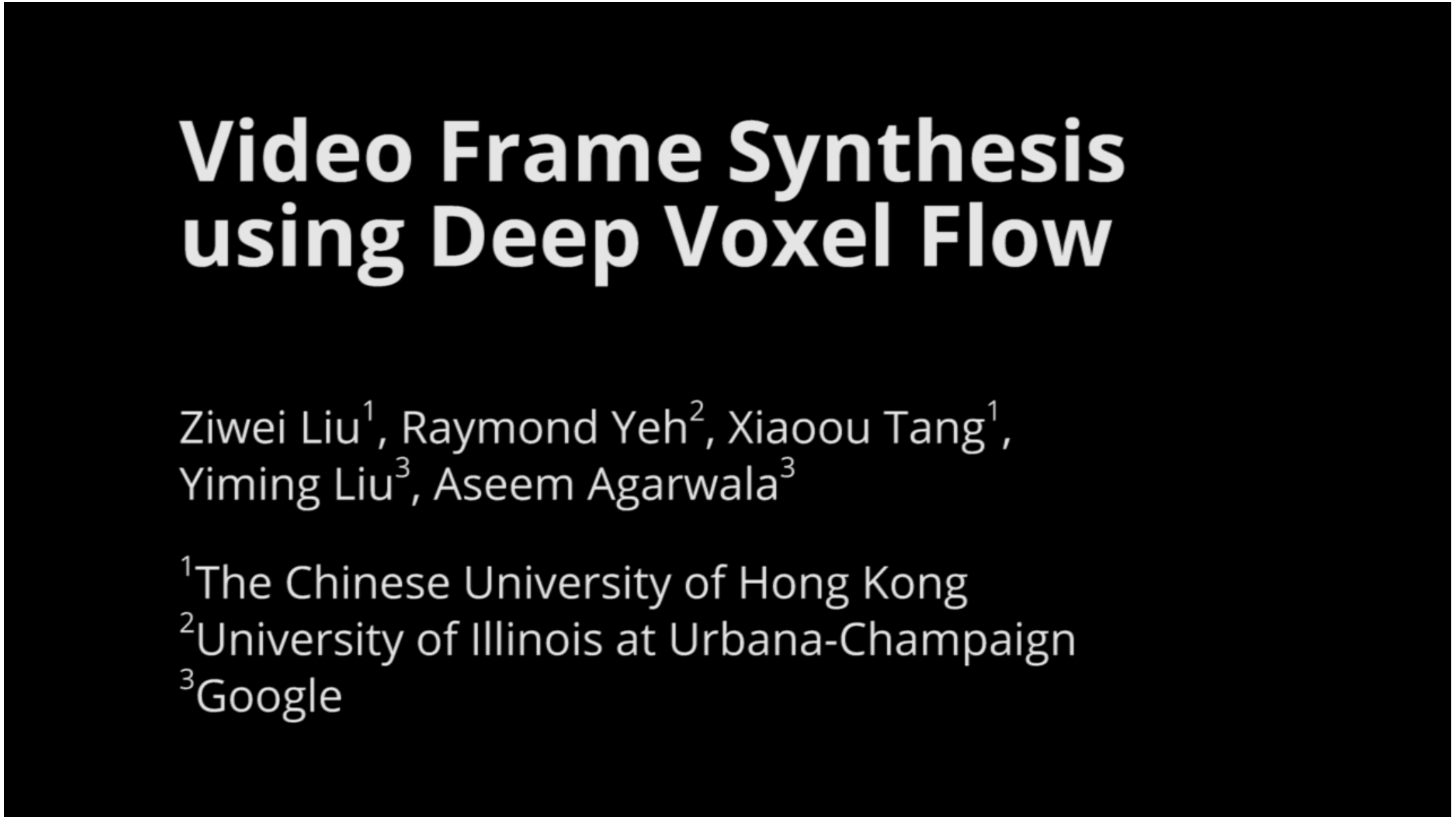
Method 1 \ Method 2

(b)



Slow-Motion Effect

[Liu et al., 2017]



Slow-Motion Effect

[Li et al., 2021]

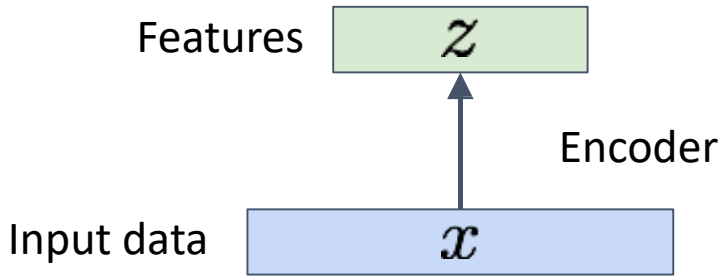
Li et al., Deep Animation Video Interpolation in the Wild, CVPR 2021.

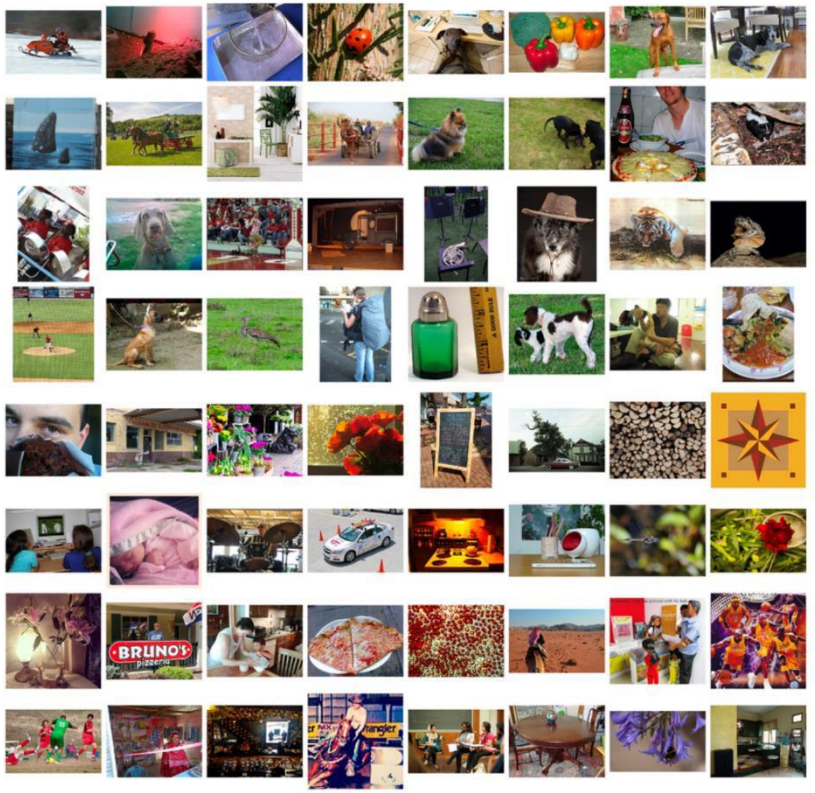
AE as Representation Learning

Autoencoders (recap)

Goal: Meaningful features that capture the main factors of variation in the dataset

- These are good for classification, clustering, exploration, generation, …
- We have no ground truth for them



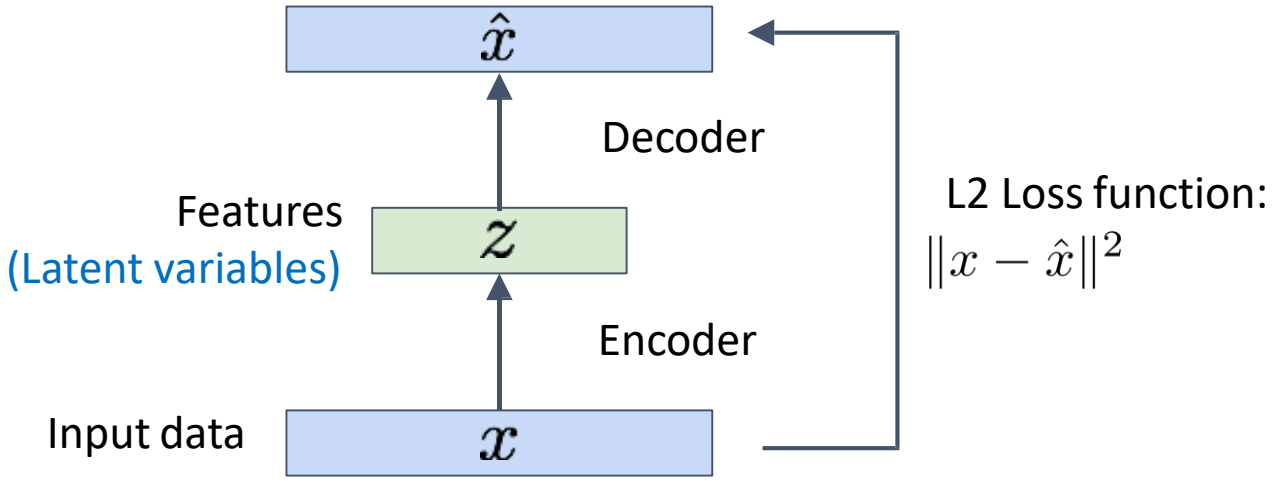


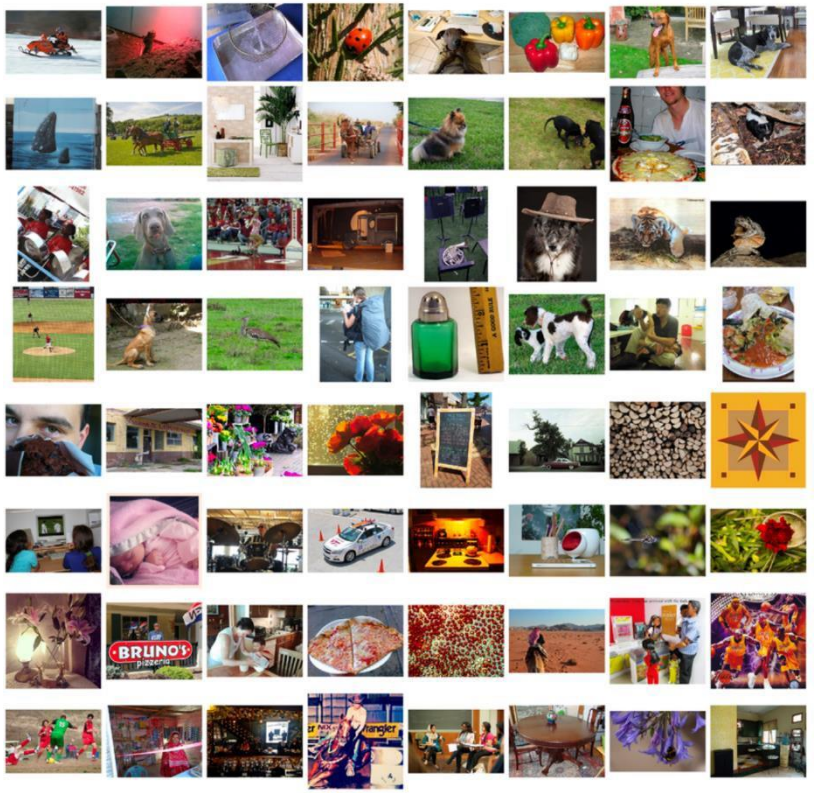
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Autoencoders (recap)

Goal: Meaningful features that capture the main factors of variation

Features that can be used to reconstruct the image



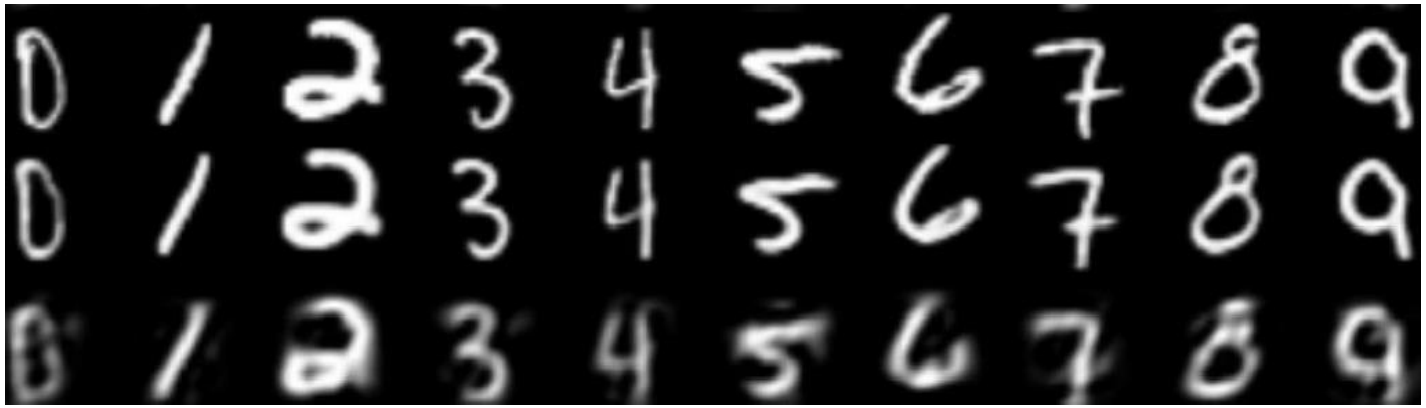


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Autoencoders (recap)

Linear Transformation for Encoder and Decoder give result close to PCA

Deeper networks give better reconstructions, since basis can be non-linear



Original

Autoencoder

PCA

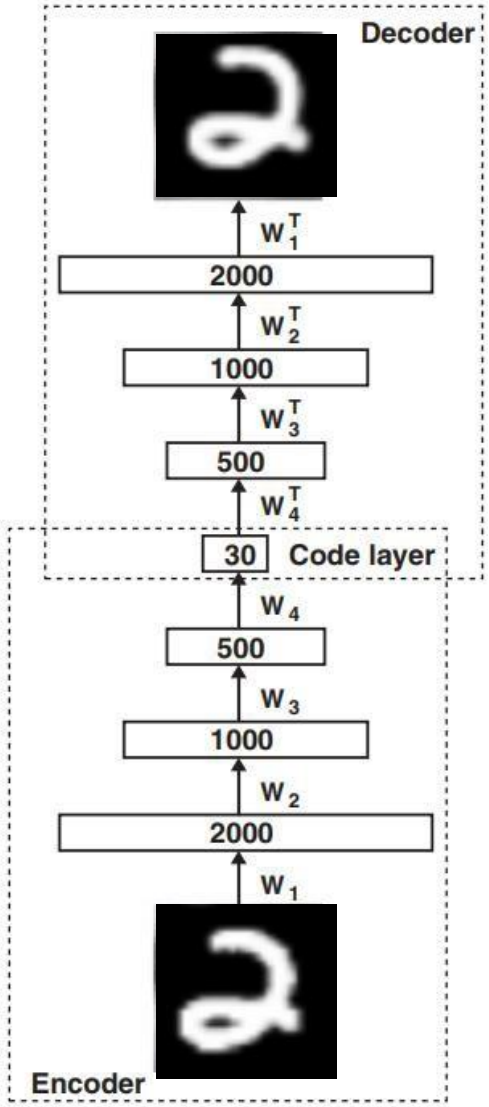
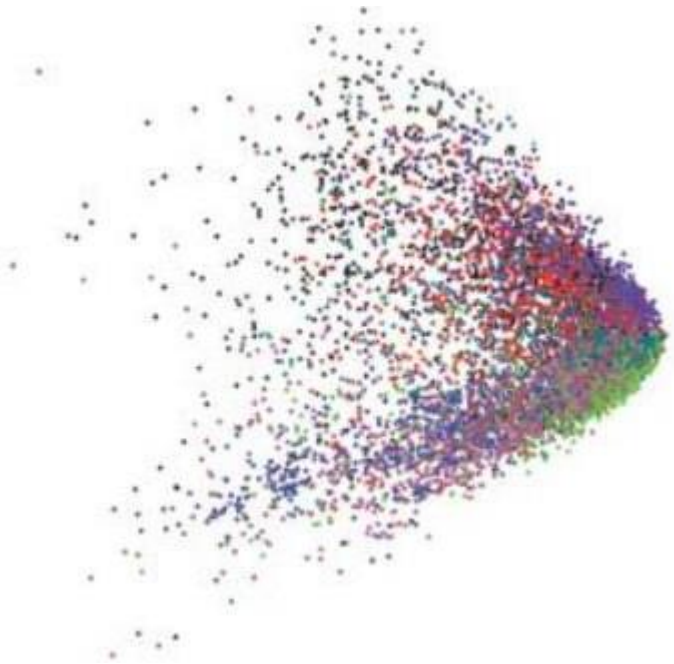


Image Credit: Reducing the Dimensionality of Data with Neural Networks, . Hinton and Salakhutdinov, Science 2016

Example: Document Word → 2D Code

LSA (based on PCA)



Autoencoder

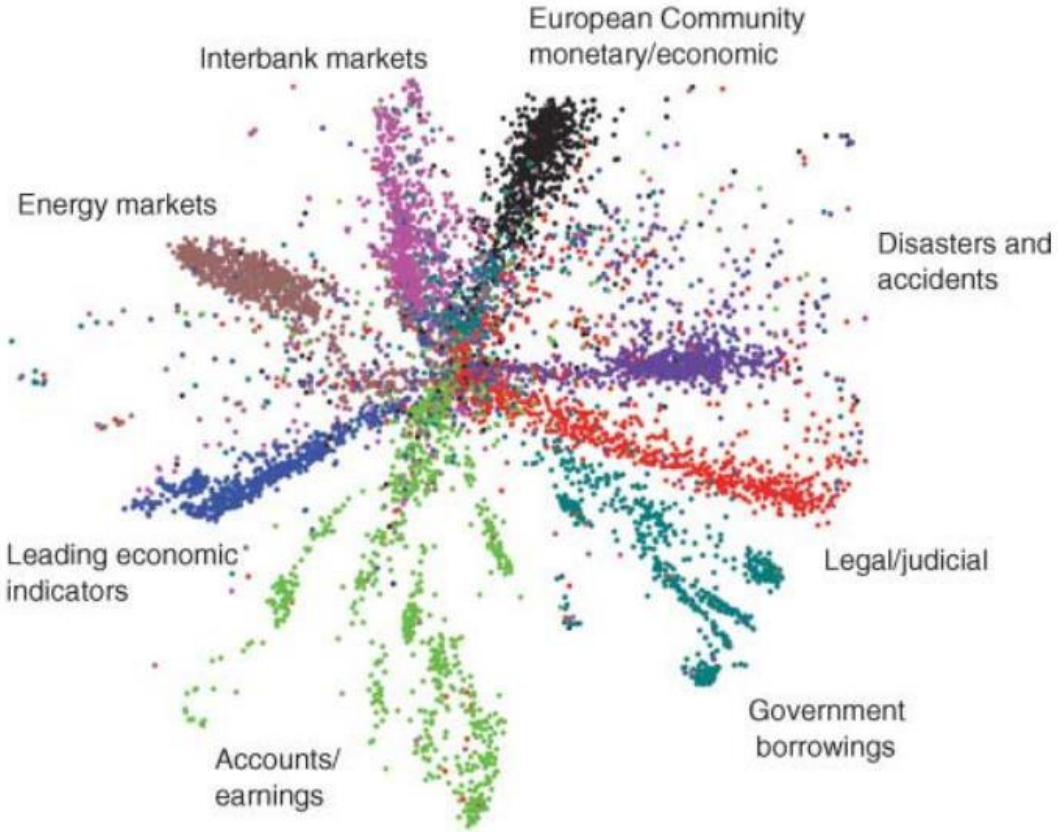
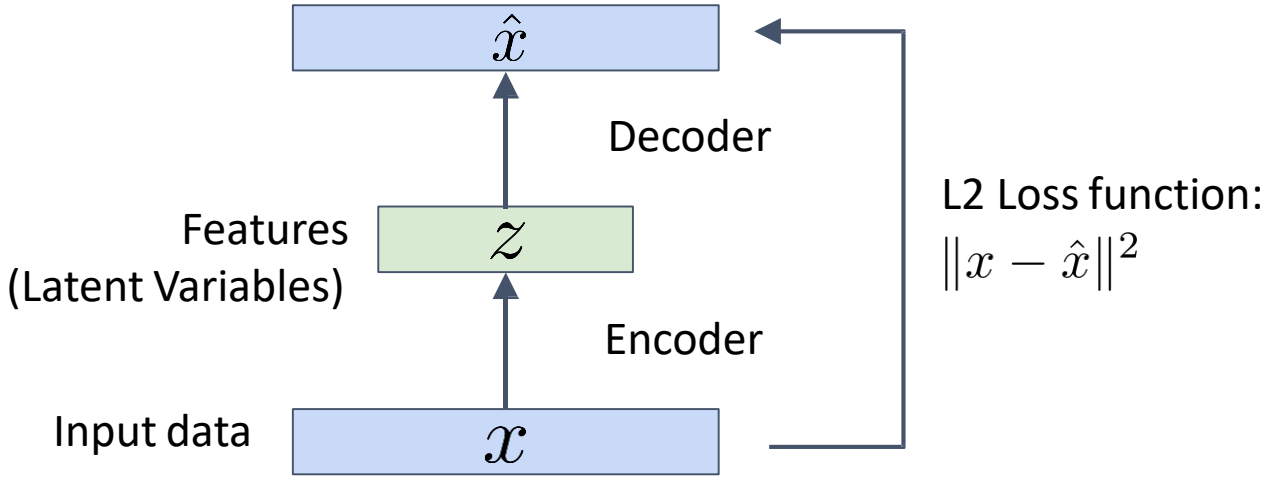


Image Credit: Reducing the Dimensionality of Data with Neural Networks, . Hinton and Salakhutdinov, Science 2016

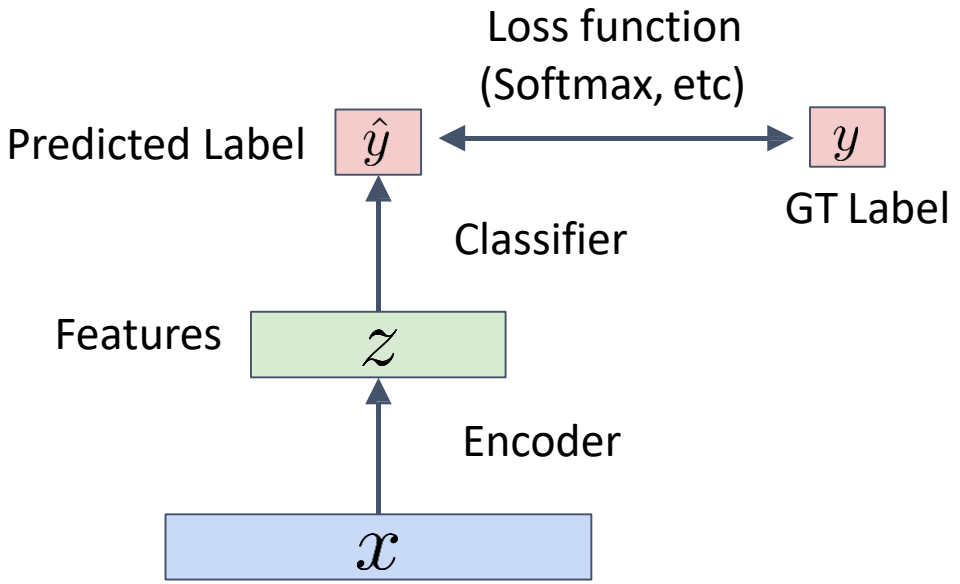
Example: Semi-Supervised Classification

- Many images, but few ground truth labels

start unsupervised
train autoencoder on many images

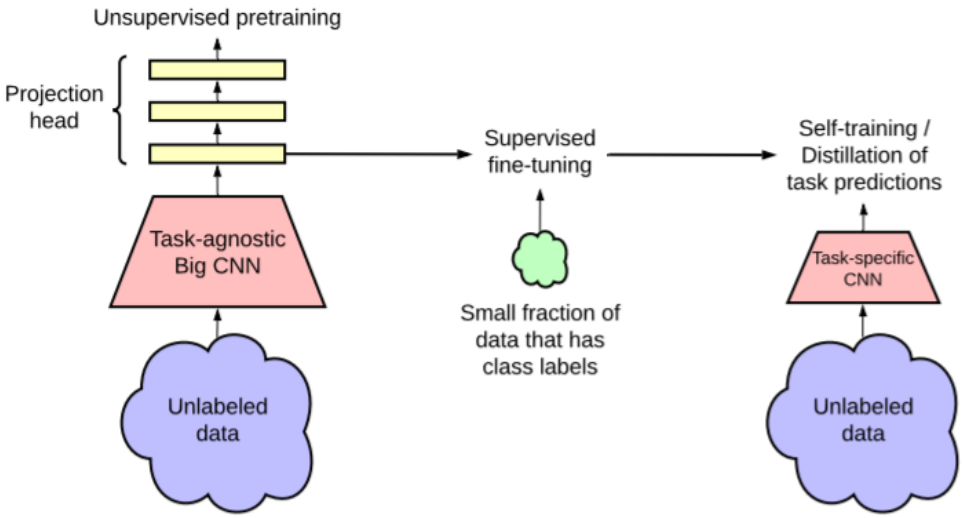


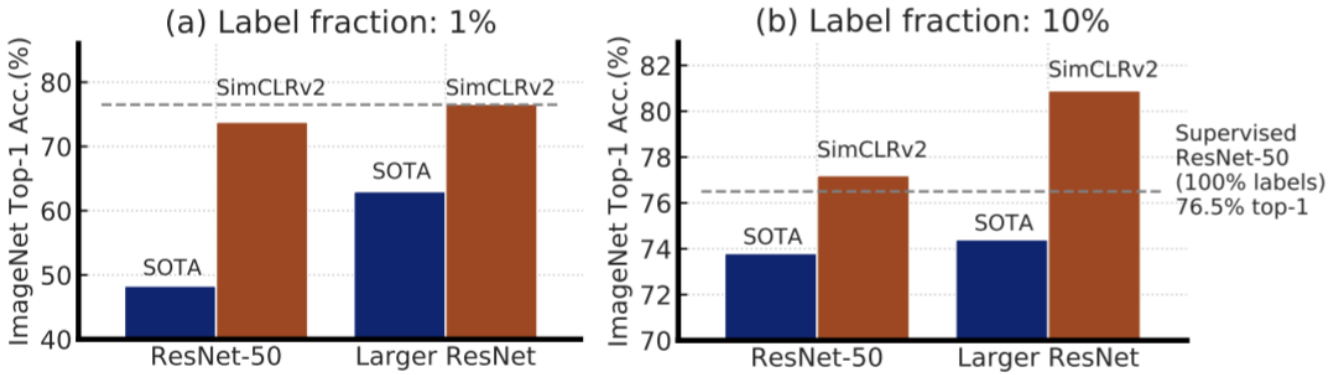
supervised fine-tuning
train classification network on labeled images



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Example: Semi-Supervised Classification





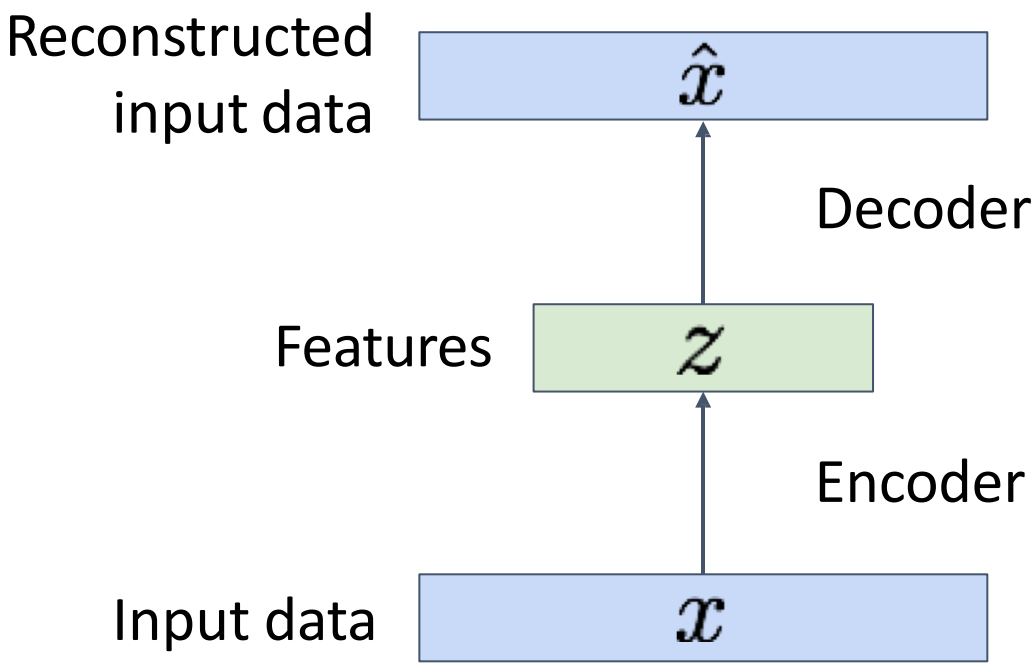
Chen et al., Big Self-Supervised Models are Strong Semi-Supervised Learners, NeurIPS 2020.

Autoencoders: What's Next

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

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.

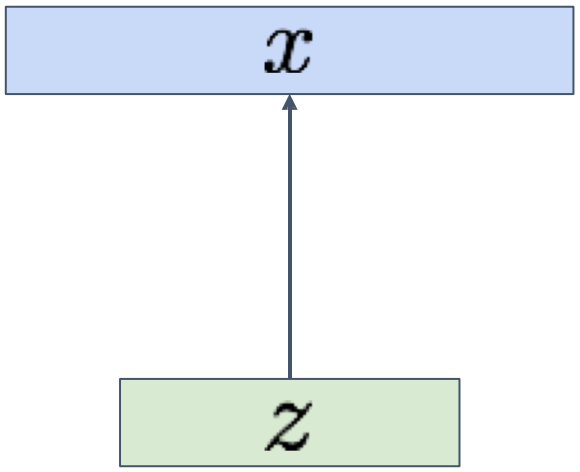
After training, sample new data like this:

Sample from conditional

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

Sample z from prior

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



Variational Autoencoders

Probabilistic spin on autoencoders:

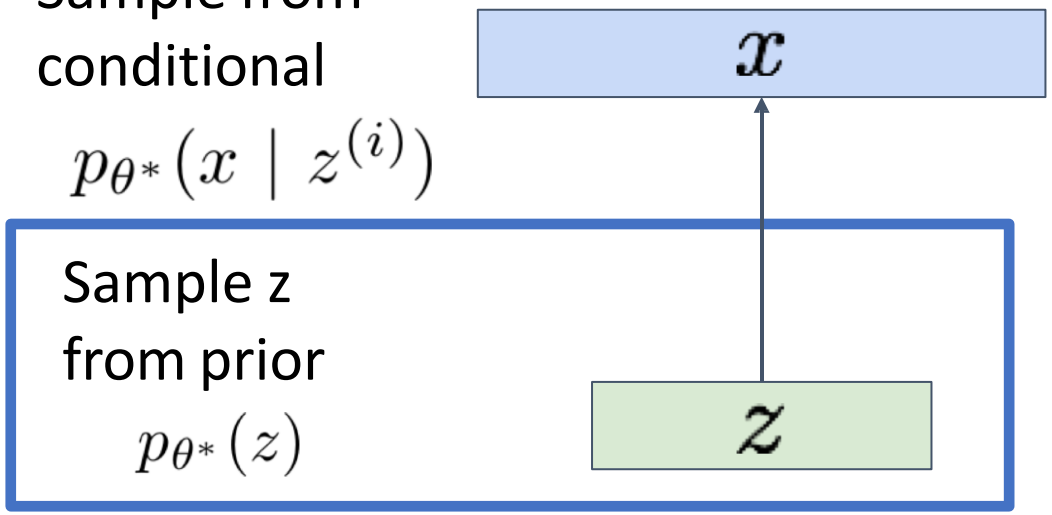
1. Learn latent features z from raw data
2. Sample from the model to generate new data

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.

After training, sample new data like this:

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



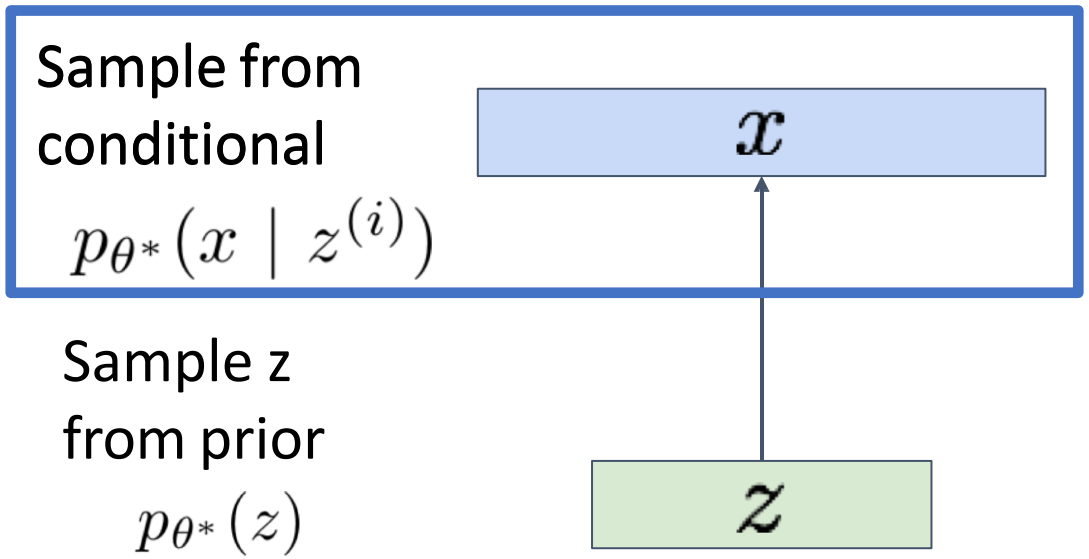
Assume simple prior p(z), e.g. Gaussian

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 autoencoder)

Variational Autoencoders

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

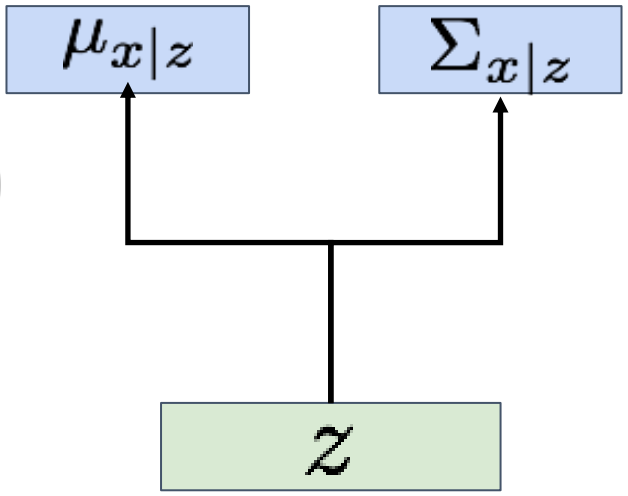
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

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**

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 $\sum_{x|z}$

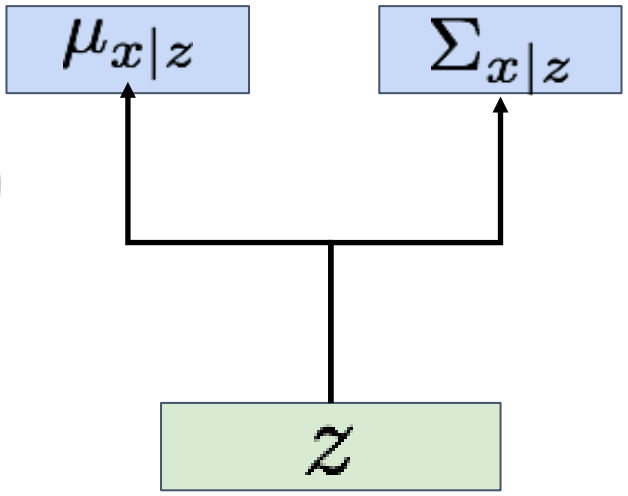
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

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**

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

Variational Autoencoders

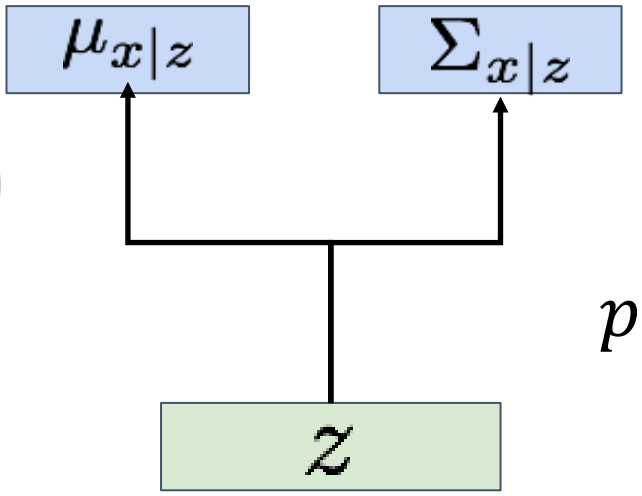
Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

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

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

How to train this model?

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



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

Basic idea: **maximize likelihood of data**

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

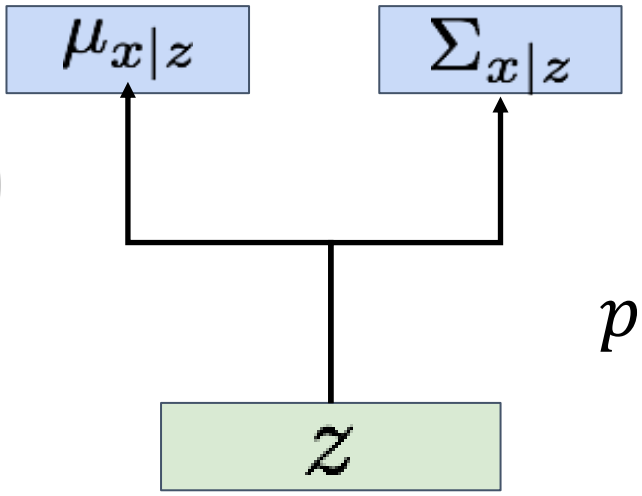
$$p_\theta(x) = \int p_\theta(x,z)dz = \int p_\theta(x|z)p_\theta(z)dz$$

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

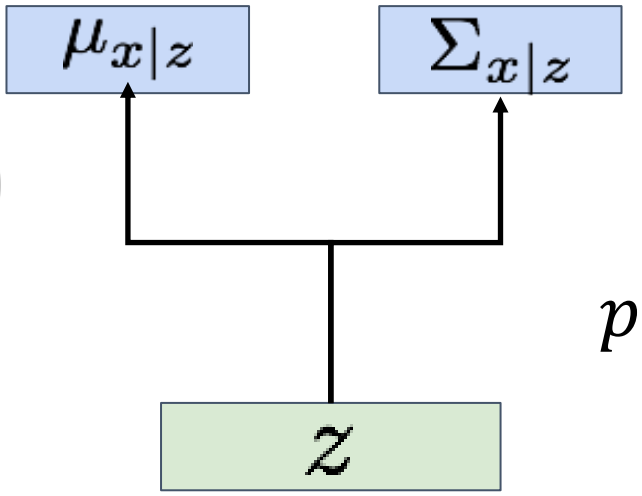
Variational Autoencoders

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

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, we assumed Gaussian prior for z

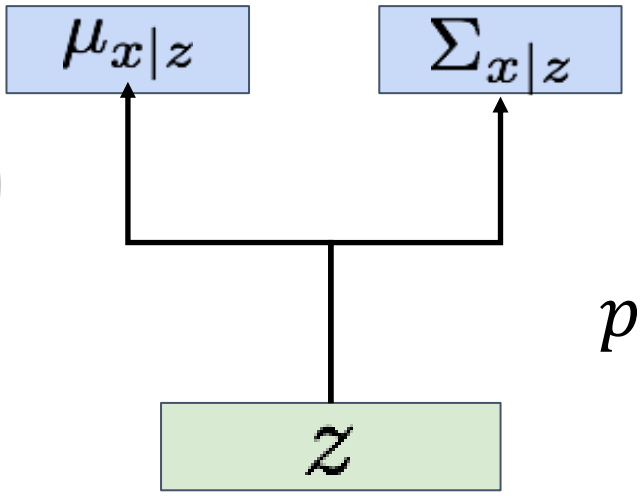
Variational Autoencoders

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:

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

Problem: Impossible to integrate over all z!

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

Variational Autoencoders

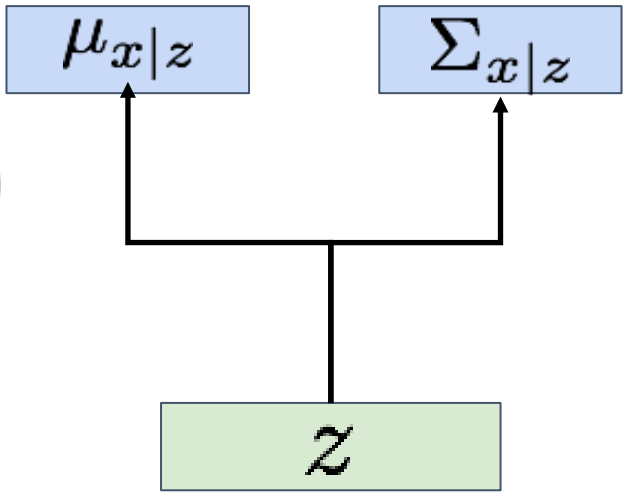
Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

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

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

How to train this model?

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



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

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

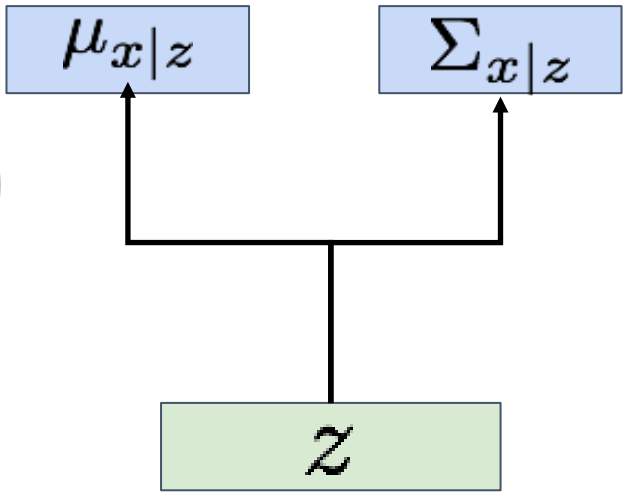
Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

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

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

How to train this model?

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



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

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

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

Variational Autoencoders

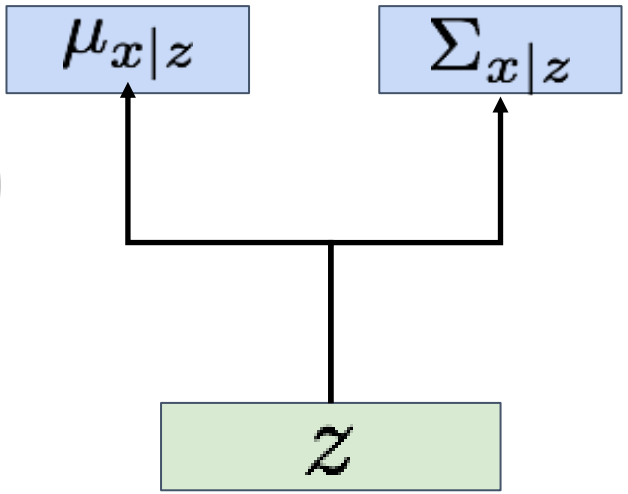
Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

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

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

How to train this model?

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



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

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, we assumed Gaussian prior

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

Variational Autoencoders

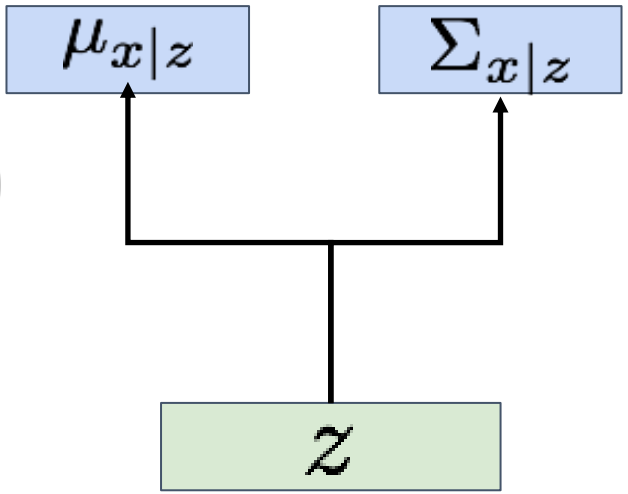
Decoder must be **probabilistic**:
Decoder inputs z, outputs mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

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

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

How to train this model?

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



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

Basic idea: **maximize likelihood of data**

Another idea: Try Bayes' Rule:

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

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 $\sum_{x|z}$

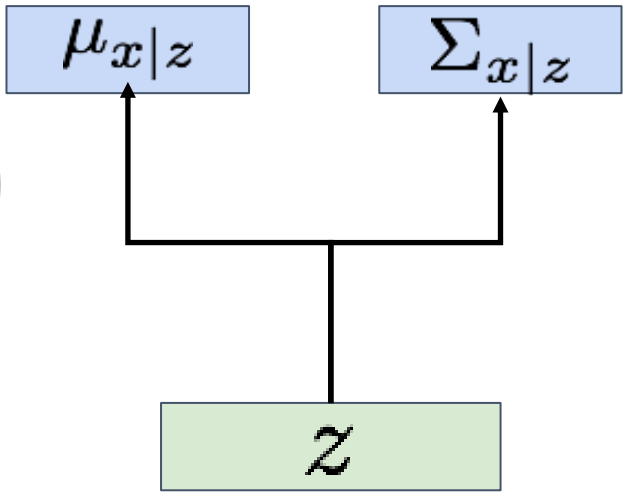
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

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

How to train this model?

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

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



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 $\sum_{x|z}$

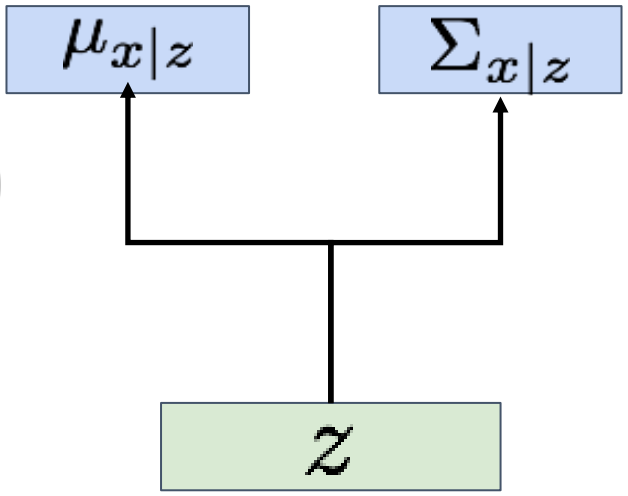
Sample x from Gaussian with mean $\mu_{x|z}$ and (diagonal) covariance $\sum_{x|z}$

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

How to train this model?

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

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



Basic idea: **maximize likelihood of data**

Another idea: Try Bayes' Rule:

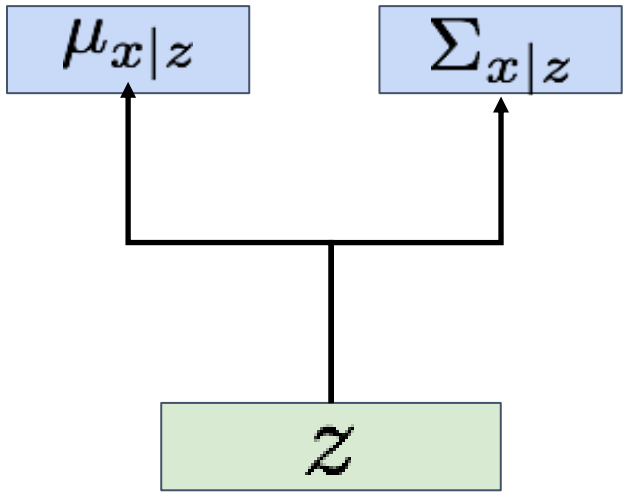
$$p_\theta(x) = \frac{p_\theta(x \mid z)p_\theta(z)}{p_\theta(z \mid x)} \approx \frac{p_\theta(x \mid z)p_\theta(z)}{q_\phi(z \mid x)}$$

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

Variational Autoencoders

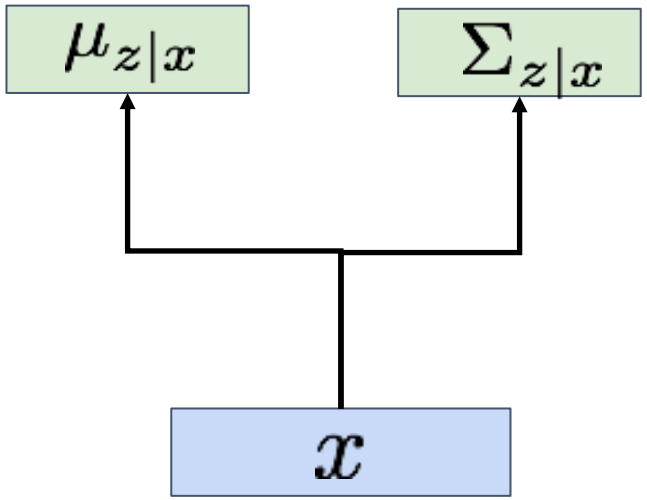
Decoder network inputs latent code z, gives distribution over data x

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



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

Reparameterization Trick

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

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

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

Split up using rules for logarithms

Variational Autoencoders

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

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

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

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

Variational Autoencoders

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

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

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

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

Variational Autoencoders

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

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

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

Data reconstruction

Variational Autoencoders

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

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

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

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

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

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

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

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

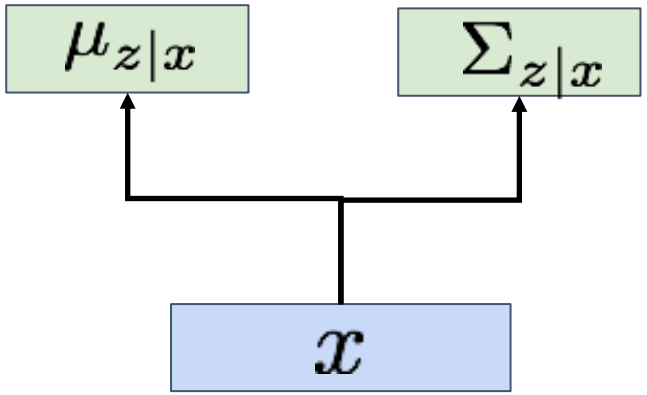
Variational Autoencoders

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

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

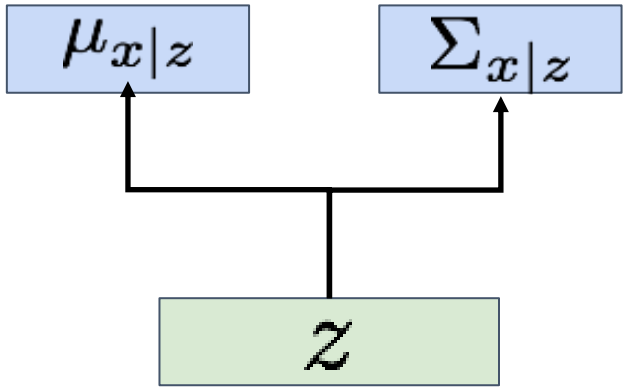
Encoder Network

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



Decoder Network

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



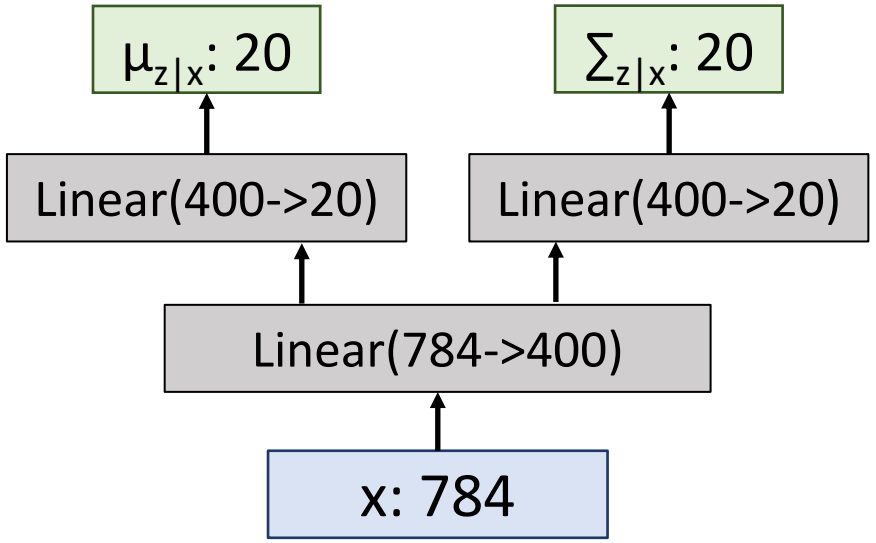
Example: Fully-Connected VAE

x: 28x28 image, flattened to 784-dim vector

z: 20-dim vector

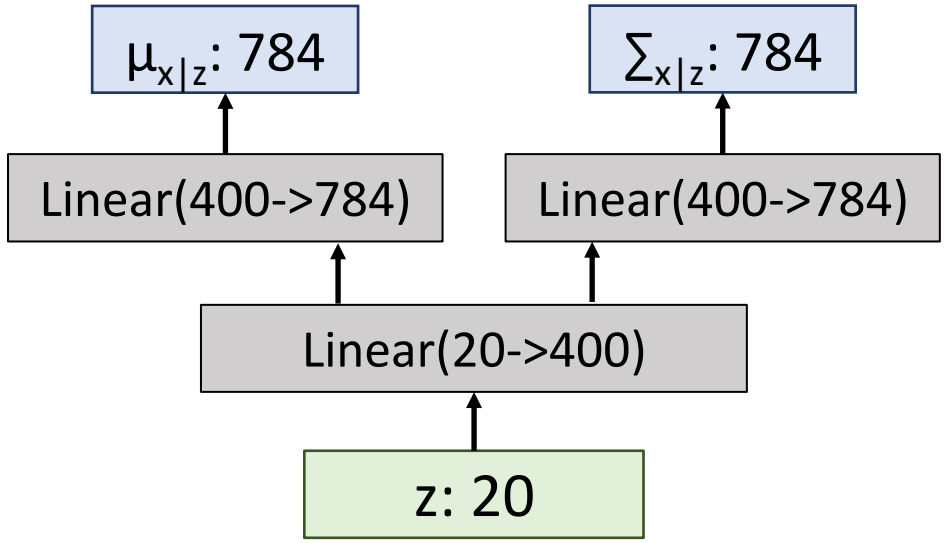
Encoder Network

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



Decoder Network

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

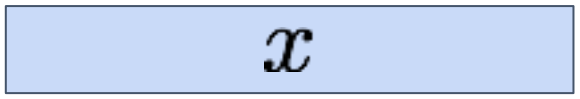


Variational Autoencoders

Train by maximizing the
variational lower bound

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

Input Data

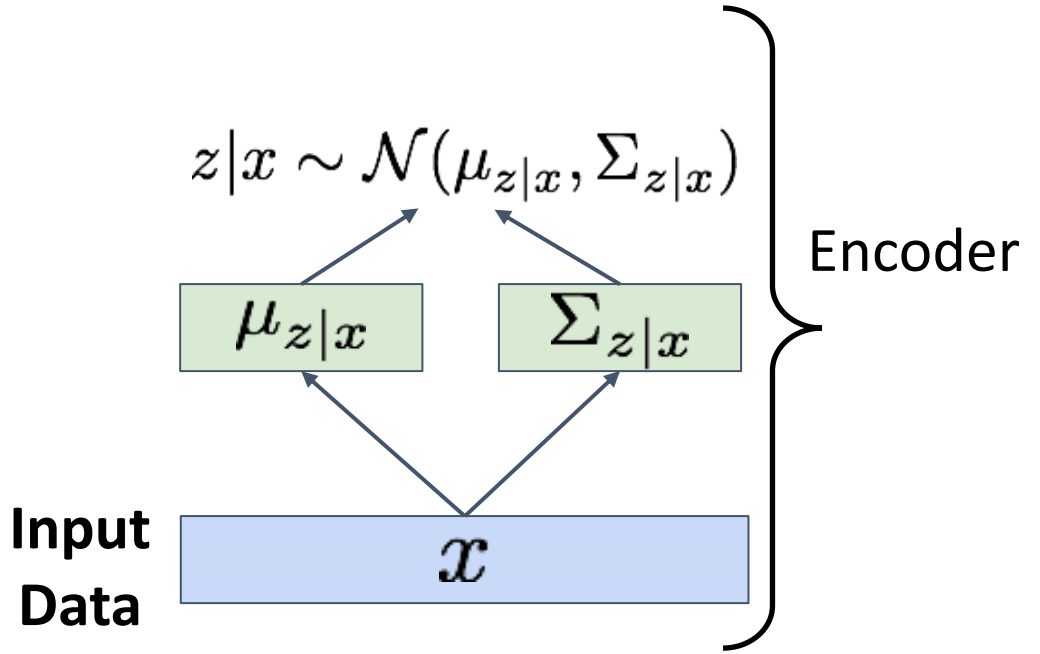


Variational Autoencoders

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

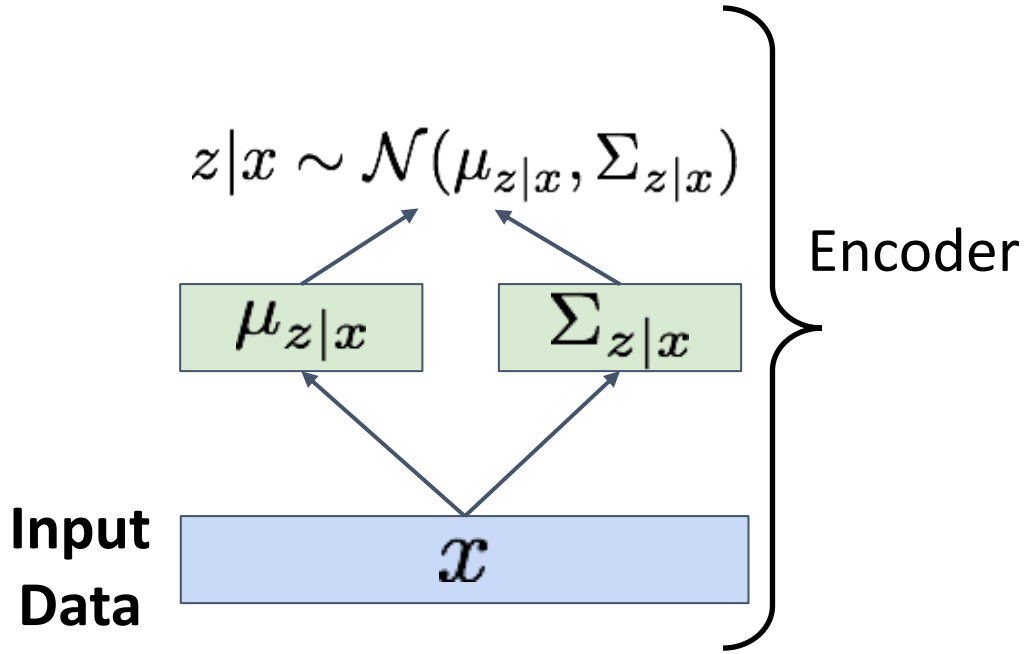


Variational Autoencoders

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

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

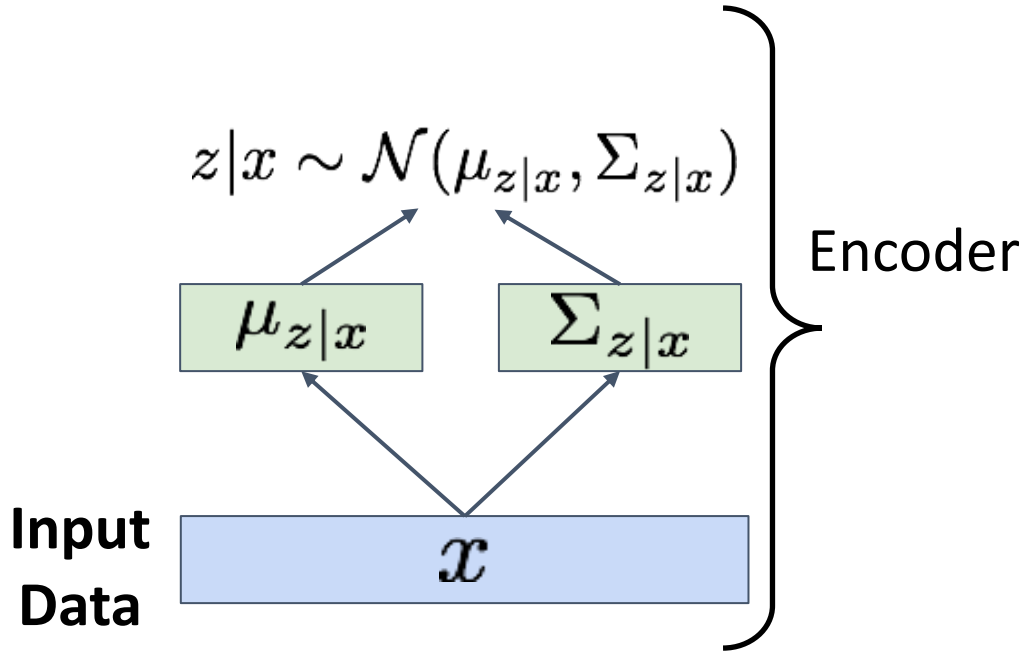
Closed form solution when q_ϕ is diagonal Gaussian and p is unit Gaussian!
(Assume z has dimension J)

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

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

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

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

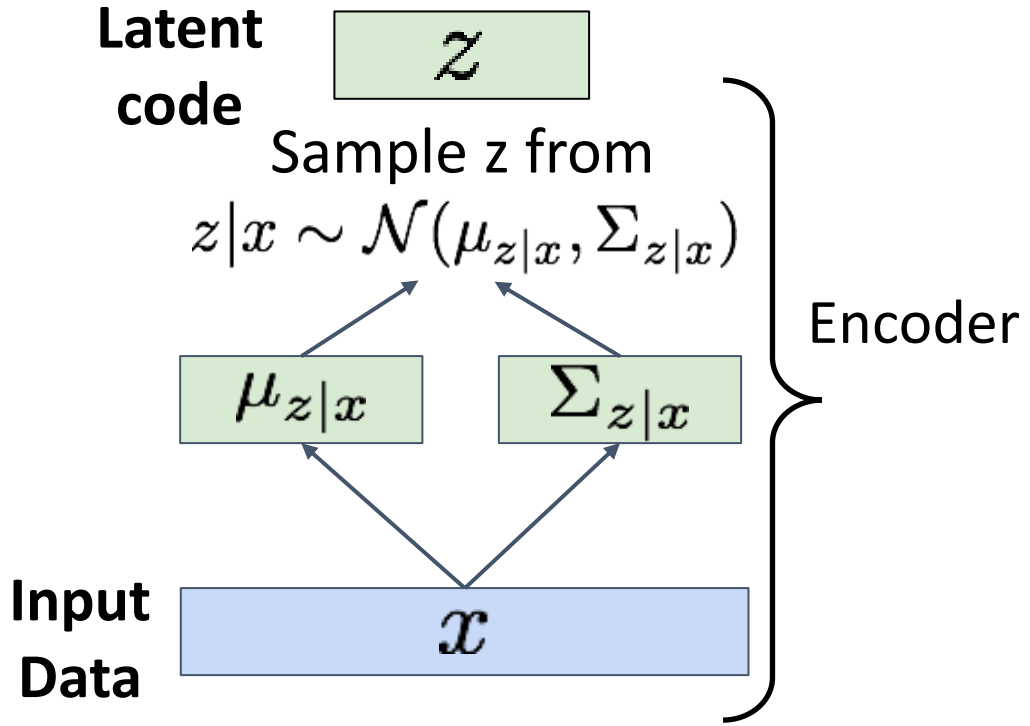


Variational Autoencoders

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

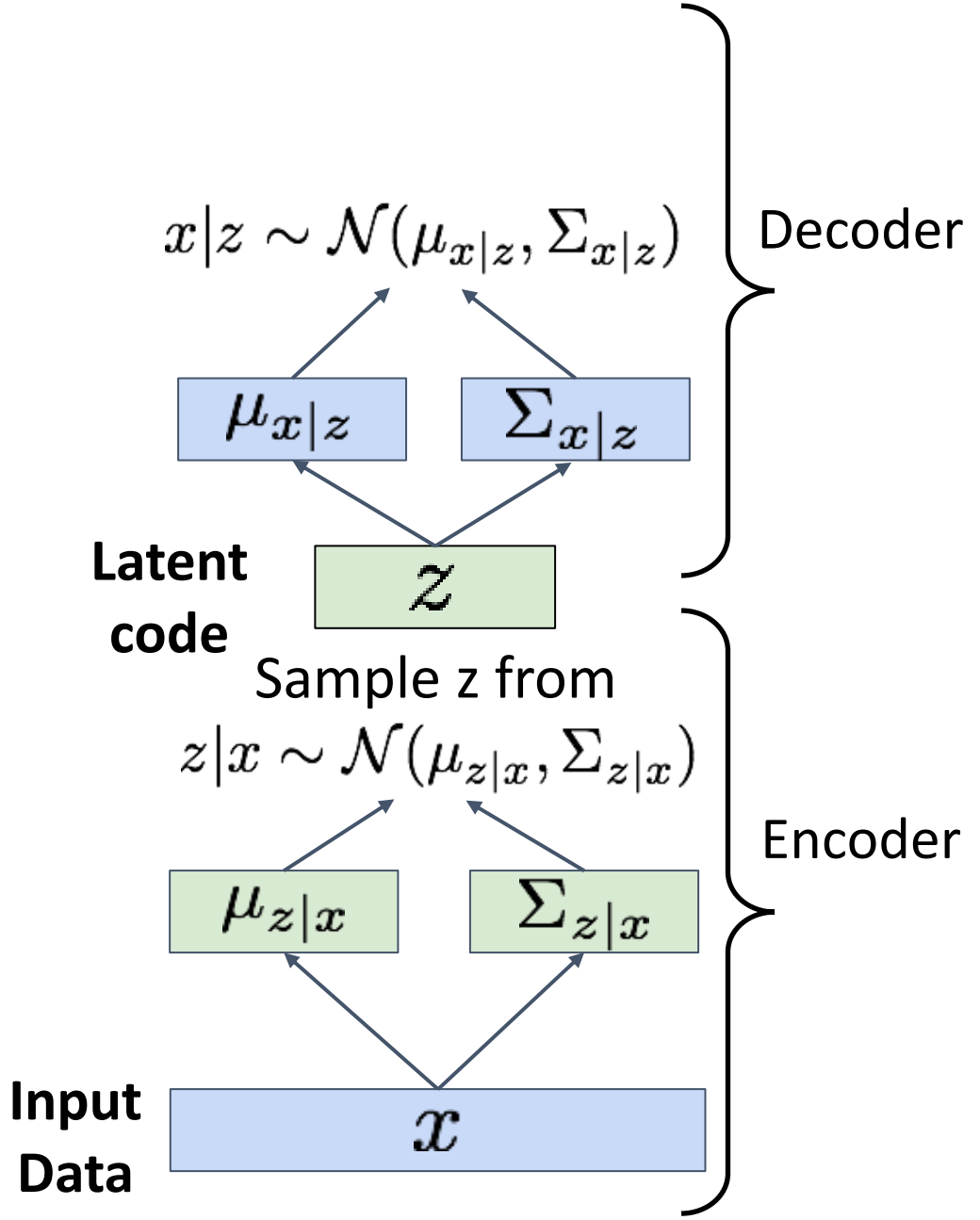


Variational Autoencoders

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

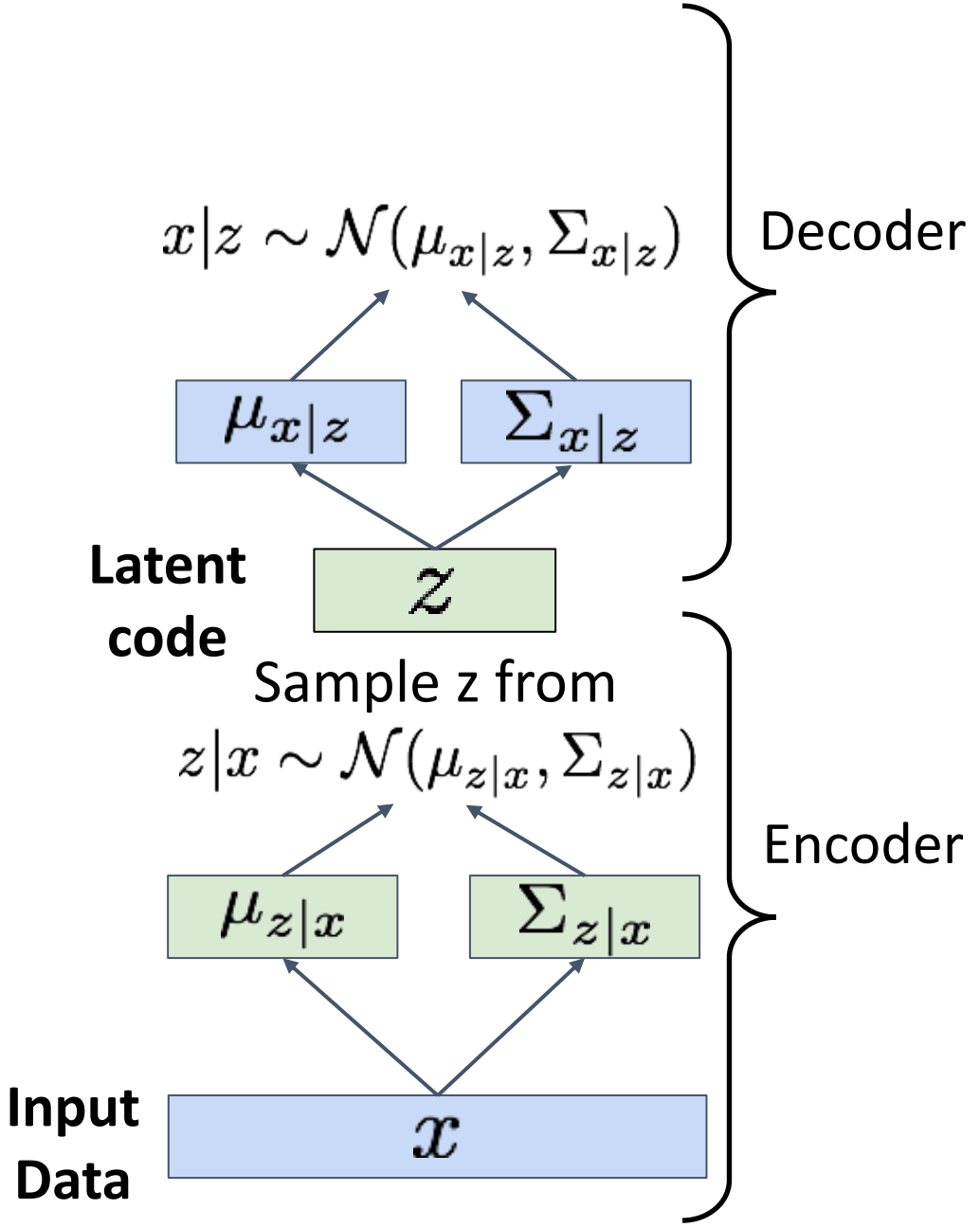


Variational Autoencoders

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)!**

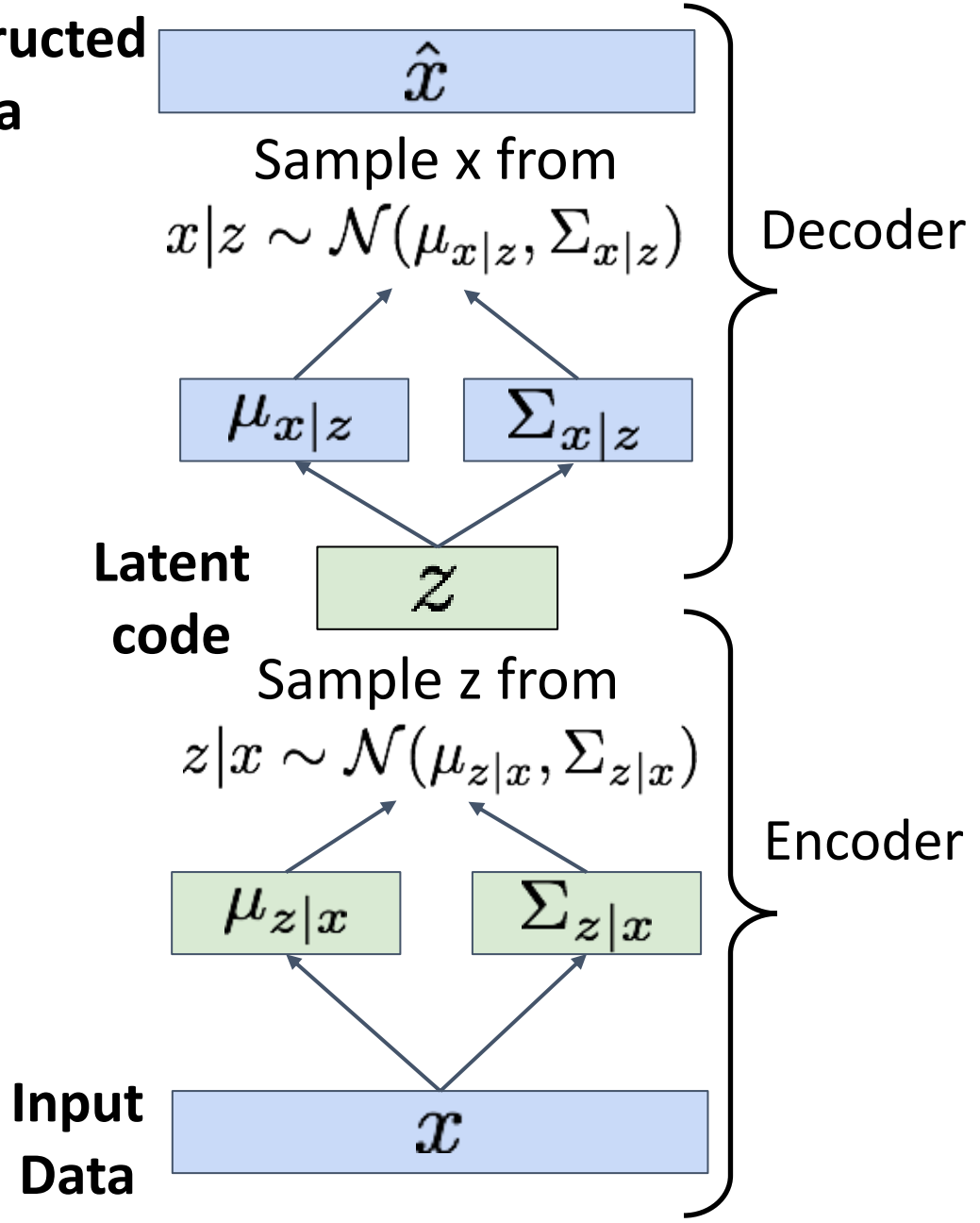


Variational Autoencoders

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)

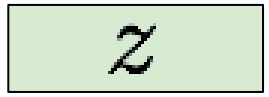


Variational Autoencoders: Generating Data

After training we can generate new data!

1. Sample z from prior p(z)

Latent code

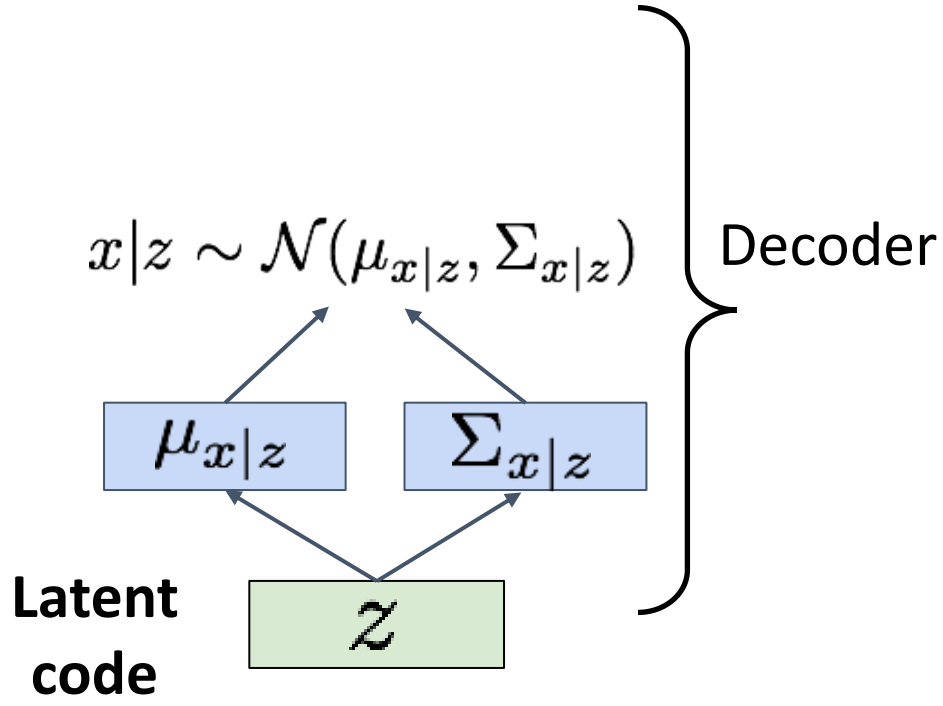


Sample z from prior p(z)

Variational Autoencoders: Generating Data

After training we can generate new data!

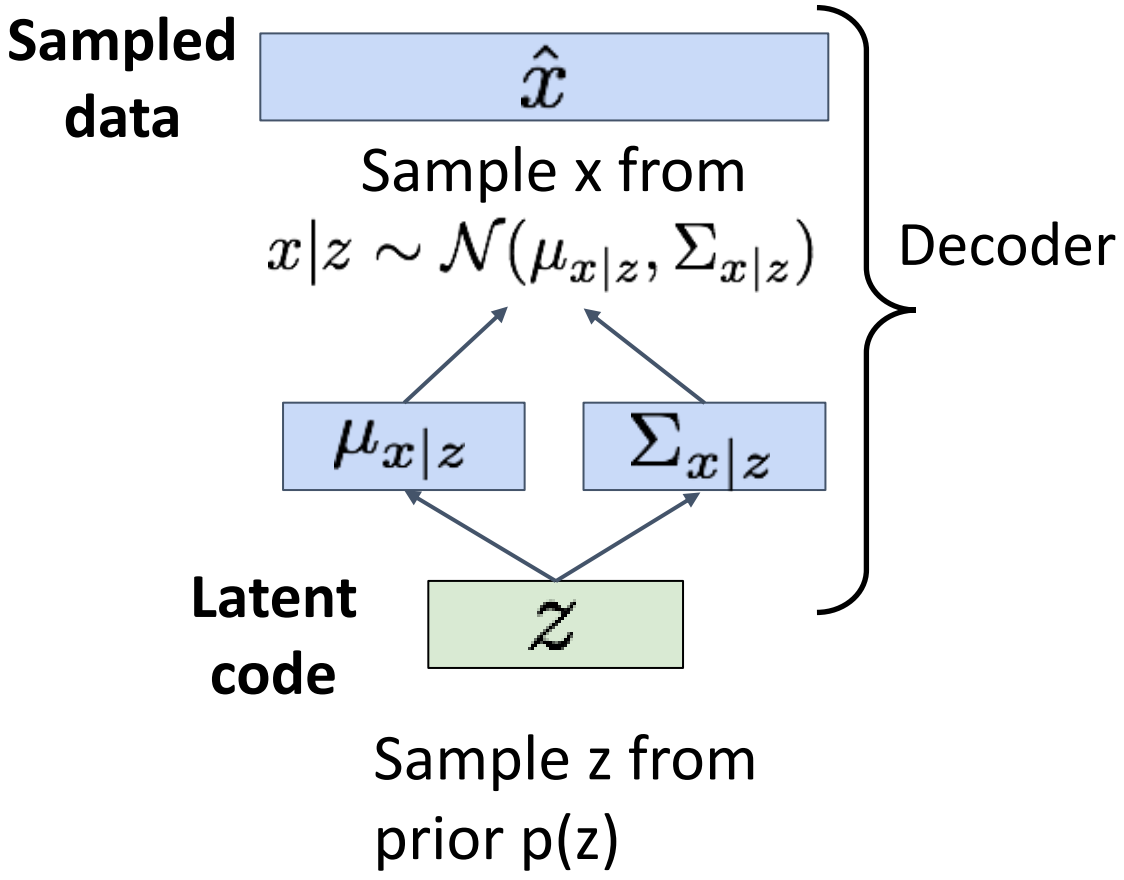
1. Sample z from prior p(z)
2. Run sampled z through decoder to get distribution over data x



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

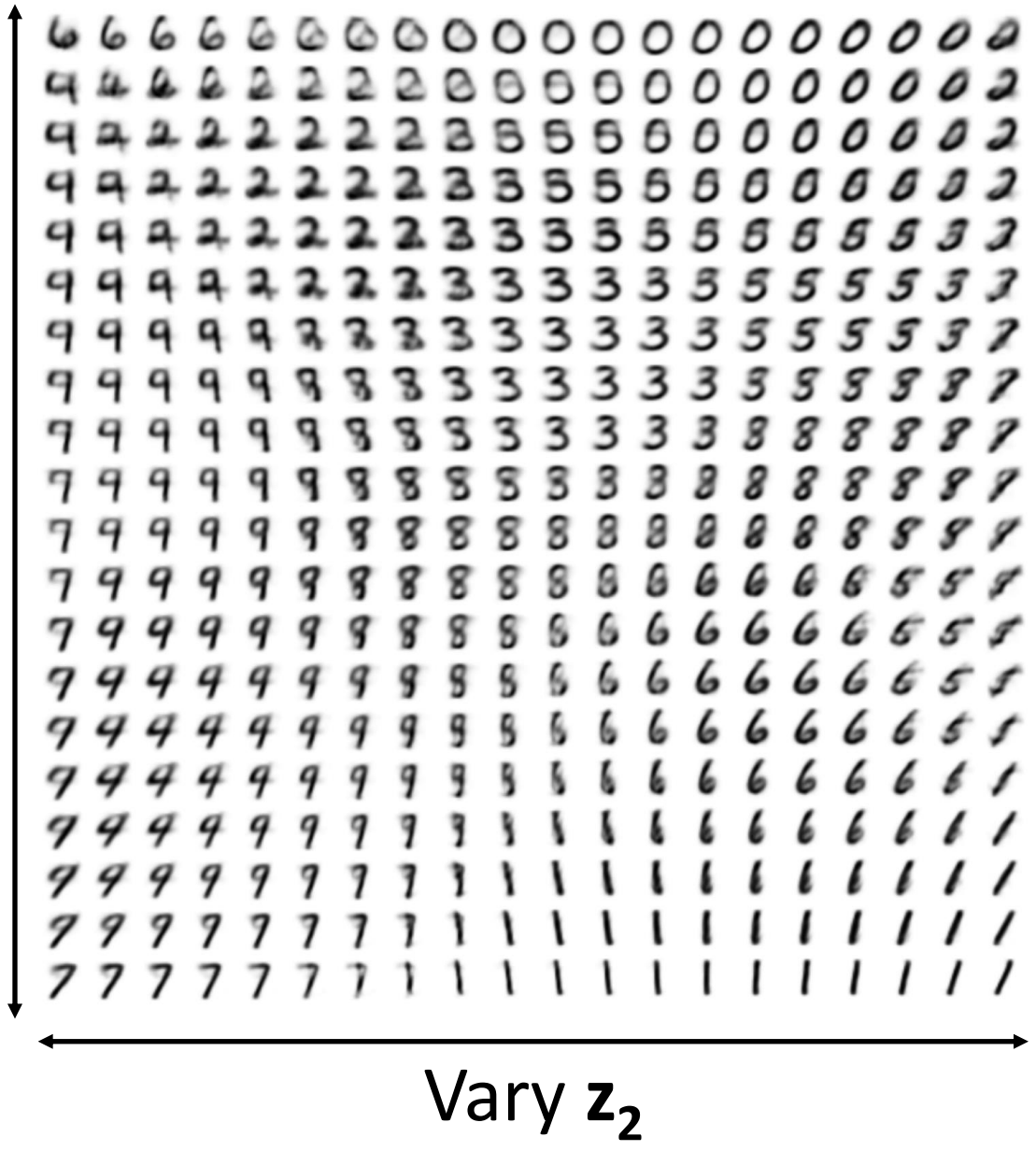
Figures from (L) Dirk Kingma et al. 2016; (R) Anders Larsen et al. 2017.

Disentanglement in VAE

Variational Autoencoders

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

“Disentangling factors of variation”

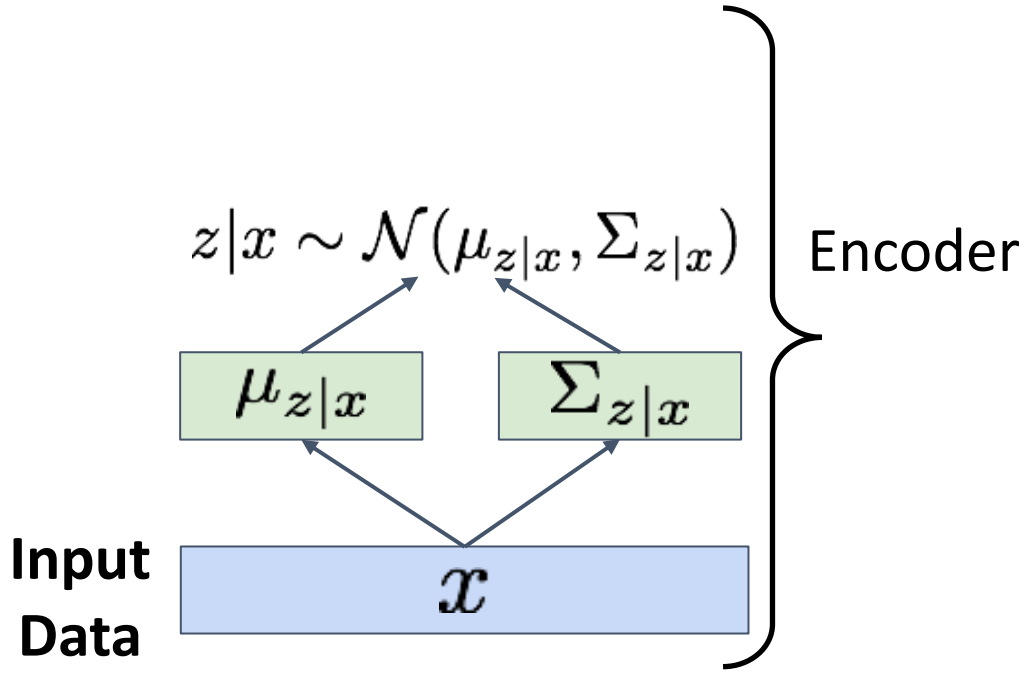


Kingma and Welling, Auto-Encoding Variational Beyes, ICLR 2014

Variational Autoencoders

After training we can **edit images**

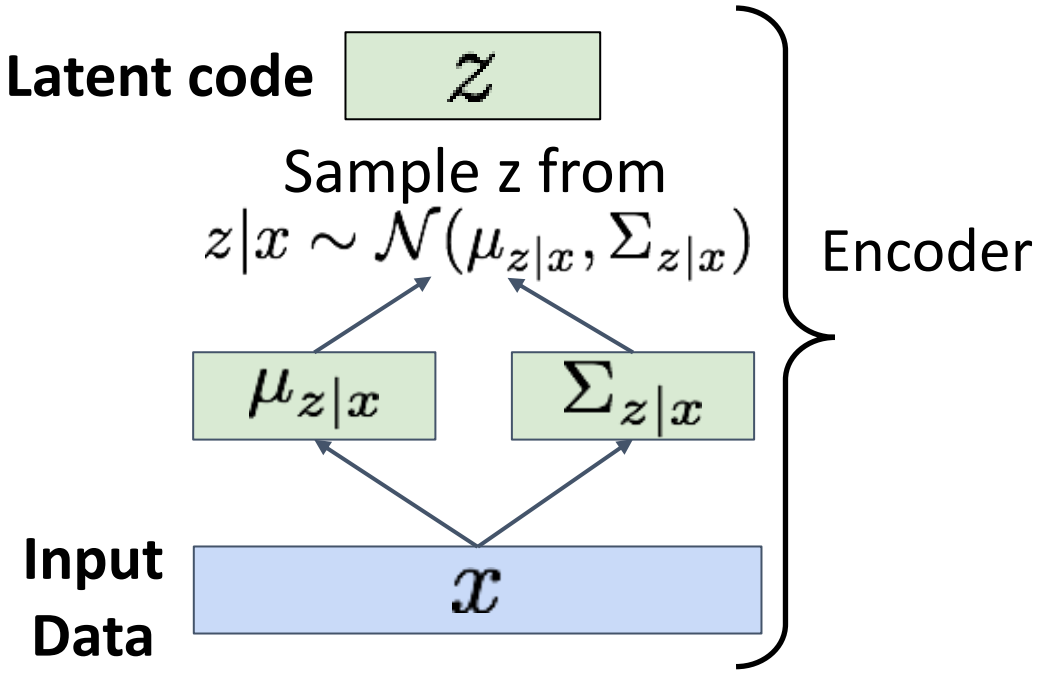
1. Run input data through **encoder** to get a distribution over latent codes



Variational Autoencoders

After training we can **edit images**

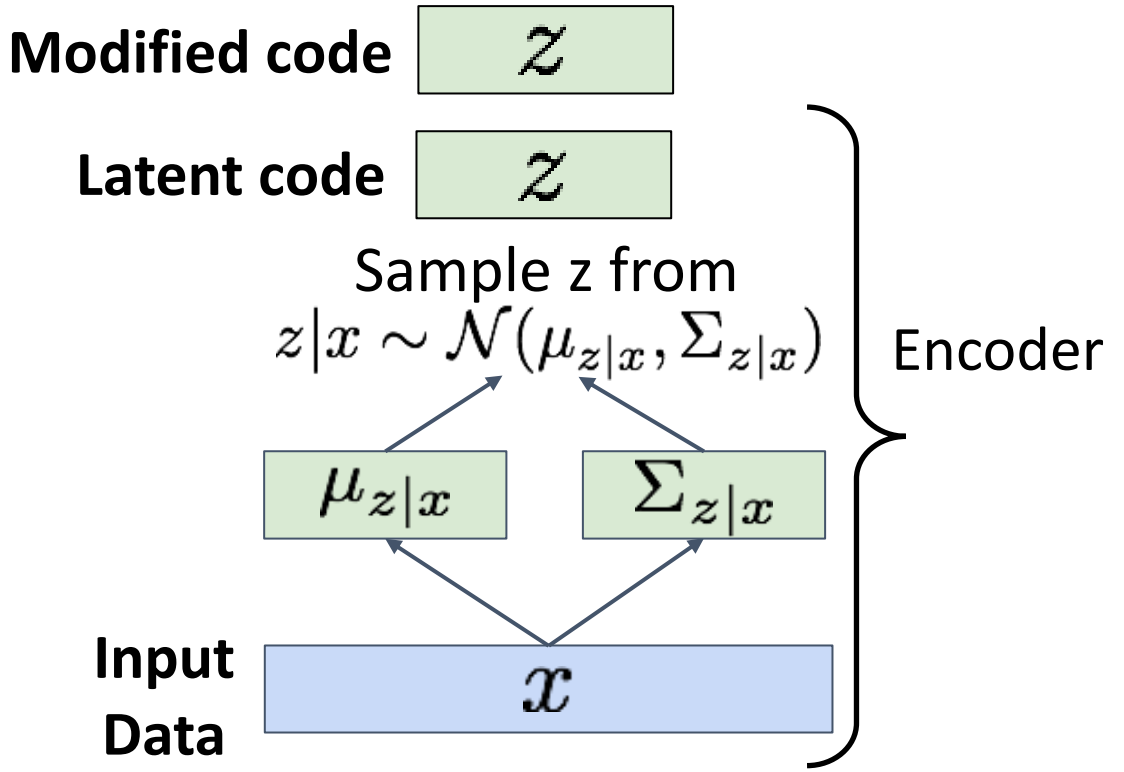
1. Run input data through **encoder** to get a distribution over latent codes
2. Sample code z from encoder output



Variational Autoencoders

After training we can **edit images**

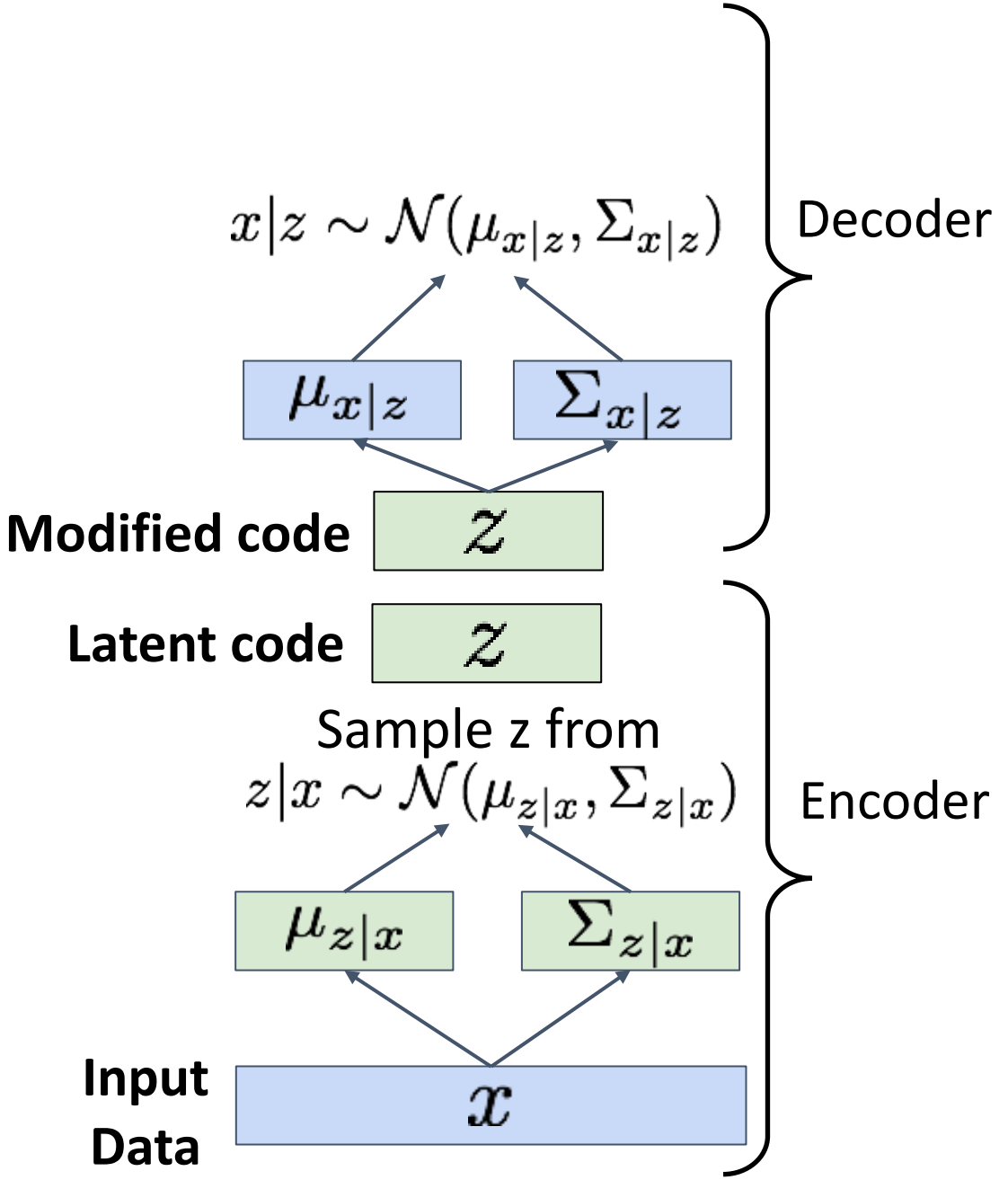
1. Run input data through **encoder** to get a distribution over latent codes
2. Sample code z from encoder output
3. Modify some dimensions of sampled code



Variational Autoencoders

After training we can **edit images**

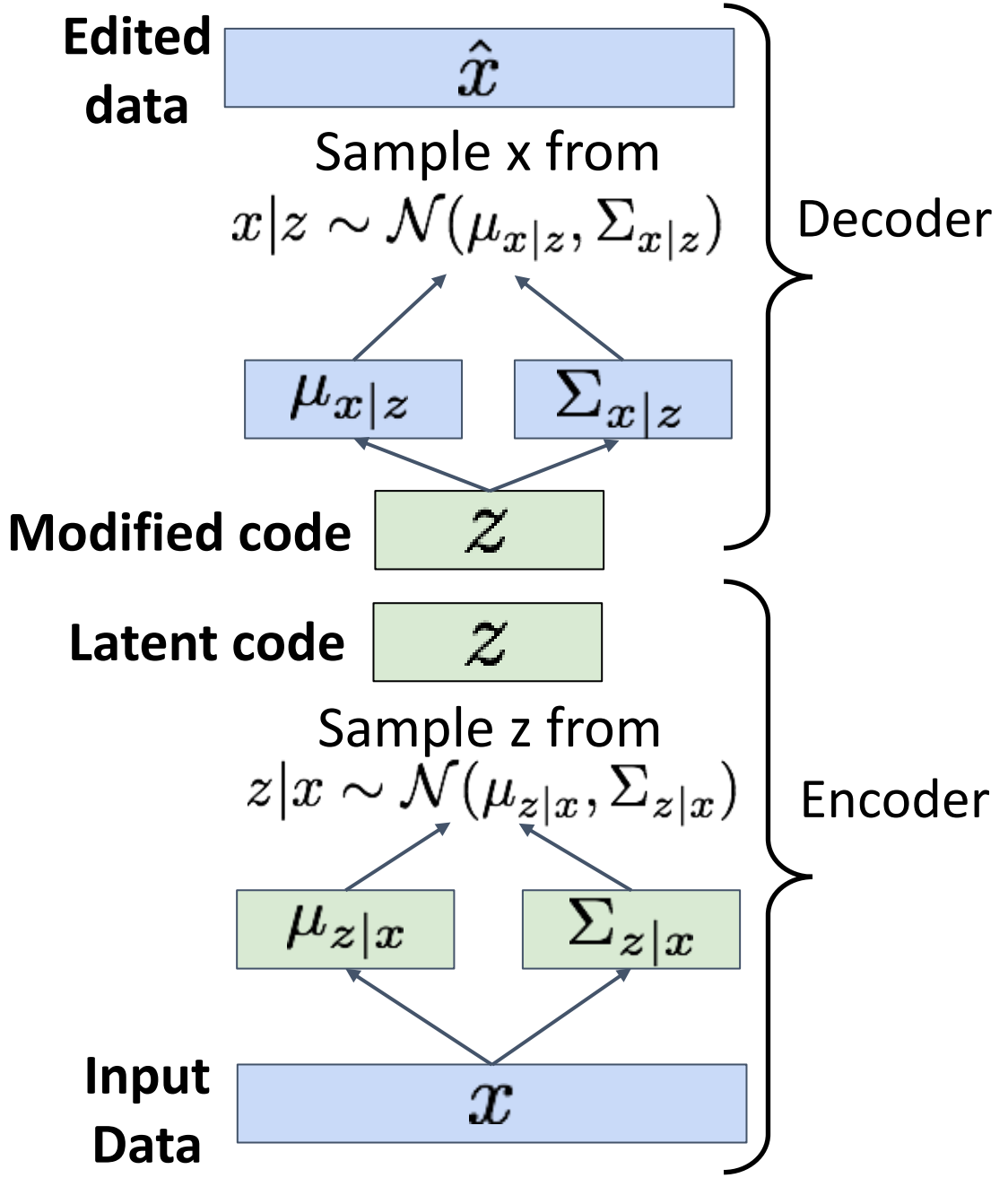
1. Run input data through **encoder** to get a distribution over latent codes
2. Sample code z from encoder output
3. Modify some dimensions of sampled code
4. Run modified z through **decoder** to get a distribution over data sample



Variational Autoencoders

After training we can **edit images**

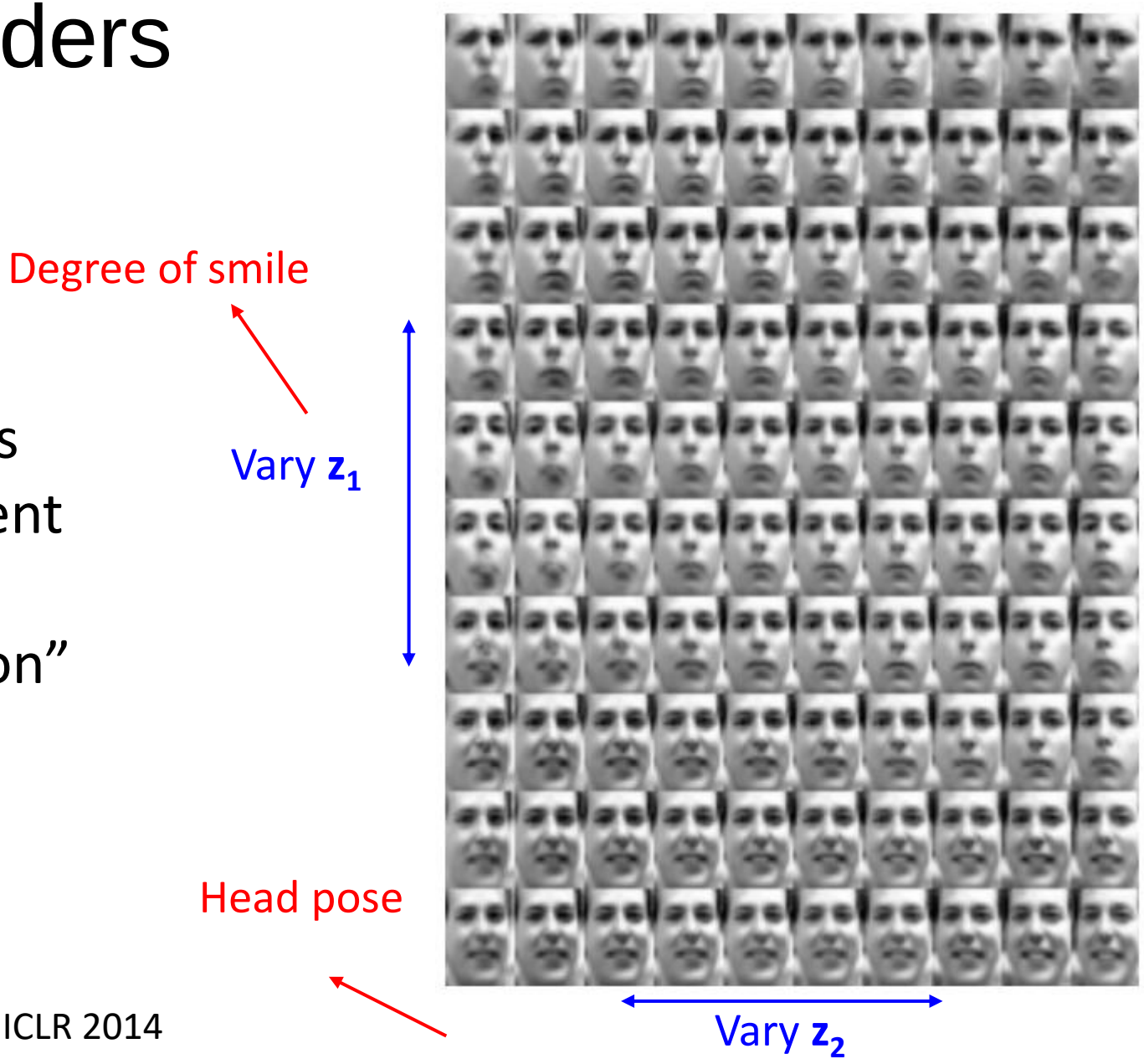
1. Run input data through **encoder** to get a distribution over latent codes
2. Sample code z from encoder output
3. Modify some dimensions of sampled code
4. Run modified z through **decoder** to get a distribution over data samples
5. Sample new data from (4)



Variational Autoencoders

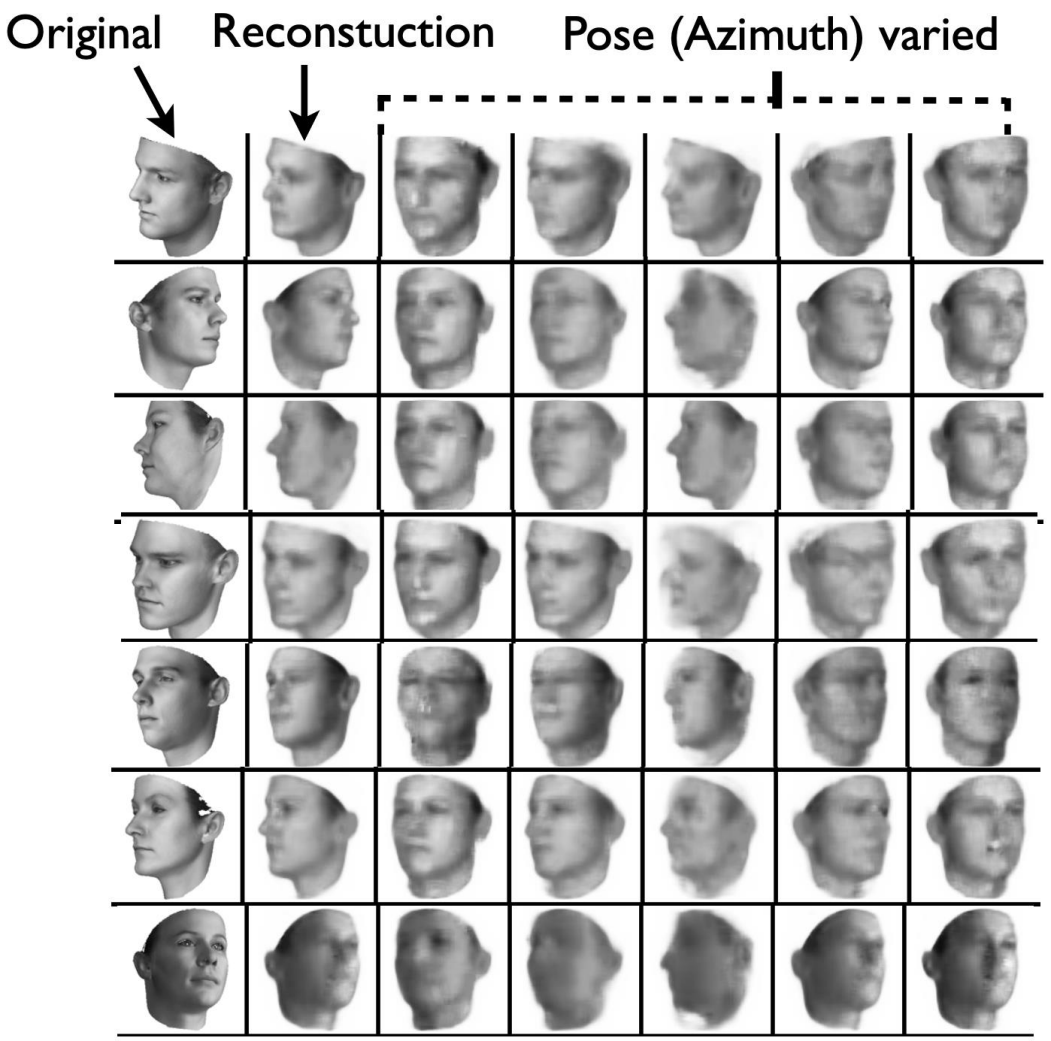
The diagonal prior on p(z) causes dimensions of z to be independent

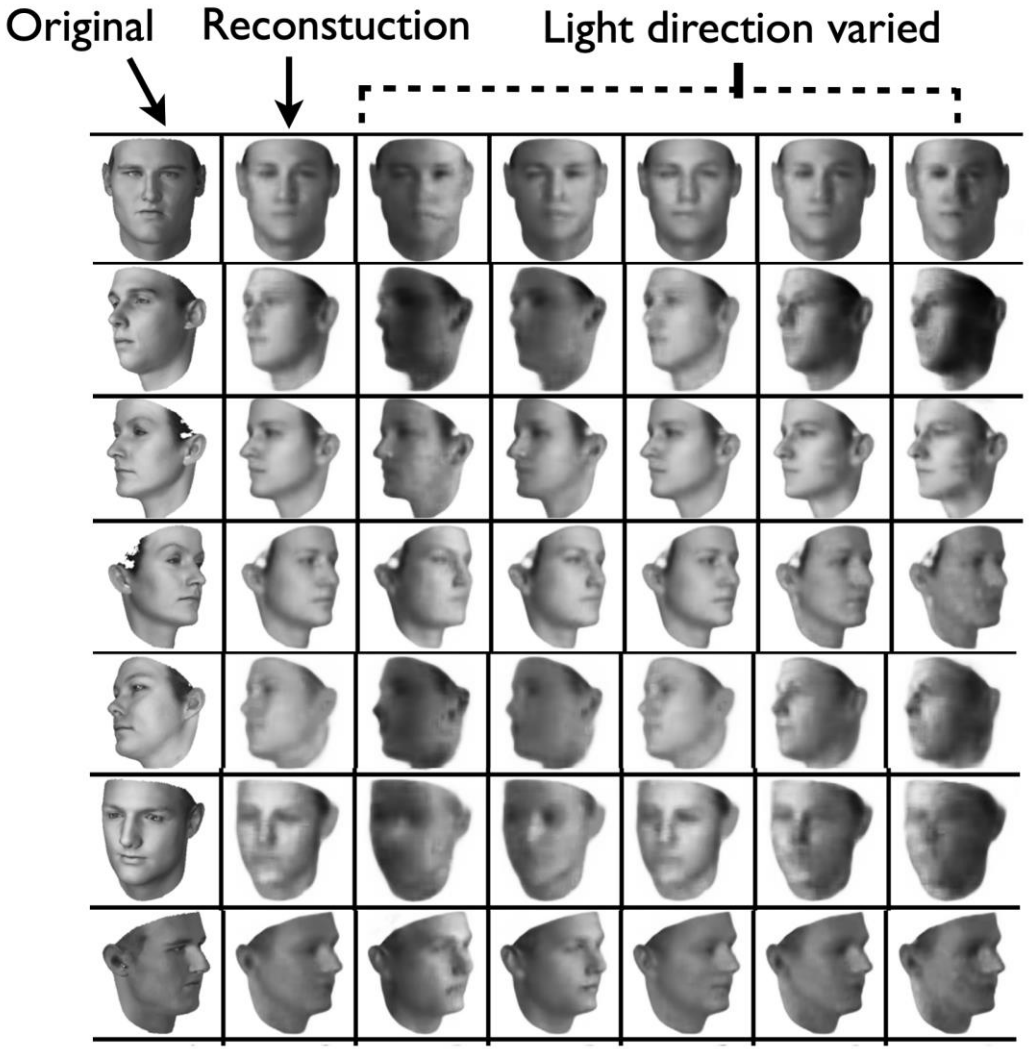
“Disentangling factors of variation”



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

Variational Autoencoders: 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 autoregressive models (PixelCNN/PixelRNN)
- Samples blurrier and lower quality compared to state-of-the-art (GANs)

Active areas of research:

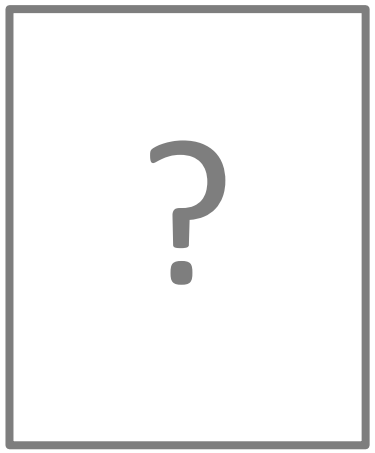
- More flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian, e.g., Gaussian Mixture Models (GMMs)
- Incorporating structure in latent variables, e.g., Categorical Distributions

VAE as Generative Model

Generative Models (recap)

- Assumption: the dataset are samples from an unknown distribution $p_{\text{data}}(x)$
- Goal: create a new sample from $p_{\text{data}}(x)$ that is not in the dataset

Dataset



Generated

Image credit: *Progressive Growing of GANs for Improved Quality, Stability, and Variation,* Karras et al.

Generative Models (recap)

- Assumption: the dataset are samples from an unknown distribution $p_{\text{data}}(x)$
- Goal: create a new sample from $p_{\text{data}}(x)$ that is not in the dataset

Dataset

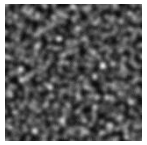
Generated

Image credit: *Progressive Growing of GANs for Improved Quality, Stability, and Variation,* Karras et al.

Generative Models (recap)

$$p_\theta(x) \approx p_{\text{data}}(x)$$

Generator with parameters θ



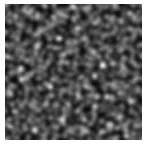
$$p(z)$$

known and
easy to sample from

Generative Models (recap)

$p_\theta(x) \approx p_{\text{data}}(x)$

Generator with parameters θ



$p(z)$

known and easy to sample from

How to measure similarity of $p_\theta(x)$ and $p_{\text{data}}(x)$?

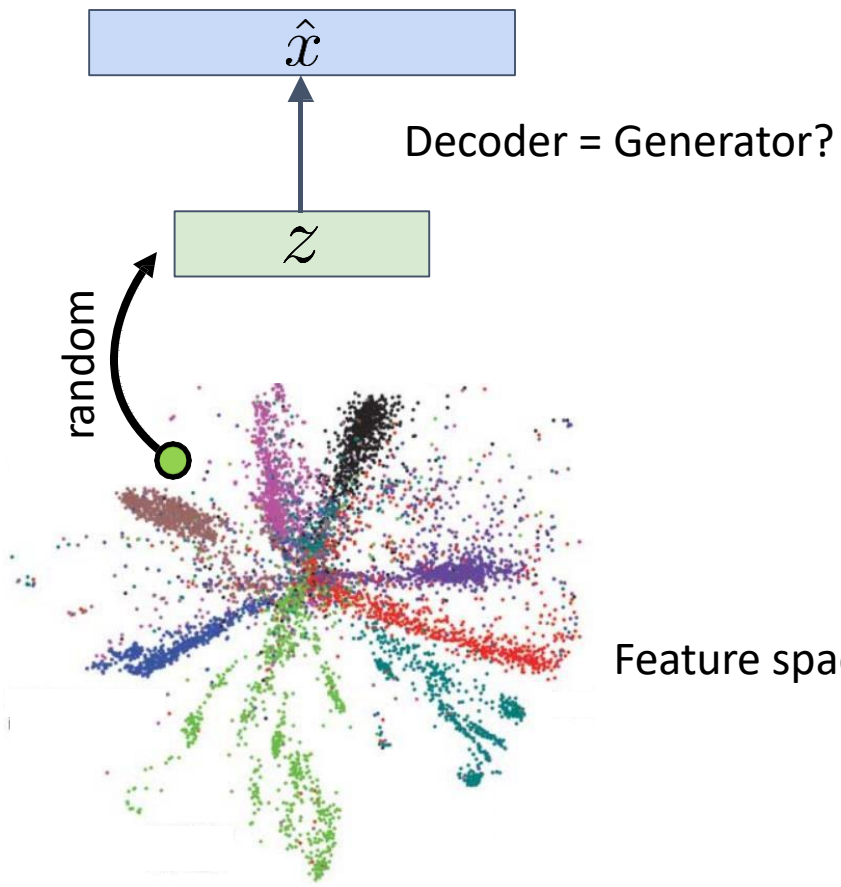
1) Likelihood of data in $p_\theta(x)$

Variational Autoencoders (VAEs)

2) Adversarial game:
Discriminator distinguishes $p_\theta(x)$ and $p_{\text{data}}(x)$ vs *Generator* makes it hard to distinguish

Generative Adversarial Networks (GANs)

Autoencoders as Generative Models?

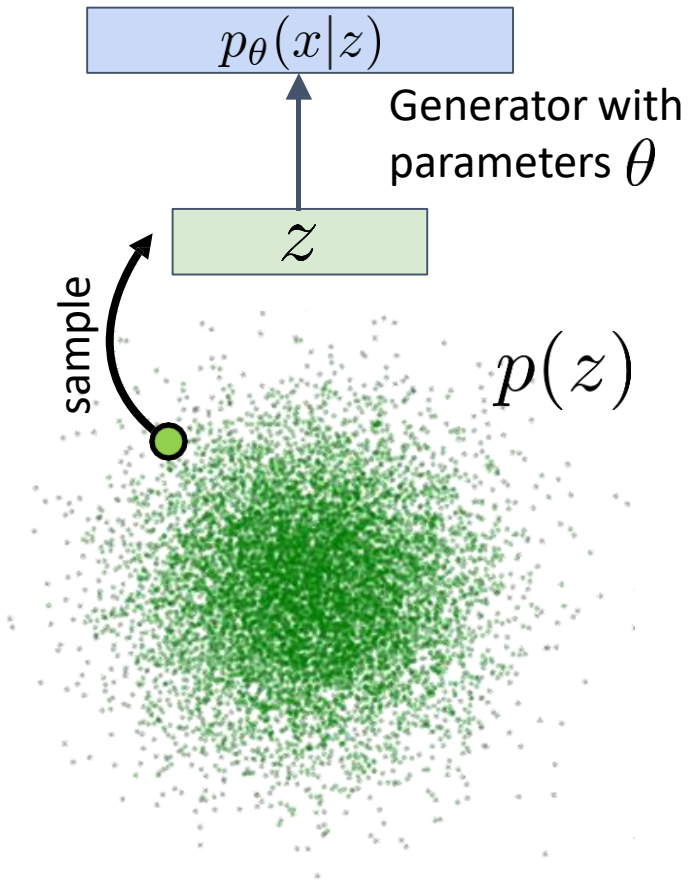


Feature space / latent space

- A trained decoder transforms some features z to approximate samples from $p_{\text{data}}(x)$
- What happens if we pick a random z?
- We do not know the distribution $p(z)$ of features that decode to likely samples

Image Credit: *Reducing the Dimensionality of Data with Neural Networks*, Hinton and Salakhutdinov

Variational Autoencoders (VAEs)



- Pick a parametric distribution $p(z)$ for features
- The generator maps $p(z)$ to an image distribution $p_\theta(x)$ (where θ are parameters)

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

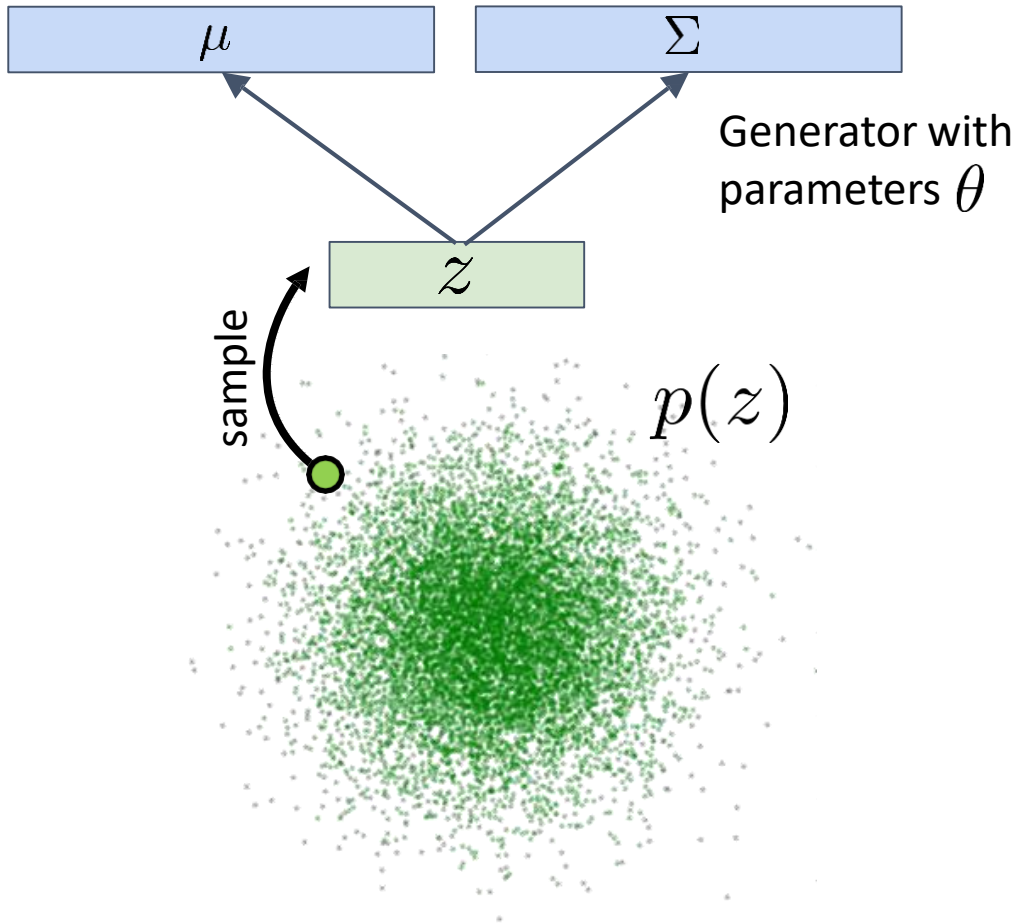
- Train the generator to maximize the likelihood of the data in $p_\theta(x)$:

$$\max_\theta \sum_{x \in \text{data}} \log p_\theta(x)$$

Outputting a Distribution

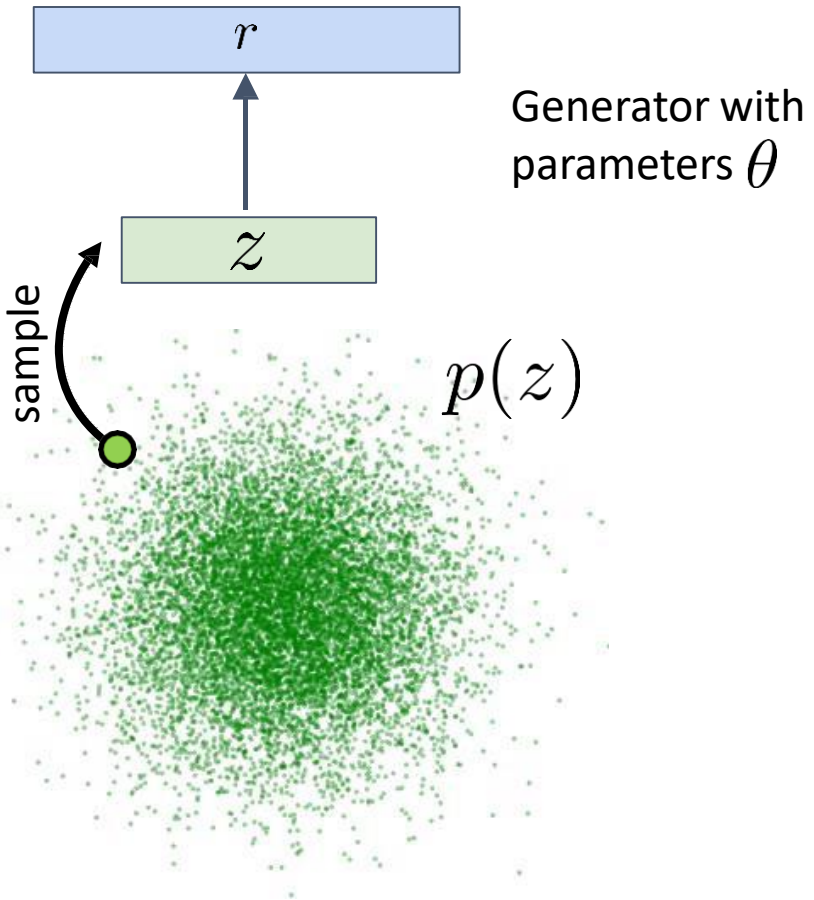
Normal distribution

$$p_\theta(x|z) = N(x; \mu(z), \Sigma(z))$$



Bernoulli distribution

$$p_\theta(x|z) = Bern(x; r(z))$$



Variational Autoencoders (VAEs): Naïve Sampling (Monte-Carlo)

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

$$\max_\theta \sum_{x \in \text{data}} \log p_\theta(x)$$

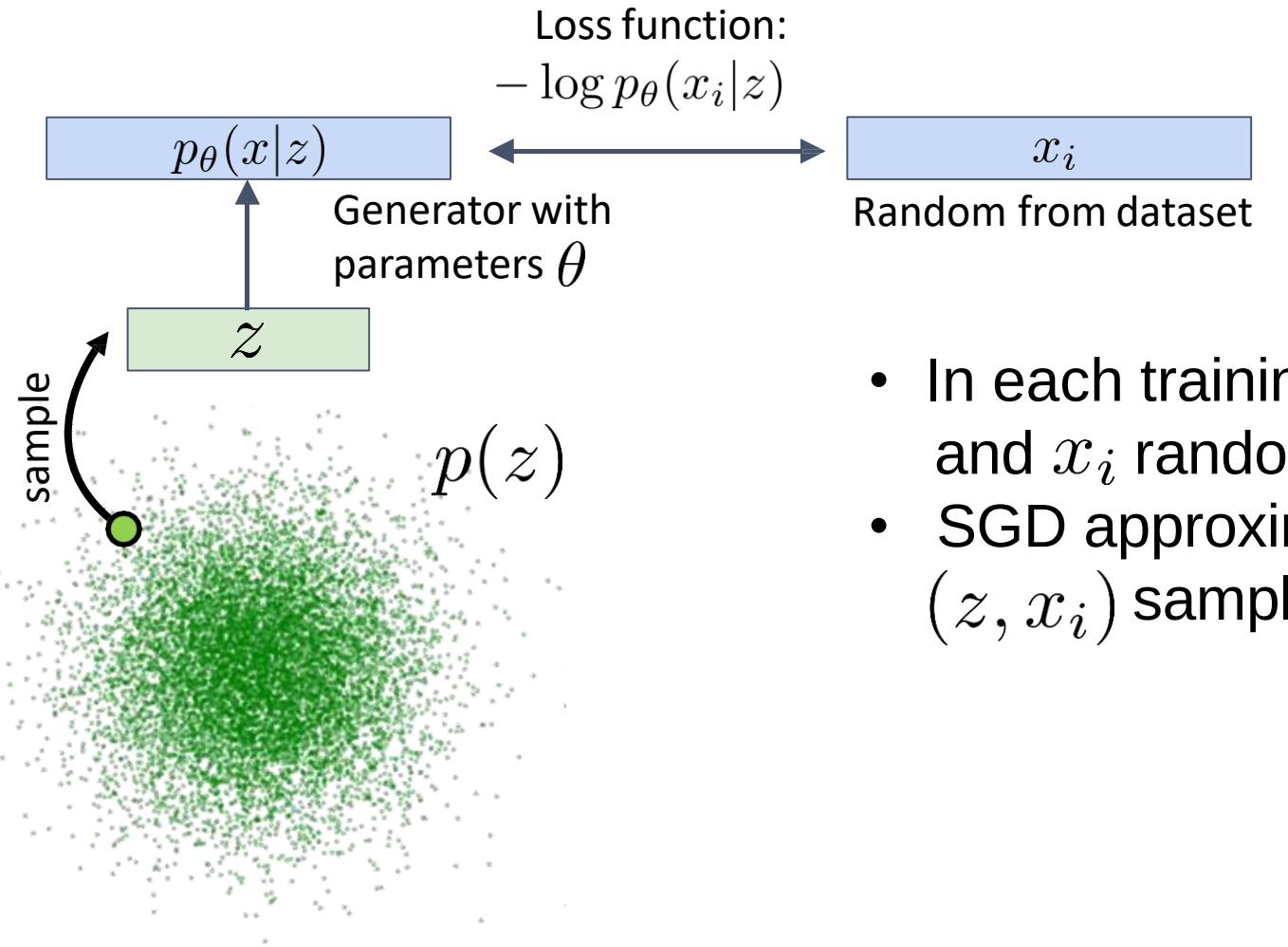
$$\theta^* = \arg\max_\theta \sum_{x \in \text{data}} \log \int p_\theta(x|z)\ p(z)\ dz$$

$$\theta^* \approx \arg\max_\theta \mathbb{E}_{x_i \sim p_{\text{data}}(x)} \mathbb{E}_{z \sim p(z)} \log p_\theta(x_i|z)$$

- SGD approximates the expected values over (z, x_i) samples
- In each training iteration, sample z from $p(z)$ …
- … and x_i randomly from the dataset, and maximize:

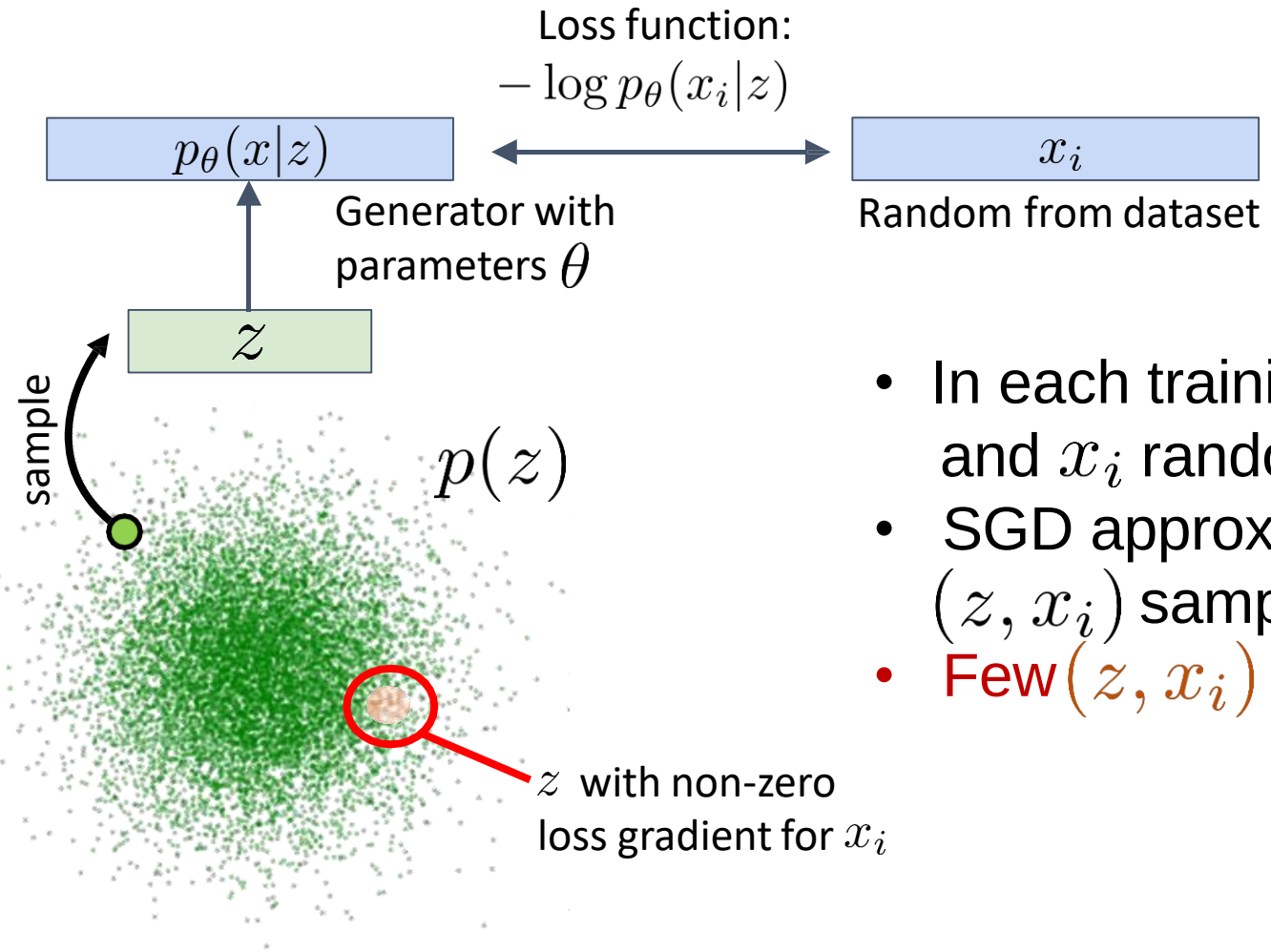
$$\max_\theta \log p_\theta(x_i|z)$$

Variational Autoencoders (VAEs): Naïve Sampling (Monte-Carlo)



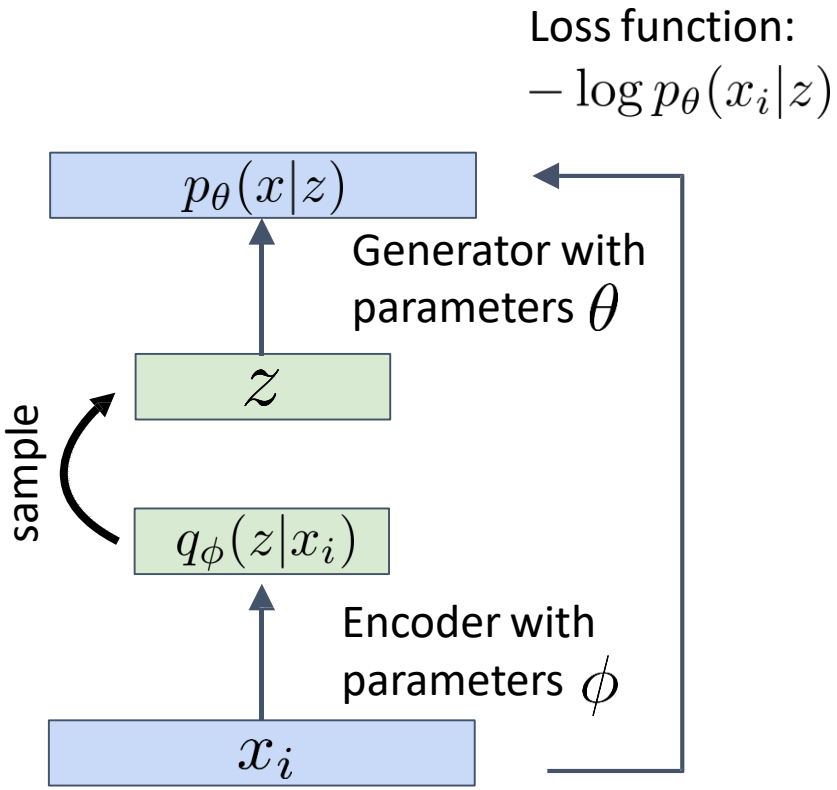
- In each training iteration, sample z from $p(z)$ and x_i randomly from the dataset
- SGD approximates the expected values over (z, x_i) samples

Variational Autoencoders (VAEs): Naïve Sampling (Monte-Carlo)



- In each training iteration, sample z from $p(z)$ and x_i randomly from the dataset
- SGD approximates the expected values over (z, x_i) samples
- Few (z, x_i) pairs have non-zero gradients

Variational Autoencoders (VAEs): The Encoder



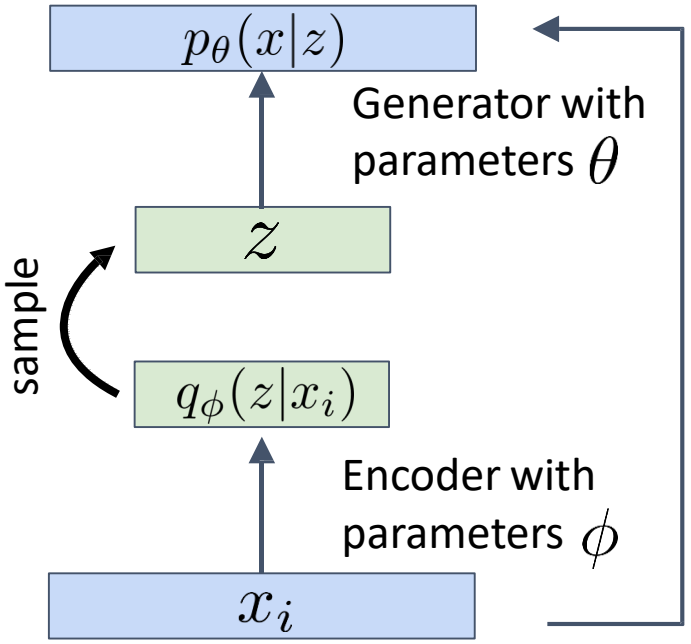
$$p_\theta(x) = \int p_\theta(x|z)\ p(z)\ dz$$

- During training, another network can guess a good z for a given x_i
- $q_\phi(z|x_i)$ should be much smaller than $p(z)$
- This also gives us the data point x_i

Variational Autoencoders (VAEs): The Encoder

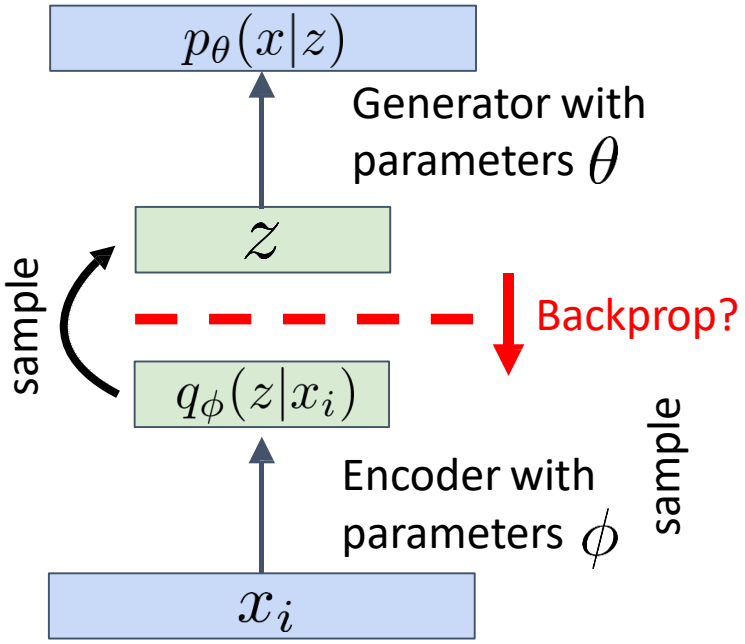
Loss function:

$$-\log p_\theta(x_i|z) + KL(\ q_\phi(z|x_i)\ \|\ p(z)\ )$$



- Can we still easily sample a new z?
- Need to make sure $q_\phi(z|x_i)$ approximates $p(z)$
- Regularize with KL-divergence
- Negative loss can be shown to be a lower bound for the likelihood, and equivalent if $q_\phi(z|x) = p_\theta(z|x)$

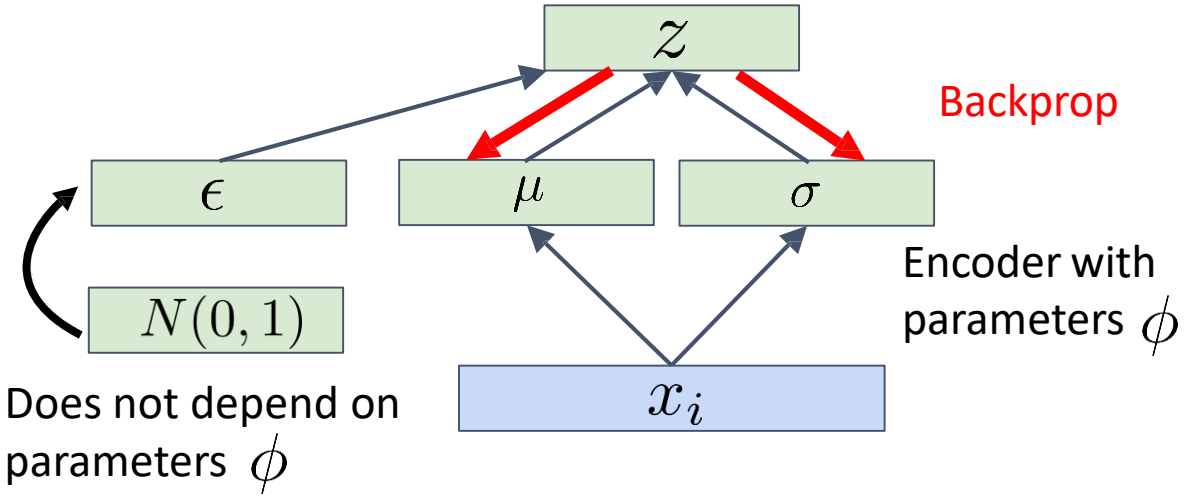
Reparameterization Trick



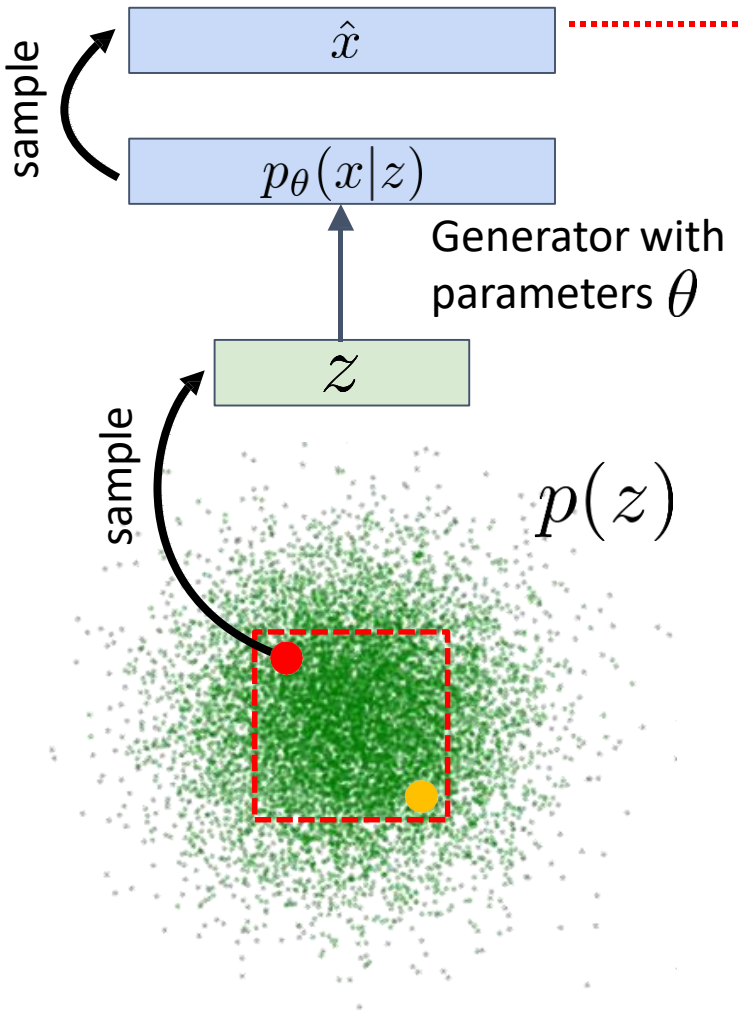
Example when $q_\phi(z|x_i) = N(z; \mu(x_i), \sigma(x_i))$:

$z = \sigma + \mu \cdot \epsilon$, where $\epsilon \sim N(0,1)$

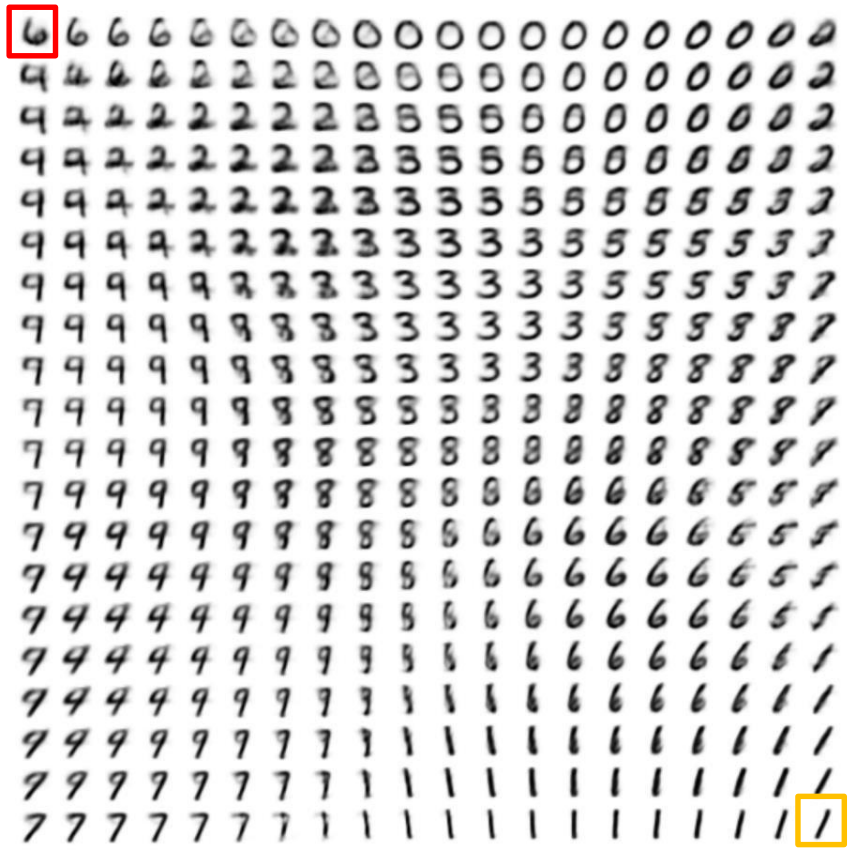
$$\frac{\partial z}{\partial \phi} = \frac{\partial \mu}{\partial \phi} + \frac{\partial \sigma}{\partial \phi} \cdot \epsilon$$



Generating Data



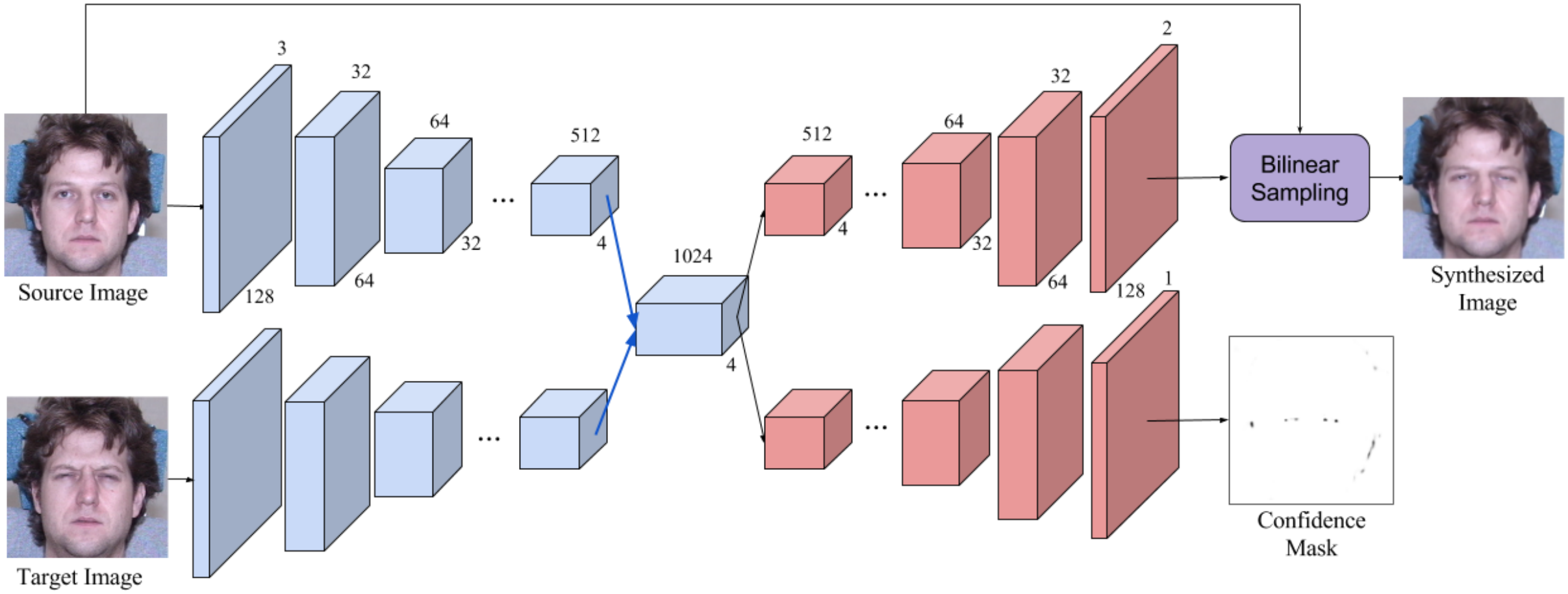
MNIST



Frey Faces

Image Credit: *Auto-Encoding Variational Bayes*, Kingma and Welling

Facial Expression Editing

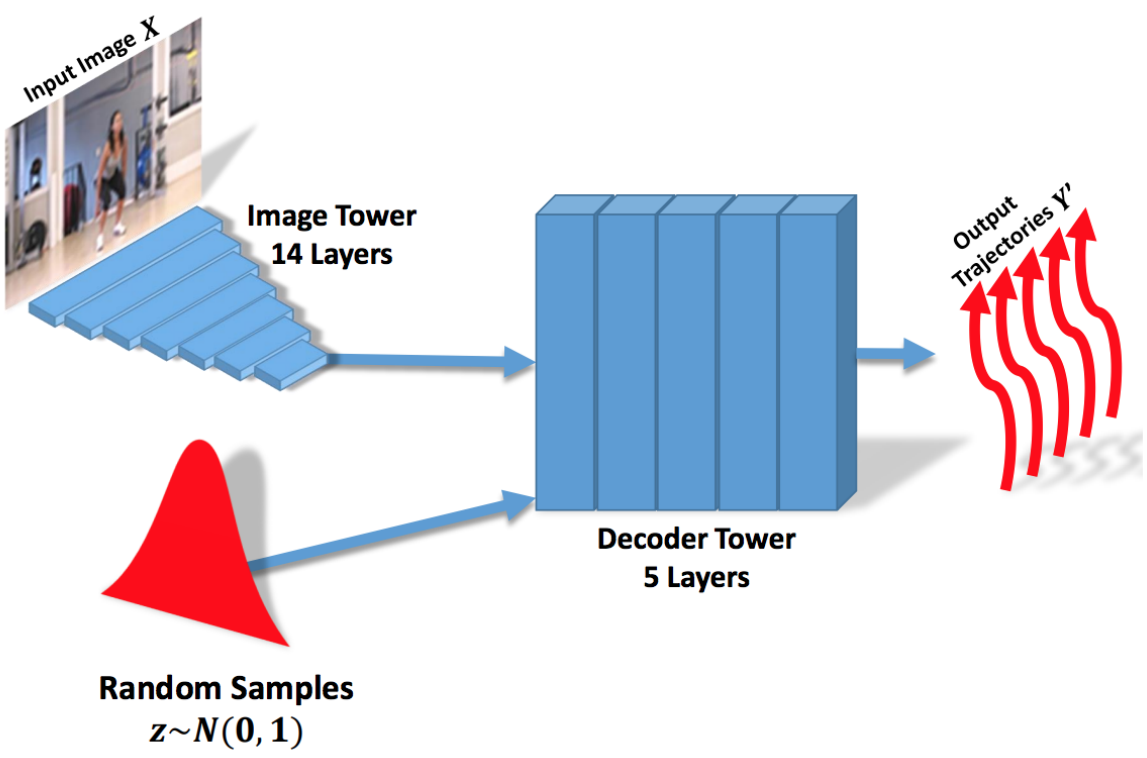


Yeh et al., Semantic Facial Expression Editing using Autoencoded Flow

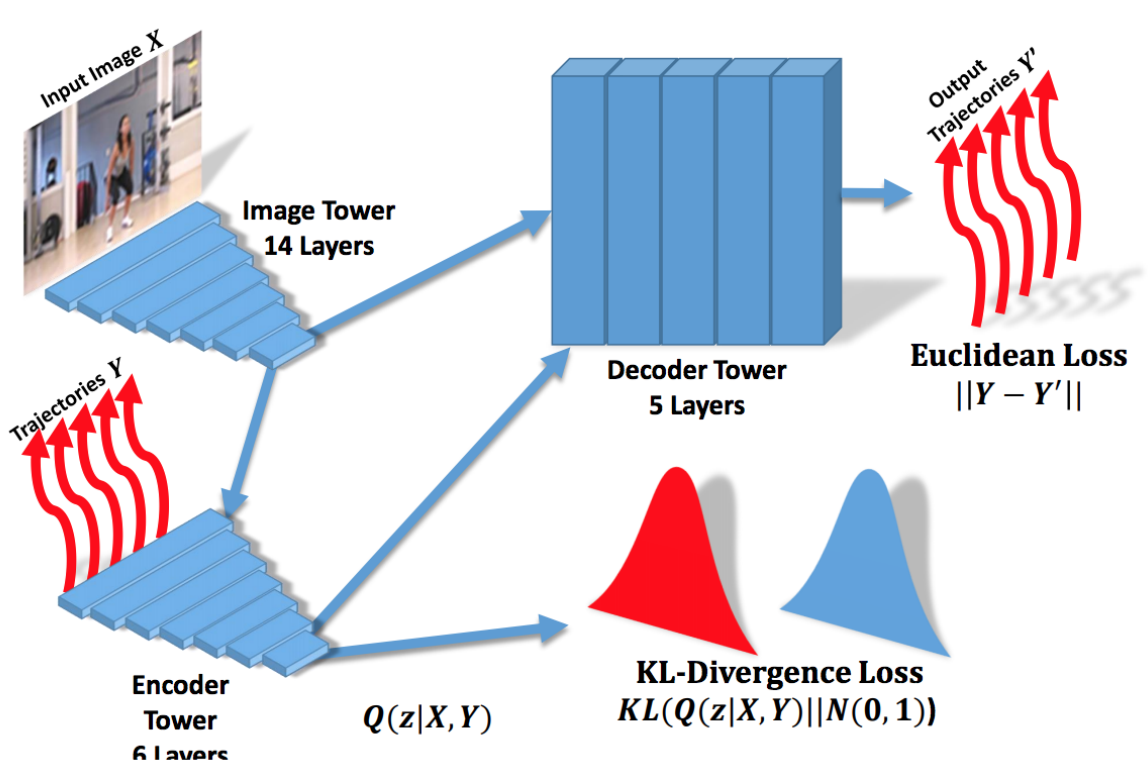
Facial Expression Editing

Yeh et al., Semantic Facial Expression Editing using Autoencoded Flow

Future Forecasting from Static Images



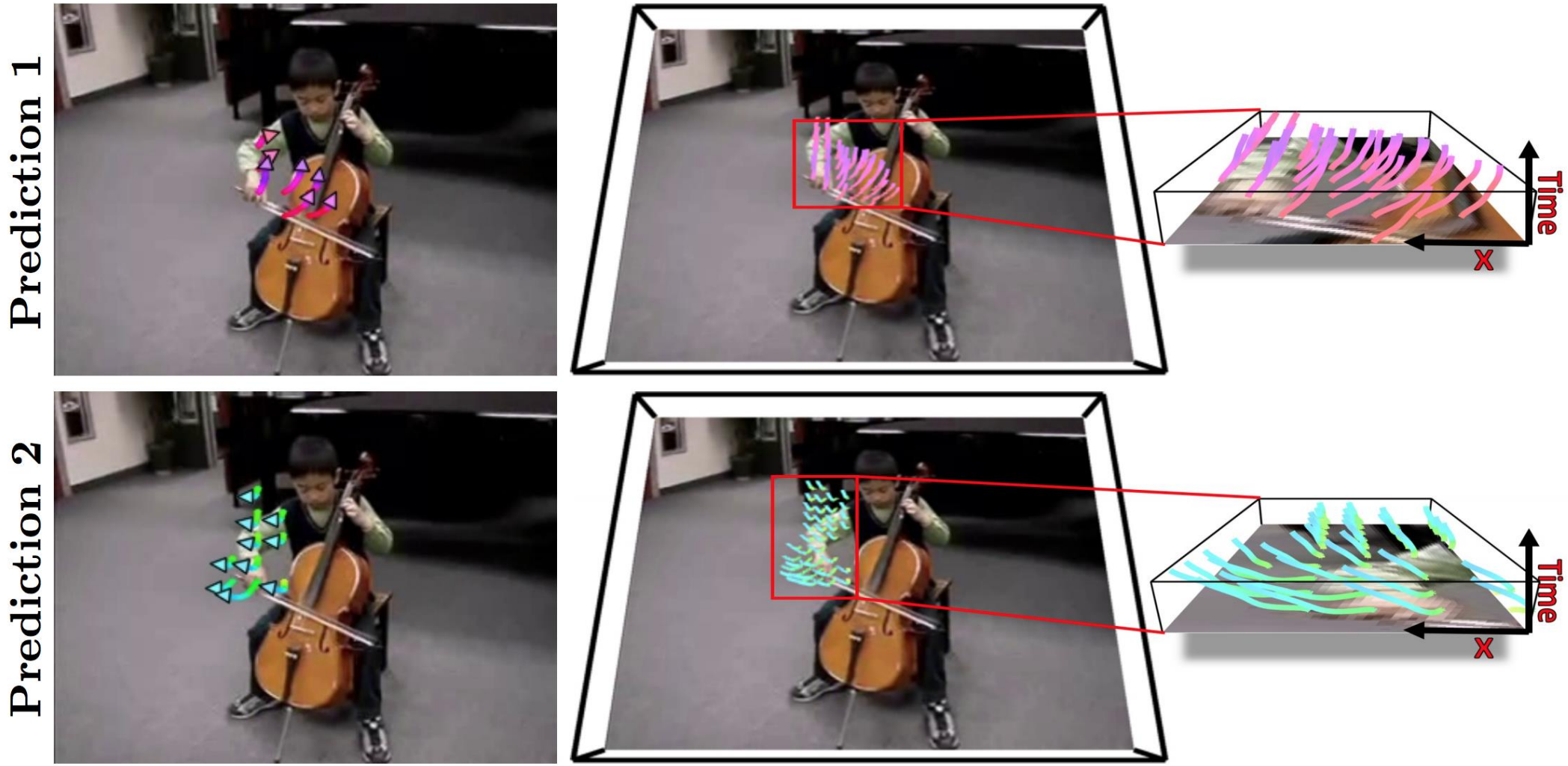
(a) Testing Architecture



(b) Training Architecture

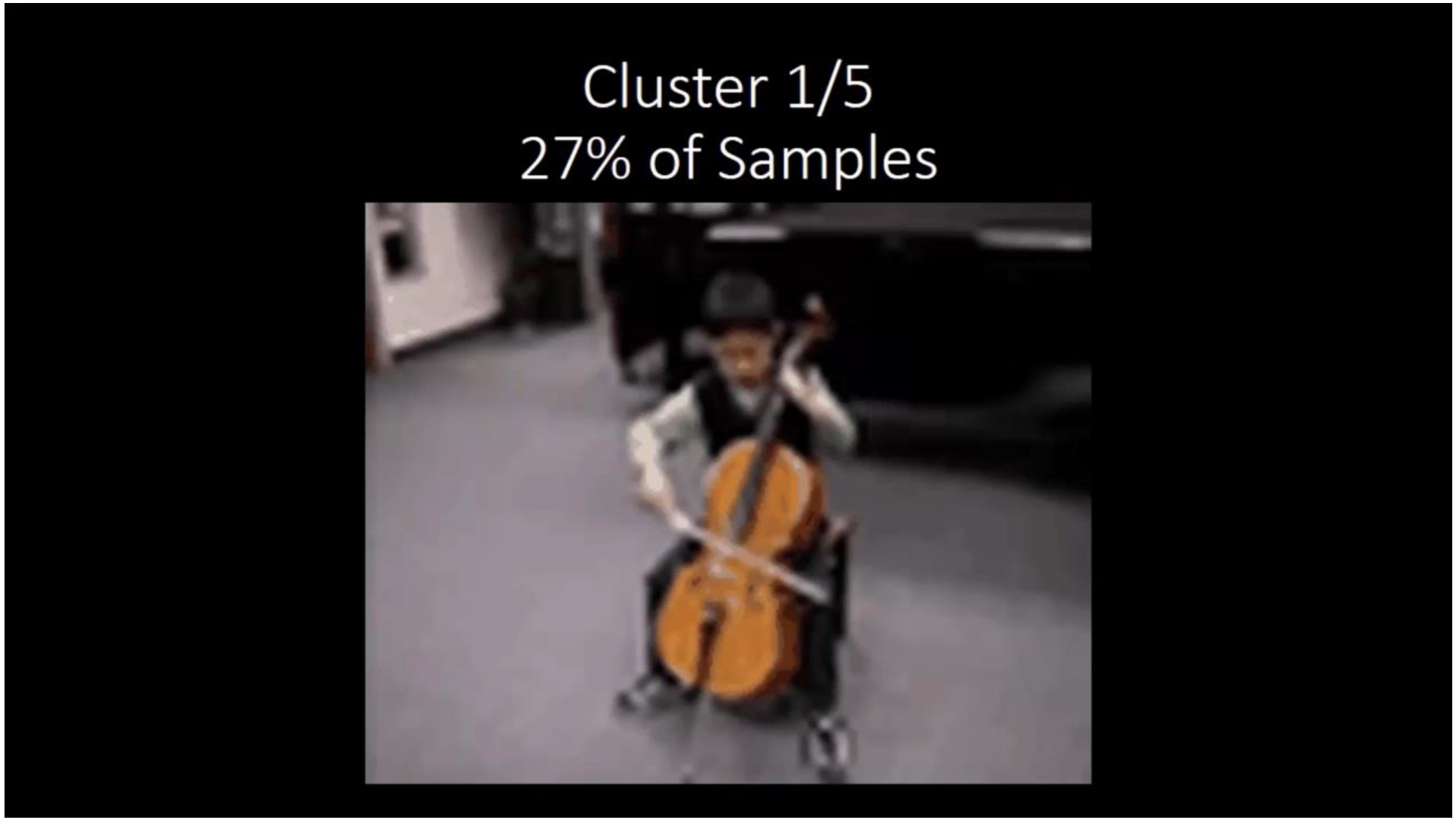
Walker et al., An Uncertain Future: Forecasting from Static Images using Variational Autoencoders

Future Forecasting from Static Images



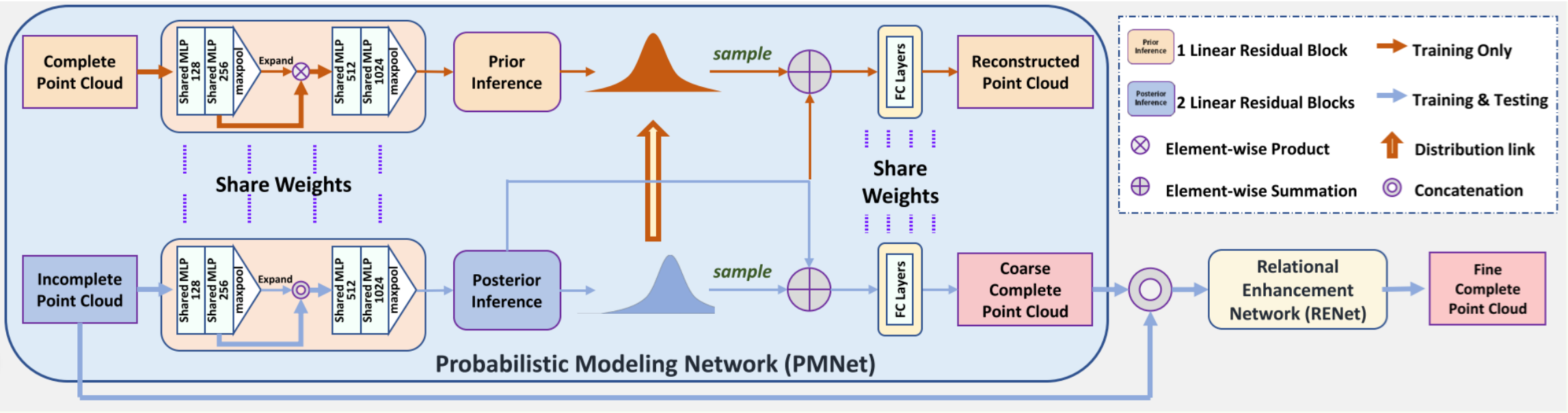
Walker et al., An Uncertain Future: Forecasting from Static Images using Variational Autoencoders

Future Forecasting from Static Images



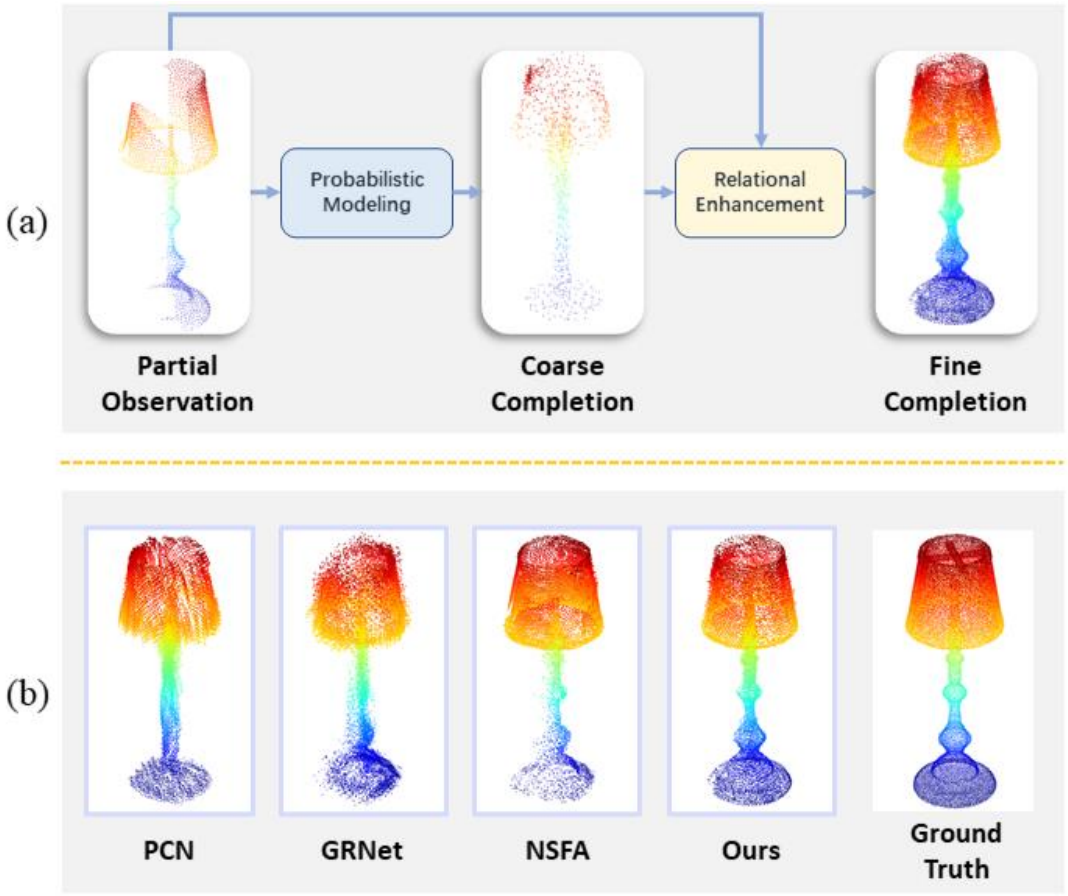
Walker et al., An Uncertain Future: Forecasting from Static Images using Variational Autoencoders

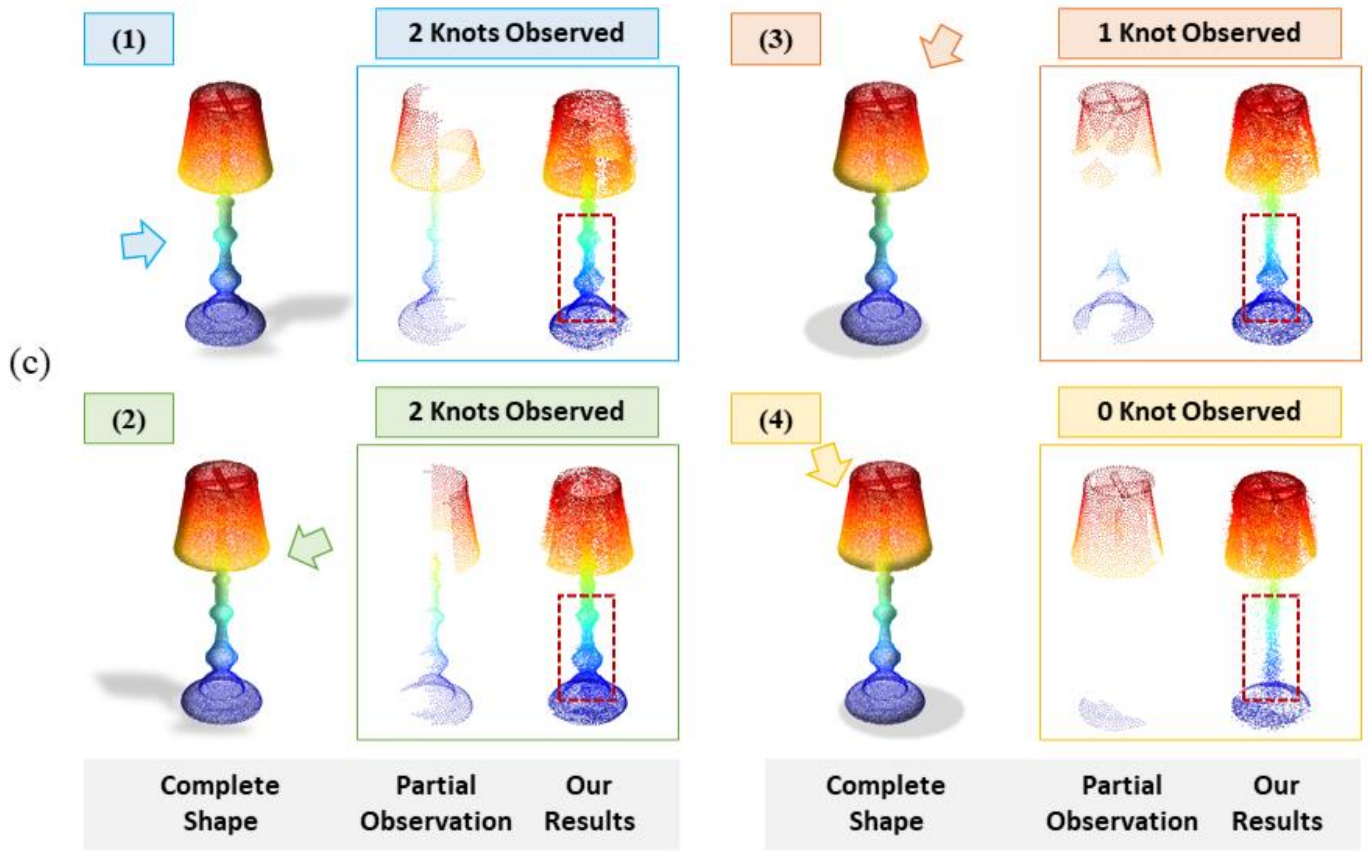
Point Cloud Completion



Pan et al., Variational Relational Point Completion Network, CVPR 2021.

Point Cloud Completion





Open Problems

Open Problem I:
The Rise of VQ-VAE

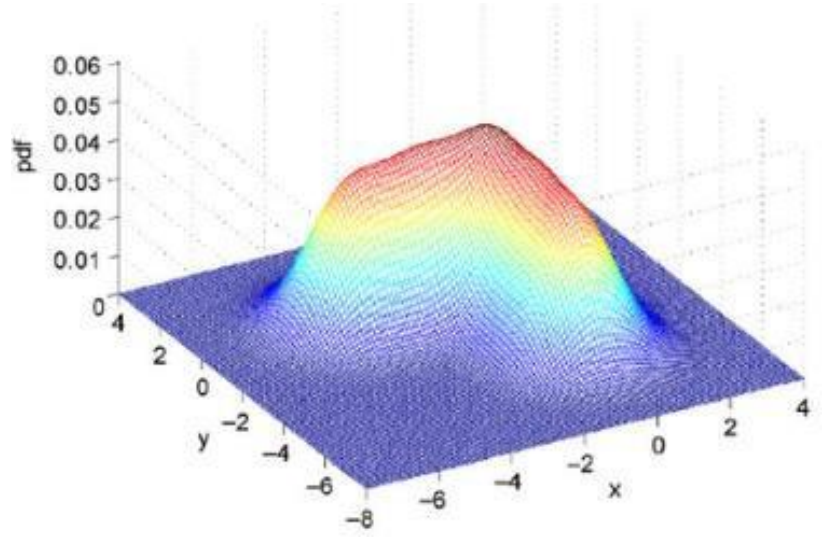
Neural Discrete Representation Learning

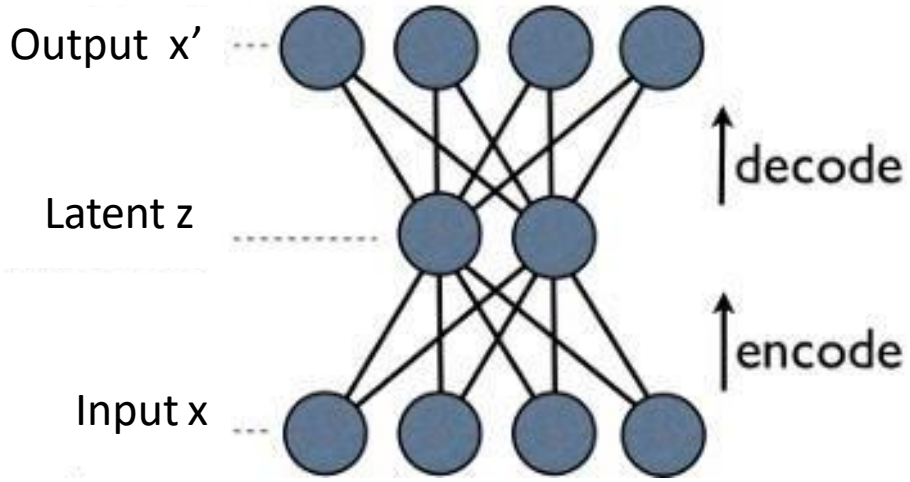
1. What is the task?
2. Comparison & Contribution
3. VQ-VAE Model
4. Results
    1. Density estimation & Reconstruction
    2. Sampling
    3. Speech
5. Discussion & Conclusion

A van den Oord et al., Neural Discrete Representation Learning, NeurIPS 2017.

What is the task?

- Task 1: Density estimation: learn p(x)
- Task 2: Extract meaningful latent variable (unsupervised)
- Task 3: Reconstruct input

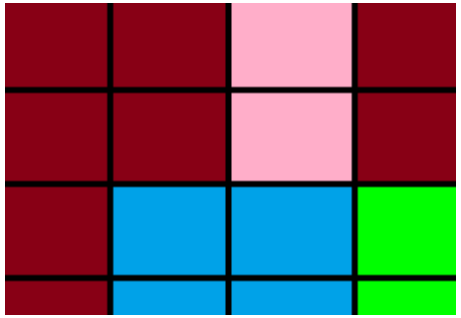




Comparison & Contribution

1. Bounds p(x), but does not require variational approximation
2. Train using maximum likelihood (stable training)
3. First to use **discrete latent** variables successfully
4. Uses whole latent space (avoid ‘posterior collapse’)

A little girl sitting on a bed with a teddybear

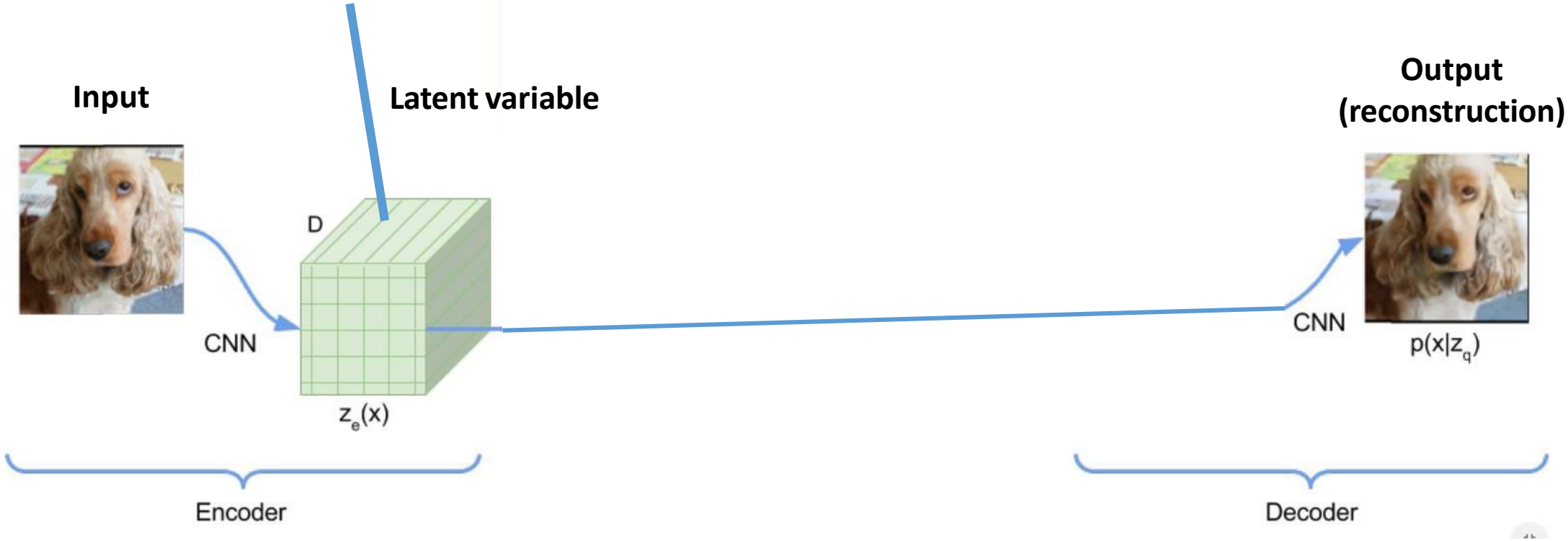


Why is discrete nice? More natural representation for humans, avoids posterior collapse (because you can more easily manage your latent space using your dictionary), compresseable, easier to learn a prior over a discrete latent space (more tractable than a continuous latent space).

Auto Encoder

How to discretize?

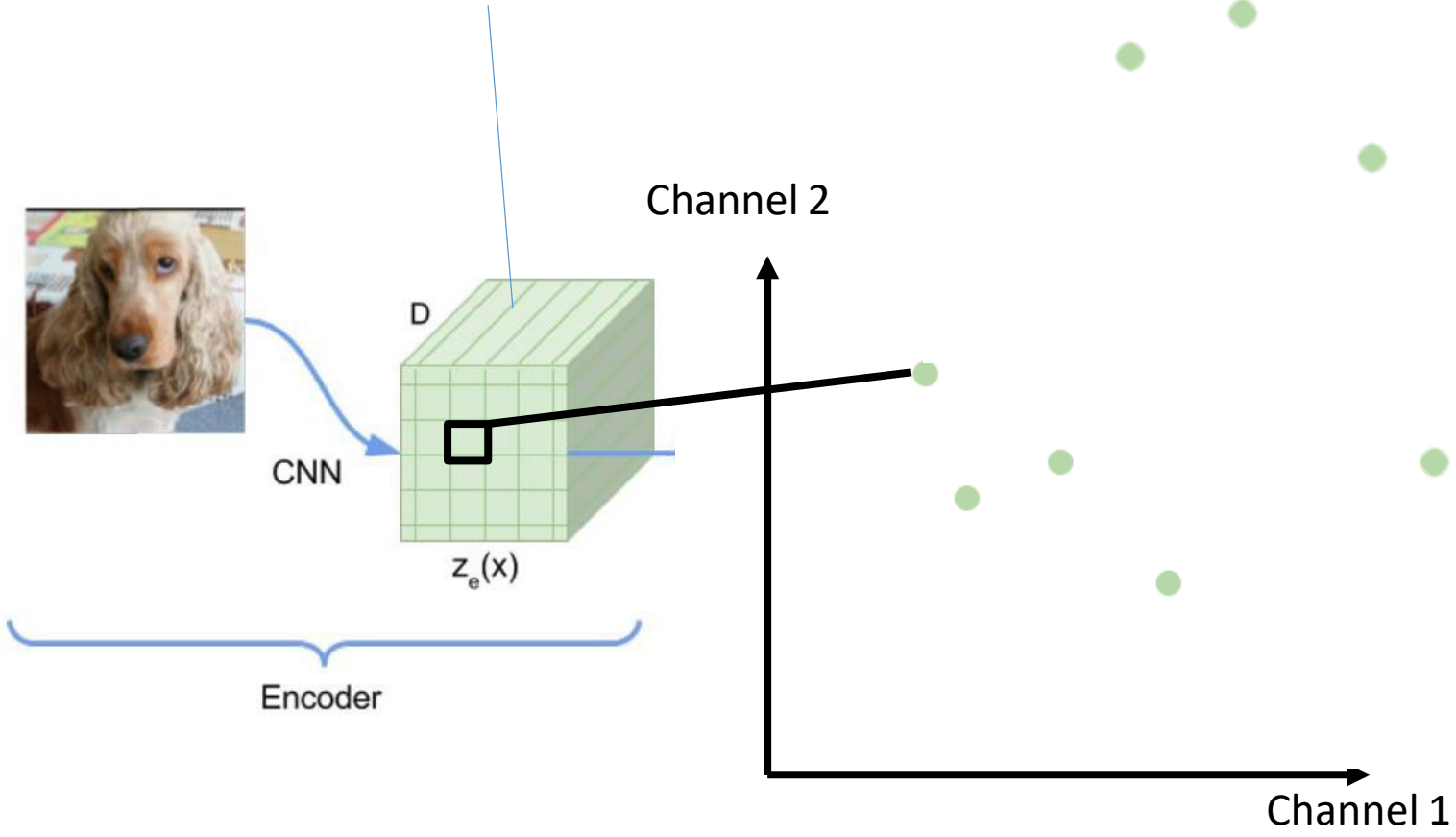
For the example:
We take this to be a 4 x 4 image
with 2 channels.



We can train this system end-to-end
using MSE (reconstruction loss)

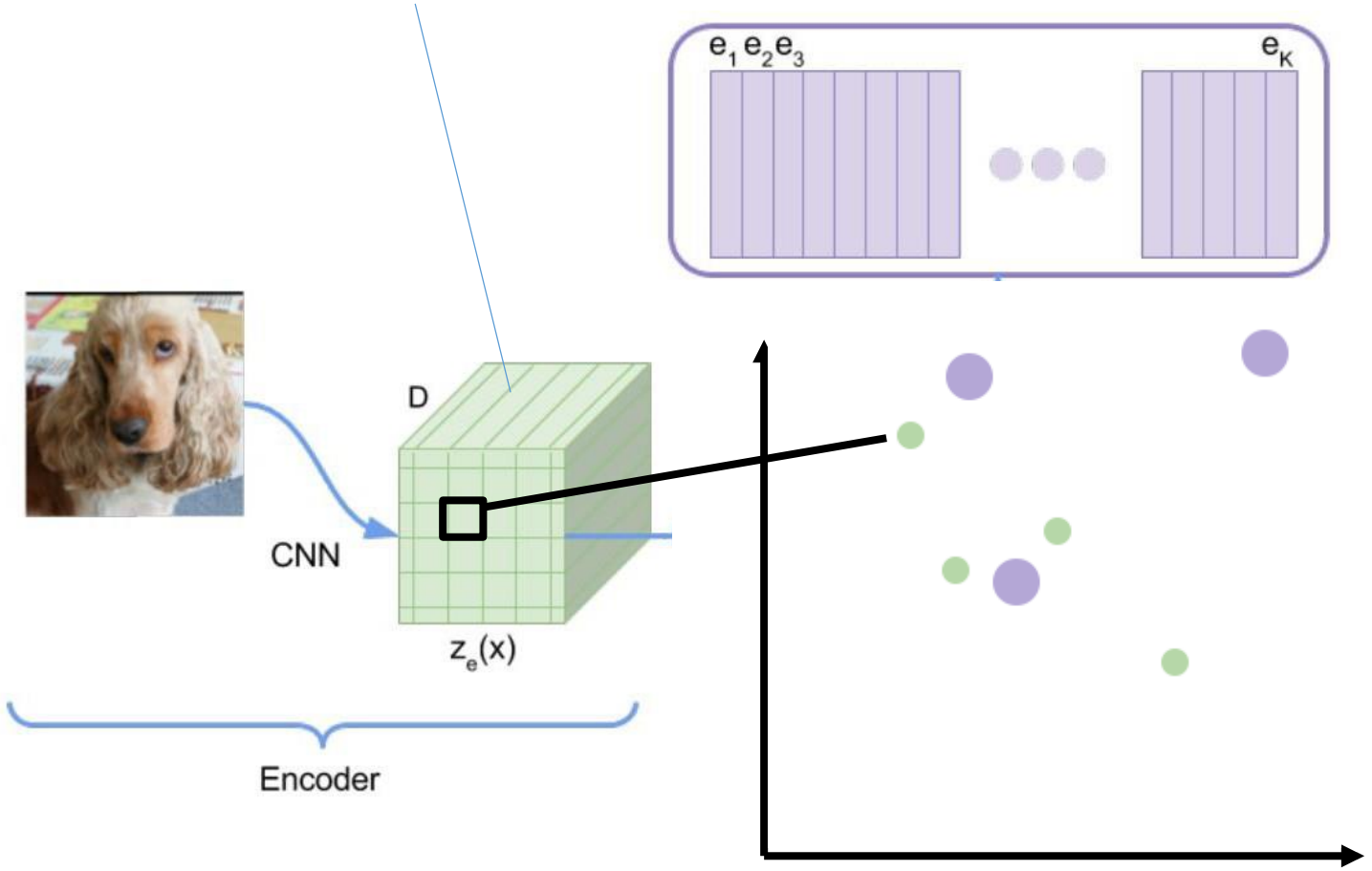
How to Discretize?

4 x 4 image with 2 channels.
We plot all pixel values (16) in 2D
(since we have 2 channels)



How to Discretize?

4 x 4 image with 2 channels.

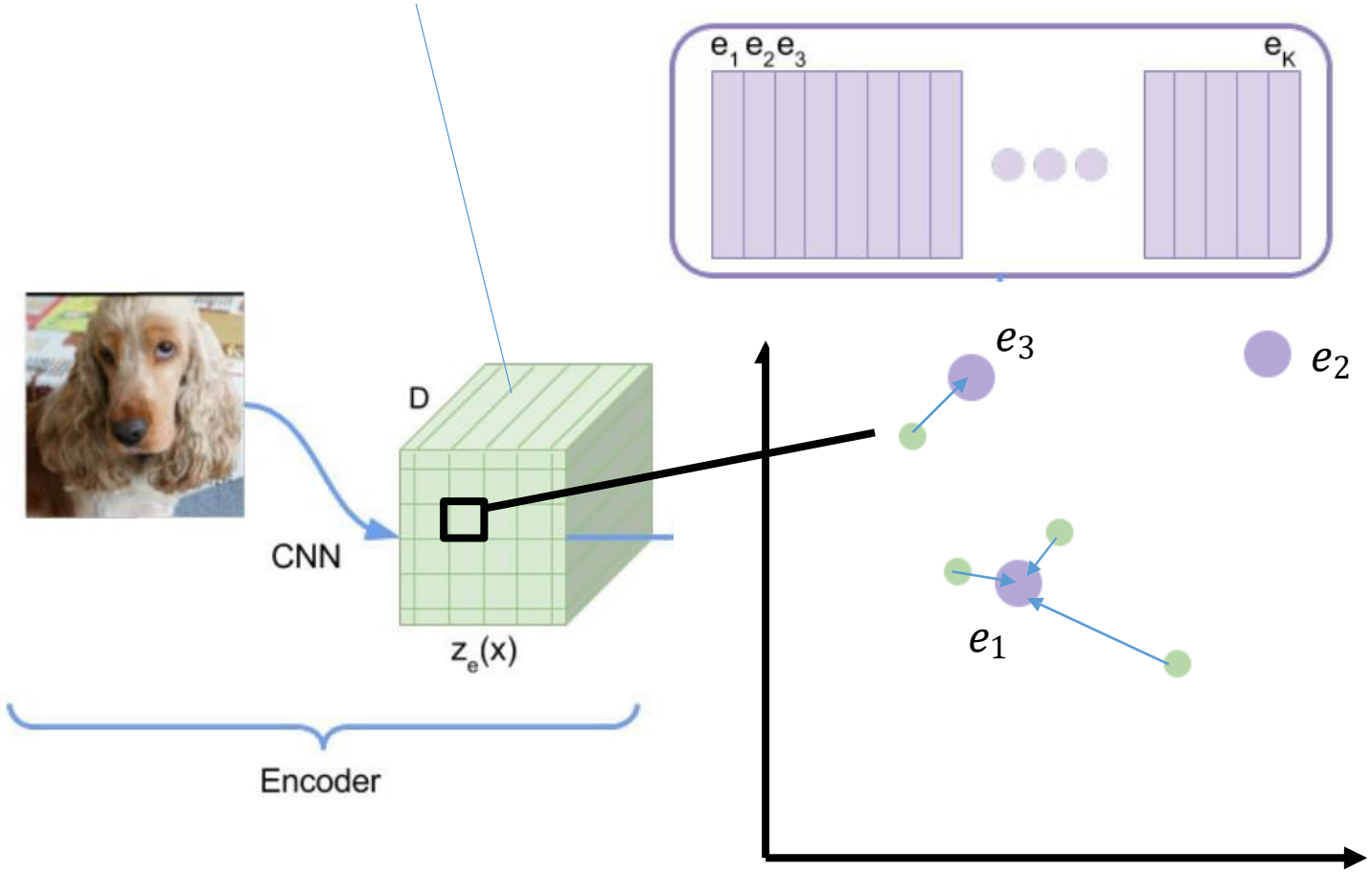


Make dictionary of vectors $e_1, \ldots, e_K$

Each e_i has 2 dimensions.

How to Discretize?

4 x 4 image with 2 channels.



Make dictionary of vectors $e_1, \ldots, e_K$

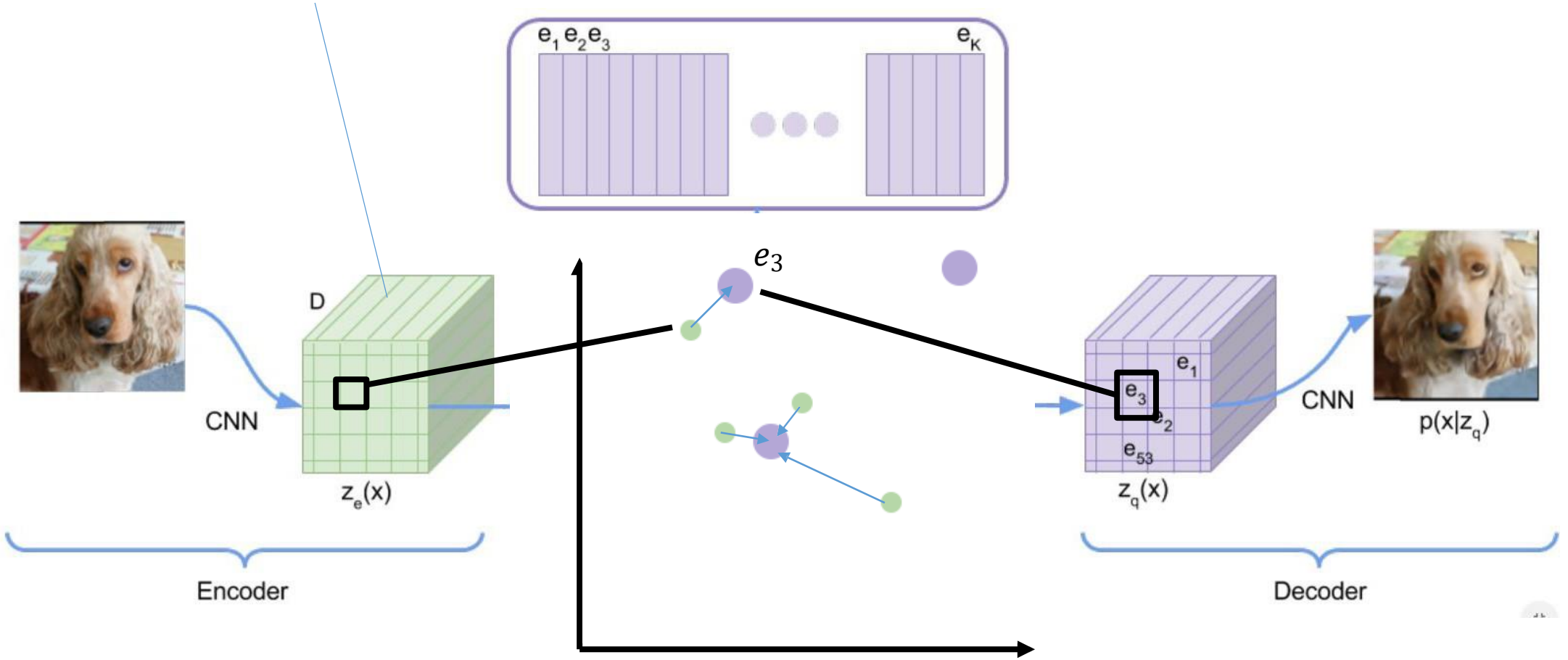
Each e_i has 2 dimensions.

For each latent pixel, look up nearest dictionary element e

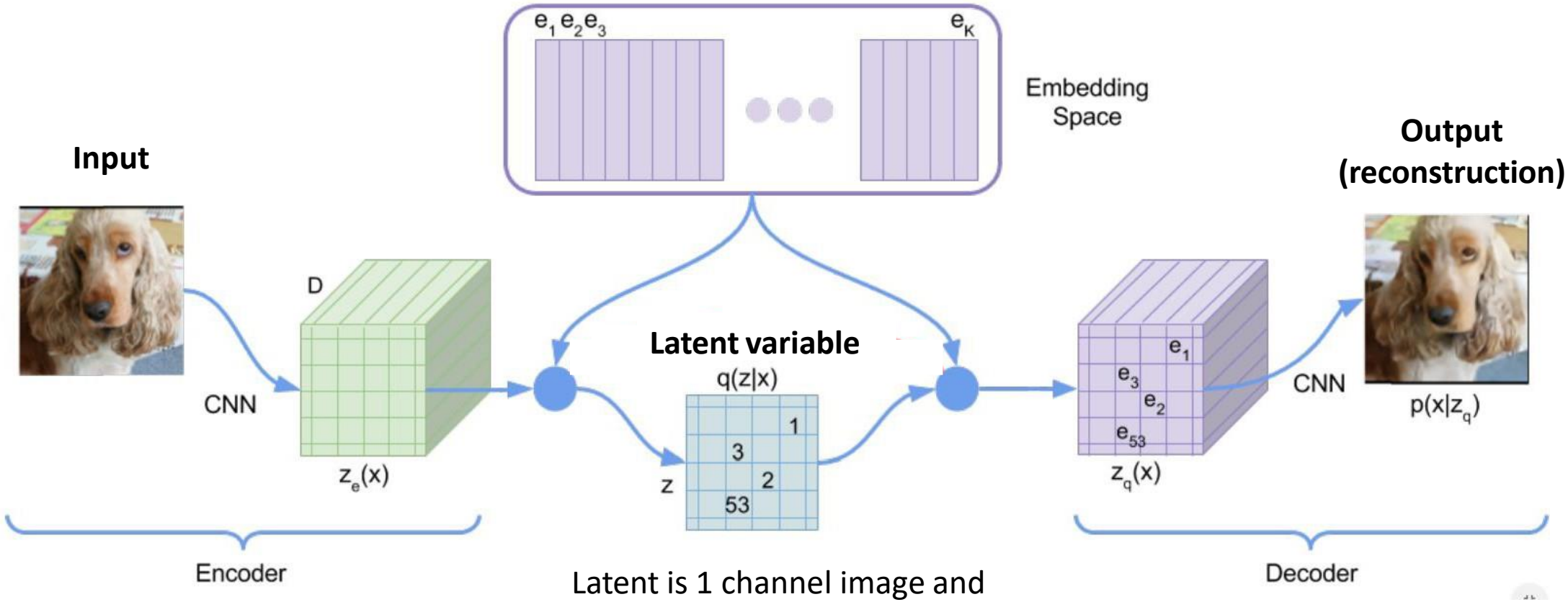
How to Discretize?

4 x 4 image with 2 channels.

Each e_i has 2 dimensions.



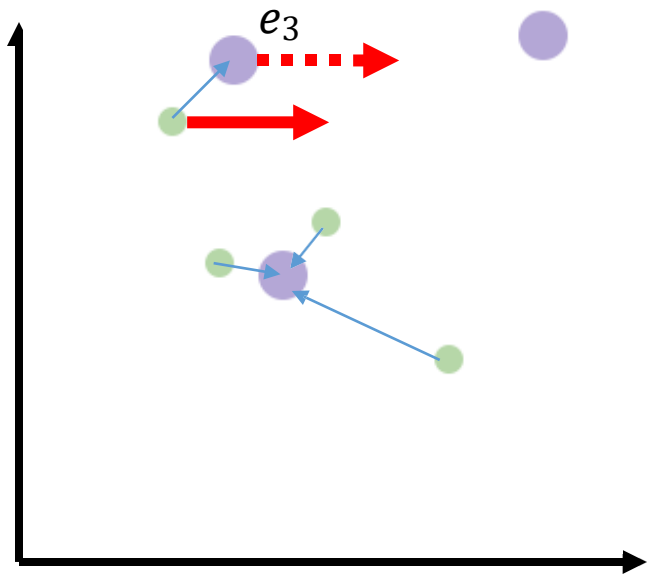
VQ-VAE Model

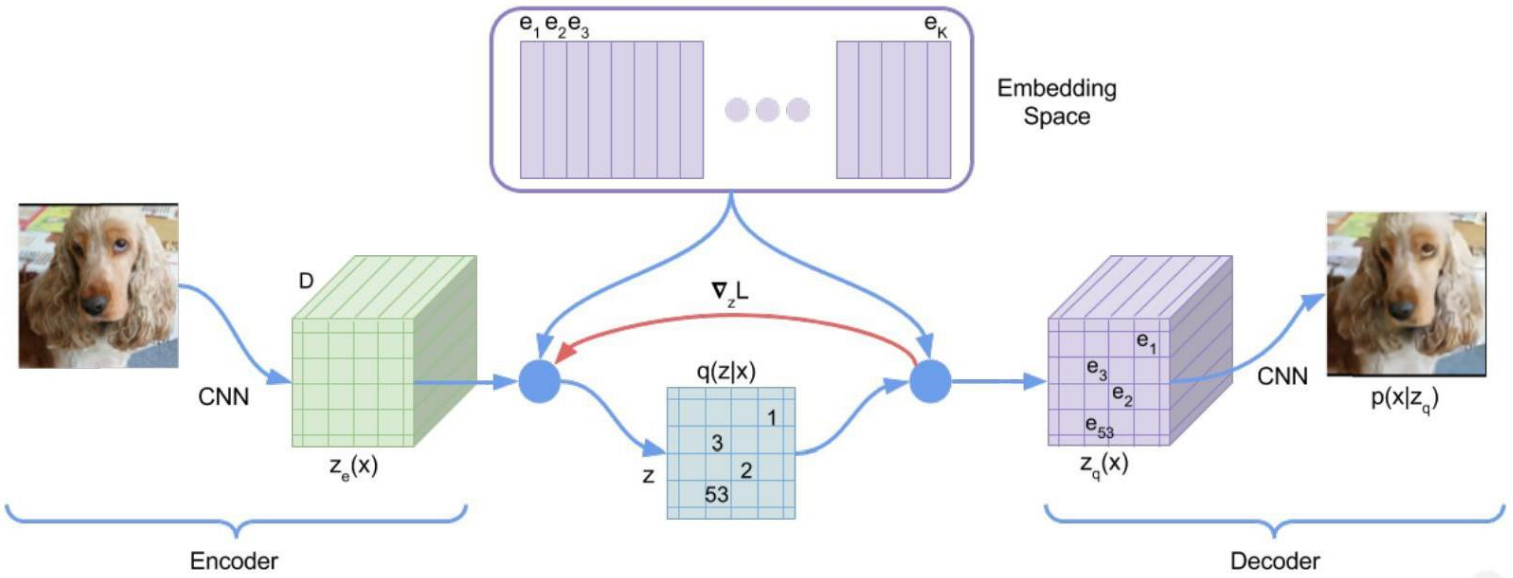


Latent is 1 channel image and contains the id of each e for each pixel (**discrete**).

How to train? (1/2)

- How to backpropegate through the discretization?
  - Lets say a gradient is incoming to a dictionary vector
  - We do not update the dictionary vector (fixed)
  - Instead we apply the gradient of e to the non-discretized vector





How to train? (2/2)

- Loss part 1: reconstruction error (dictionary fixed)

- Loss part 2: to update the dictionary

Density Estimation & Reconstructions

- Comparable with VAE on CIFAR-10 in terms of density estimation
- Reconstructions on ImageNet are very good

VQ-VAE

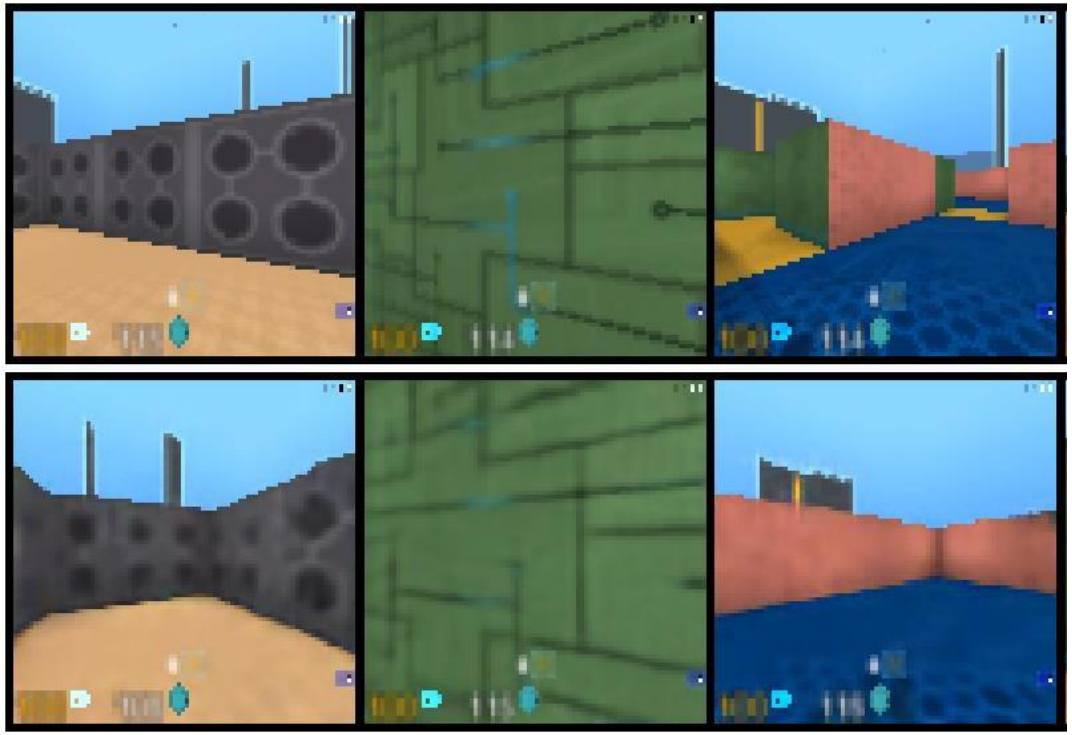
VAE

Density Estimation & Reconstructions

Stacking VQ-VAE

- VQ-VAE stacked to get higher level latents
  - Use DeepMind lab (artificial images)
  - Errors: sharpness and global mismatch
  - Latents seem ‘useful’: can generate coherent video from latent space (input first 6 images, output: video)
  - No quantative experiment

Original (21168 bits = 3 Kb)



Reconstruction (27 bits)

Stacking VQ-VAE

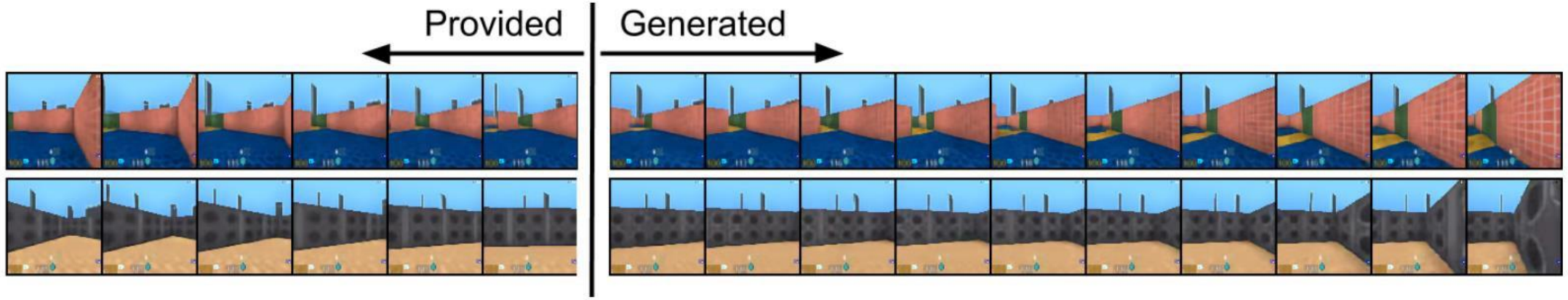


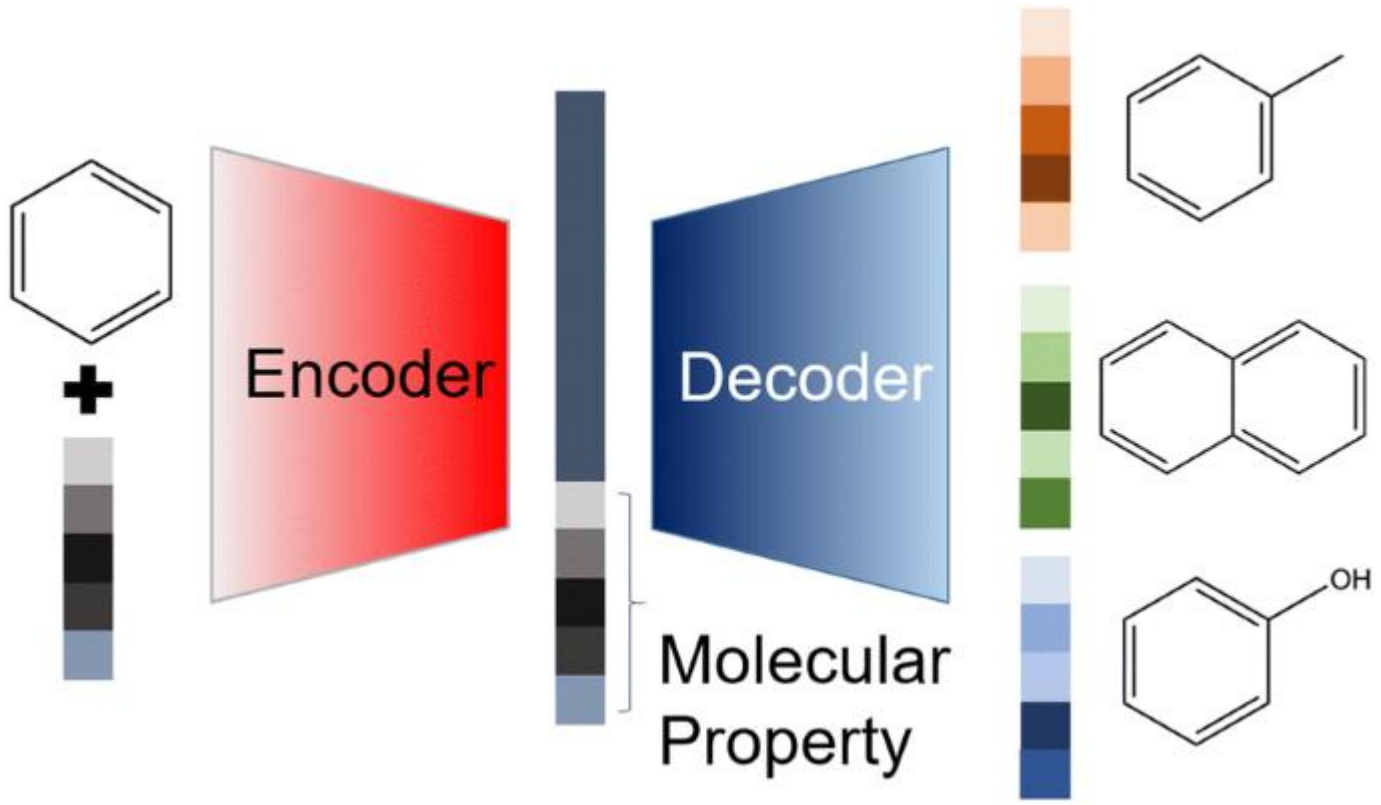
Figure 7: First 6 frames are provided to the model, following frames are generated conditioned on an action. Top: repeated action "move forward", bottom: repeated action "move right".

Discussion & Conclusion

	GAN	Variational Autoencoder	Pixel CNN	VQ-VAE
Compute exact likelihood p(x)	✗	✗	✓	✗
Has latent variable z	✓	✓	✗	✓
Compute latent variable z (inference)	✗	✓	✗	✓
Discrete latent variable	✗	✗	✗	✓
Stable training?	✗	✓	✓	✓
Sharp images?	✓	✗	✓	✓

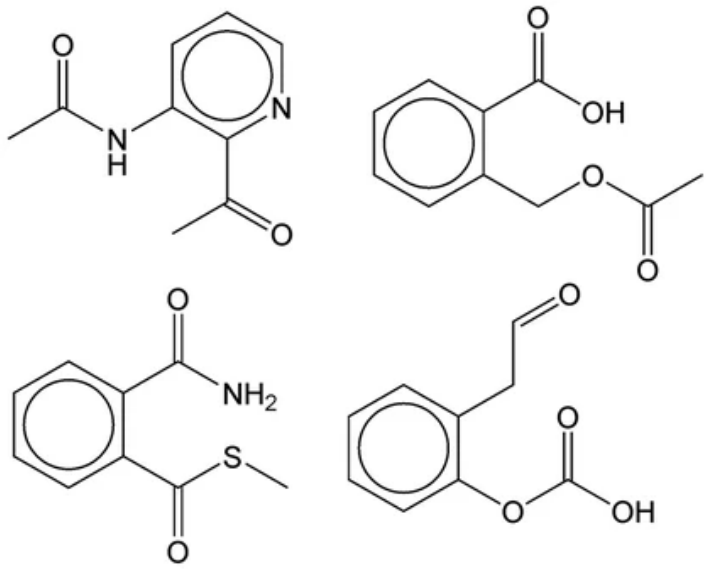
Open Problem II:
VAE in Natural Science

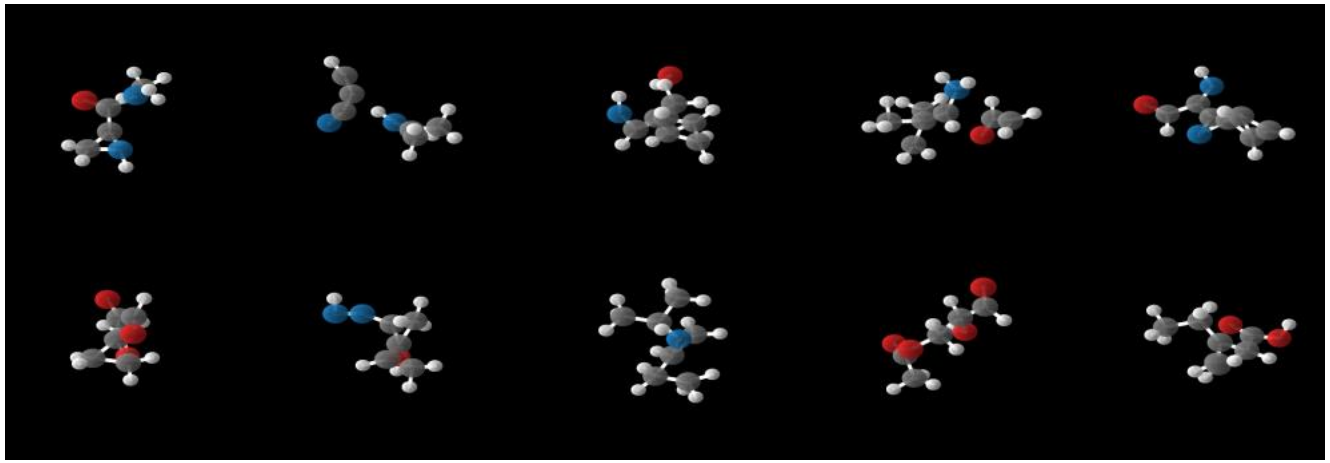
Molecular Design



Lim et al., Molecular generative model based on conditional variational autoencoder for de novo molecular design, Journal of Cheminformatics 2018.

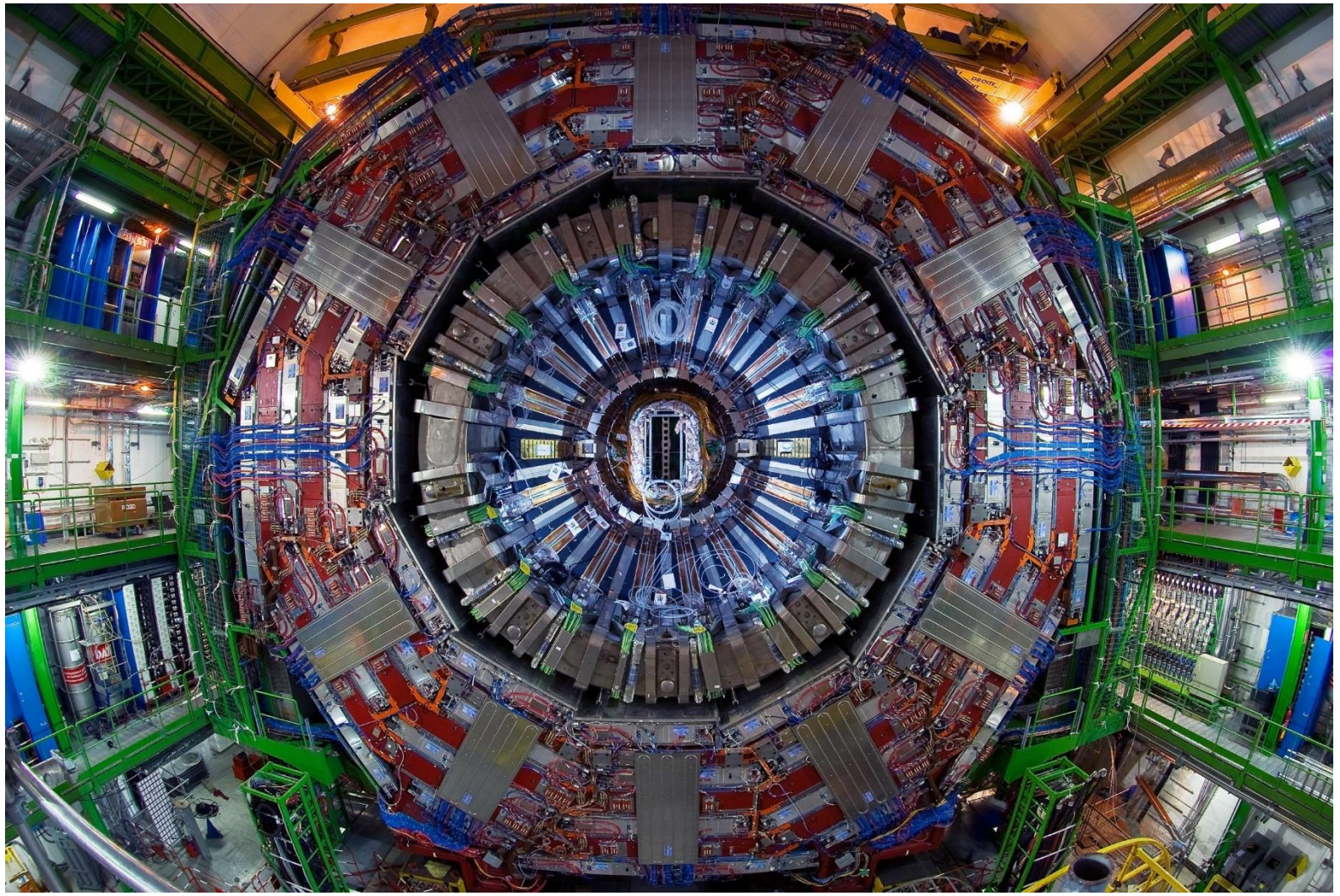
Molecular Design





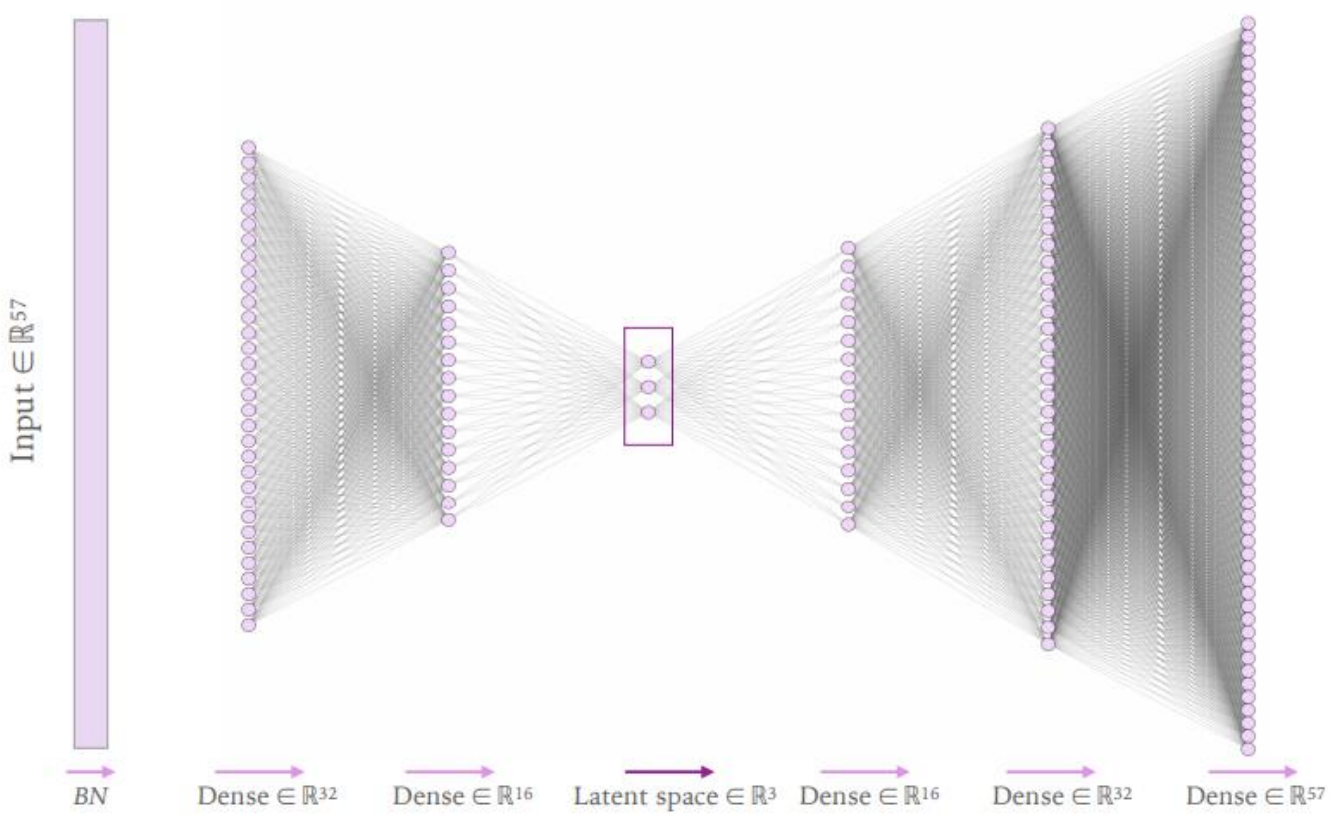
Molecules generated by the CVAE with the condition vector made of the five target properties of Aspirin and the latent vector slightly modified from that of Aspirin

New Particle Detection



Large Hadron Collider

New Particle Detection



Govorkova et al., Autoencoders on FPGAs for real-time, unsupervised new physics detection at 40 MHz at the Large Hadron Collider, ArXiv 2021.

Summary

- Autoencoders
  - Can infer useful latent representation for a dataset
  - Bad generators
- VAEs
  - Can infer a useful latent representation for a dataset
  - Better generators due to latent space regularization
  - Lower quality reconstructions and generated samples (usually blurry)

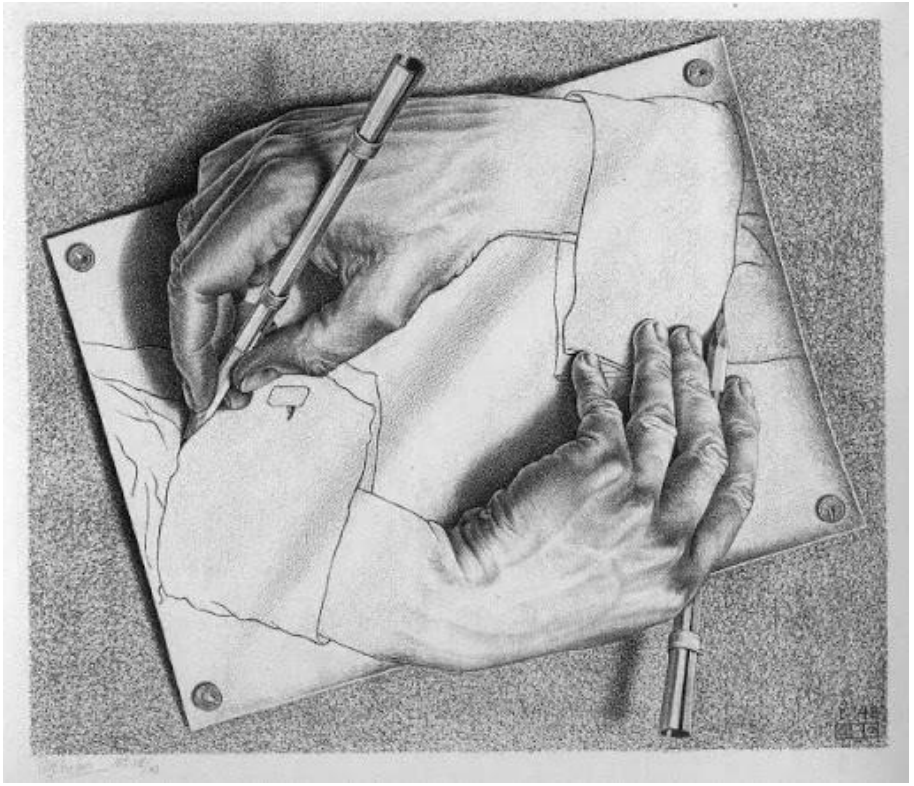
Summary

- Autoencoders
  - Can infer useful latent representation for a dataset
  - Bad generators
- VAEs
  - Can infer a useful latent representation for a dataset
  - Better generators due to latent space regularization
  - Lower quality reconstructions and generated samples (usually blurry)
- GANs
  - Can not find a latent representation for a given sample (no encoder)
  - Usually better generators than VAEs
  - Currently unstable training (active research)

Resources

- VAE on Digits: http://dpkingma.com/sgvb_mnist_demo/demo.html

- VAE on Faces: http://vdumoulin.github.io/morphing_faces/online_demo.html



Next Time:

Generative Adversarial Networks (GANs)