

# Deep Learning for Data Science

## DS 542

<https://dl4ds.github.io/fa2025/>

Unsupervised Learning and Variational Autoencoders

# Plan for Today

- Unsupervised Learning
- Latent Variables
- Probabilistic Generative Models
- Variational Autoencoders (VAEs)
- VAE math
- Examples of VAEs

# Kinds of Learning

## Supervised learning

Any time that

- We are provided input/output pairs
- And asked to build a model generalizing them

## Unsupervised learning

Everything else? Not quite.

- Self/semi-supervised learning used inconsistently.
- Sometimes partially supervised.
- Sometimes deriving targets for unsupervised data.

Also ignoring reinforcement learning.

# Unsupervised Learning

- Learning problems where an input/output relation was not provided.
  - Often not a specific function to learn.
  - No input/output training data!
- General task is “learn the distribution”.
  - Calculate mean and standard deviation technically qualifies.
  - But usually we want something that can match the distribution a lot better.

# Unsupervised Learning → Supervised Learning?

Previously saw next token prediction with LLMs

- Was this supervised or unsupervised?

# Unsupervised Learning → Supervised Learning?

Previously saw next token prediction with LLMs

- Was this supervised or unsupervised?
  - Unsupervised data set - lots of text.
  - Extracted lots of supervised problems - pieces of text and next tokens.
  - Fine tuning GPT 4 → ChatGPT has more explicit supervision.

Generation by discriminating what to generate next 🤔

# Supervised vs. Self/Unsupervised Learning

## Supervised Learning

**Data:**  $(x, y)$

$x$  is data,  $y$  is a label

**Goal:** Learn function to map  
 $x \rightarrow y$

**Applications:** Classification, regression, object detection, semantic segmentation, etc.

## Self/Unsupervised Learning

**Data:**  $x$

$x$  is data, no labels! Or labels part of the data

**Goal:** Learn the hidden or underlying structure of the data.

**Applications:** Clustering, dimensionality reduction, compression, find outliers, generating new examples, denoising, interpolating between data points, etc.

Related split: did humans decide the labels or targets?

# Any Questions?



**Moving on**

- Unsupervised Learning
- Latent Variables
- Probabilistic Generative Models
- Variational Autoencoders (VAEs)
- VAE math
- Examples of VAEs



# Latent Variables

- What is a latent variable?
  - Invisible but underlying truth behind what's going on?
- Latent variable  $\rightarrow$  observations?
  - Often lower dimension than our observations.
  - Observation  $\sim f(\text{latent})$
  - But not always.
- Observation  $\rightarrow$  latent variable?
  - K-means mapping data to cluster id
  - Often will want to infer latents from observations (like inverting GAN)

Will be saying  
“observation” a lot today to  
distinguish “visible” data  
from inferred latents.

# Generative Models

If you have

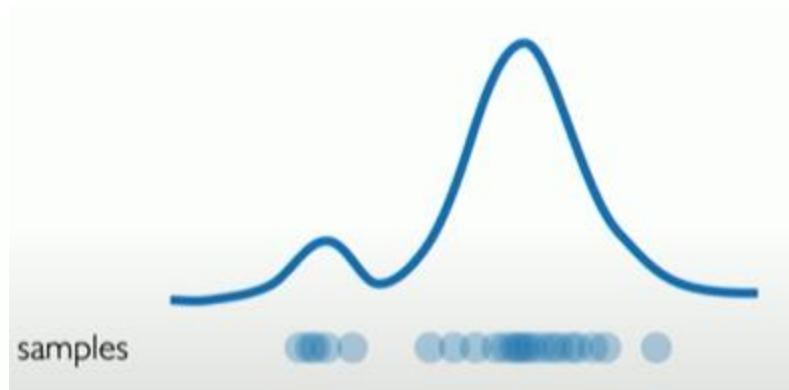
1. A probability distribution of latent variables, and
2. A function mapping latent variables to observations

You basically have a generative model.

# Generative Modeling

Goal: Take as input training samples from some distribution and learn a model that represents that distribution

## Probability Density Estimation

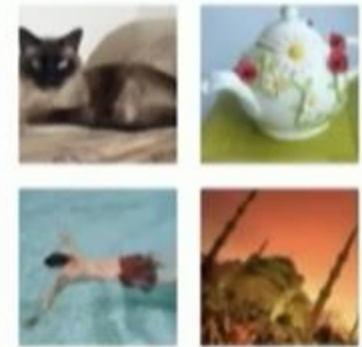


## Sample Generations



Input samples

Training data  $\sim P_{data}(x)$



Generated samples

Generated  $\sim P_{model}(x)$

How can we learn  $P_{model}(x)$  similar to  $P_{data}(x)$ ?

# Why generative models? Debiasing

Capable of uncovering **underlying features** in a dataset



Homogeneous skin color, pose

VS



Diverse skin color, pose, illumination

How can we use this information to create fair and representative datasets?

# Why generative models? Outlier detection

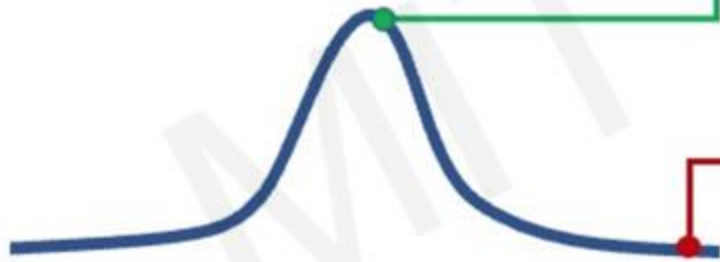
- **Problem:** How can we detect when we encounter something new or rare?
- **Strategy:** Leverage generative models, detect outliers in the distribution
- Use outliers during training to improve even more!

95% of Driving Data:

(1) sunny, (2) highway, (3) straight road



Detect outliers to avoid unpredictable behavior when training



Edge Cases



Harsh Weather



Pedestrians

# More outlier examples



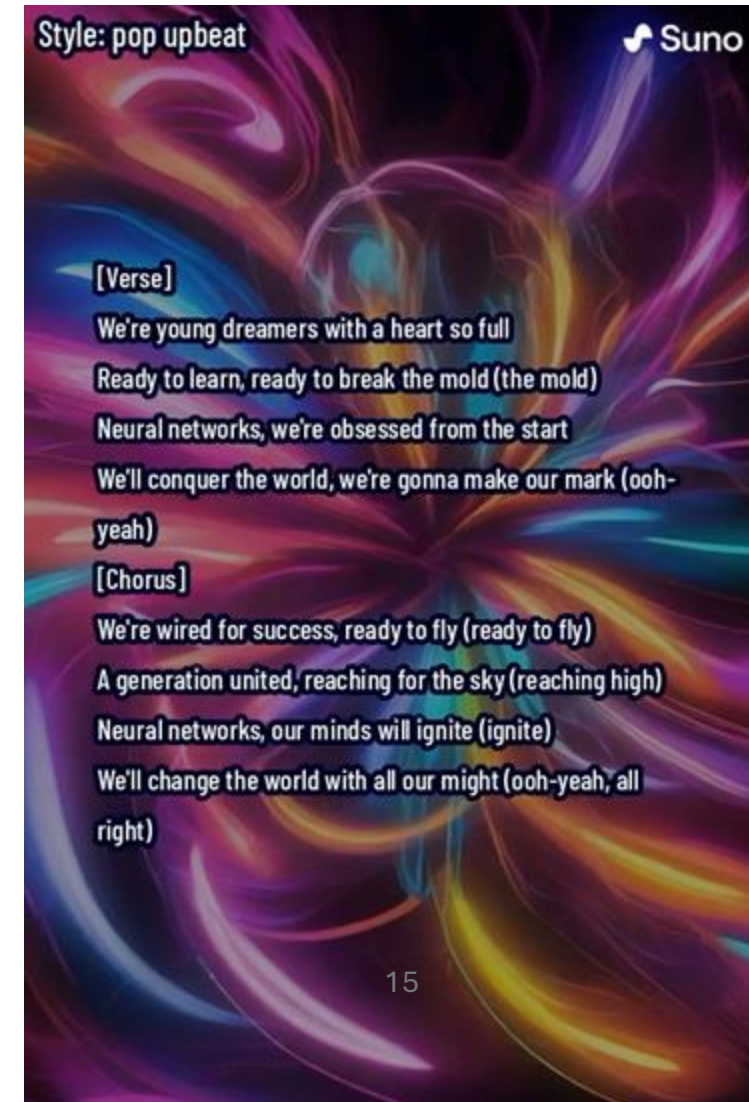


# Why generative models?

## image, video and audio creation



A teenage superhero fighting crime in an urban setting shown in the style of claymation.

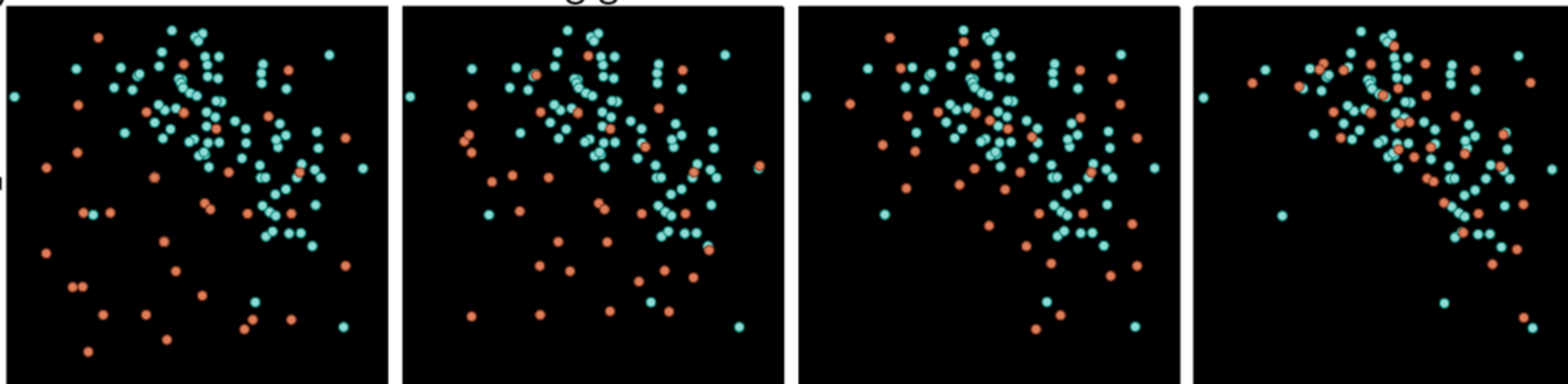


Write a short pop song about students wanting to learn about neural networks and do great things with them.

a)

Fitting generative adversarial networks

$x_2$



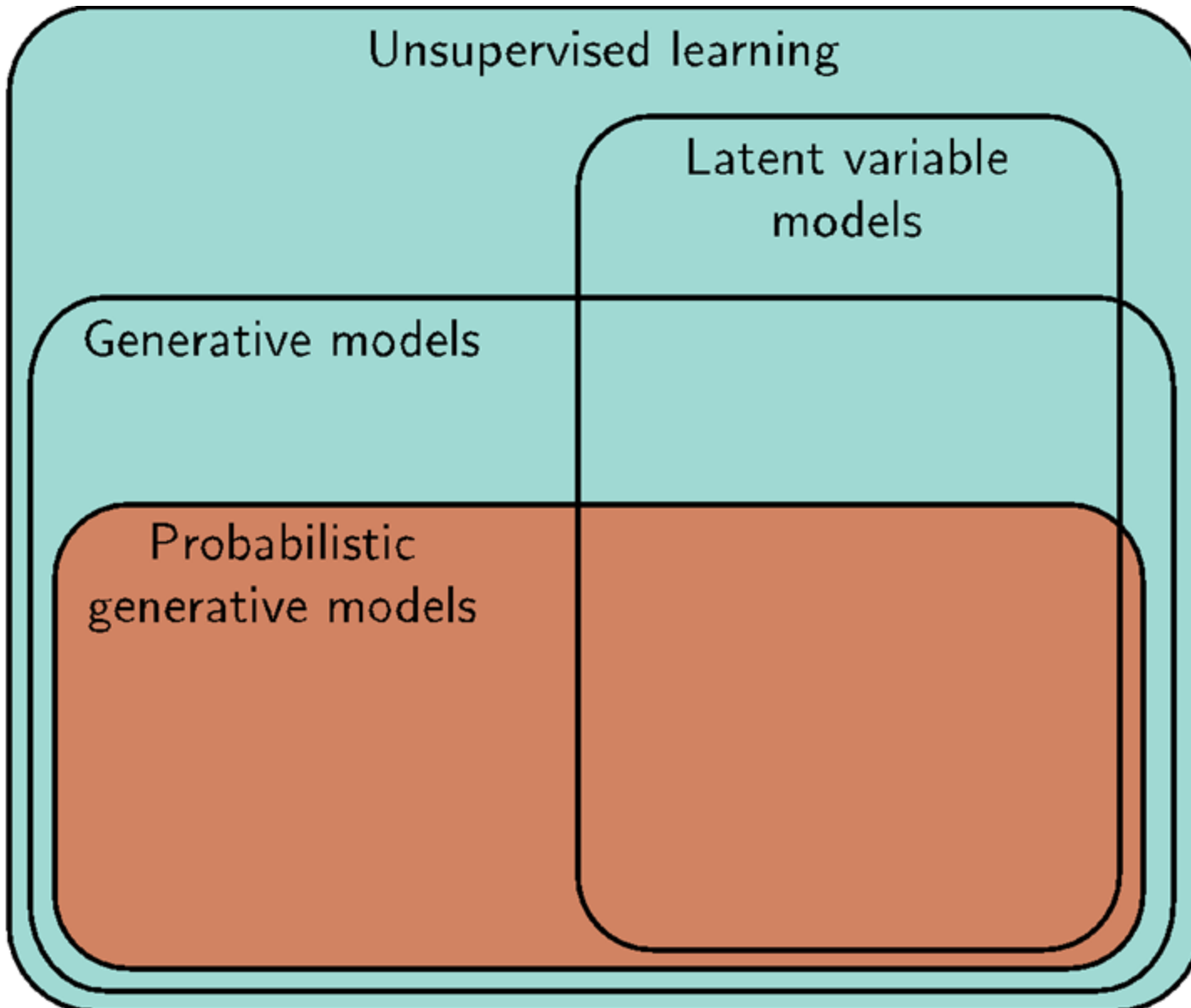


# Any Questions?



## Moving on

- Unsupervised Learning
- Latent Variables
- Probabilistic Generative Models
- Variational Autoencoders (VAEs)
- VAE math
- Examples of VAEs



Generative = can generate new examples

Probabilistic = can assign probability to data examples

# Probabilistic Generative Models

## Key distinction

- Can assign probability to observations (conditioned on model parameters)

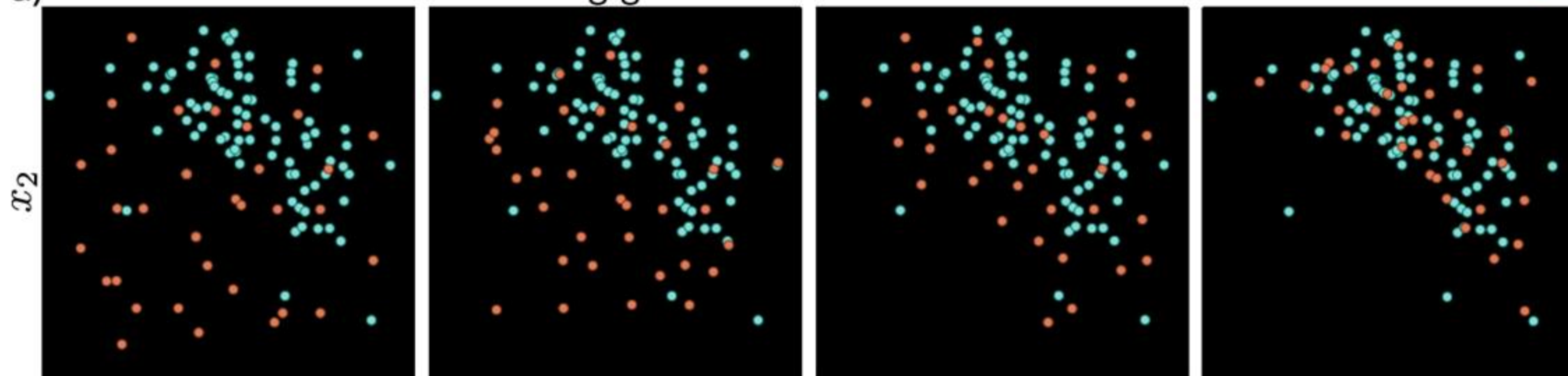
Can't you get this from the latent probabilities and latent to observation mapping?

Not always easy to invert...

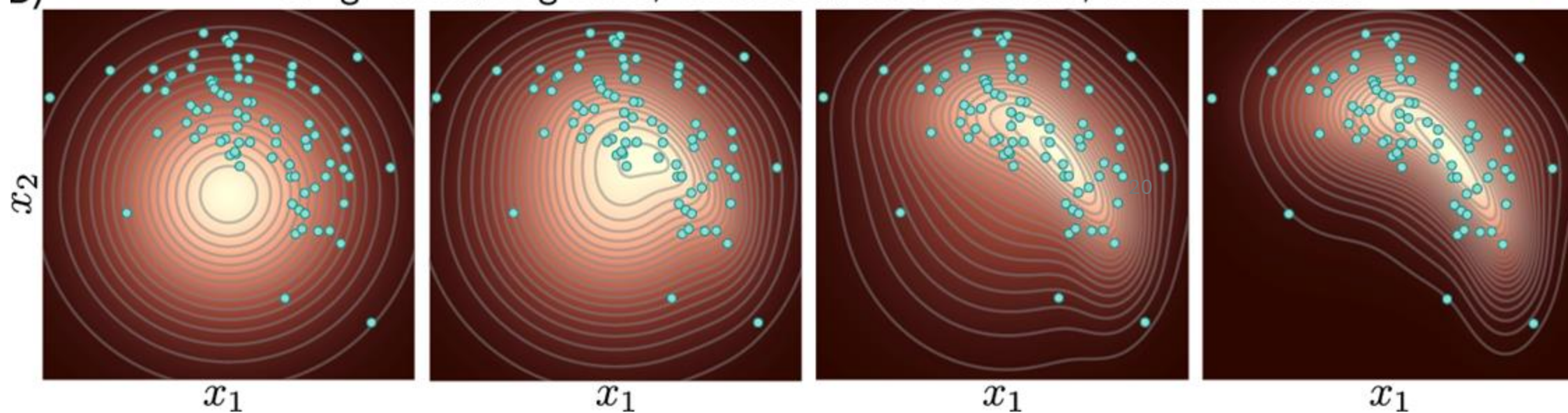
## Standard optimization:

- Maximize probability of observations
- Requires direct calculation of observation probability from model parameters?
- Implicitly suppresses dissimilar possibilities...

a) — Fitting generative adversarial networks —→



b) — Fitting normalizing flows, variational auto-encoders, diffusion models —→



# Probabilistic Generative Models

Since we can calculate probabilities for observations,

- We can compare different models
  - Which model makes the test data more likely?
- We can quantify how unlikely an observation is...
  - So is it an outlier?

# Examples of Probabilistic Generative Models

- ~~Generative adversarial networks~~ (missing probabilities)
- Variational autoencoders (today)
- Diffusion models (next week)

# Probabilistic models

- Maximize log likelihood of training data

$$\hat{\phi} = \operatorname{argmax}_{\phi} \left[ \sum_{i=1}^I \log[\operatorname{Pr}(x_i | \phi)] \right]$$

- Find the parameters,  $\phi$ , of some parametric probability distribution so that the training data is most likely under that distribution

# What makes a good model?

- Efficient sampling:
  - Generating samples from the model should be computationally inexpensive and take advantage of the parallelism of modern hardware.



# What makes a good model?

- High-quality sampling:
  - The samples should be indistinguishable from the real data that the model was trained with.
  - This is broadly getting better as we train bigger models.

# What makes a good model?

- Coverage:
  - Samples should represent the entire training distribution. It is insufficient to only generate samples that all look like a subset of the training data.
  - GANs have trouble with this since their generator training does not directly see the training data...

# What makes a good model?

- Well-behaved latent space:
  - Every latent variable  $z$  should correspond to a plausible data example  $x$  and smooth changes in  $z$  should correspond to smooth changes in  $x$ .
  - Usually this is the case. Just ignore the 6 fingered hands?

# What makes a good model?

- Interpretable latent space:
  - Manipulating each dimension of  $z$  should correspond to changing an interpretable property of the data. For example, in a model of language, it might change the topic, tense or degree of verbosity.
  - This is stronger than having a well-behaved latent space, since changes in a particular direction need to be semantically similar.

# What makes a good model?

- Efficient likelihood computation:
  - If the model is probabilistic, we would like to be able to calculate the probability of new examples efficiently and accurately.
- WTB: a probability calculator that identifies fake news as low probability.

# Do we have good models?

	GANs	VAEs	Flows	Diffusion
Efficient sampling	✓	✓	✓	✗
High quality	✓	✗	✗	✓
Coverage	✗	?	?	?
Well-behaved latent space	✓	✓	✓	✗
Interpretable latent space	?	?	?	✗
Efficient likelihood	n/a	✗	✓	✗

How to measure performance within or between categories?

- Open research area.

# Quantifying Performance - Test Likelihood

How likely is the the test data given our model? (Throwback to loss functions)

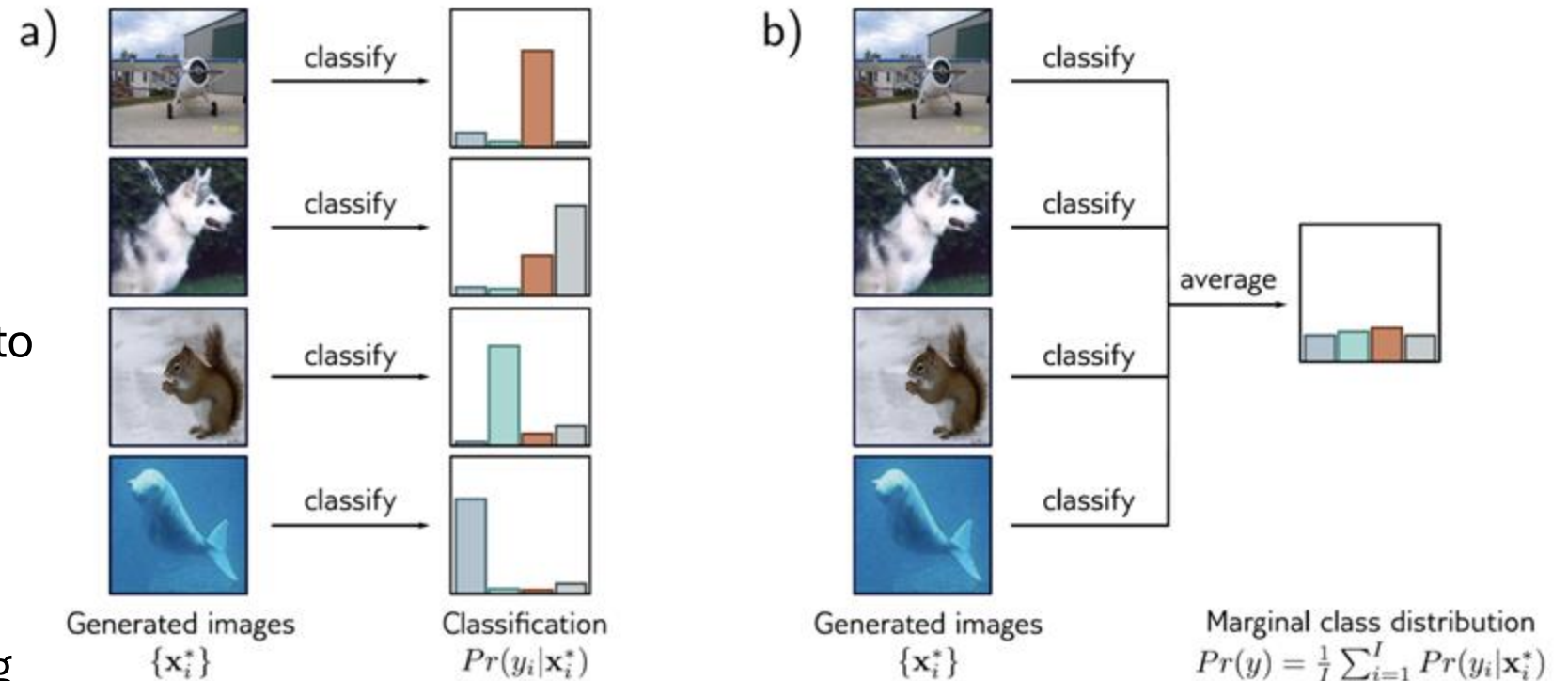
$$\sum_{i=1}^I \log[\text{Pr}(x_i | \phi)]$$

See also perplexity if working with text.

# Quantifying Performance - Inception Score

Grading via another model

- Usually the Inception model for ImageNet
- Want generated images to have a single very likely classification.
- But average flat classification across generated images.
- Formal formula checking KL-divergence between those on a per-generated image basis...



**Figure 14.4** Inception score. a) A pretrained network classifies the generated images. If the images are realistic, the resulting class probabilities  $Pr(y_i | \mathbf{x}_i^*)$  should be peaked at the correct class. b) If the model generates all classes equally frequently, the marginal (average) class probabilities should be flat. The inception score measures the average distance between the distributions in (a) and the distribution in (b). Images from Deng et al. (2009).

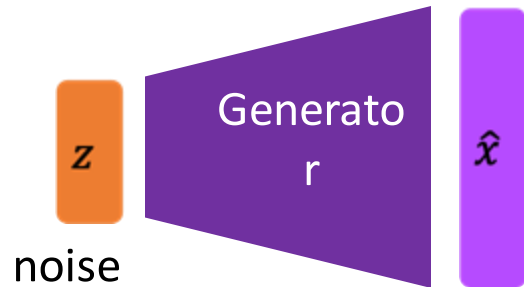


# Quantifying Performance - Fréchet Inception Distance

Another visual similarity metric based on Inception model (others can be used).

- Map generated images to distribution of Inception features.
- Model the distribution of Inception features as a multivariate normal distribution.
- Compare two such distributions with the Wasserstein distance (metric)
  - Also called “earth mover’s distance”
  - Smaller is better.
  - Closed form solution from multivariate normal assumption.

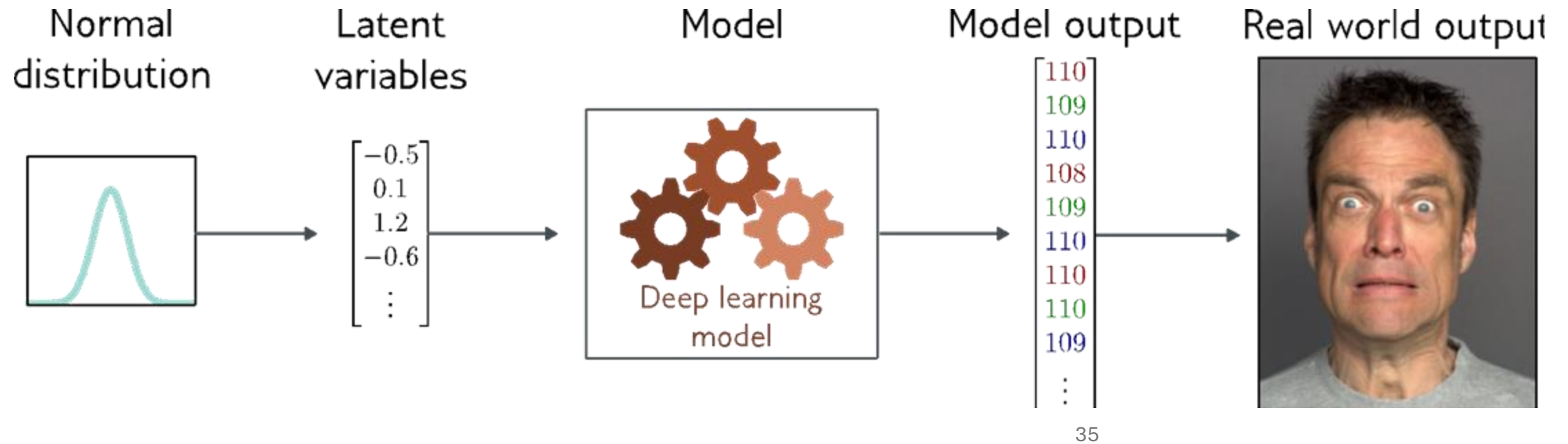
# General Idea of GANs



- Don't try to build a probability model directly
- Learn a transformation from a sample of noise to look similar to training data distribution

Left GANs vulnerable to mode collapse where only some of the distribution is replicated.

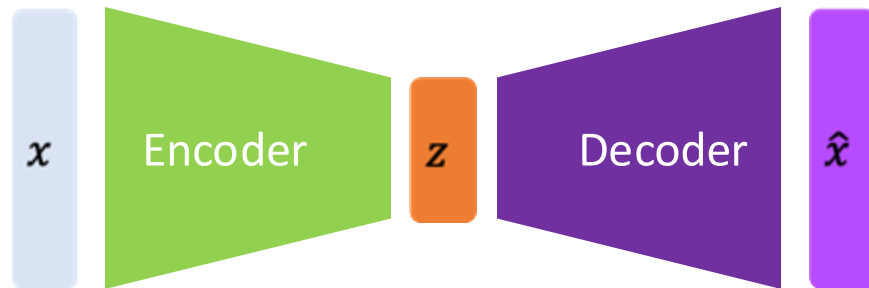
# Latent variable models



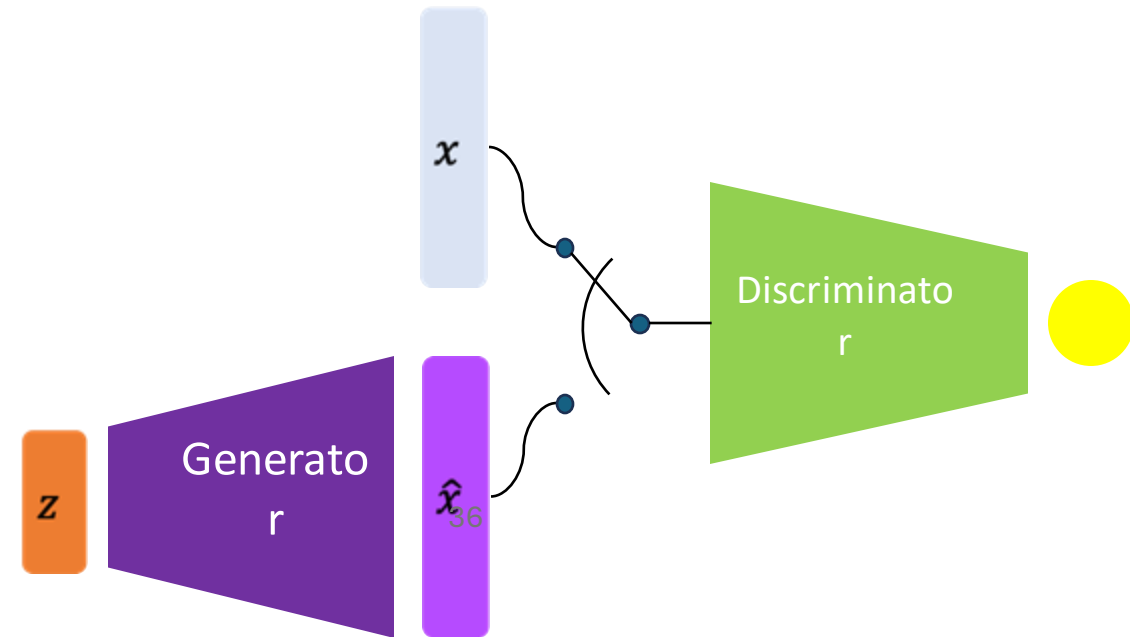
Latent variable models map a random “latent” variable to create a new data sample

# Latent Variable Models

Autoencoders and  
Variational Autoencoders  
(VAEs)



Generative Adversarial Networks



# Latent Variable Models

Informally speaking, different levels of latent variables...

- Latent variable directly determines observations
  - e.g.  $x = f(z)$
- Latent variable determines distribution of observations
  - e.g.  $x \sim \text{Norm}[f_\mu(z), f_{\sigma^2}(z)]$
- These levels aren't really different -
  - An extremely tight distribution  $\sim$  a fixed prediction
  - A fixed prediction + noise  $\sim$  a distribution

# Any Questions?



**Moving on**

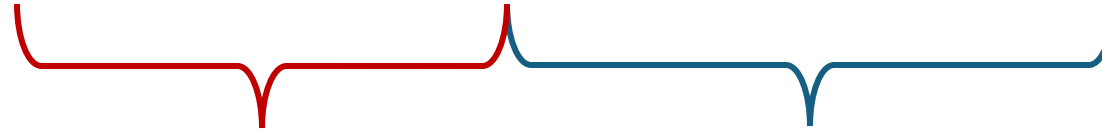
- Unsupervised Learning
- Latent Variables
- Probabilistic Generative Models
- Variational Autoencoders (VAEs)
- VAE math
- Examples of VAEs

# Variational Autoencoders (VAEs)

Goal is to learn the probability distribution from observed data

Can sample the distribution, but not evaluate probabilities exactly.

# Variational Autoencoder



Variational Inference: A method from machine learning that approximates probability densities through optimization.

Autoencoder: A type of artificial neural network used to learn efficient codings of unlabeled data in an unsupervised manner.

VAE is an autoencoder whose encodings distribution is regularized during the training to ensure that its latent space has good properties allowing us to generate new data.



# Auto-Encoding Variational Bayes

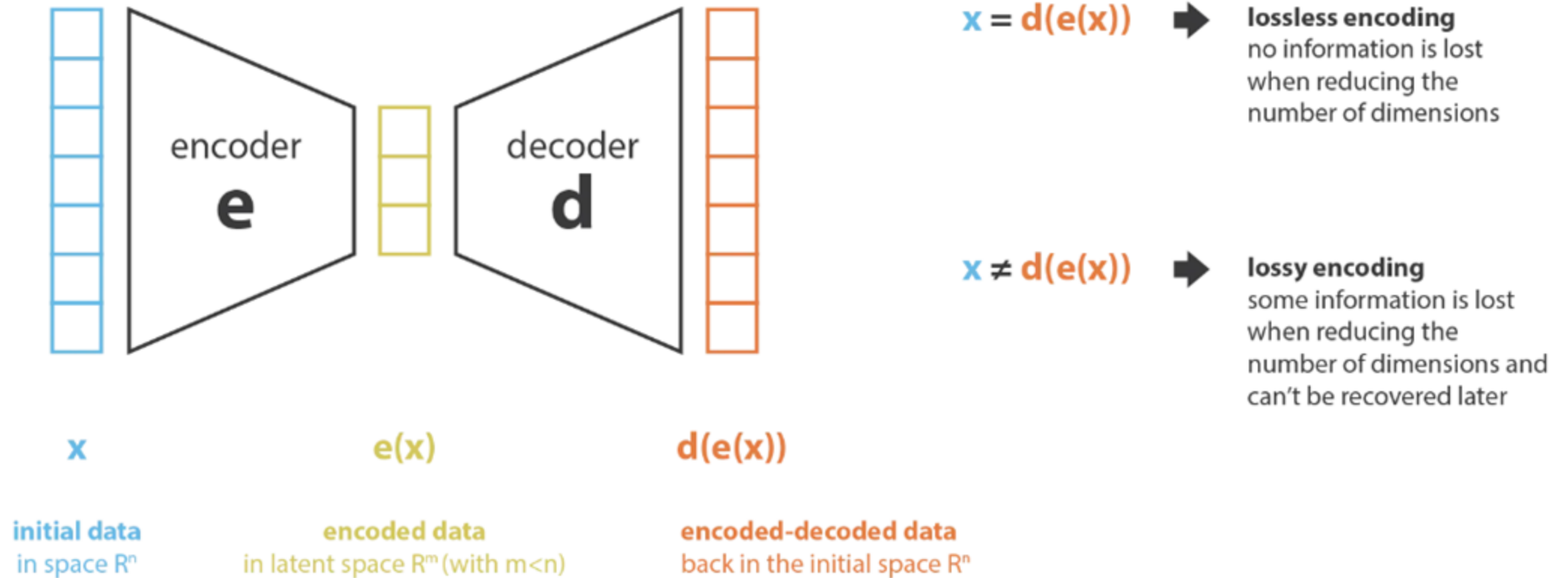
Autoencoder: A type of artificial neural network used to learn efficient codings of unlabeled data in an unsupervised manner.

Variational Inference: A method from machine learning that approximates probability densities through optimization.

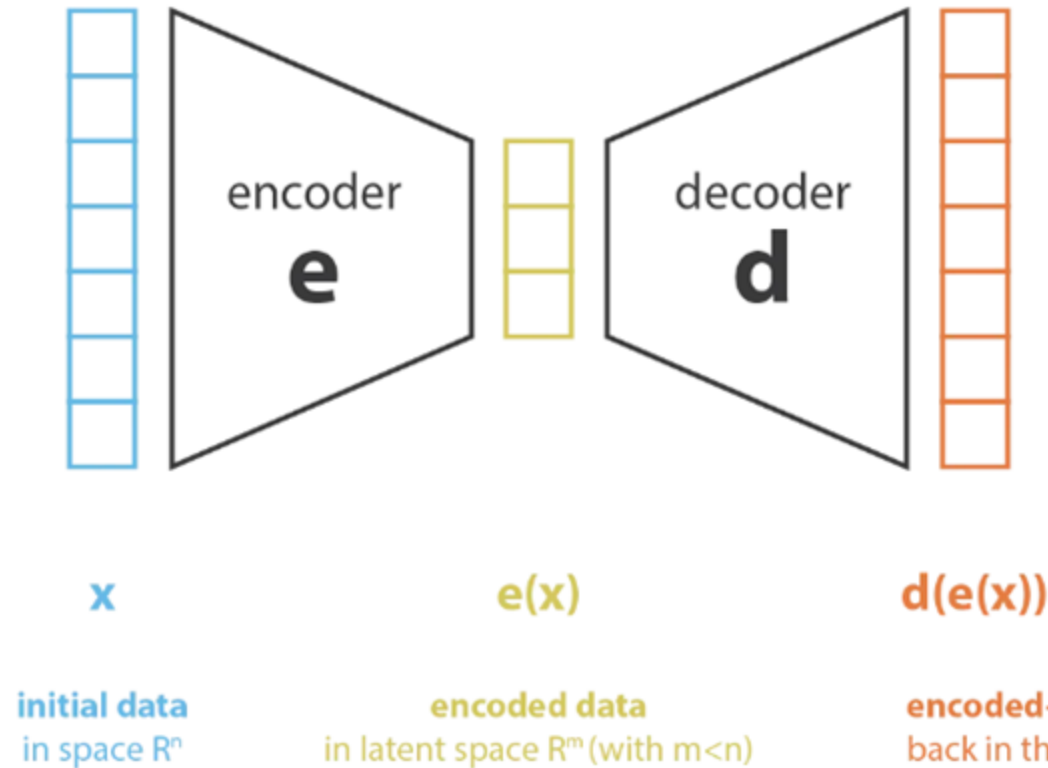
Bayesian since joint density is decomposed into prior and posterior density distributions using Bayes Rule:

$$p(\mathbf{z}, \mathbf{x}) = p(\mathbf{x}|\mathbf{z}) p(\mathbf{z})$$

# Dimensionality reduction with an autoencoder



# Dimensionality reduction with an autoencoder



We want to find the best encoder,  $e$ , and decoder,  $d$ , to minimize the error between  $x$  and  $d(e(x))$ .

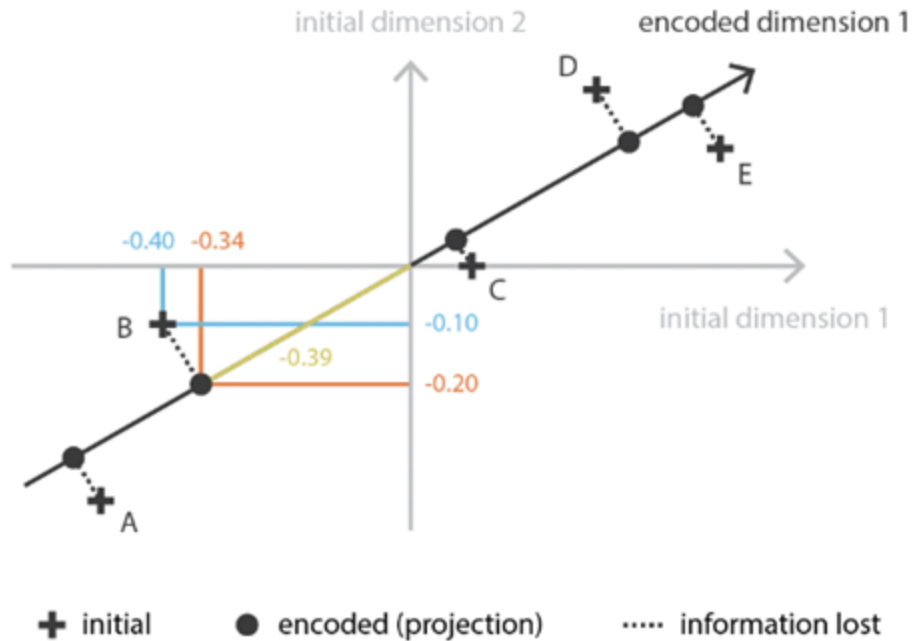
$$(e^*, d^*) = \operatorname{argmin}_{(e, d) \in E \times D} \epsilon(x, d(e(x)))$$

where

$$\epsilon(x, d(e(x)))$$

is the reconstruction error.

# Dimensionality reduction with Principal Component Analysis (PCA)

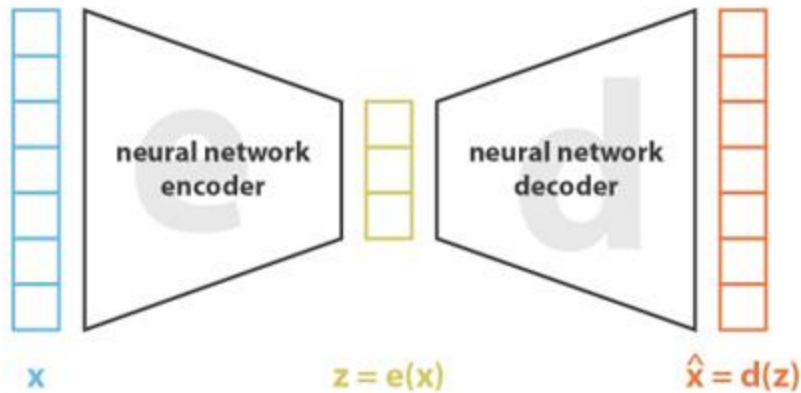


$n_d = 2 \quad n_e = 1$			
Point	Initial	Encoded	Decoded
A	(-0.50, -0.40)	-0.63	(-0.54, -0.33)
B	(-0.40, -0.10)	-0.39	(-0.34, -0.20)
C	(0.10, 0.00)	0.09	(0.07, 0.04)
D	(0.30, 0.30)	0.41	(0.35, 0.21)
E	(0.50, 0.20)	0.53	(0.46, 0.27)

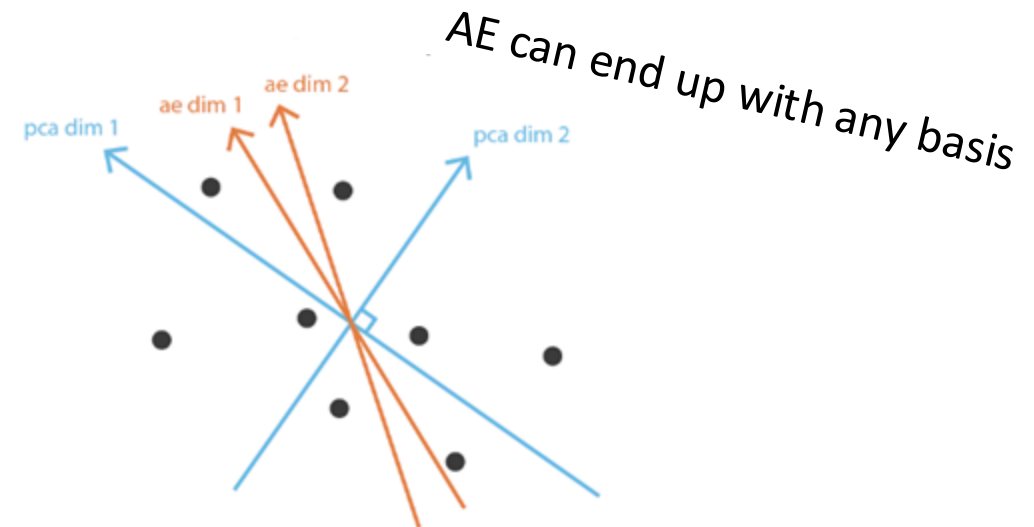
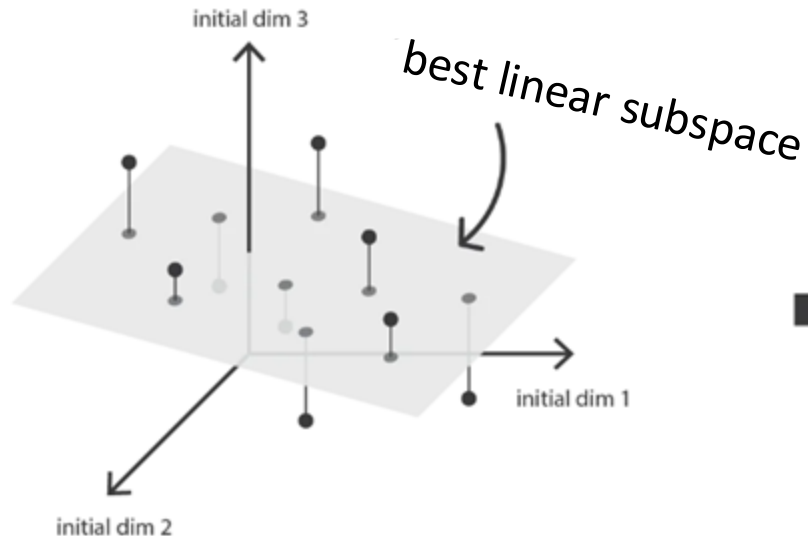
Project the  $n_d$ -dimensional features onto an orthogonal  $n_e$ -dimensional subspace that minimizes Euclidean distance.

# Linear Transformation!!

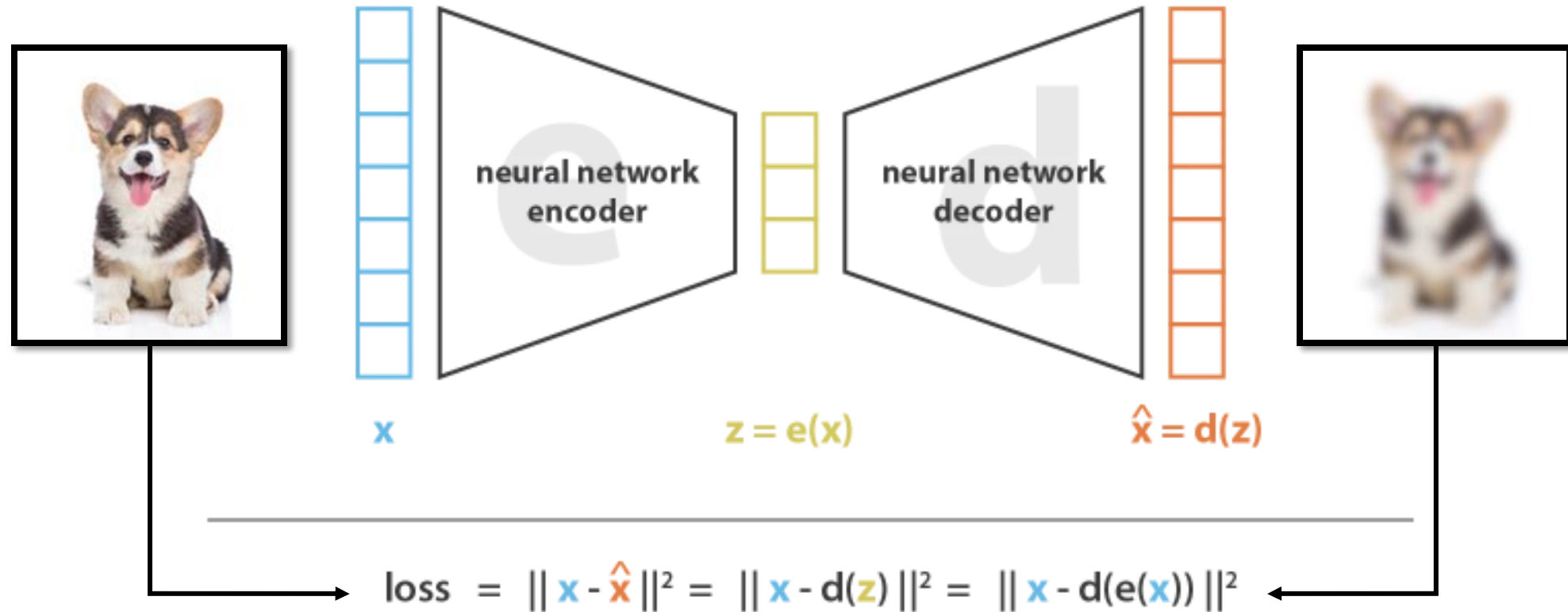
# Neural Network Autoencoder – 1 Linear Layer



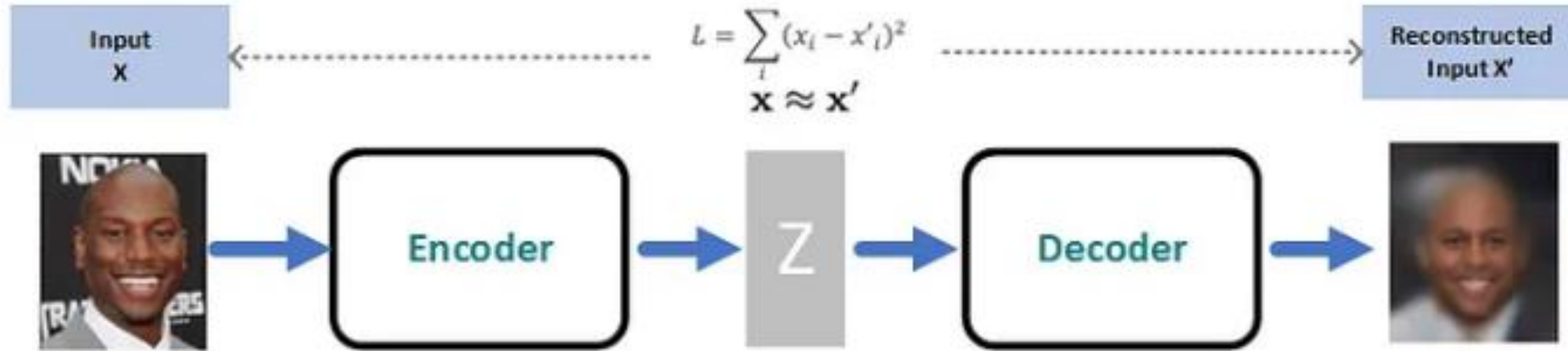
We could define encoder and decoder to each have one linear layer (no activation function), but it wouldn't necessarily converge during training to PCA solution.



# Neural Network Autoencoder

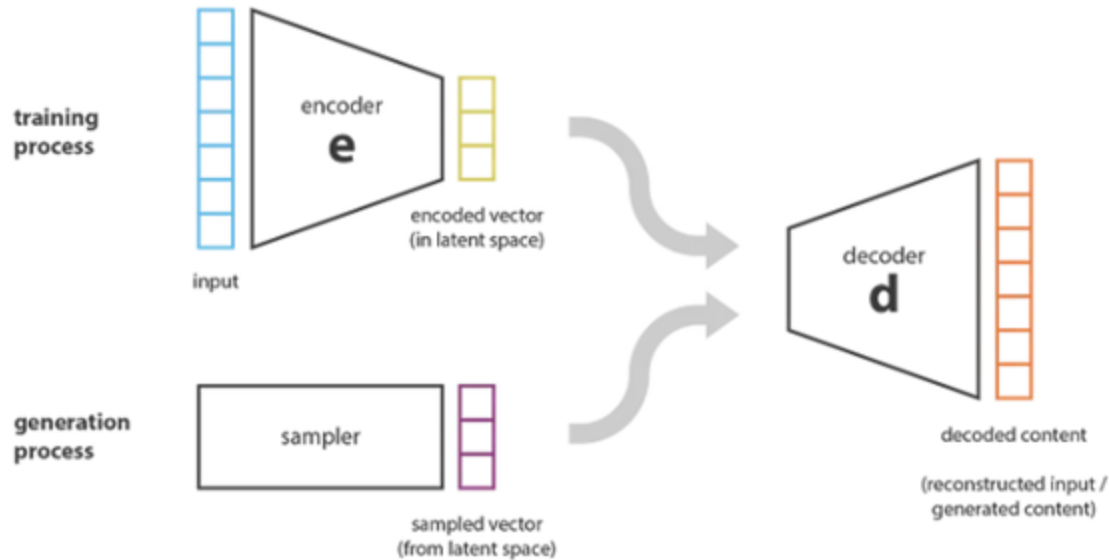


# Autoencoder Reconstruction



Trained on CelebA dataset.

# Can we generate new samples with autoencoder?



Train encoder and decoder as autoencoder.

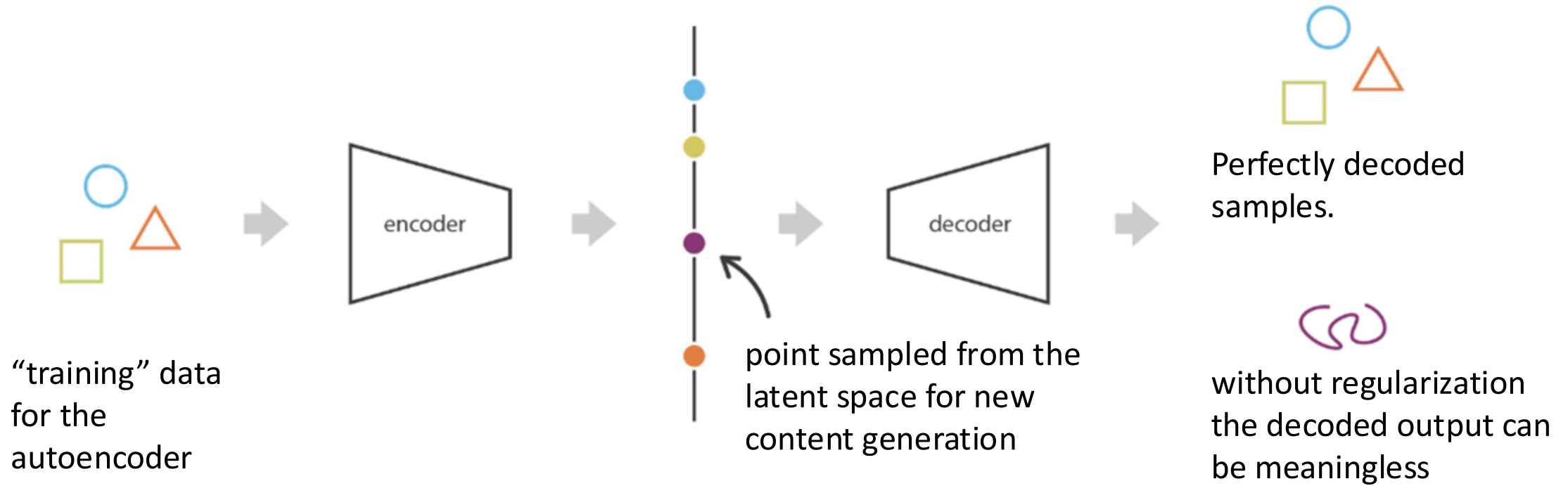
Randomly select a different point in the latent space.

Provide as input to the decoder to generate an output.

Will this produce a good quality output?  
Why?



# Extreme case: Memorization



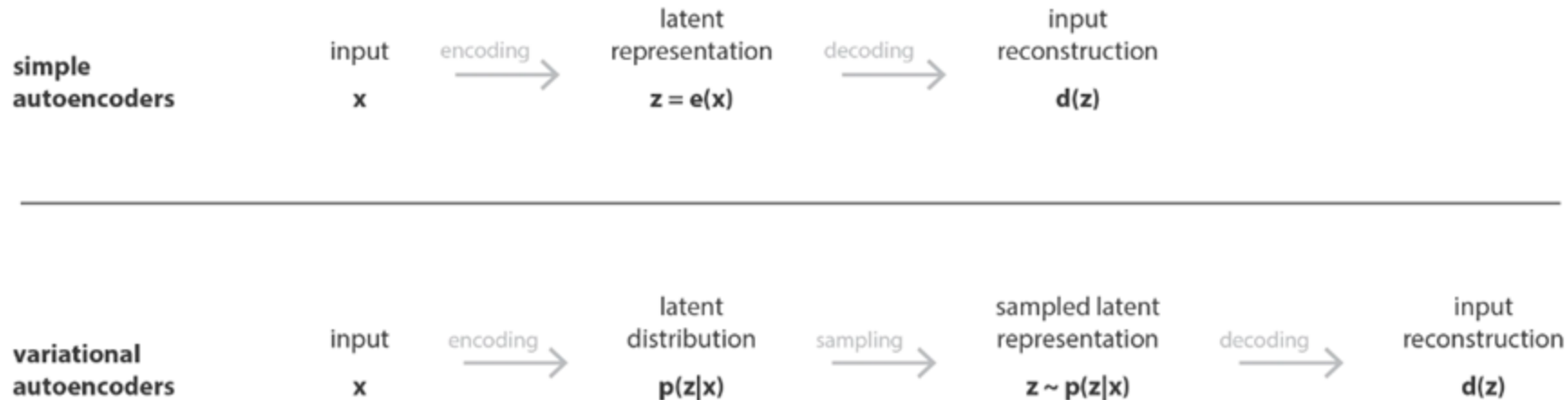
Encoder and decoder are so powerful that they can fully memorize the data.

# Variational Autoencoder...

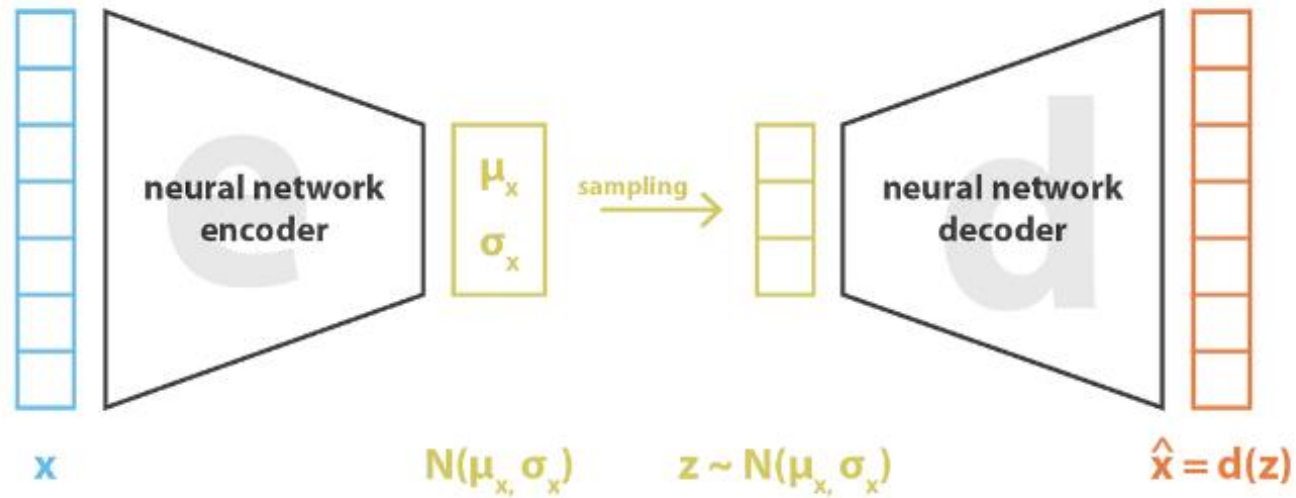
...is an autoencoder whose training is *regularized* to avoid overfitting and ensure that the *latent space has good properties* that enable generative process.

Instead of encoding as a *single point*, encode it as a *distribution* over the latent space.

Assume distributions are normal.



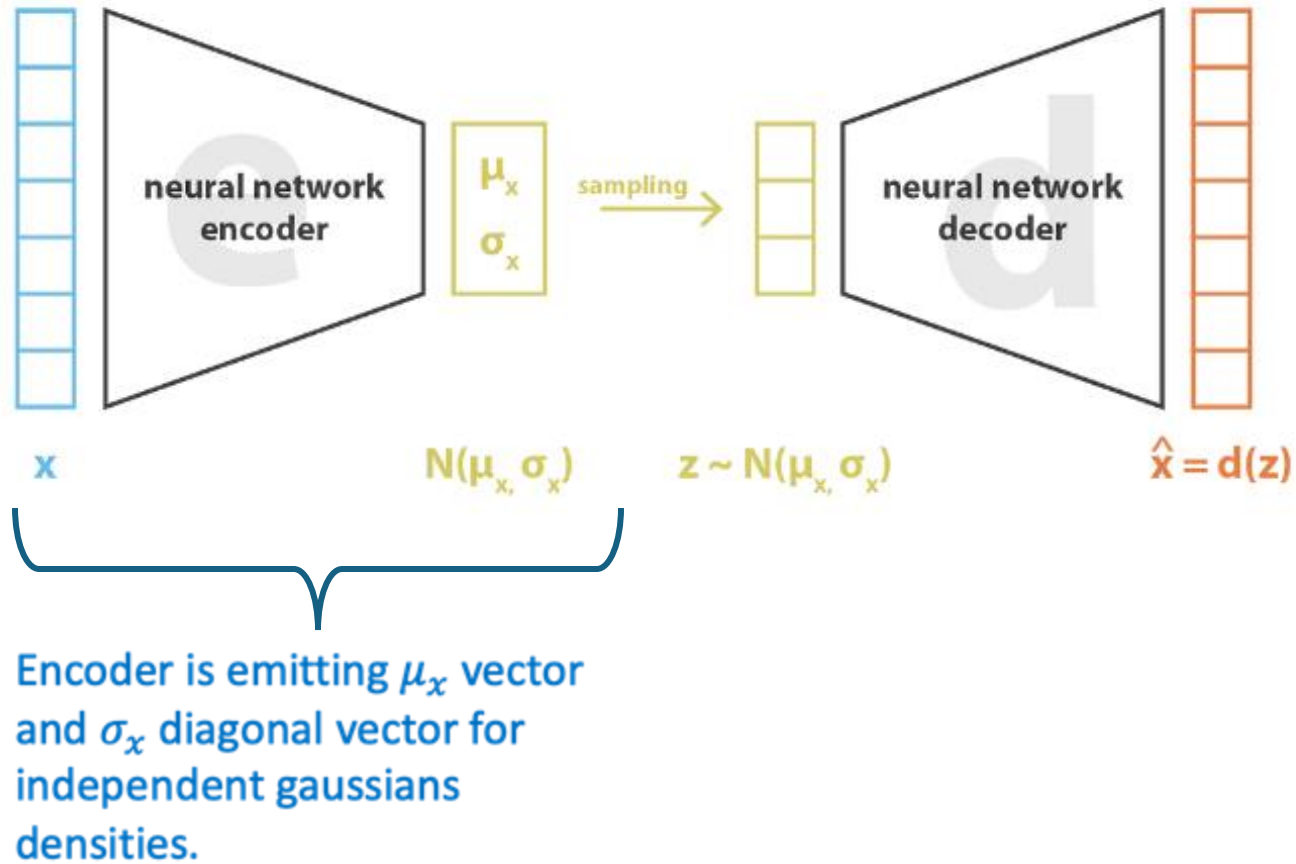
# Variational Autoencoder



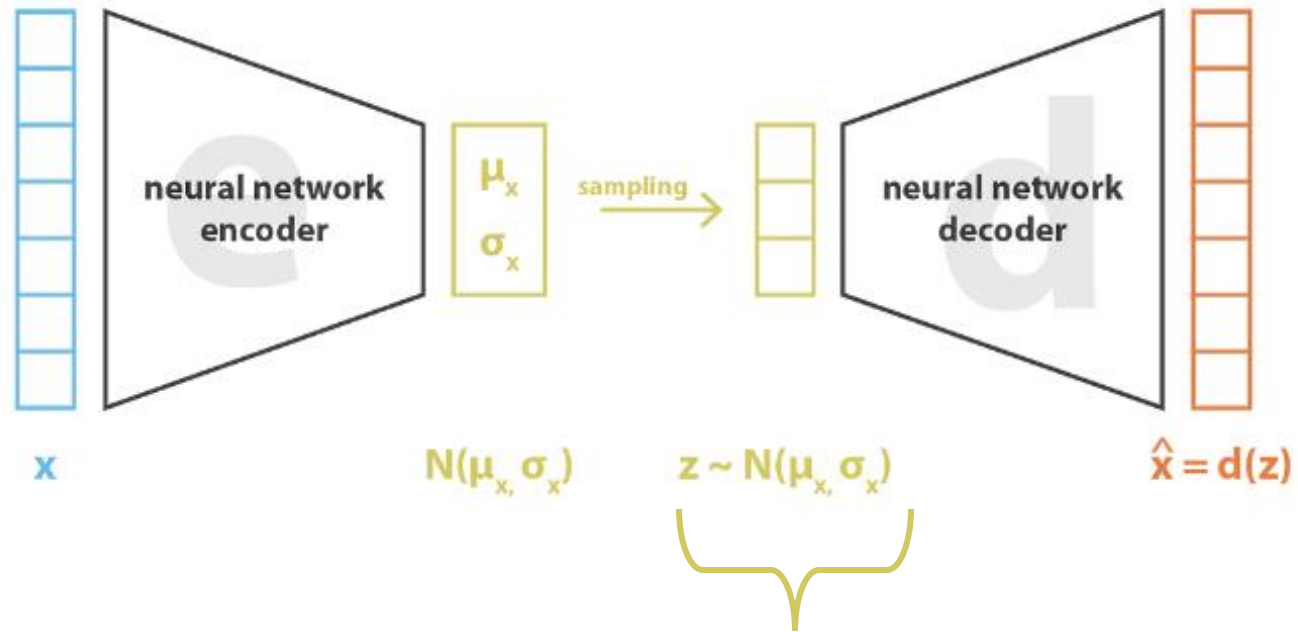
---

$$\text{loss} = ||x - \hat{x}||^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)] = ||x - d(z)||^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)]$$

# Variational Autoencoder

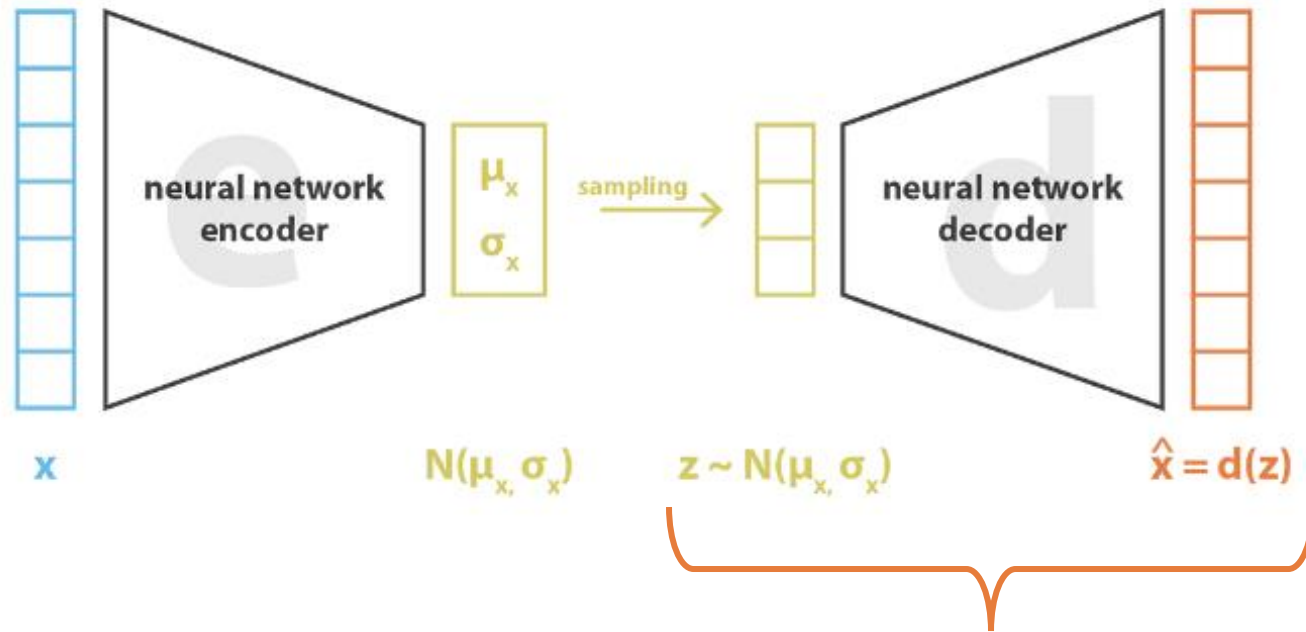


# Variational Autoencoder



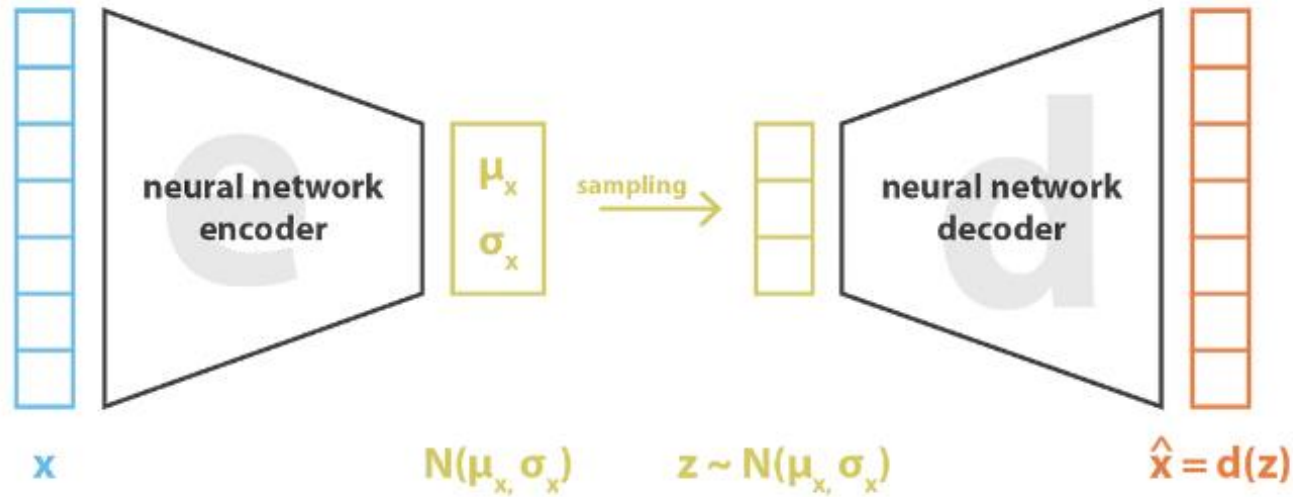
We then sample  $z$  from the multivariate Normal.

# Variational Autoencoder



Then input  $z$  to the decoder network to produce output.

# Variational Autoencoder

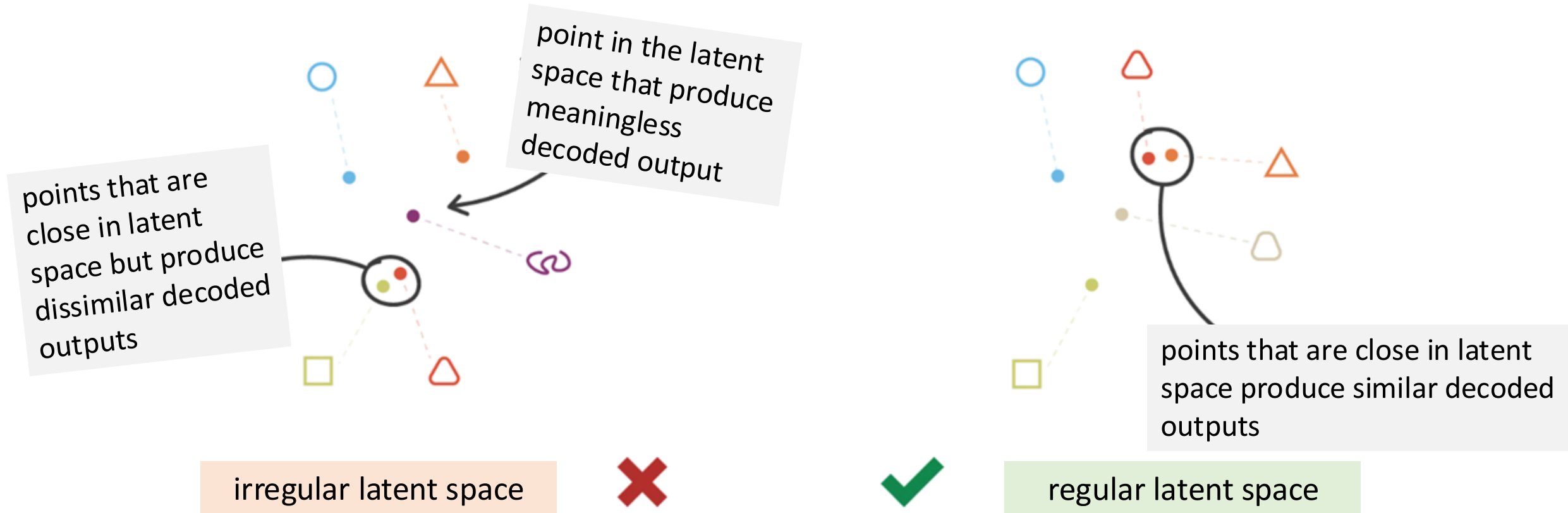


$$\text{loss} = \underbrace{\|x - \hat{x}\|^2}_{\text{L2 Loss}} + \underbrace{\text{KL}[N(\mu_x, \sigma_x), N(0, I)]}_{\text{Kulback-Leibler divergence}} = \|x - d(z)\|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)]$$

The loss is now the L2 loss as with the autoencoder, but with an additional KL-divergence term as regularizer.

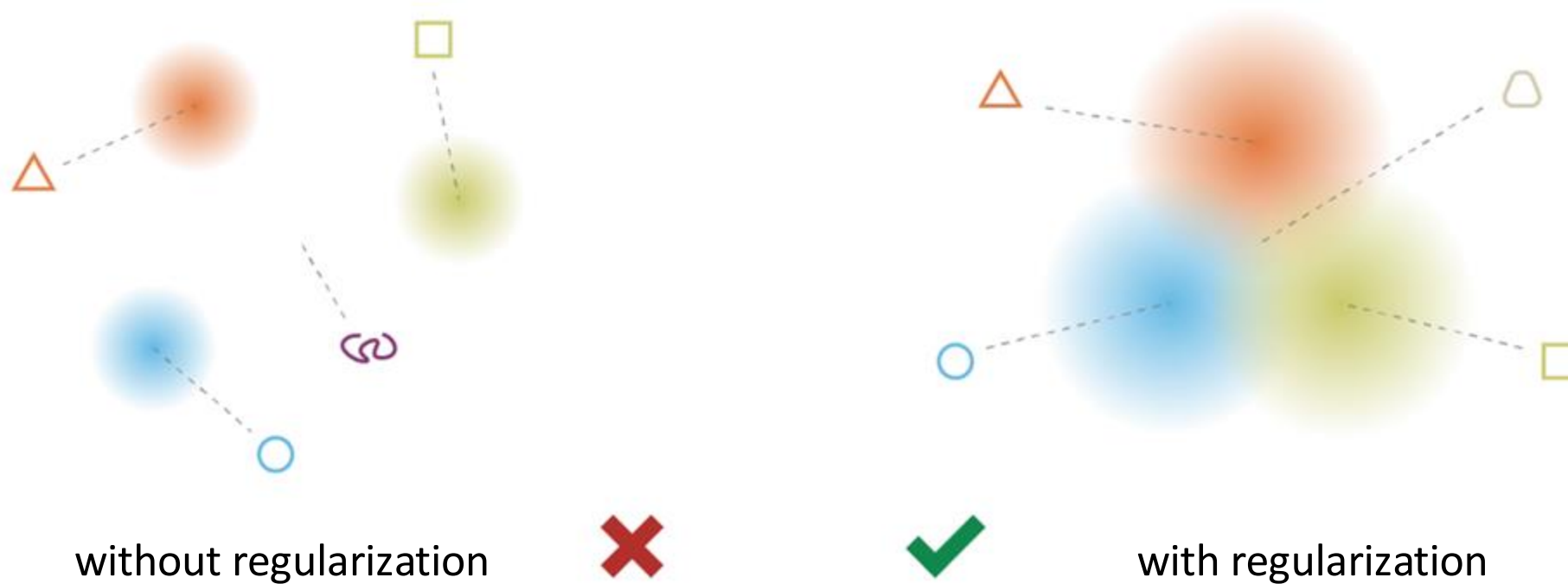
$$D_{\text{KL}}(P||Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)}$$

# Intuitions about Regularization





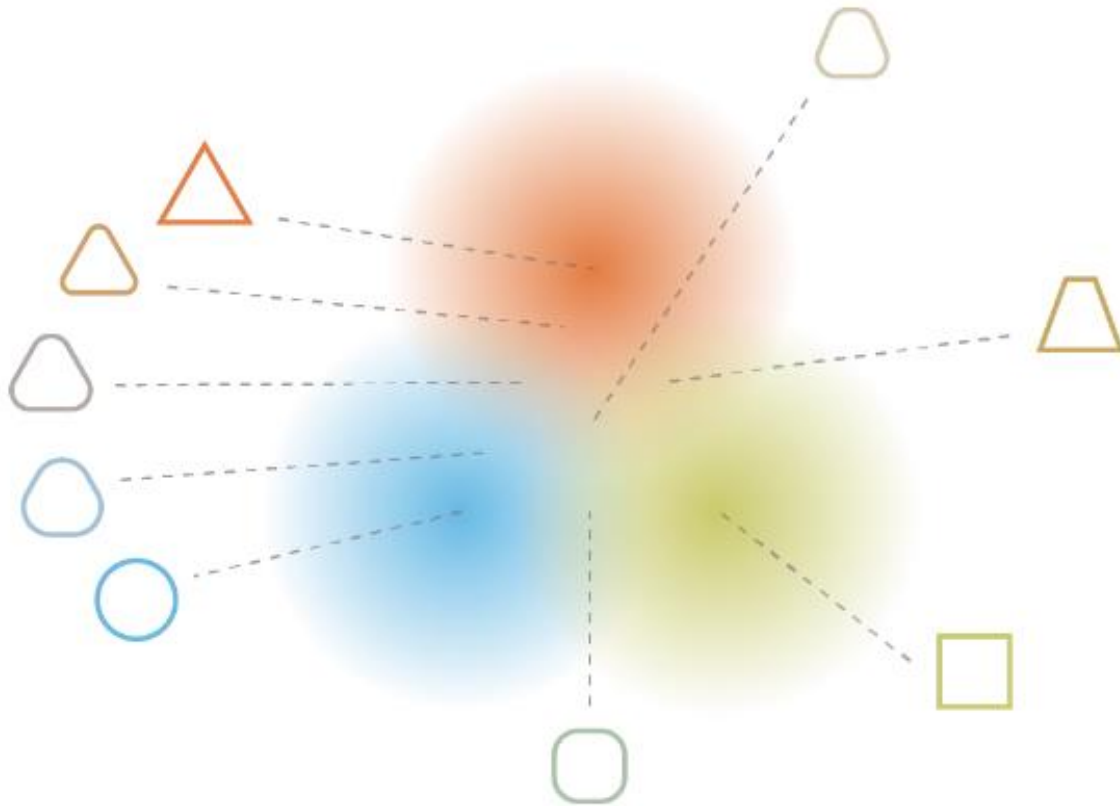
# Encoding to Normal distributions is not enough



We have to regularize the means and the covariances too!  
Regularize to a standard normal.

$$\text{loss} = ||\mathbf{x} - \hat{\mathbf{x}}||^2 + \text{KL}[N(\boldsymbol{\mu}_x, \boldsymbol{\sigma}_x), N(\mathbf{0}, \mathbf{I})]$$

# Benefit of regularization



The continuity and completeness obtained from regularization tends to create a “gradient” over the information encoded in latent space.

# Any Questions?



**Moving on**

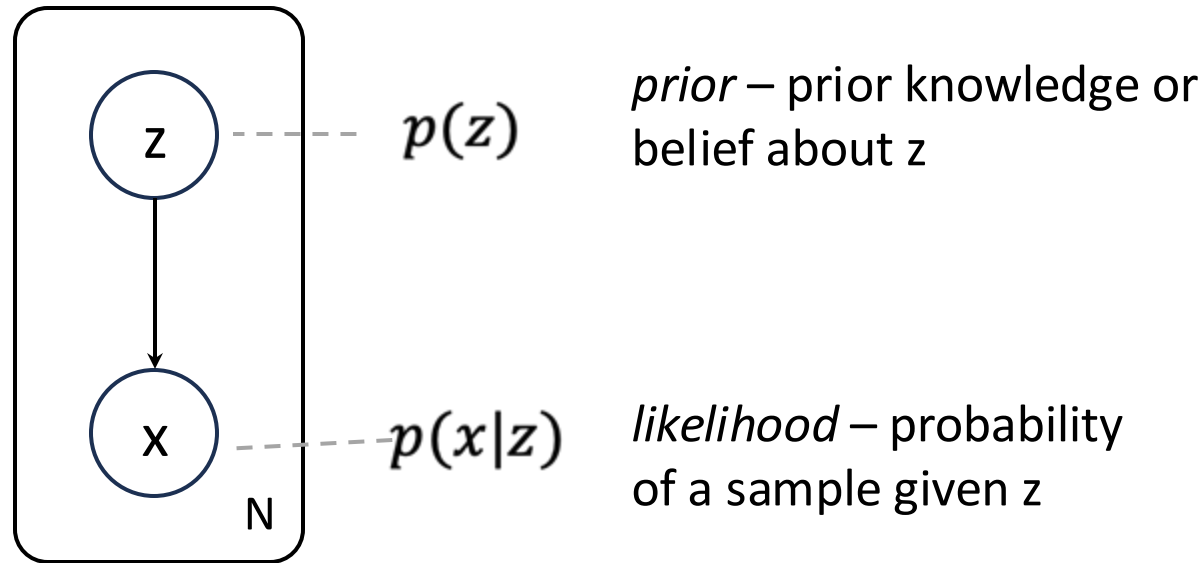
- Unsupervised Learning
- Latent Variables
- Probabilistic Generative Models
- Variational Autoencoders (VAEs)
- VAE math
- Examples of VAEs



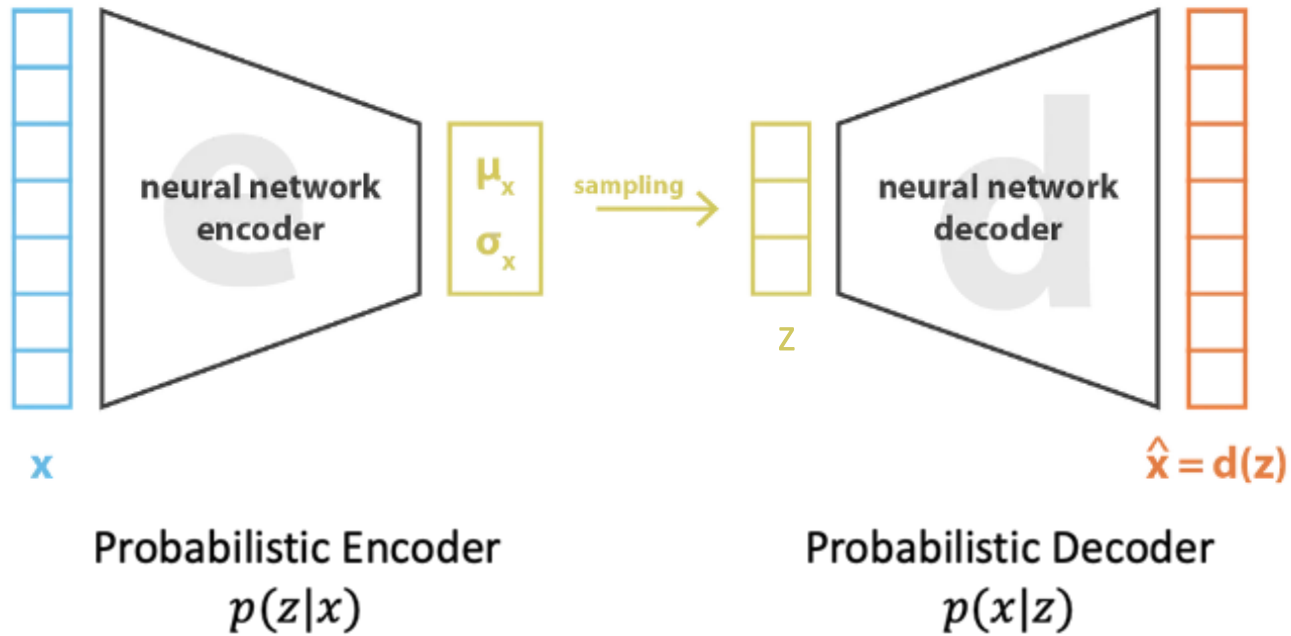




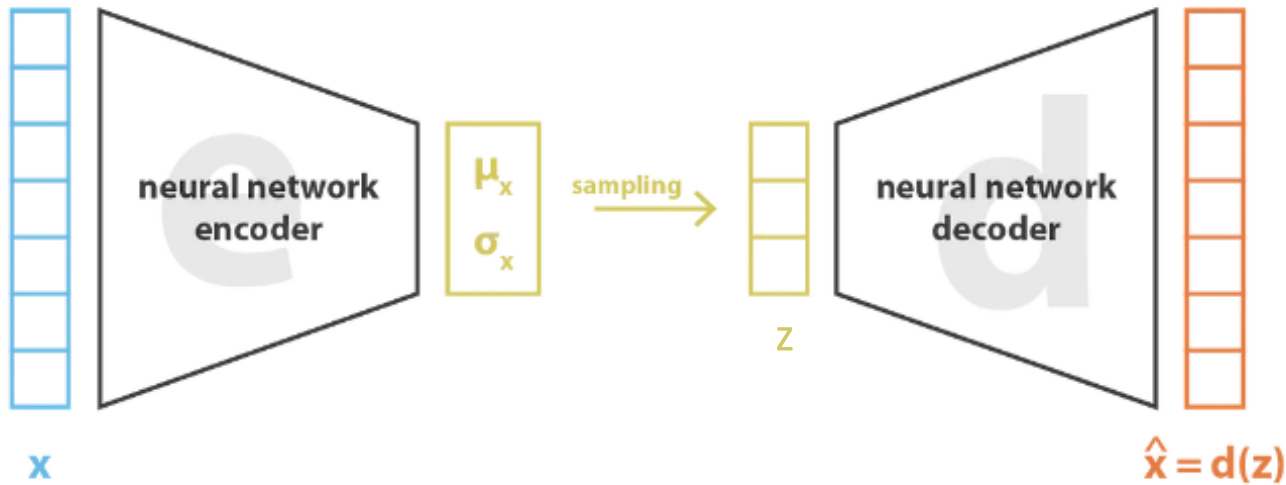
# Preliminaries: Bayesian Models



# Bayesian Inference



# Bayesian Inference



Probabilistic Encoder  
 $p(z|x)$

*posterior* – update our knowledge of  $z$  given a new sample

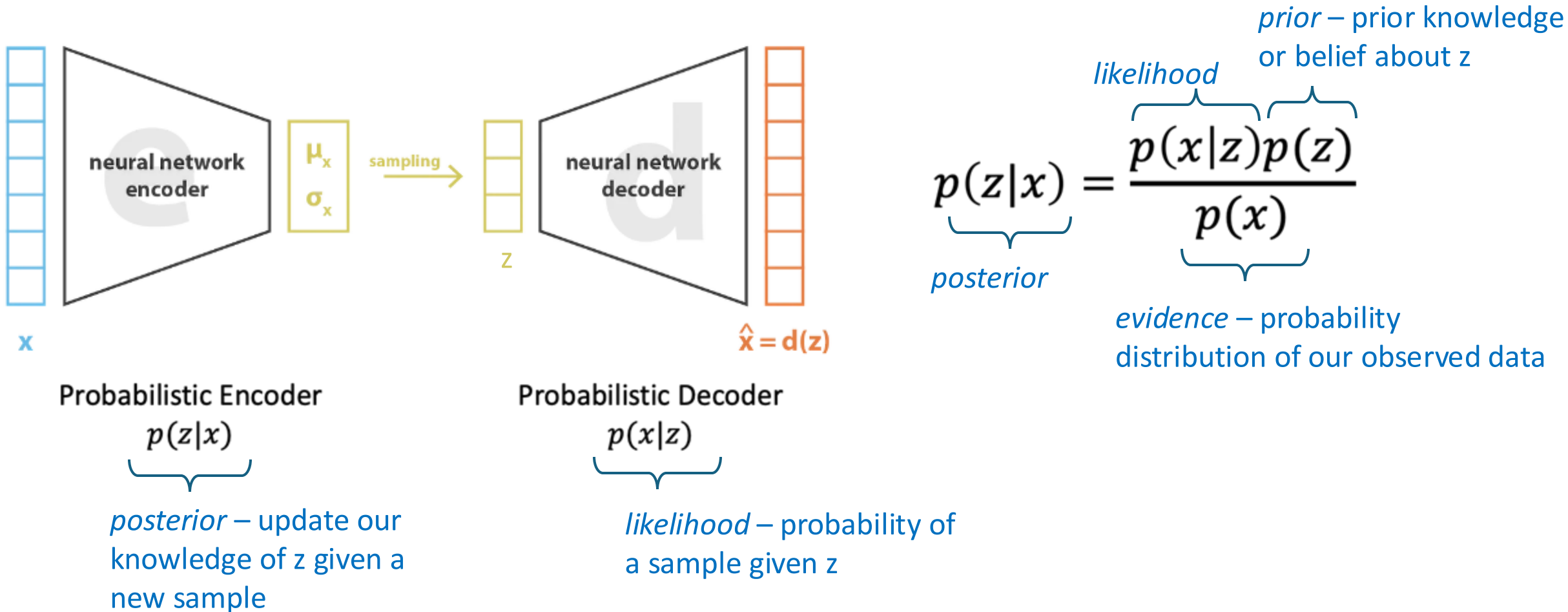
Probabilistic Decoder  
 $p(x|z)$

*likelihood* – probability of a sample given  $z$

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

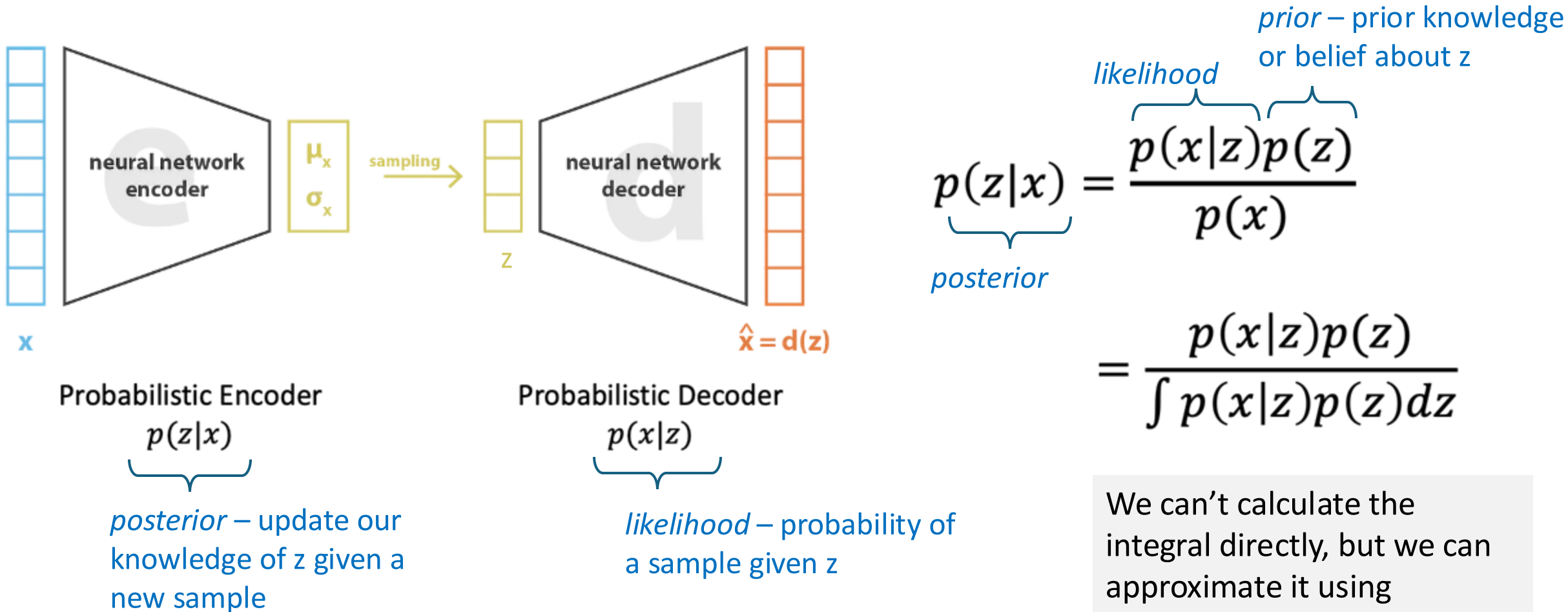
We can relate the *posterior* to the *likelihood* via **Bayes Theorem**.

# Bayesian Inference

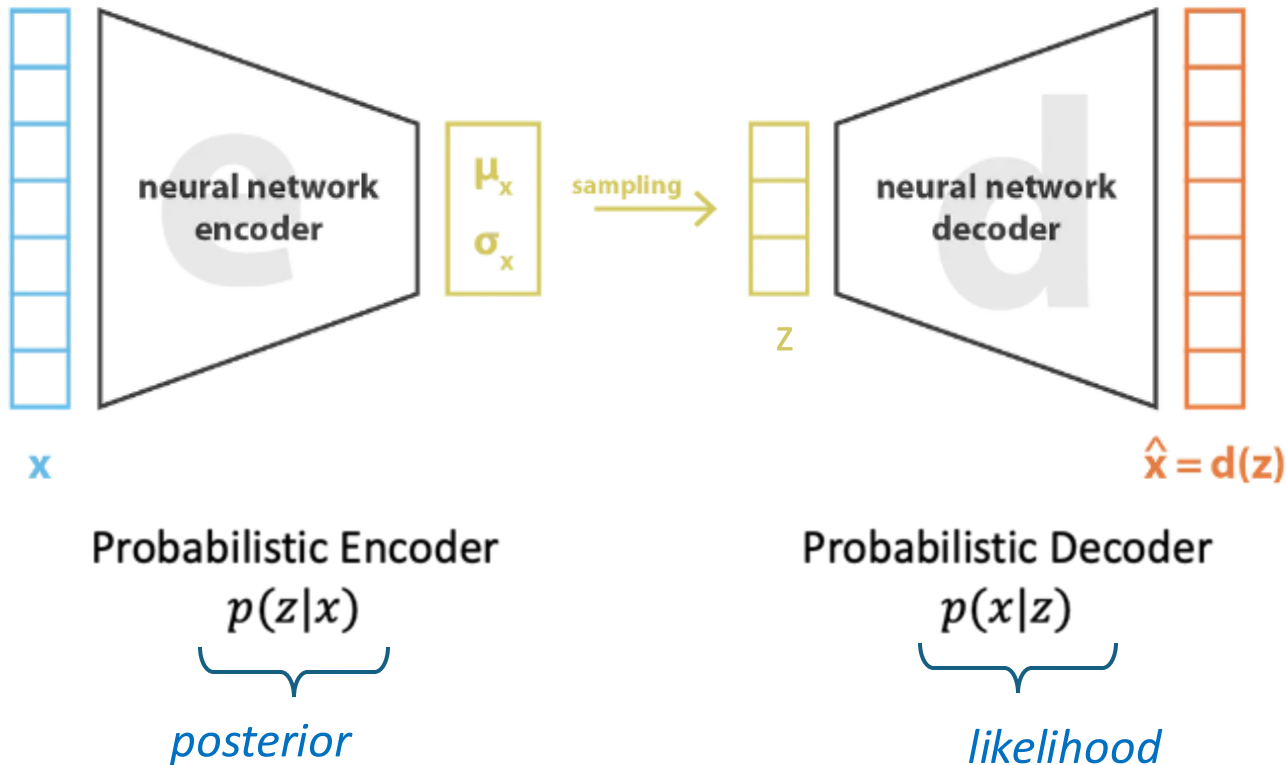




# Bayesian Inference



# Simplifying Assumptions



Assume that the *prior* is a standard Gaussian

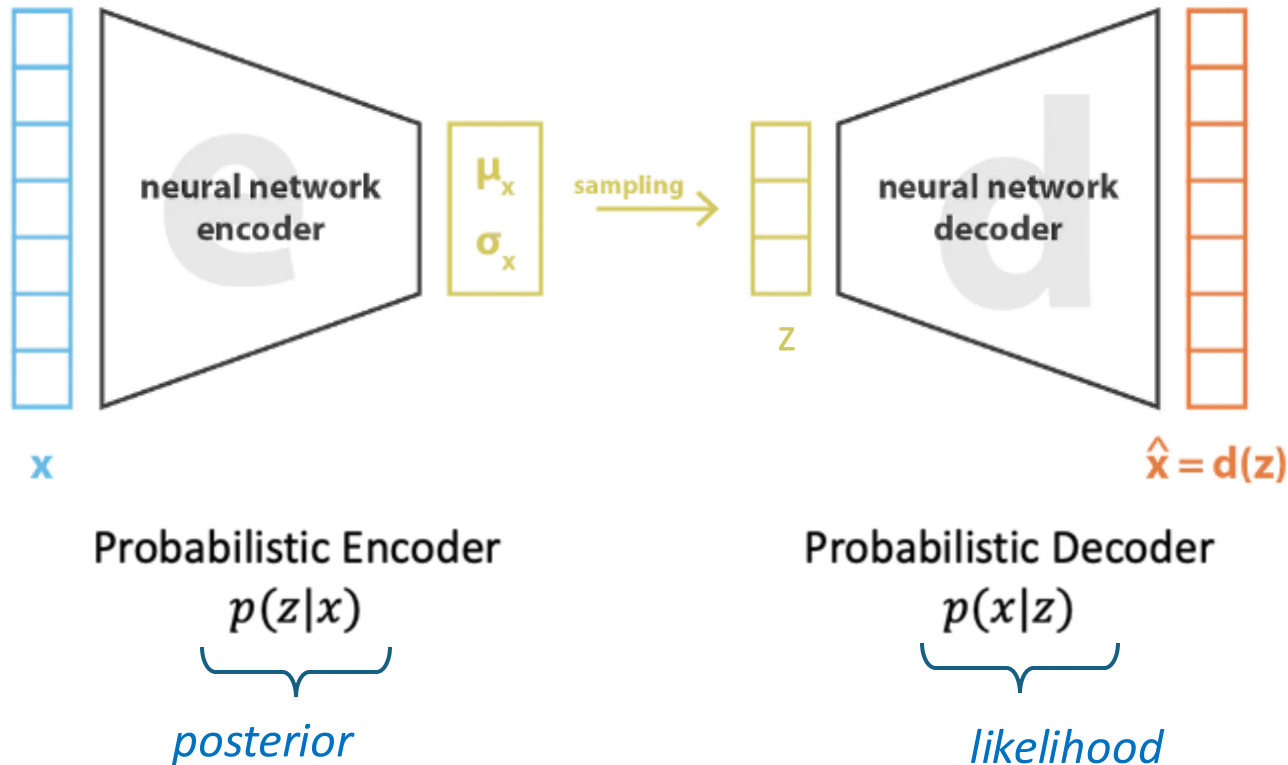
$$p(z) \equiv \mathcal{N}(0, I)$$

And *likelihood* is a Gaussian

$$p(x|z) \equiv \mathcal{N}(f(z), cI)$$

where  $f \in F$  is a family of functions we will specify later and  $c > 0$ .

# Variational Inference Formulation



We are going to approximate *posterior* to parameterized set of Gaussians.

Approximate  $p(z|x)$  by a Gaussian  $q_x(z)$ .

$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

where  $g \in G$  and  $h \in H$  are a family of functions we will define shortly.

$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

# Variational Inference

$$(g^*, h^*) = \arg \min_{(g, h) \in G \times H} KL(q_x(z), p(z|x))$$

We want to find the best functions,  $g$  and  $h$ , to minimize the KL-divergence from the posterior  $p(z|x)$ .

## C.5.1 Kullback-Leibler divergence

The most common measure of distance between probability distributions  $p(x)$  and  $q(x)$  is the *Kullback-Leibler* or KL divergence and is defined as:

$$D_{KL}[p(x)||q(x)] = \int p(x) \log \left[ \frac{p(x)}{q(x)} \right] dx. \quad (\text{C.28})$$

$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

# Variational Inference

$$\begin{aligned}(g^*, h^*) &= \arg \min_{(g, h) \in G \times H} KL(q_x(z), p(z|x)) \\ &= \arg \min_{(g, h) \in G \times H} \left( \mathbb{E}_{z \sim q_x} (\log q_x(z)) - \mathbb{E}_{z \sim q_x} \left( \log \frac{p(x|z)p(z)}{p(x)} \right) \right)\end{aligned}$$

- Rewriting KL divergence as Expectation,
- log of division is difference of the logs
- substituting for the posterior using Bayes Theorem

$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

# Variational Inference

$$\begin{aligned}(g^*, h^*) &= \arg \min_{(g, h) \in G \times H} KL(q_x(z), p(z|x)) \\&= \arg \min_{(g, h) \in G \times H} \left( \mathbb{E}_{z \sim q_x} (\log q_x(z)) - \mathbb{E}_{z \sim q_x} \left( \log \frac{p(x|z)p(z)}{p(x)} \right) \right) \\&= \arg \min_{(g, h) \in G \times H} (\mathbb{E}_{z \sim q_x} (\log q_x(z)) - \mathbb{E}_{z \sim q_x} (\log p(z)) - \mathbb{E}_{z \sim q_x} (\log p(x|z)) + \mathbb{E}_{z \sim q_x} (\log p(x)))\end{aligned}$$

- log of product becomes sum of logs
- log of division becomes difference of logs

$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

# Variational Inference

$$\begin{aligned}(g^*, h^*) &= \arg \min_{(g, h) \in G \times H} KL(q_x(z), p(z|x)) \\&= \arg \min_{(g, h) \in G \times H} \left( \mathbb{E}_{z \sim q_x} (\log q_x(z)) - \mathbb{E}_{z \sim q_x} \left( \log \frac{p(x|z)p(z)}{p(x)} \right) \right) \\&= \arg \min_{(g, h) \in G \times H} (\mathbb{E}_{z \sim q_x} (\log q_x(z)) - \mathbb{E}_{z \sim q_x} (\log p(z)) - \mathbb{E}_{z \sim q_x} (\log p(x|z)) + \mathbb{E}_{z \sim q_x} (\log p(x))) \\&= \arg \max_{(g, h) \in G \times H} (\mathbb{E}_{z \sim q_x} (\log p(x|z)) - KL(q_x(z), p(z)))\end{aligned}$$

- negating and converting from argmin to argmax
- collecting terms to form KL divergence

$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

# Variational Inference

$$\begin{aligned}(g^*, h^*) &= \arg \min_{(g, h) \in G \times H} KL(q_x(z), p(z|x)) \\&= \arg \min_{(g, h) \in G \times H} \left( \mathbb{E}_{z \sim q_x} (\log q_x(z)) - \mathbb{E}_{z \sim q_x} \left( \log \frac{p(x|z)p(z)}{p(x)} \right) \right) \\&= \arg \min_{(g, h) \in G \times H} (\mathbb{E}_{z \sim q_x} (\log q_x(z)) - \mathbb{E}_{z \sim q_x} (\log p(z)) - \mathbb{E}_{z \sim q_x} (\log p(x|z)) + \mathbb{E}_{z \sim q_x} (\log p(x))) \\&= \arg \max_{(g, h) \in G \times H} \underbrace{(\mathbb{E}_{z \sim q_x} (\log p(x|z)))}_{\text{Maximize the expected log likelihood.}} - \underbrace{KL(q_x(z), p(z))}_{\text{Minimize the difference between the approximate posterior and the prior.}}\end{aligned}$$



$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

# Variational Inference

$$\begin{aligned}
 (g^*, h^*) &= \arg \min_{(g, h) \in G \times H} KL(q_x(z), p(z|x)) \\
 &= \arg \min_{(g, h) \in G \times H} \left( \mathbb{E}_{z \sim q_x} (\log q_x(z)) - \mathbb{E}_{z \sim q_x} \left( \log \frac{p(x|z)p(z)}{p(x)} \right) \right) \\
 &= \arg \min_{(g, h) \in G \times H} (\mathbb{E}_{z \sim q_x} (\log q_x(z)) - \mathbb{E}_{z \sim q_x} (\log p(z)) - \mathbb{E}_{z \sim q_x} (\log p(x|z)) + \mathbb{E}_{z \sim q_x} (\log p(x))) \\
 &= \arg \max_{(g, h) \in G \times H} (\mathbb{E}_{z \sim q_x} (\log p(x|z)) - KL(q_x(z), p(z))) \\
 &= \arg \max_{(g, h) \in G \times H} \left( \underbrace{\mathbb{E}_{z \sim q_x} \left( -\frac{\|x - f(z)\|^2}{2c} \right)}_{\text{Log of the Gaussian likelihood } p(x|z) \equiv \mathcal{N}(f(z), cI)} - KL(q_x(z), p(z)) \right)
 \end{aligned}$$

Log of the Gaussian likelihood  $p(x|z) \equiv \mathcal{N}(f(z), cI)$ .

This brings our function,  $f$ , into the equation, so...

$$q_x(z) \equiv \mathcal{N}(g(x), h(x))$$

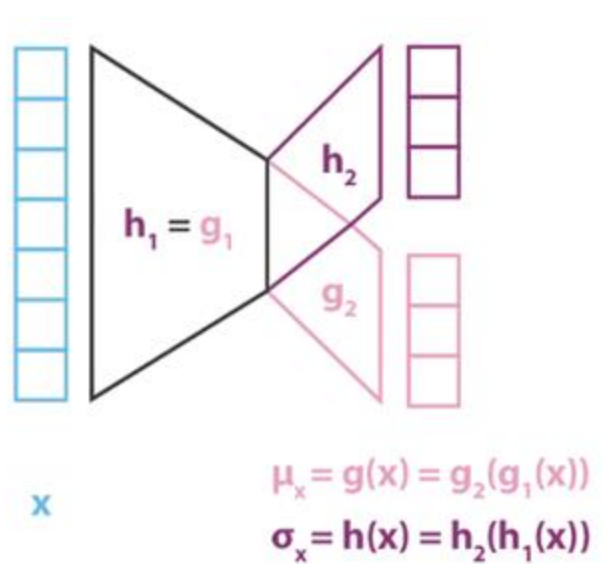
# Variational Inference

We are looking for optimal  $f^*$ ,  $g^*$  and  $h^*$  such that

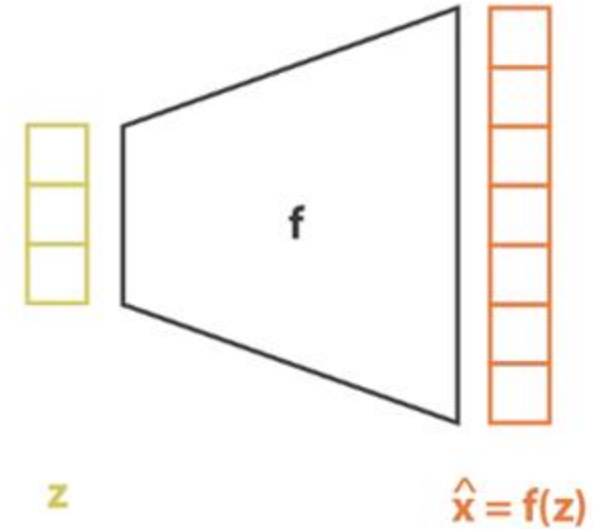
$$(f^*, g^*, h^*) = \arg \max_{(f, g, h) \in F \times G \times H} \left( \mathbb{E}_{z \sim q_x} \left( -\frac{\|x - f(z)\|^2}{2c} \right) - KL(q_x(z), p(z)) \right)$$

Note that the constant,  $c$ , determines the balance between reconstruction error and the regularization term given by KL divergence.

# Enter the Neural Networks

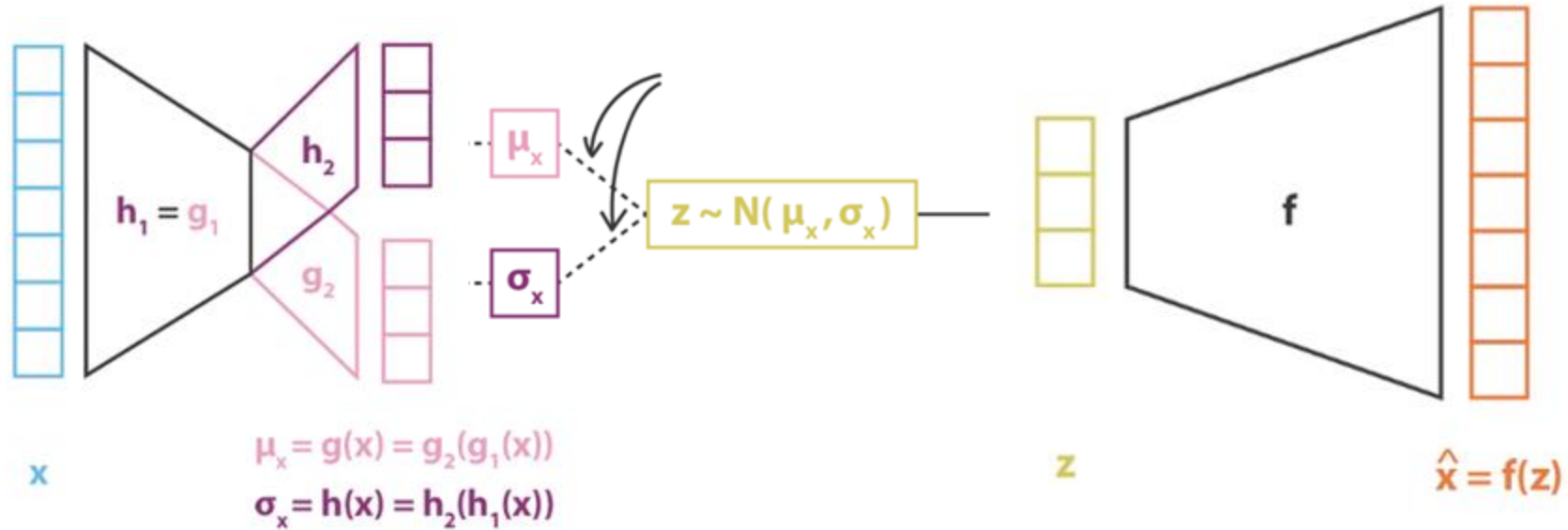


Encoder produces the mean and variance.



Decoder reconstructs the input (during training)

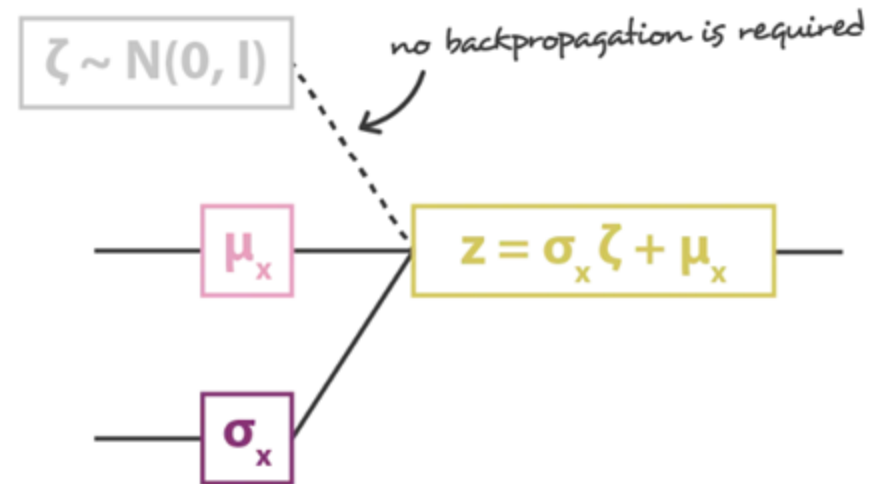
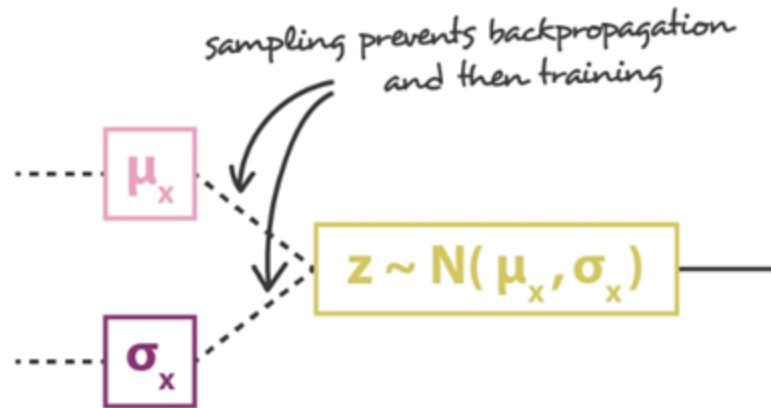
# But one more problem to solve



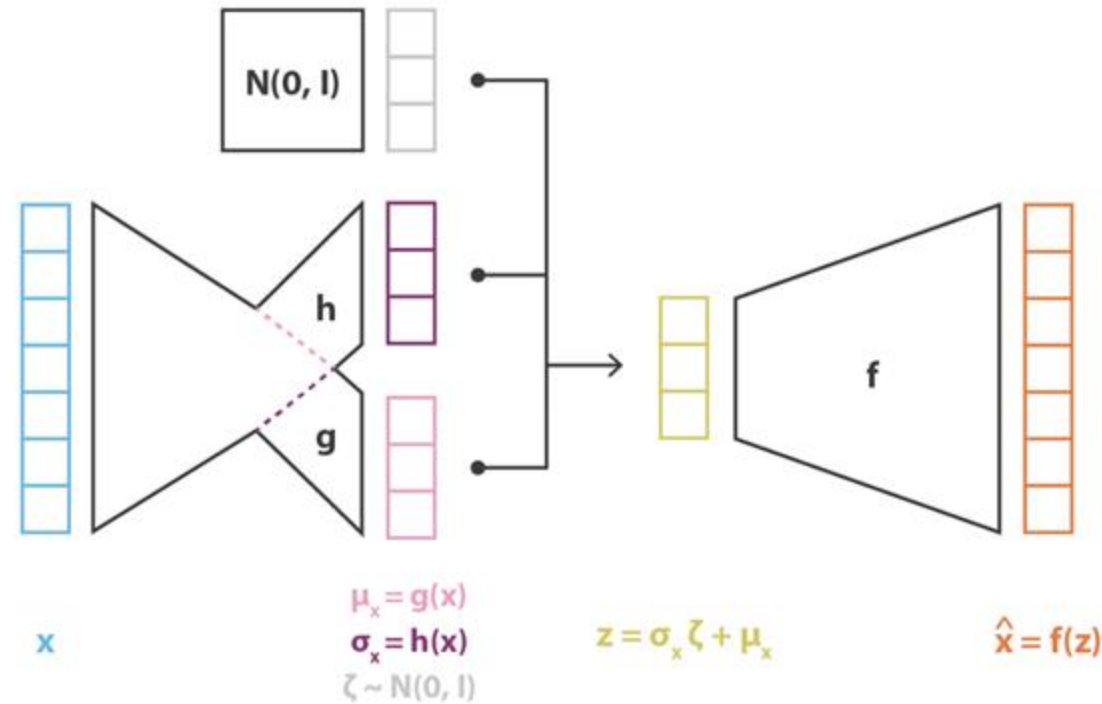
76

We can't backpropagate through the sampling step.

# Use the reparameterization trick



# Putting it all together



We use a Monte-Carlo approximation to the expectation of reconstruction loss

Convert  $C = 1/(2c)$ .

---

$$\text{loss} = C \|x - \hat{x}\|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)] = C \|x - f(z)\|^2 + \text{KL}[N(g(x), h(x)), N(0, I)]$$

We have as trainable neural network!

# Probability Distribution Divergence Measures

## C.5.1 Kullback-Leibler divergence

The most common measure of distance between probability distributions  $p(x)$  and  $q(x)$  is the *Kullback-Leibler* or KL divergence and is defined as:

$$D_{KL}[p(x)||q(x)] = \int p(x) \log \left[ \frac{p(x)}{q(x)} \right] dx. \quad (\text{C.28})$$

---

## C.5.2 Jensen-Shannon divergence

The KL divergence is not symmetric (i.e.,  $D_{KL}[p(x)||q(x)] \neq D_{KL}[q(x)||p(x)]$ ). The Jensen-Shannon divergence is a measure of distance that is symmetric by construction:

$$D_{JS}[p(x)||q(x)] = \frac{1}{2} D_{KL} \left[ p(x) \middle| \middle| \frac{p(x) + q(x)}{2} \right] + \frac{1}{2} D_{KL} \left[ q(x) \middle| \middle| \frac{p(x) + q(x)}{2} \right]. \quad (\text{C.30})$$

It is the mean divergence of  $p(x)$  and  $q(x)$  to the average of the two distributions.







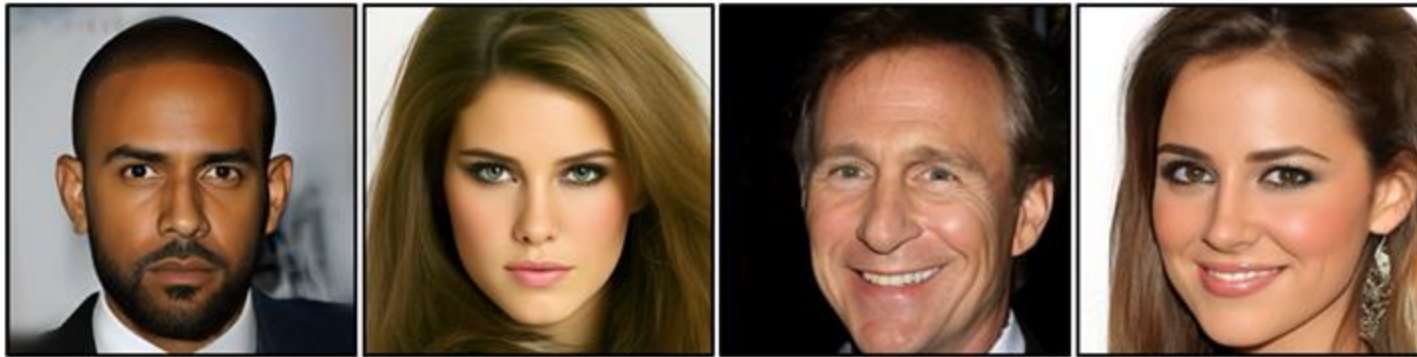
# Any Questions?



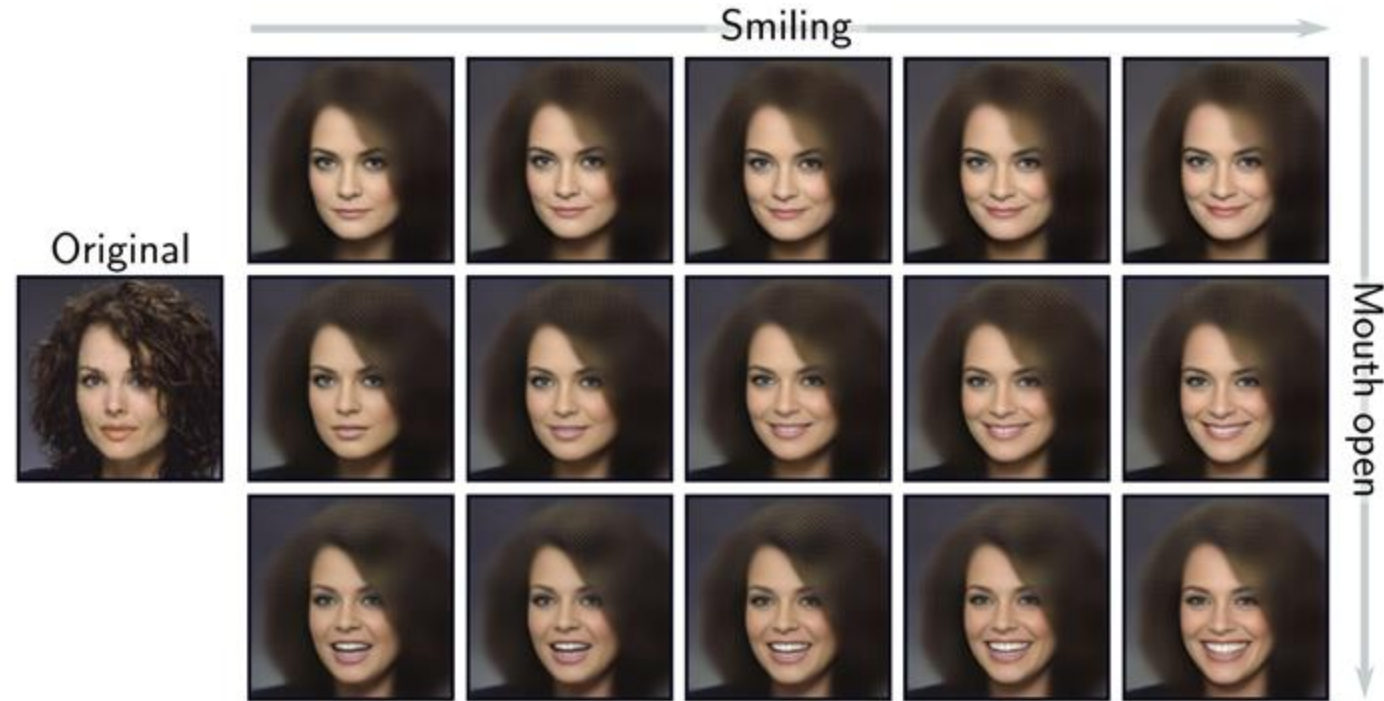
## Moving on

- Unsupervised Learning
- Latent Variables
- Probabilistic Generative Models
- Variational Autoencoders (VAEs)
- VAE math
- Examples of VAEs

# Generating high quality images

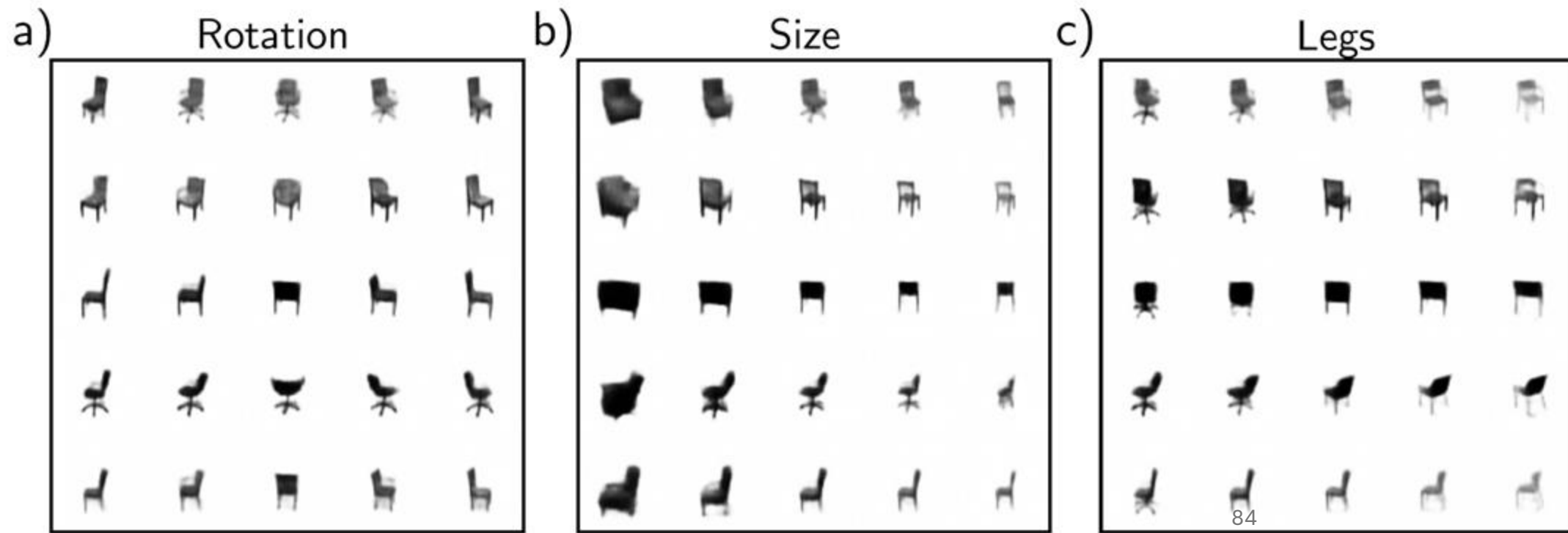


# Resynthesizing real data with changes



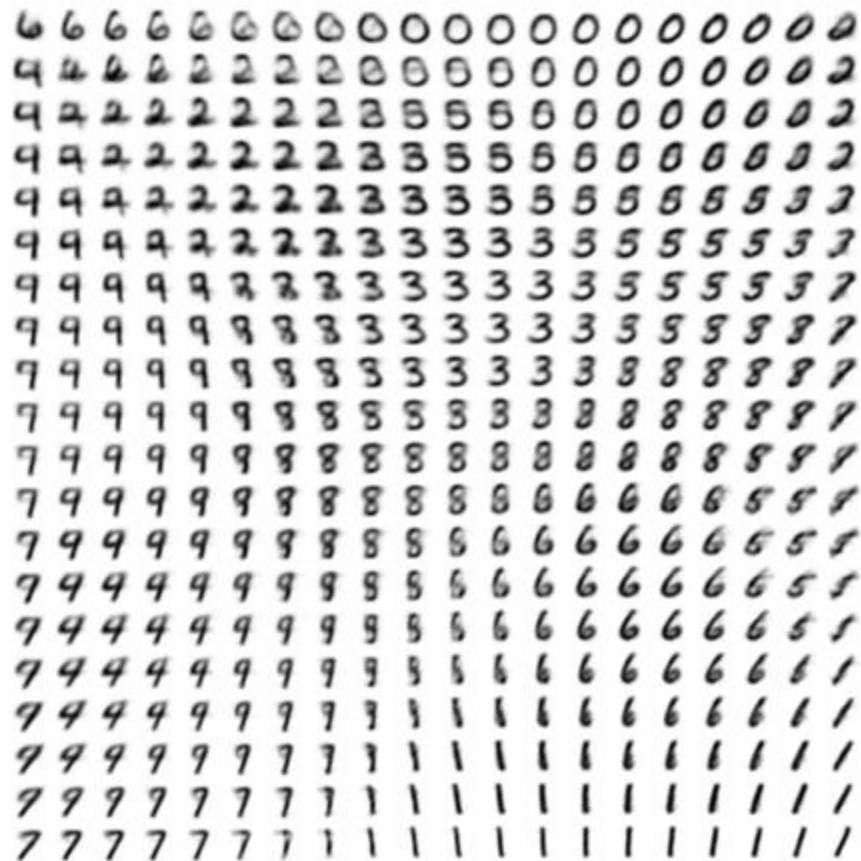
**Figure 17.13** Resynthesis. The original image on the left is projected into the latent space using the encoder, and the mean of the predicted Gaussian is chosen to represent the image. The center-left image in the grid is the reconstruction of the input. The other images are reconstructions after manipulating the latent space in directions representing smiling/neutral (horizontal) and mouth open/closed (vertical). Adapted from White (2016).

# Disentanglement of the latent space





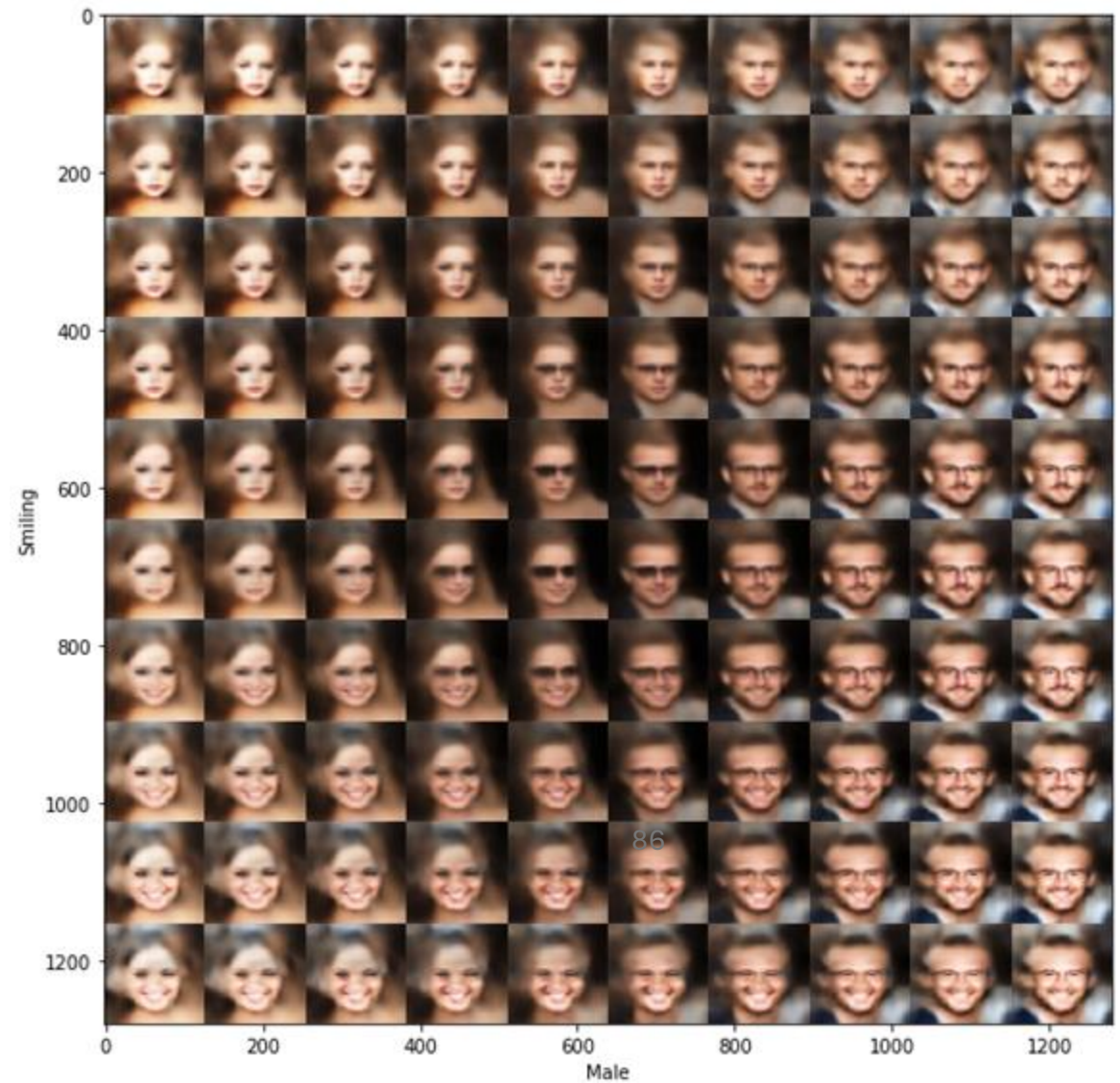
(a) Learned Frey Face manifold



(b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables  $\mathbf{z}$ . For each of these values  $\mathbf{z}$ , we plotted the corresponding generative  $p_{\theta}(\mathbf{x}|\mathbf{z})$  with the learned parameters  $\theta$ .

# Conditional VAEs



Example from <https://towardsdatascience.com/variational-autoencoders-vaes-for-dummies-step-by-step-tutorial-69e6d1c9d8e9>



# Debiasing

Capable of uncovering **underlying features** in a dataset



Homogeneous skin color, pose

VS



Diverse skin color, pose, illumination

How can we use this information to create fair and representative datasets?

# Outlier Detection

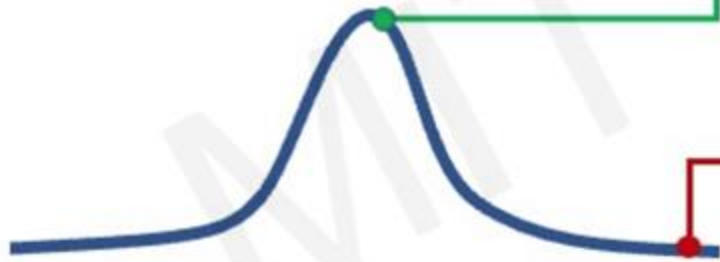
- **Problem:** How can we detect when we encounter something new or rare?
- **Strategy:** Leverage generative models, detect outliers in the distribution
- Use outliers during training to improve even more!

**95% of Driving Data:**

(1) sunny, (2) highway, (3) straight road



Detect outliers to avoid unpredictable behavior when training



Edge Cases



Harsh Weather



Pedestrians



# Any Questions?



**Moving on**

- Unsupervised Learning
- Latent Variables
- Probabilistic Generative Models
- Variational Autoencoders (VAEs)
- VAE math
- Examples of VAEs