

Generative Models I: FVBN & VAE

M. Soleymani

Sharif University of Technology

Spring 2026

Most slides are based on Fei Fei Li and colleagues lectures, cs231n and some slides from Bhiksha Raj, 11-785, CMU

Supervised Learning

- Data: (x, y)
 - x is data
 - y is label
- Goal: Learn a function to map $x \rightarrow y$
- Examples: Classification, regression, object detection, semantic segmentation, image captioning, machine translation, question answering, and etc.

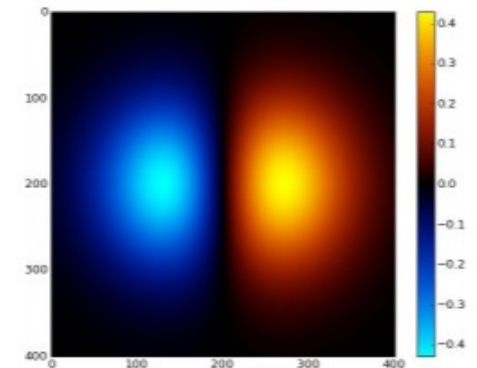
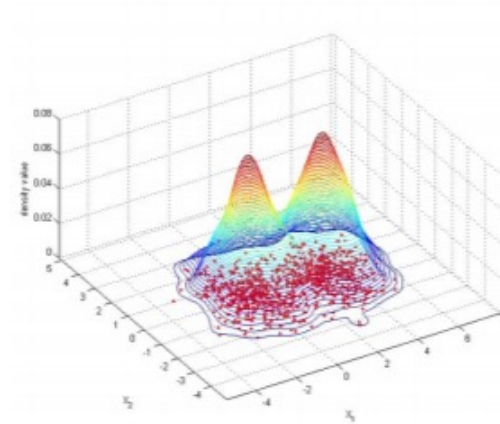
Density Estimation

- Data: x
 - Just data, no labels!
- Goal: Learn some underlying hidden structure of the data



Figure copyright Ian Goodfellow, 2016. Reproduced with permission.

1-d density estimation



2-d density estimation

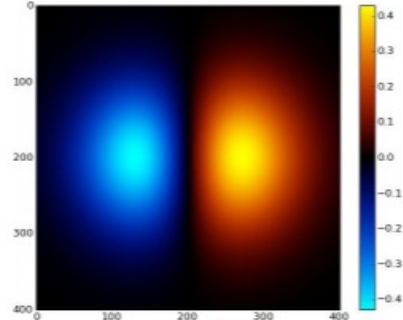
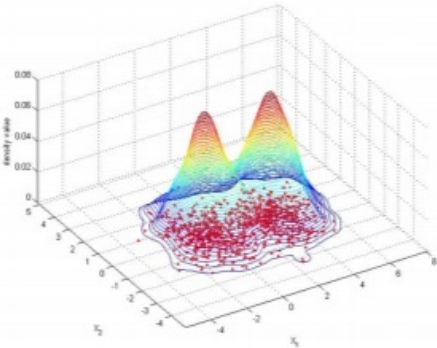
Unsupervised Learning: Generative models

Take samples from a distribution and learn the probability distribution underlying them



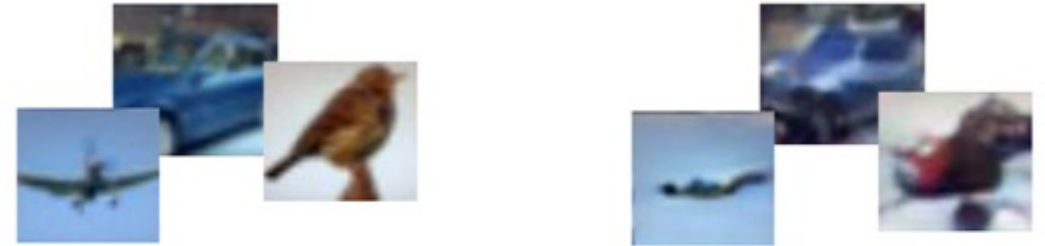
Figure copyright Ian Goodfellow, 2016. Reproduced with permission.

1-d density estimation



2-d density estimation

2-d density images [left](#) and [right](#)
are CC0 public domain



Training data $\sim p_{\text{data}}(x)$ Generated samples $\sim p_{\text{model}}(x)$

Want to learn $p_{\text{model}}(x)$ similar to $p_{\text{data}}(x)$

Generative Models

Given training data, generate new samples from same distribution



Training data $\sim p_{\text{data}}(x)$



Generated samples $\sim p_{\text{model}}(x)$

Want to learn $p_{\text{model}}(x)$ similar to $p_{\text{data}}(x)$

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

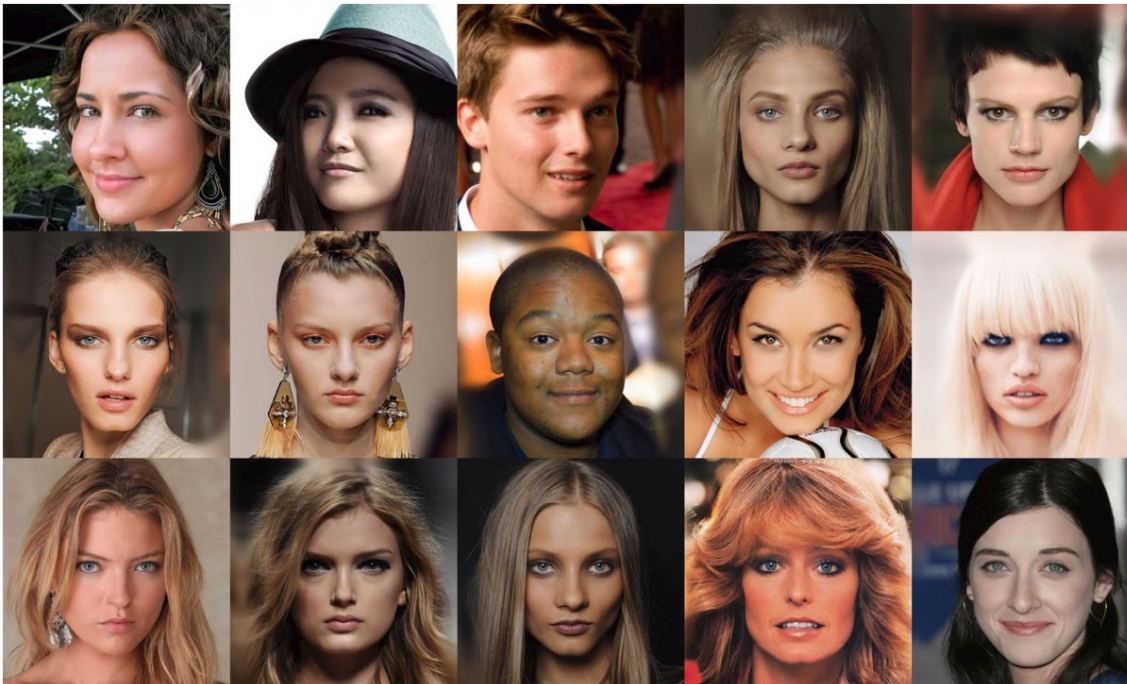
Generative Models

- Task: Given a dataset of images $\{X_1, X_2, \dots, X_N\}$ can we learn the distribution of X ?
- Given a dataset of training data, generative models find the distribution of data as $P(X)$
- However, we sometimes interested in only sampling from this distribution.
 - Instead of knowing $P(X)$ itself.

Generate samples given training examples

- Given training data, learn to generate samples from the same distribution as these ones.

some training samples



[CelebA dataset]

generated samples



Generative models

- Most of generative models are based on a latent space

$$P(x) = \int p(x|z)p(z)dz$$

- But, we first introduce Fully visible belief network (FVBN) or autoregressive generative models

$$p(x_1, \dots, x_T) = \prod_{t=1}^T P(x_t|x_1, \dots, x_{t-1})$$

Fully visible belief network

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

$$p(x_1, \dots, x_T) = \prod_{t=1}^T P(x_t | x_1, \dots, x_{t-1})$$

Likelihood of
image x

Probability of i 'th pixel value
given all previous pixels

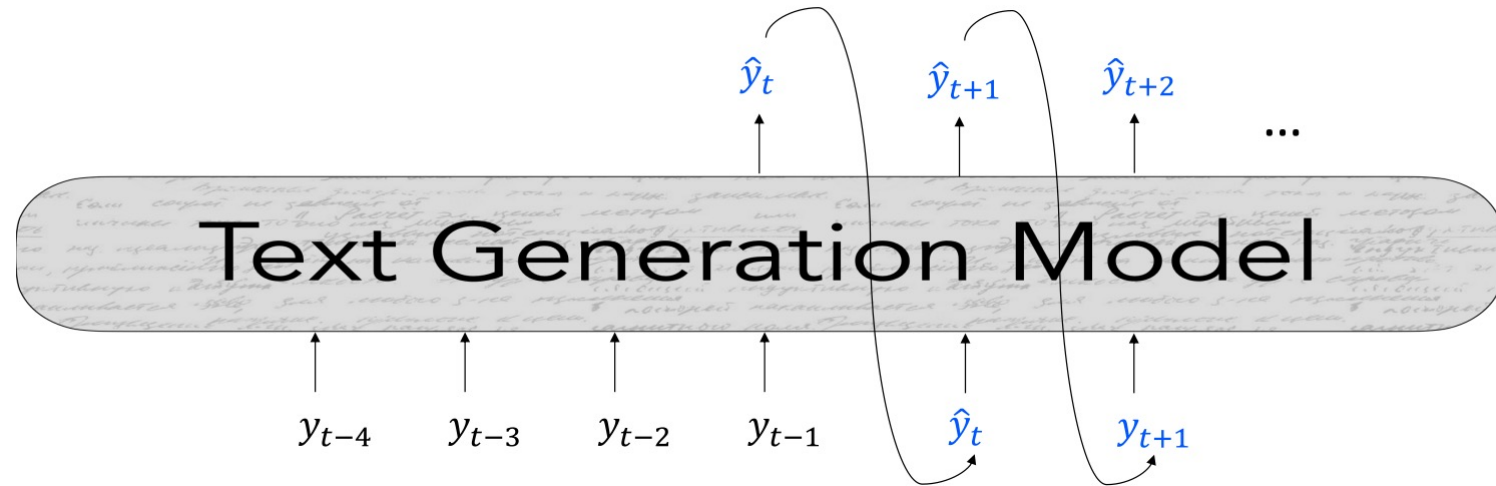
- Then maximize likelihood of training data

To model the above conditional distribution, it used a neural network

It needs ordering of variables

Autoregressive generative models

- Language modeling
 - Sequence of tokens from a finite vocab
 - We learn the distribution $P(w_{t+1}|w_1, \dots, w_t)$ by an RNN



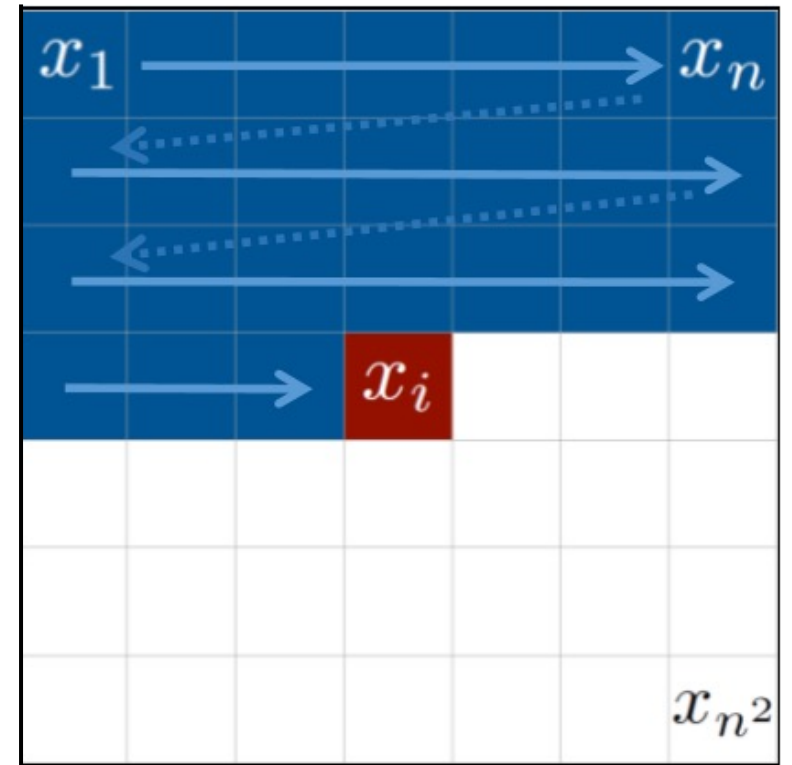
- PixelRNN and PixelCNN

Intuition

$$p(\mathbf{x}) = p(x_1, x_2, \dots, x_{n^2})$$

Chain rule:

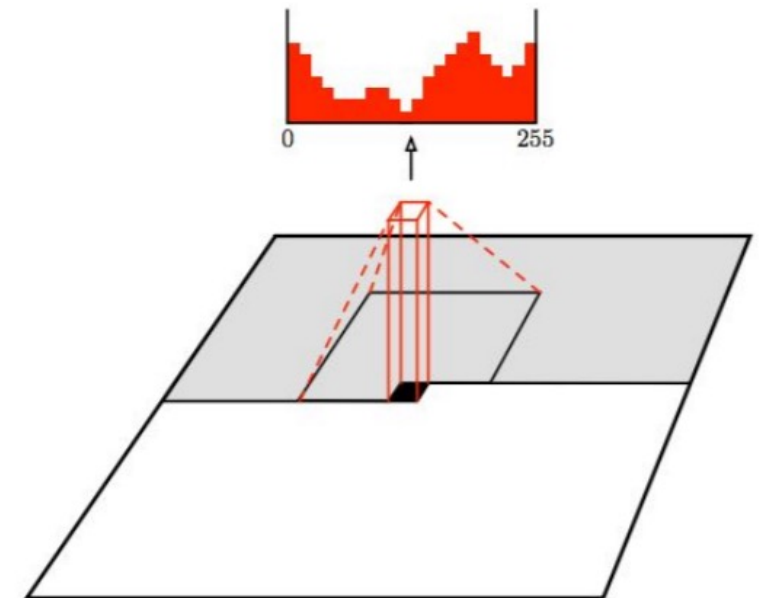
$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, \dots, x_{i-1})$$



PixelCNN

- PixelRNN has sequential process that is slow!
- Still generate image pixels starting from corner
- Dependency on previous pixels now modeled using a CNN over context region

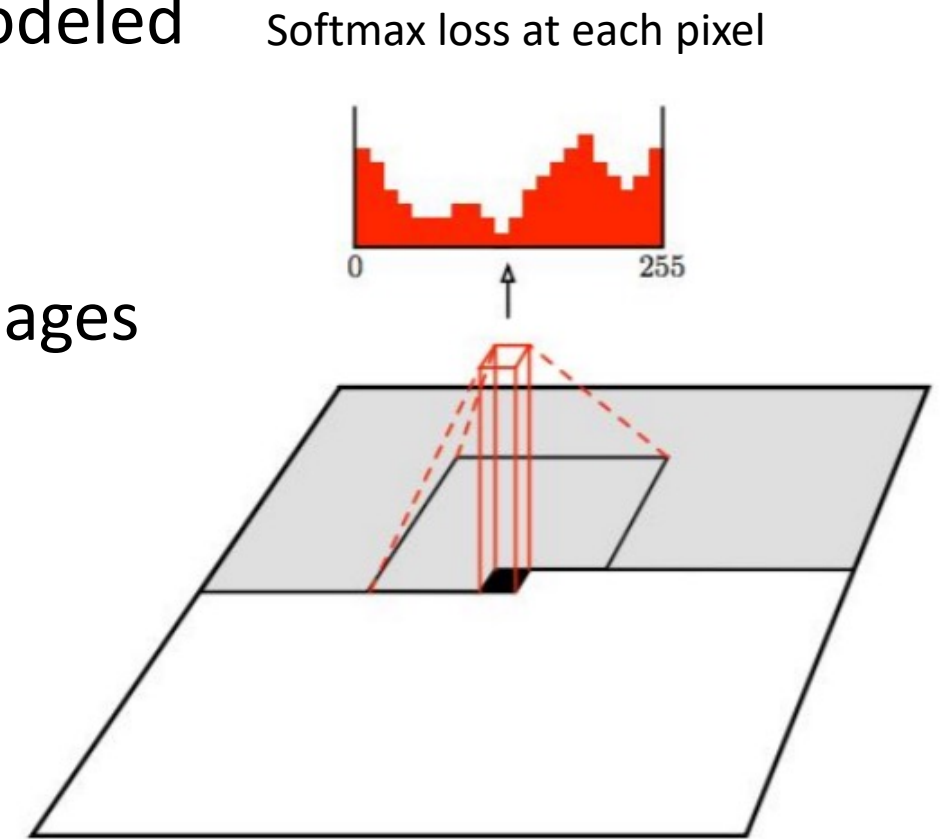
1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0



PixelCNN

- Still generate image pixels starting from corner
- Dependency on previous pixels now modeled using a CNN over context region
- Training: maximize likelihood of training images

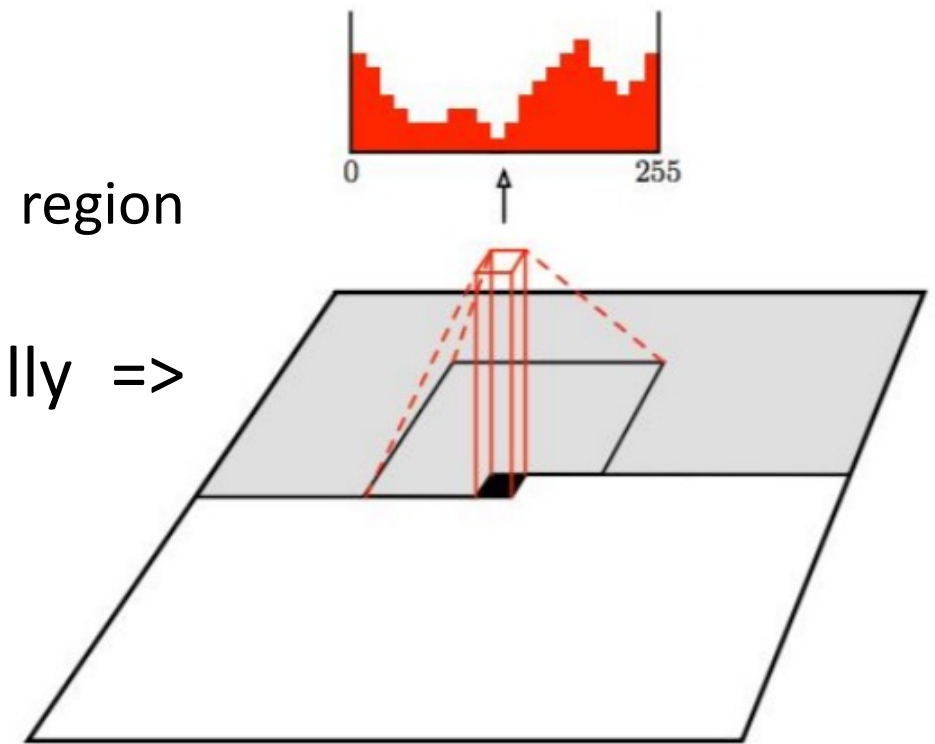
$$p(x) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1})$$



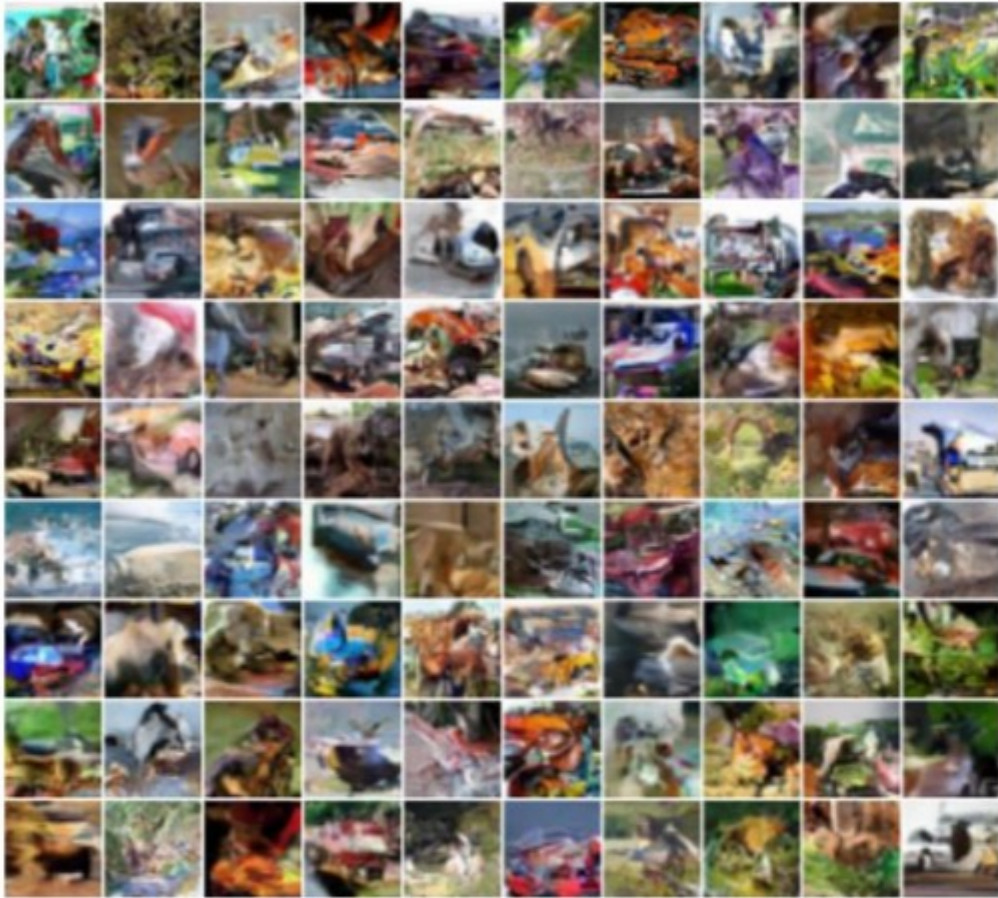
PixelCNN

- Still generate image pixels starting from corner
- Dependency on previous pixels now modeled using a CNN over context region
- Training is faster than PixelRNN
 - can parallelize convolutions since context region values known from training images
- Generation must still proceed sequentially => still slow

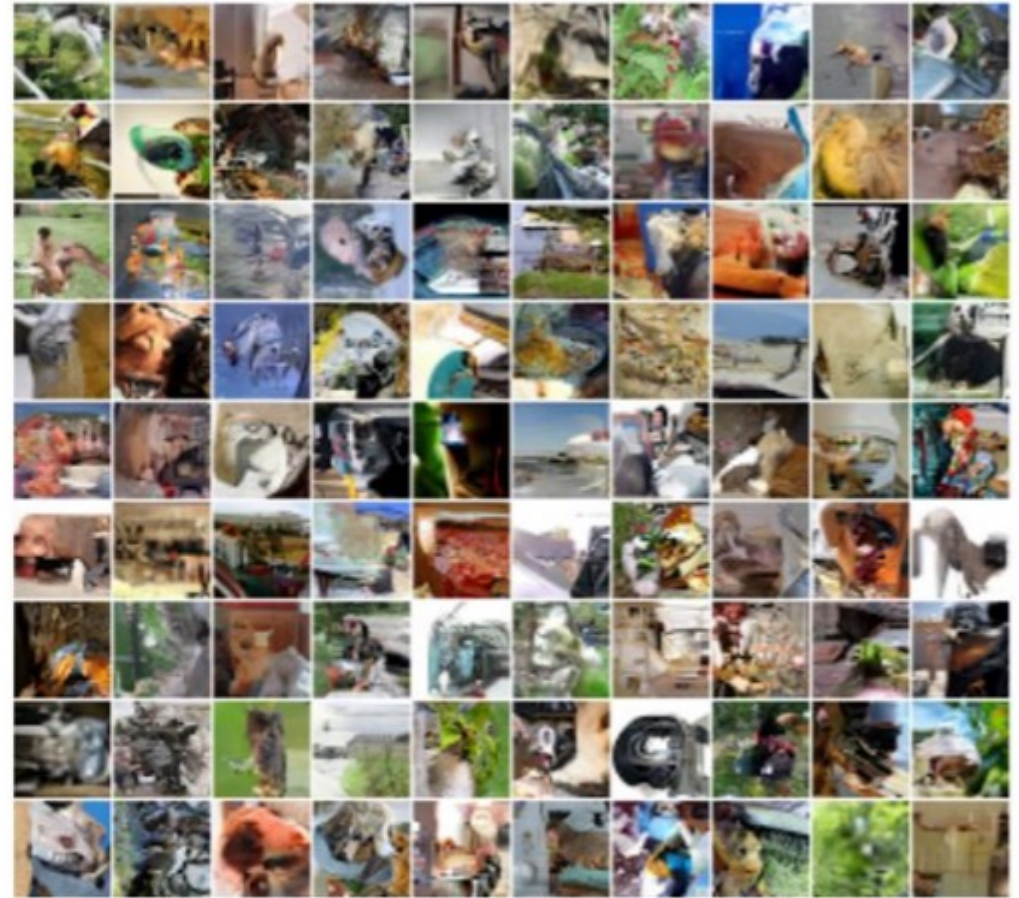
Softmax loss at each pixel



Generated Samples



32x32 CIFAR-10



32x32 ImageNet

PixelRNN and PixelCNN

Pros:

- Can explicitly compute likelihood $p(x)$
- Explicit likelihood of training data gives good evaluation metric
- Good samples

Con:

- Sequential generation => slow

Improving PixelCNN performance

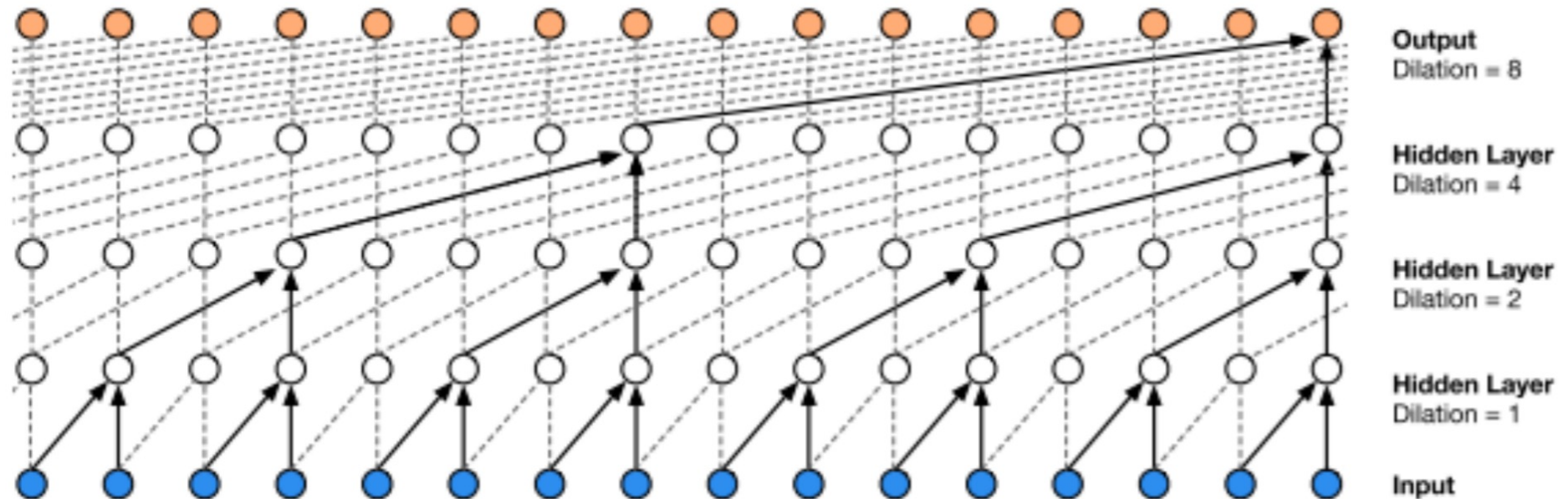
- Gated convolutional layers
- Short-cut connections
- Discretized logistic loss
- Multi-scale
- Training tricks
- Etc...

See

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

WaveNet

- A deep generative model of raw audio waveforms
- Can synthesize audio signals such as speech and music



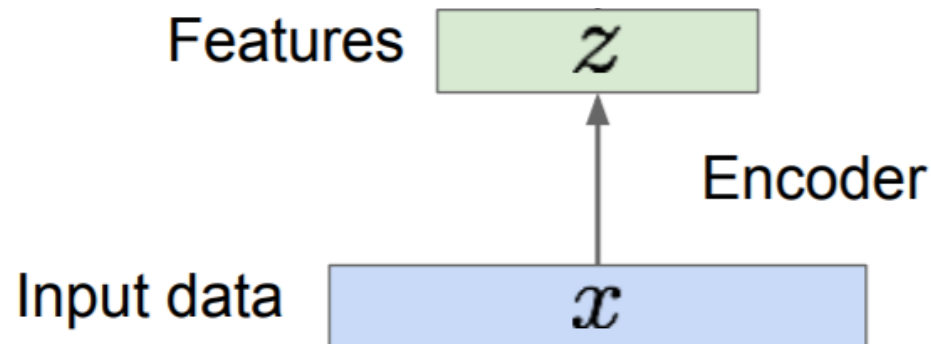
Latent variable models

What is a latent variable?

- Explanatory factors of data

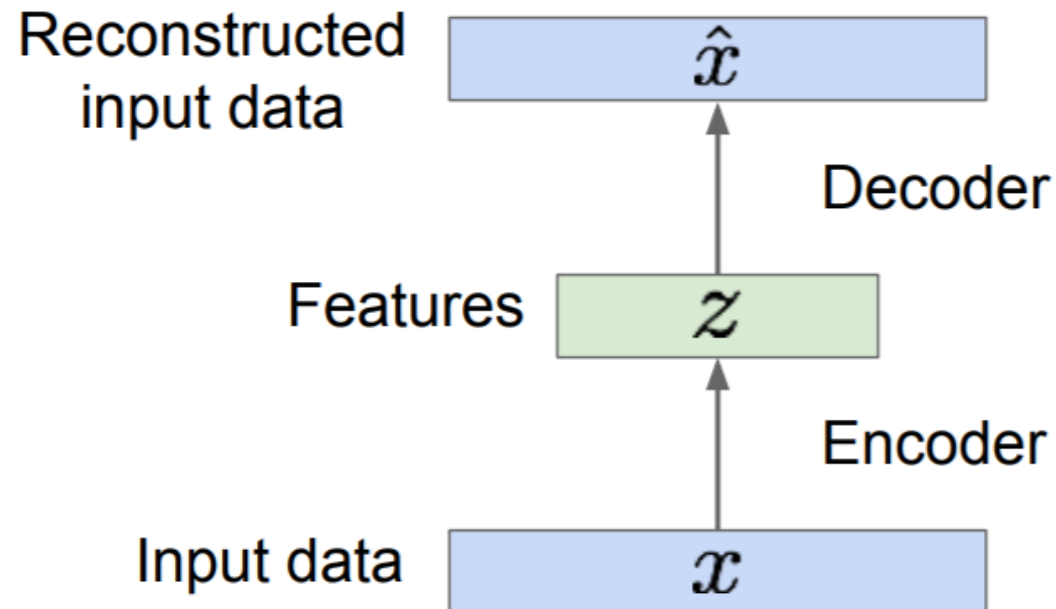
Dimensionality reduction: Autoencoders

- Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data
 - Train such that features can be used to reconstruct original data
 - “Autoencoding” - encoding itself



Autoencoders

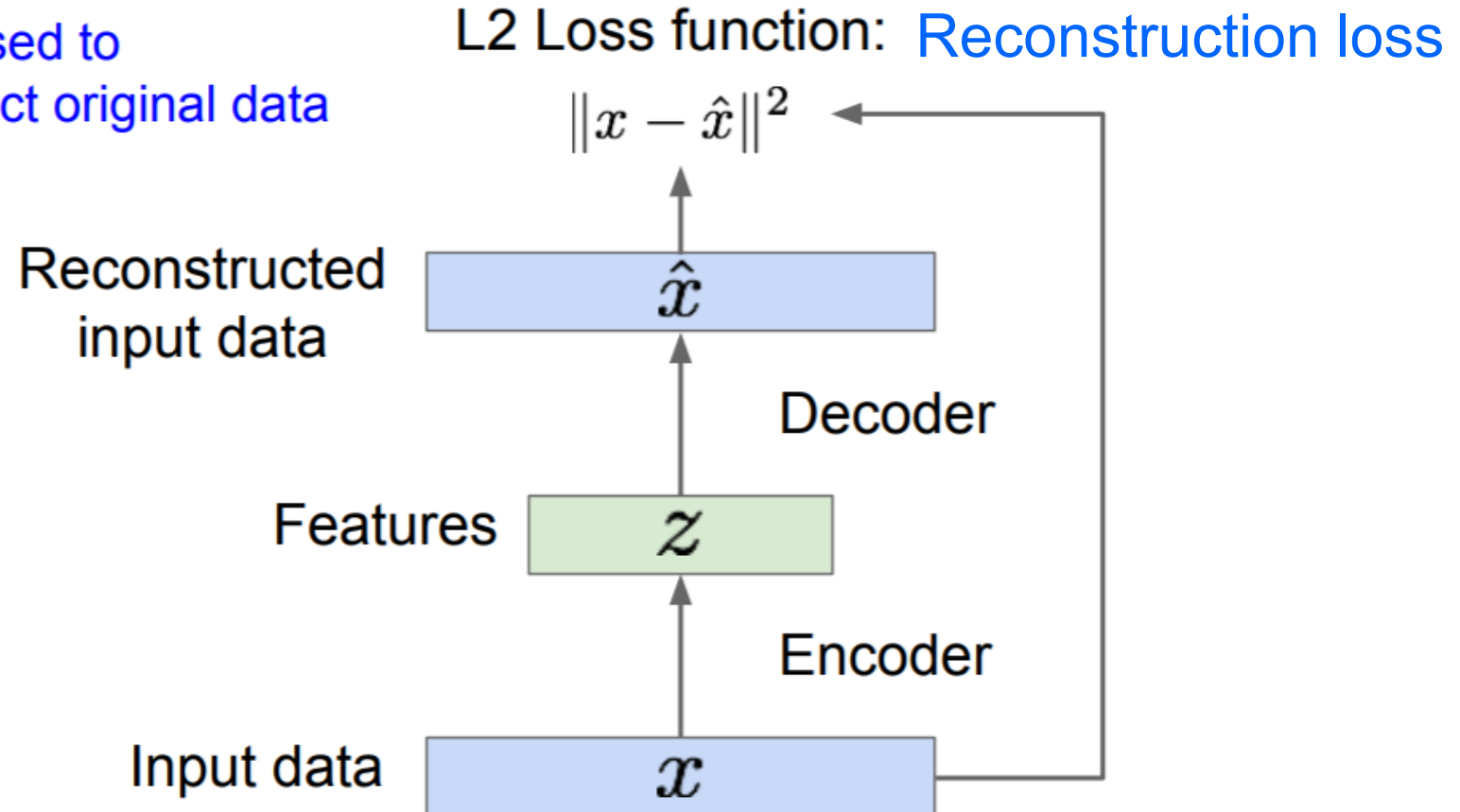
- Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data
 - Train such that features can be used to reconstruct original data
 - “Autoencoding” - encoding itself



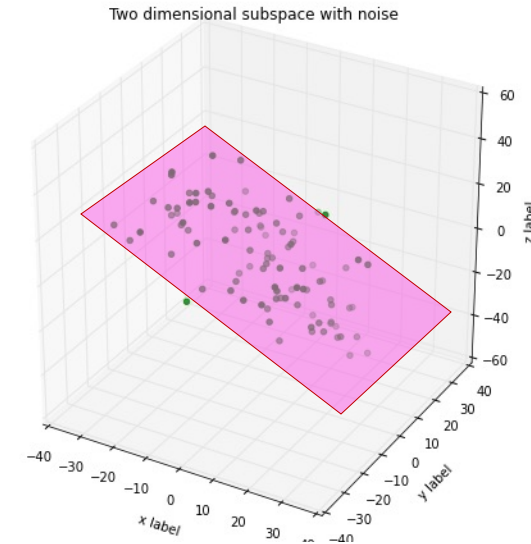
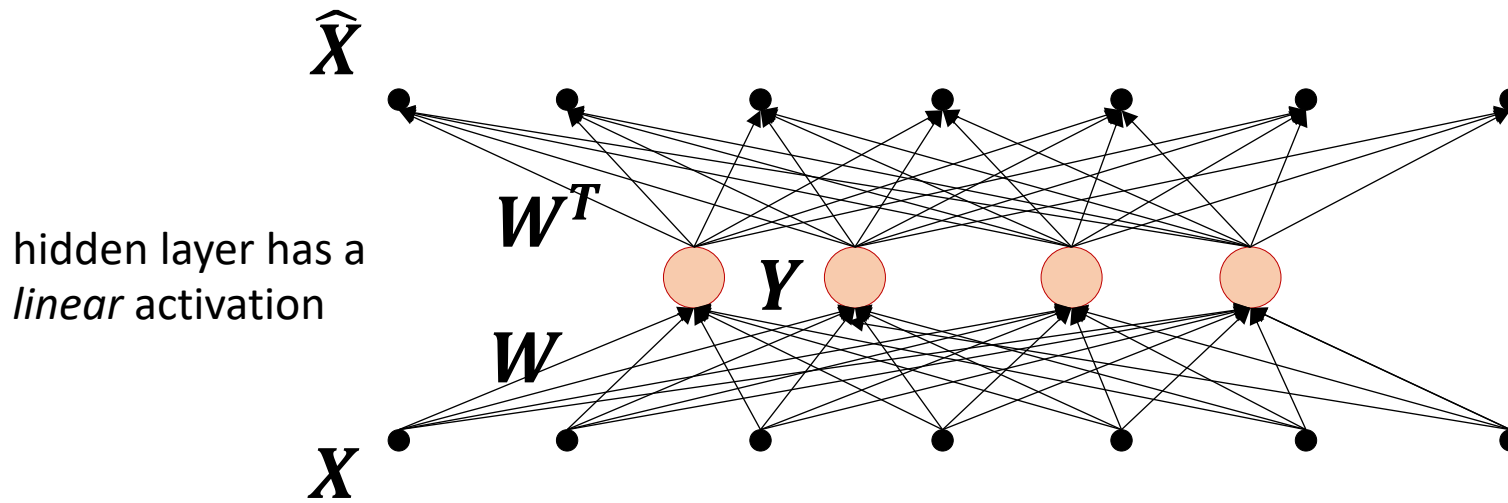
Autoencoders

- **Encoder:** The “Analysis” net which computes the hidden representation
- **Decoder:** The “Synthesis” which recomposes the data from the hidden representation

Train such that features
can be used to
reconstruct original data



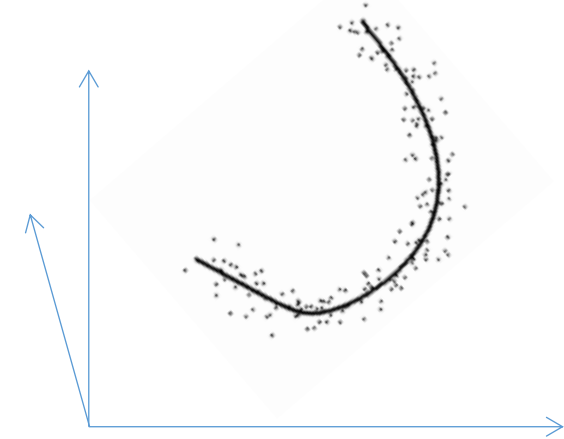
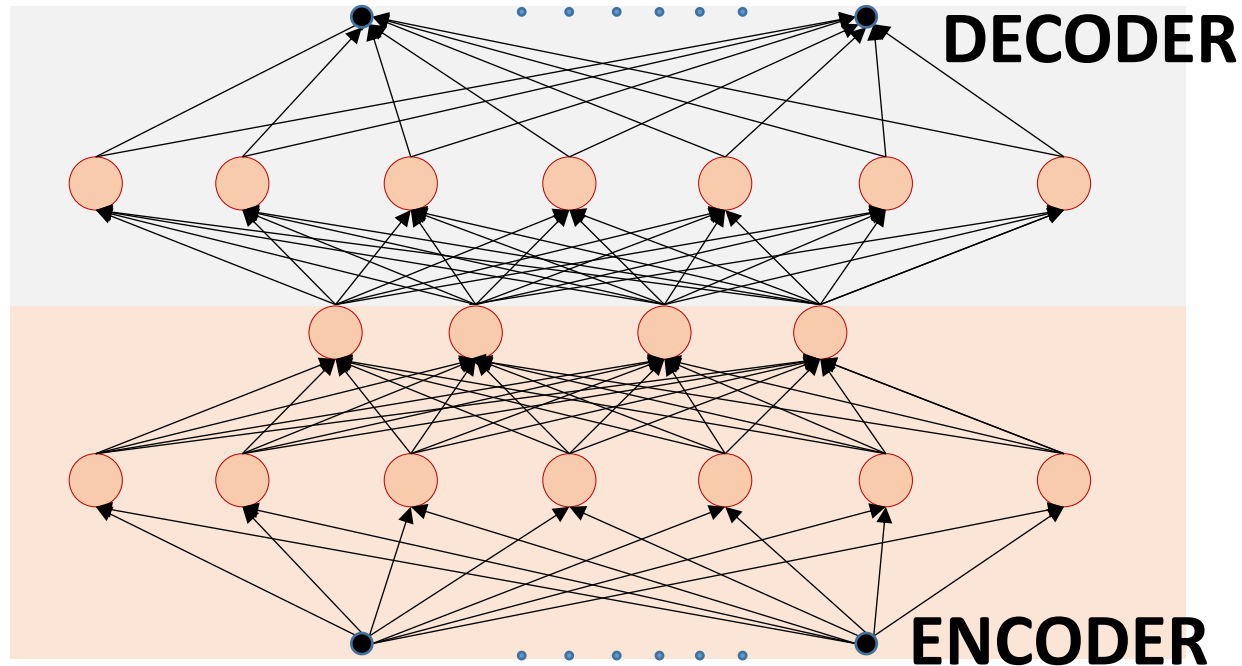
AEs without a non-linearity



$$\boxed{Y = WX} \quad \boxed{\hat{X} = W^T Y} \quad E = \|X - W^T W X\|^2 \quad \text{Find } W \text{ to minimize Avg}[E]$$

- This is similar to PCA
 - The output of the hidden layer will be in the principal subspace
 - Even if the recomposition weights are different from the “analysis” weights

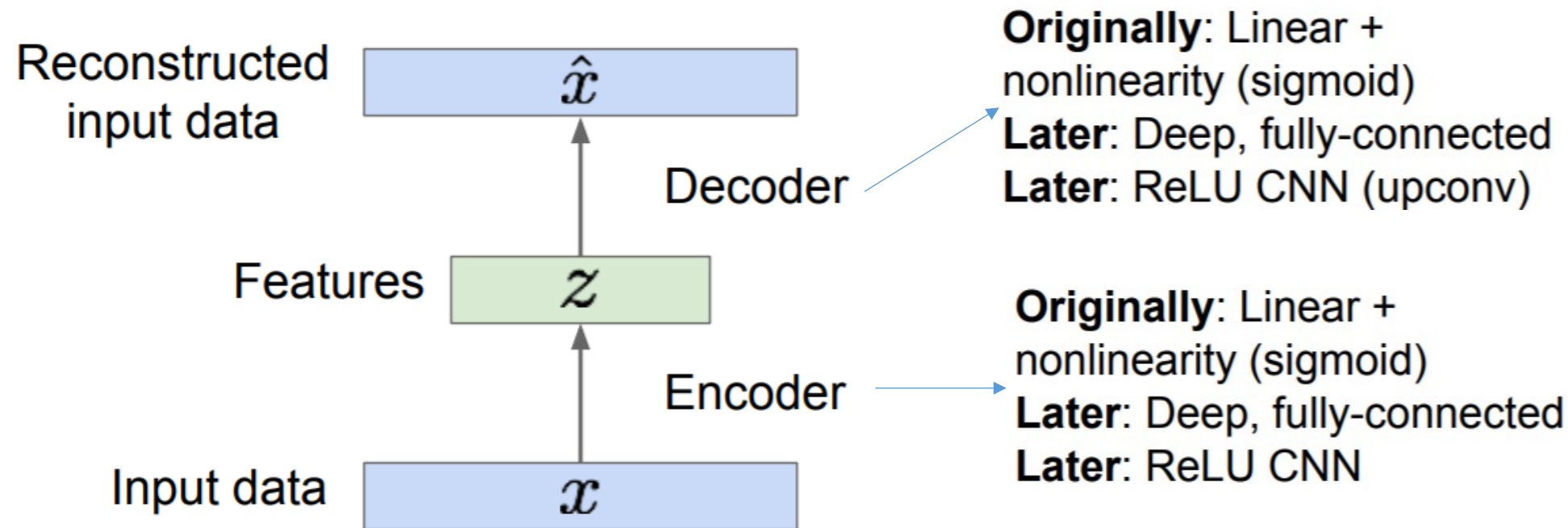
“Deep” autoencoders



- When the hidden representation is of lower dimensionality than the input, often called a “**bottleneck**” network
 - Nonlinear PCA
 - Learns the manifold for the data
 - If properly trained

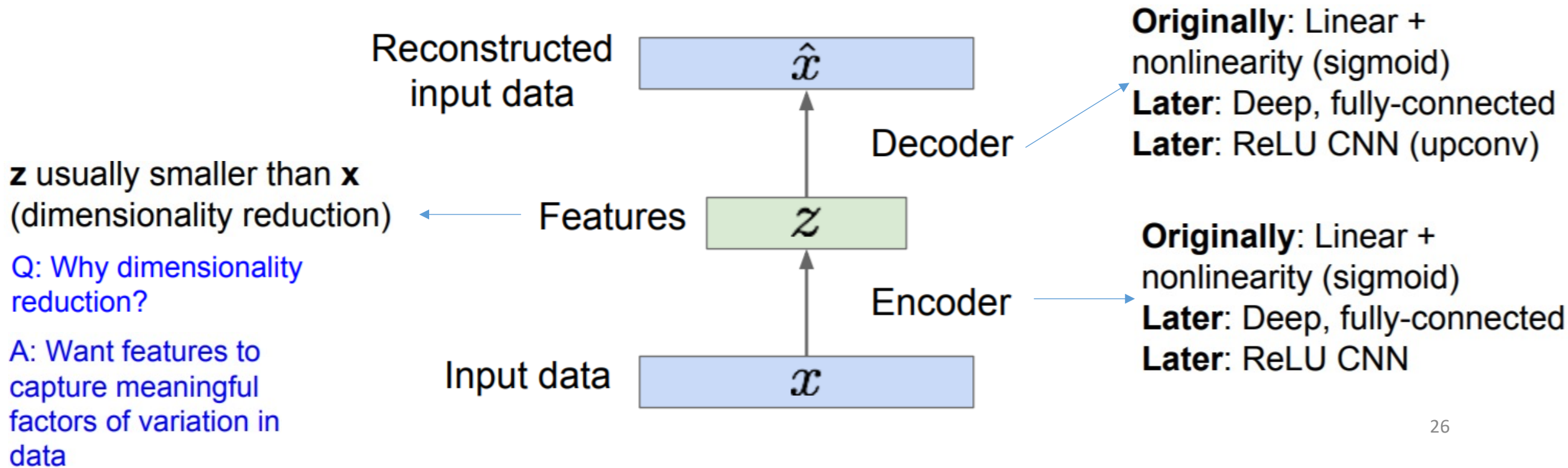
Autoencoders

- Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data
 - Train such that features can be used to reconstruct original data
 - “Autoencoding” - encoding itself



Autoencoders

- Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data
 - Train such that features can be used to reconstruct original data
 - “Autoencoding” - encoding itself

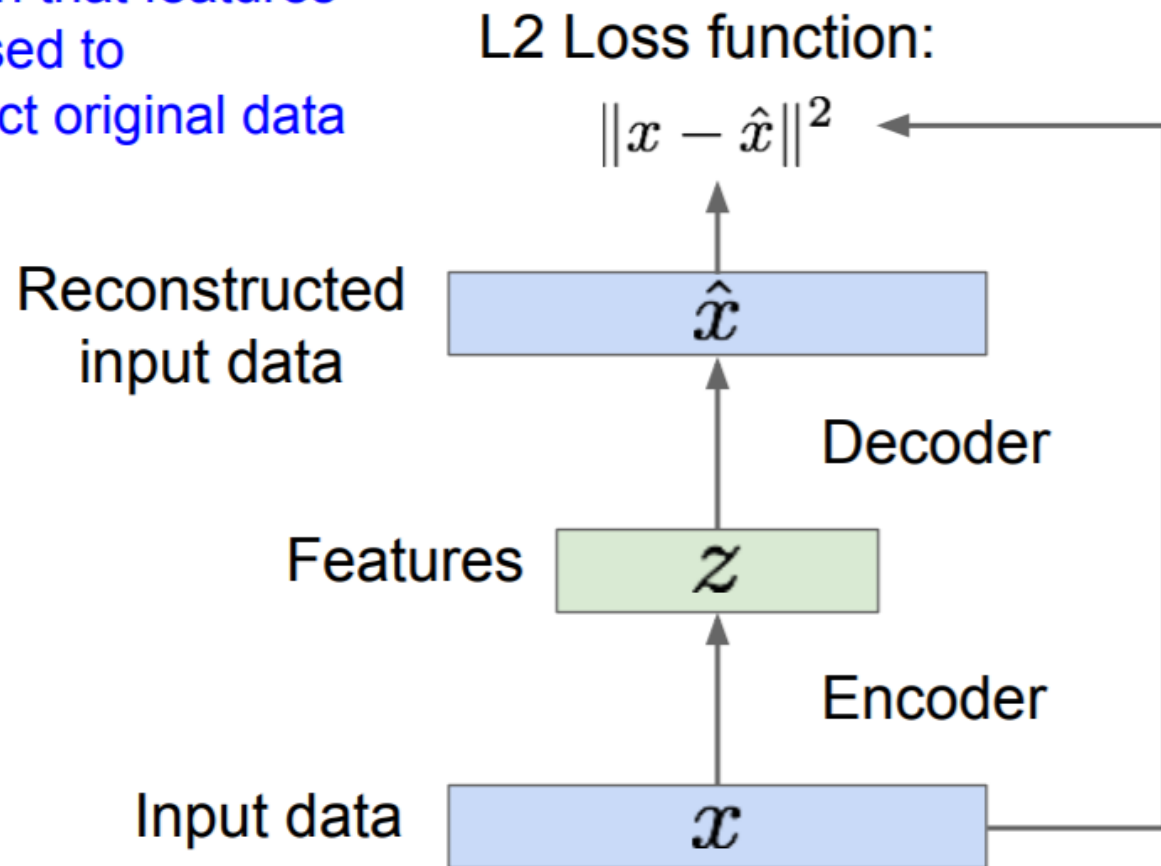


Autoencoders: Pretraining

- Pre-train the encoder with (unlabeled) data

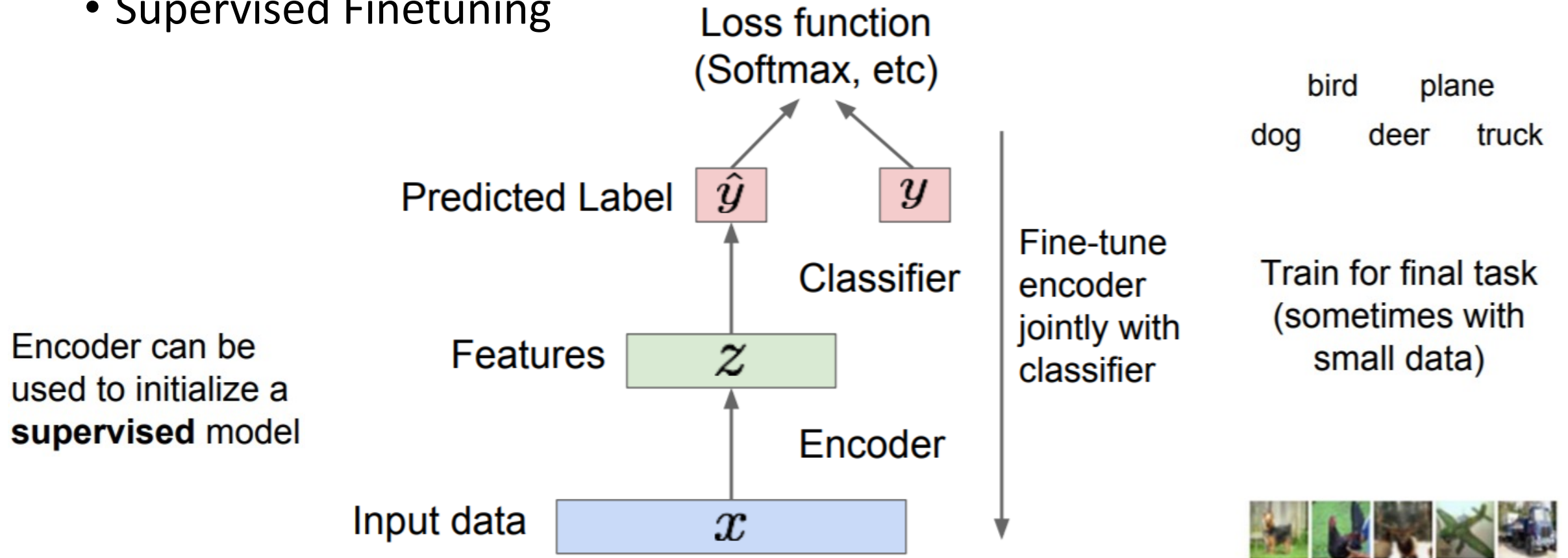
Train such that features
can be used to
reconstruct original data

Doesn't use labels!

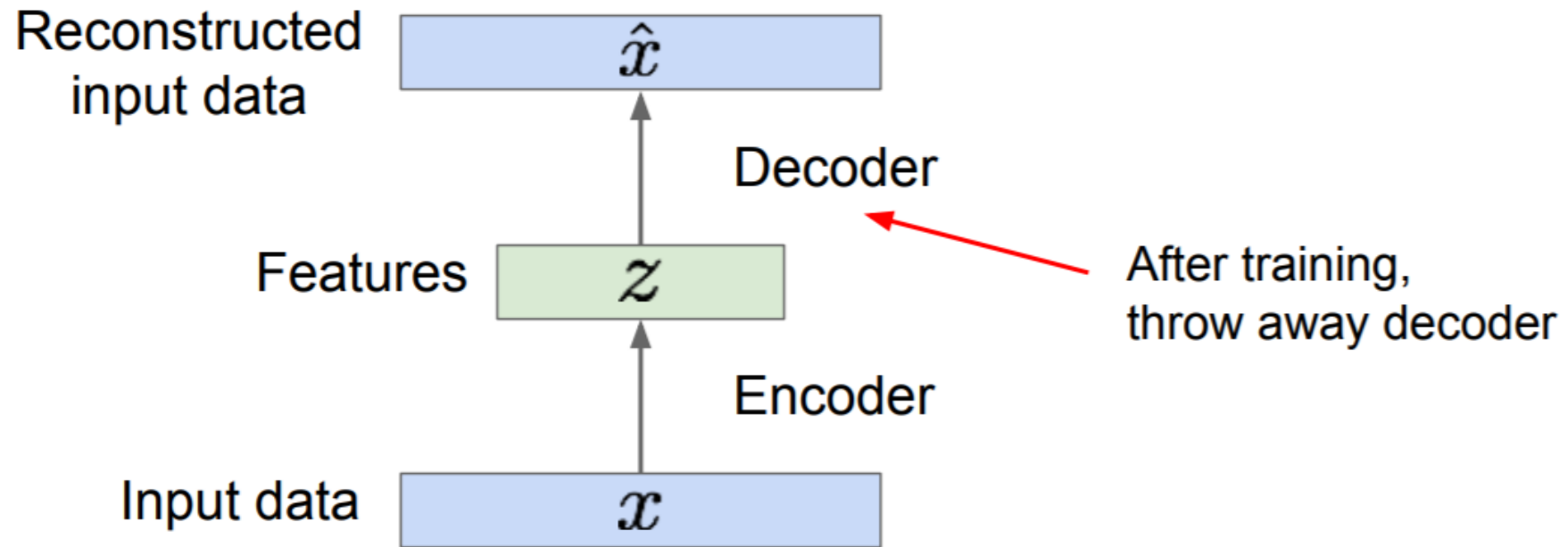


Autoencoders as pretrained models

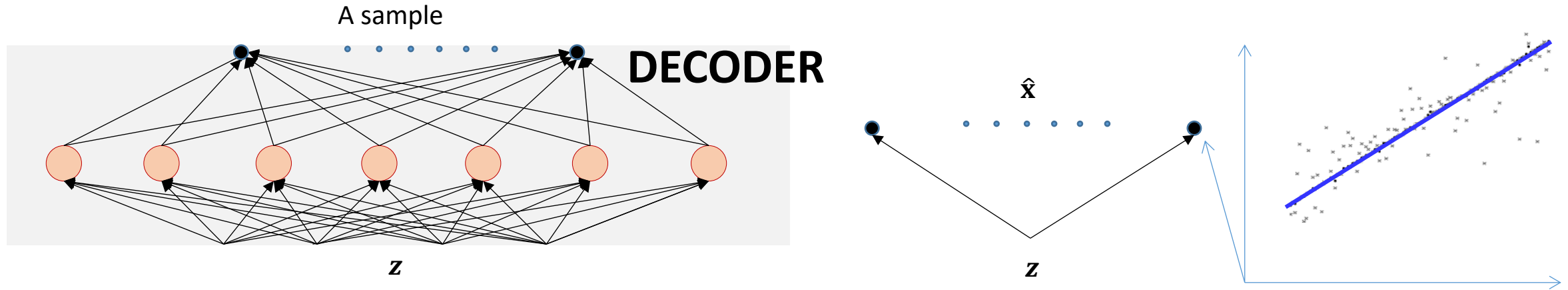
- AEs can learn features to initialize a supervised model
- Supervised Finetuning



Autoencoders

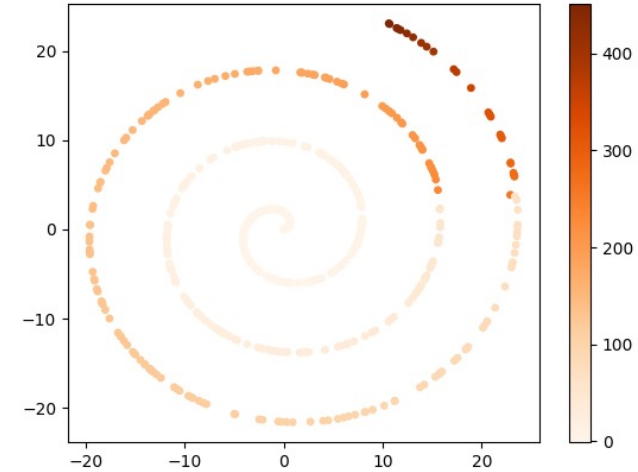
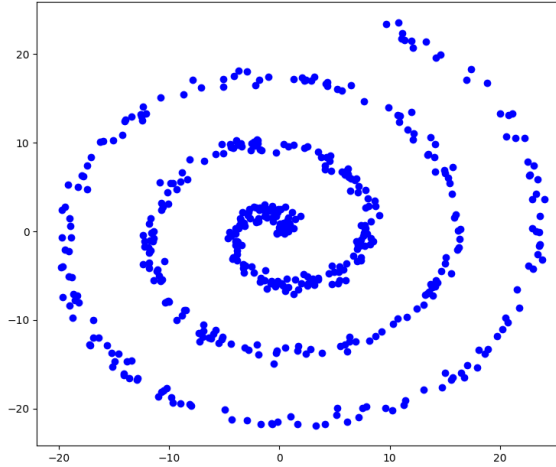
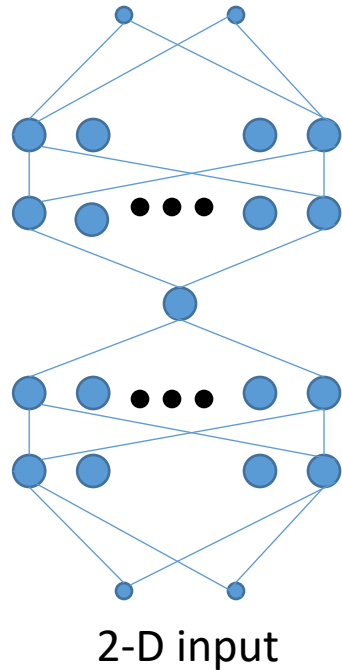


Can the decoder of AE be used for data generation?



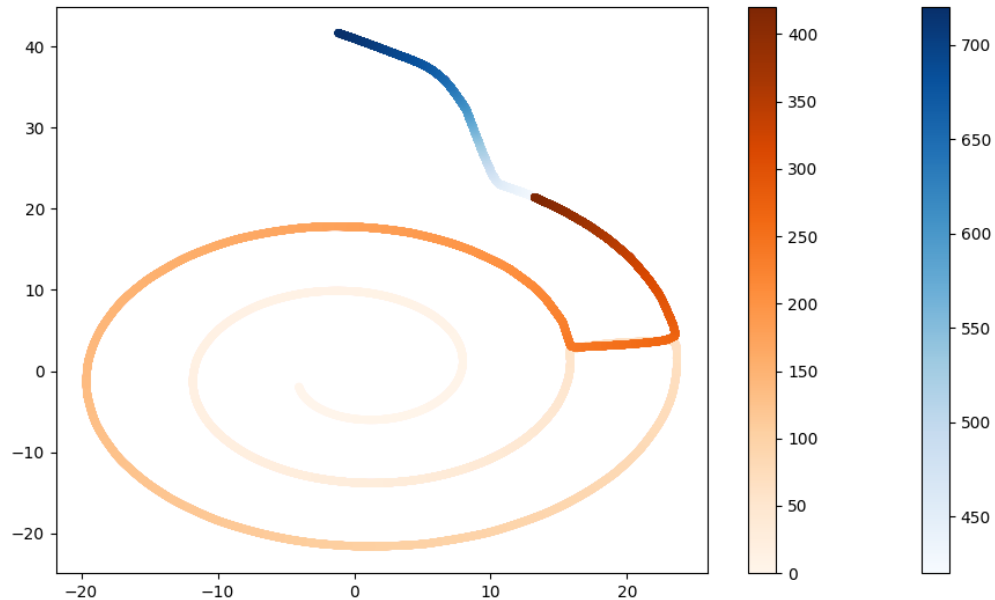
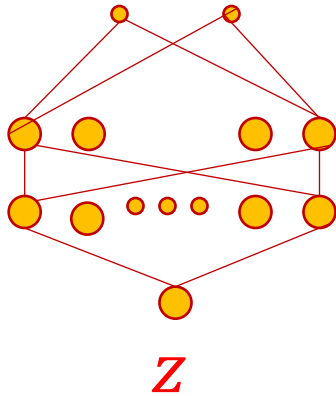
- Varying the hidden value will move along this non-linear manifold
 - Simply varying the hidden representation will result in an output that lies along the major axis
- The decoder can only generate data on the manifold that the training data lie on
- This also makes it an excellent “generator” of the distribution of the training data
 - Any values fed to the (hidden) input to the decoder will produce data similar to the training data

Some examples



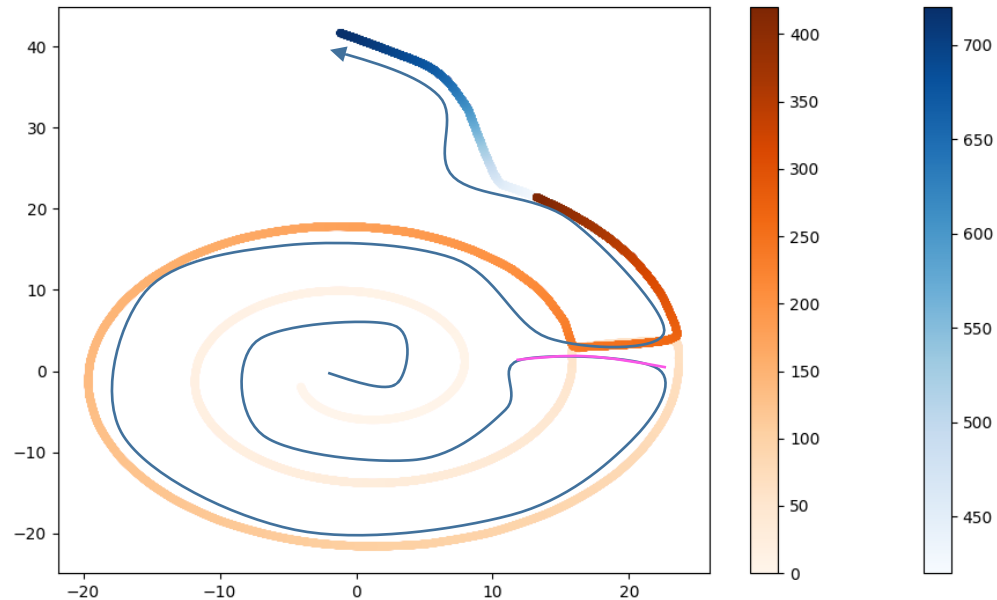
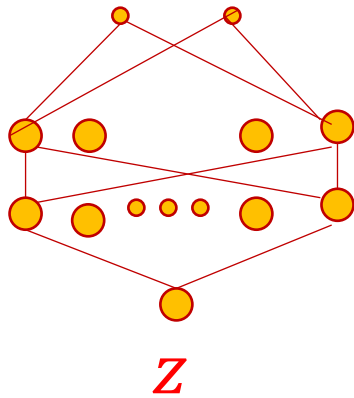
- Encoder and decoder have 2 hidden layers of 100 neurons, but hidden representation has one hidden unit
- Model seems to learn underlying helix structure

The learned manifold



- Not a “clean” function even in range of training points (Red)
 - Color shows value of z
 - z does not vary smoothly along the curve, but bounces back and forth
 - Learns manifold structure (bar) that is not represented in training data
- Does not generalize outside the range of training points (Blue)
 - Extending the range towards the center of the spiral resulted in decoded values outside the page!

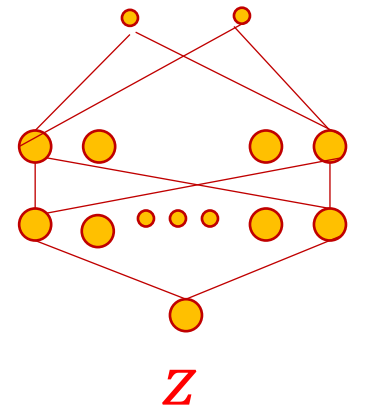
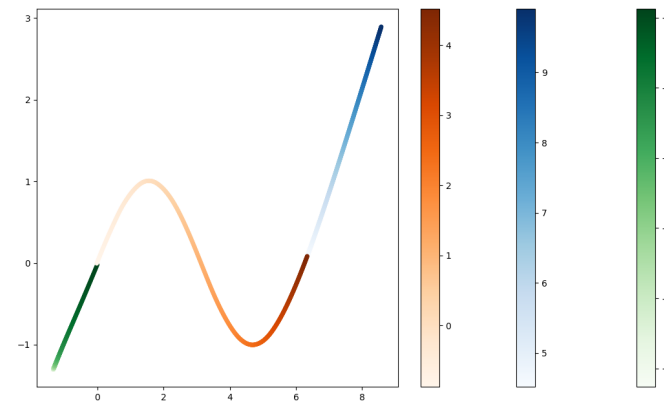
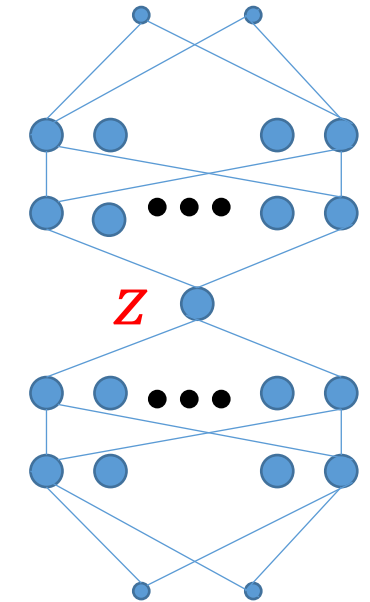
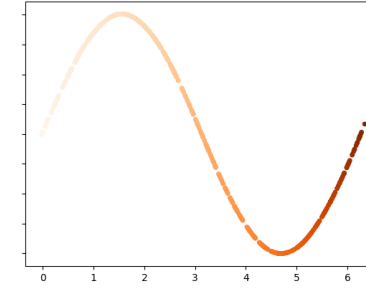
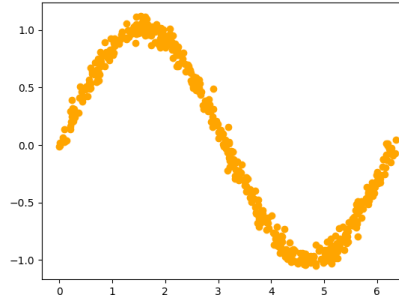
The learned manifold



- Not a “clean” function even in range of training points (Red)
 - Color shows value of z
 - z does not vary smoothly along the curve, but bounces back and forth
 - Learns manifold structure (bar) that is not represented in training data
- Does not generalize outside the range of training points (Blue)
 - Extending the range towards the center of the spiral resulted in decoded values outside the page!

Another example

- Learning to reconstruct a sinusoid
 - Input (left): data on a spiral manifold
 - Output (right): Decoded data
- AE seems to “learn” the underlying curved manifold
- The model is specific to the training data.
 - Varying the hidden layer value only generates data along the learned manifold
 - May be poorly learned



Latent space

- Neural Networks like Autoencoders can be used for representation generation
- Can auto-encoders find latent space?
 - Bottleneck hidden layer forces to learn a compressed representation
 - Reconstruction loss forces the hidden layer preserve as much as information about the data as possible
- Only generates data along the learned manifold
 - This may be **poorly learned**
 - Any and every input will result in an output along the learned manifold

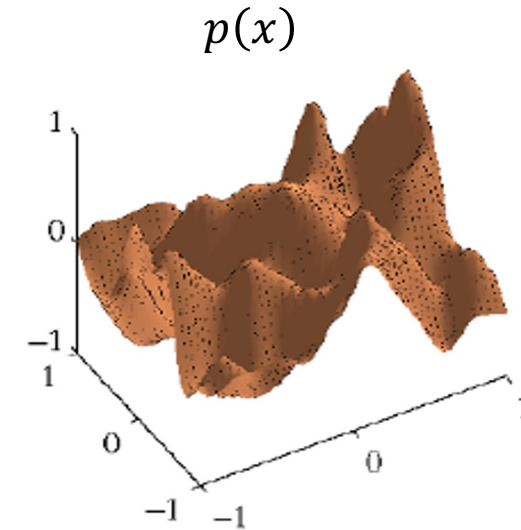
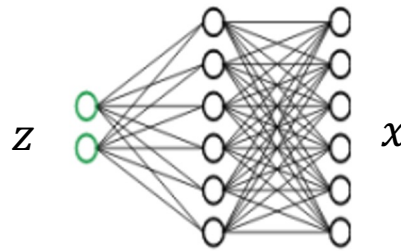
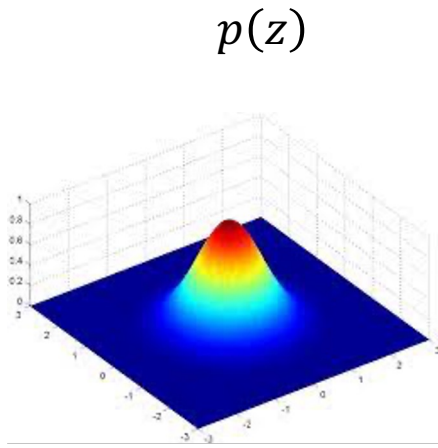
Latent space properties

- For the generation purpose, we need the latent space shows the following properties:
 - **Continuity**: Similar point in the latent space will be similar after decoding
 - **Completeness**: Samples of the latent space lead to meaningful content after decoding

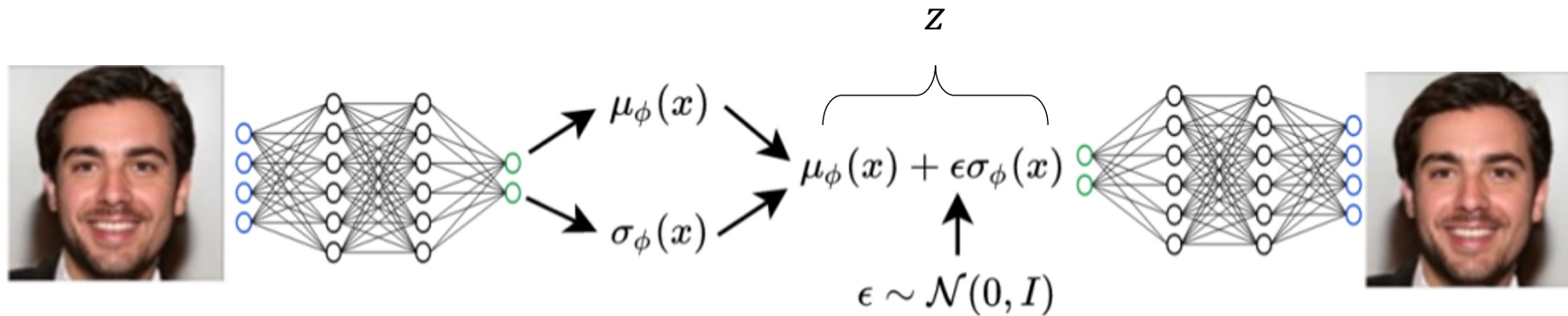
Latent variables

We define a simple distribution on the latent space, i.e., $p(z)$ and a decoder network $p_{\theta}(x|z)$ to map the latent variable to the data space

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

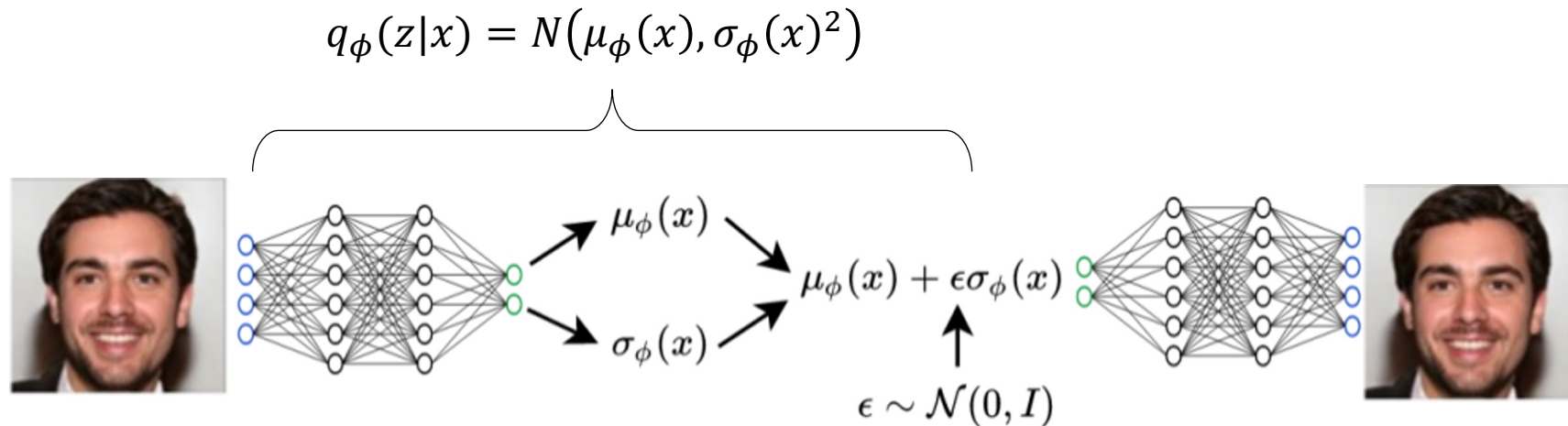


Variational Auto Encoder (VAE)



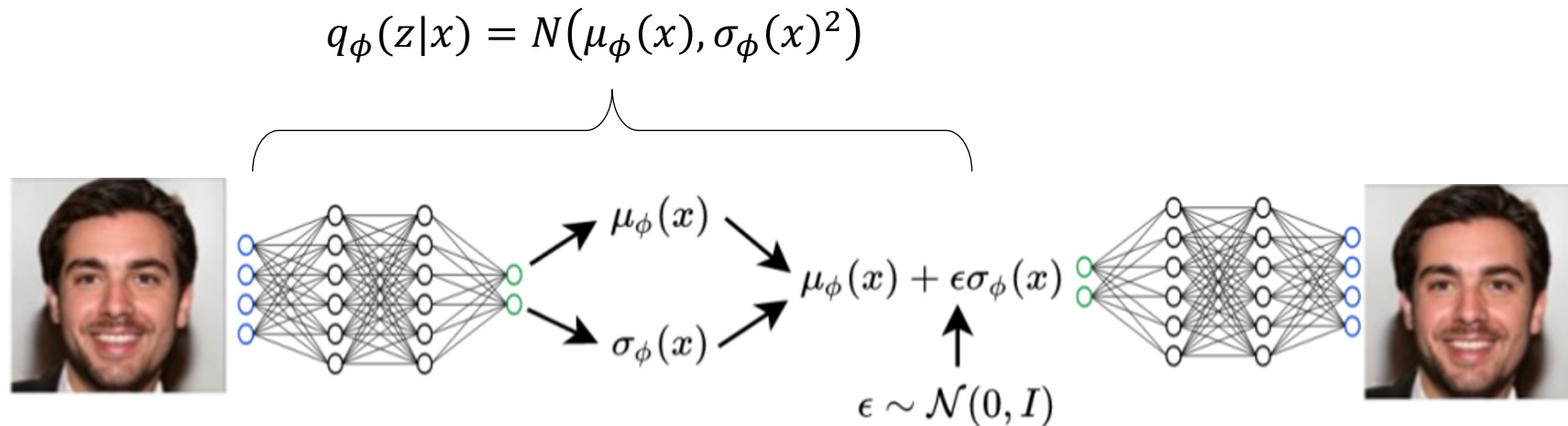
VAE loss: $\mathcal{L}(x; \theta, \phi) = \text{reconstruction loss} + \text{regularization term}$

Variational Auto Encoder (VAE)



VAE loss: $\mathcal{L}(x; \theta, \phi) = \text{reconstruction loss} + \text{regularization term}$

Variational Auto Encoder (VAE)



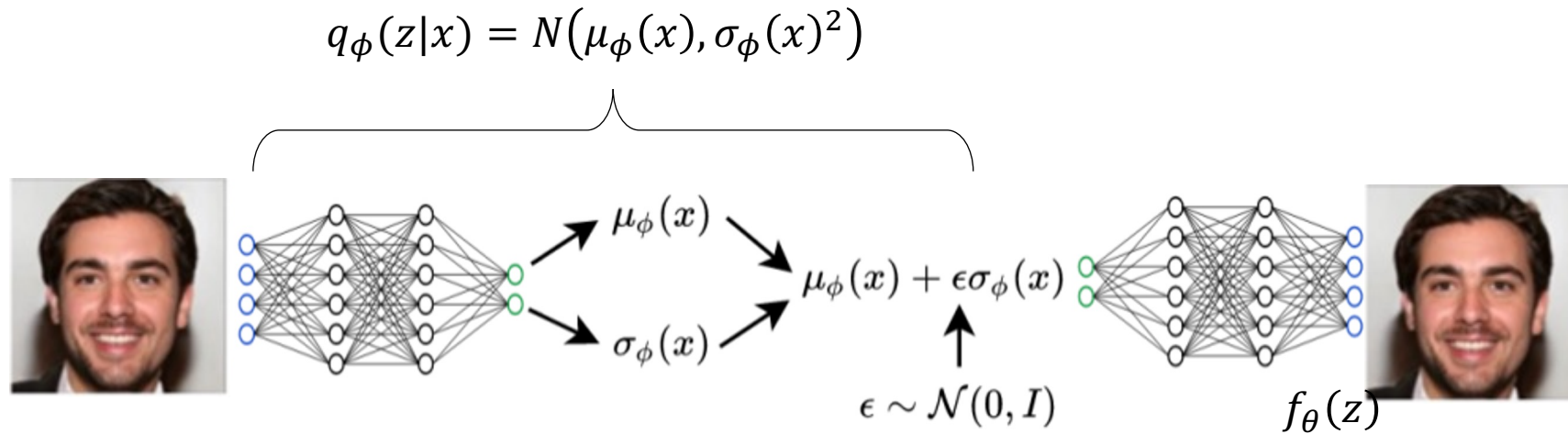
VAE loss: $\mathcal{L}(x, \theta, \phi) =$ reconstruction loss + regularization term

$$\|x^{(i)} - \hat{x}^{(i)}\|^2$$

$$KL(q_{\phi}(z|x^{(i)})||p(z))$$

Fixed prior on the latent distribution

Variational Auto Encoder (VAE)



VAE loss: $\mathcal{L}(x, \theta, \phi) = \text{reconstruction loss} + \text{regularization term}$

$$\|x^{(i)} - \hat{x}^{(i)}\|^2$$

$$KL(q_\phi(z|x^{(i)}) || N(0, I))$$

$$f_\theta(z) = f_\theta(\mu_\phi(x^{(i)}) + \epsilon\sigma_\phi(x^{(i)}))$$

Common choice of prior
on the latent distribution

KL between two Gaussians

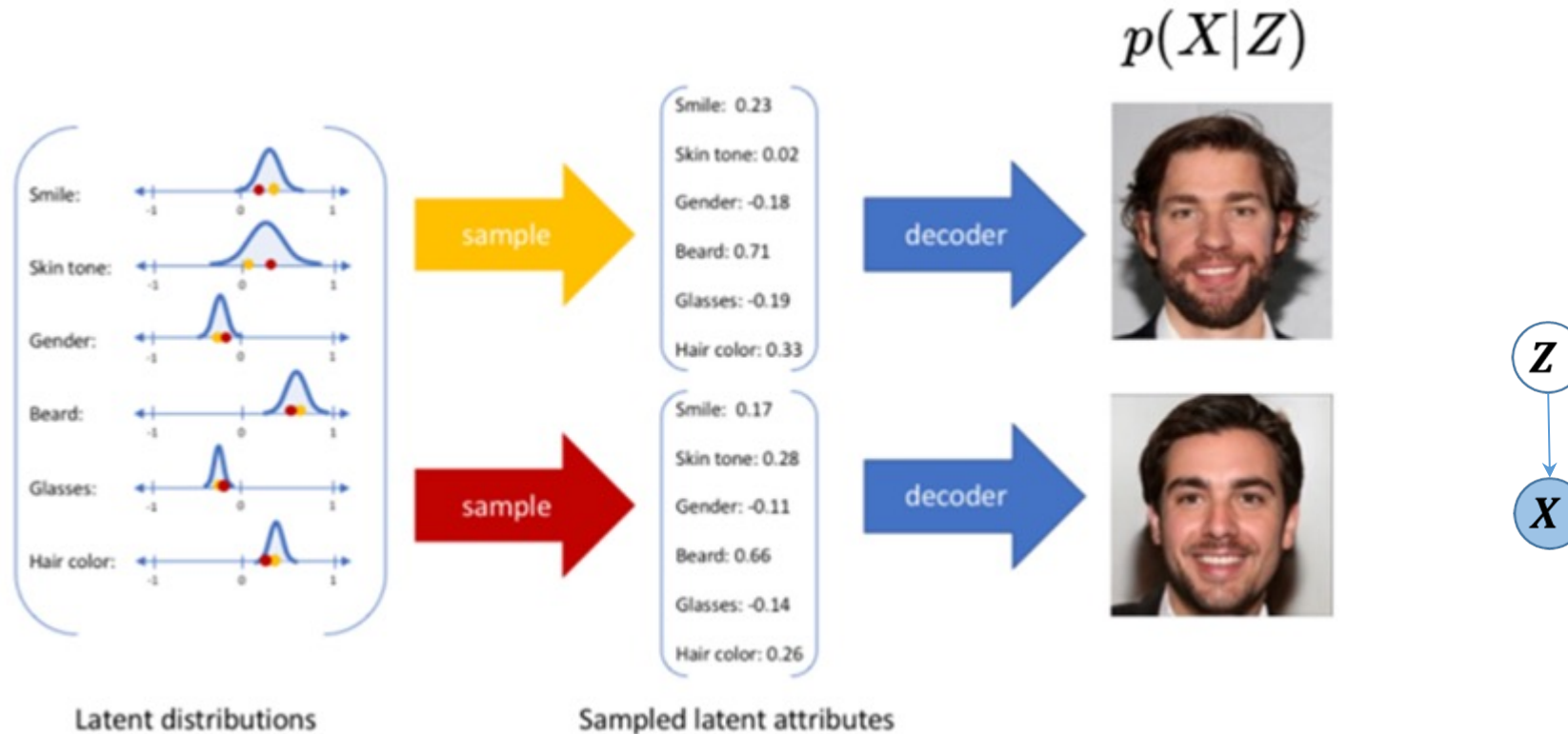
- KL between two Gaussians can be easily computed in a closed form
- KL between $q_\phi(z|x)$ and $N(0, I)$:

$$D_{KL} \left(q_\phi(z|x) || N(0, I) \right) = -\frac{1}{2} \sum_{j=1}^K (\sigma_j + \mu_j^2 - \log \sigma_j - 1)$$

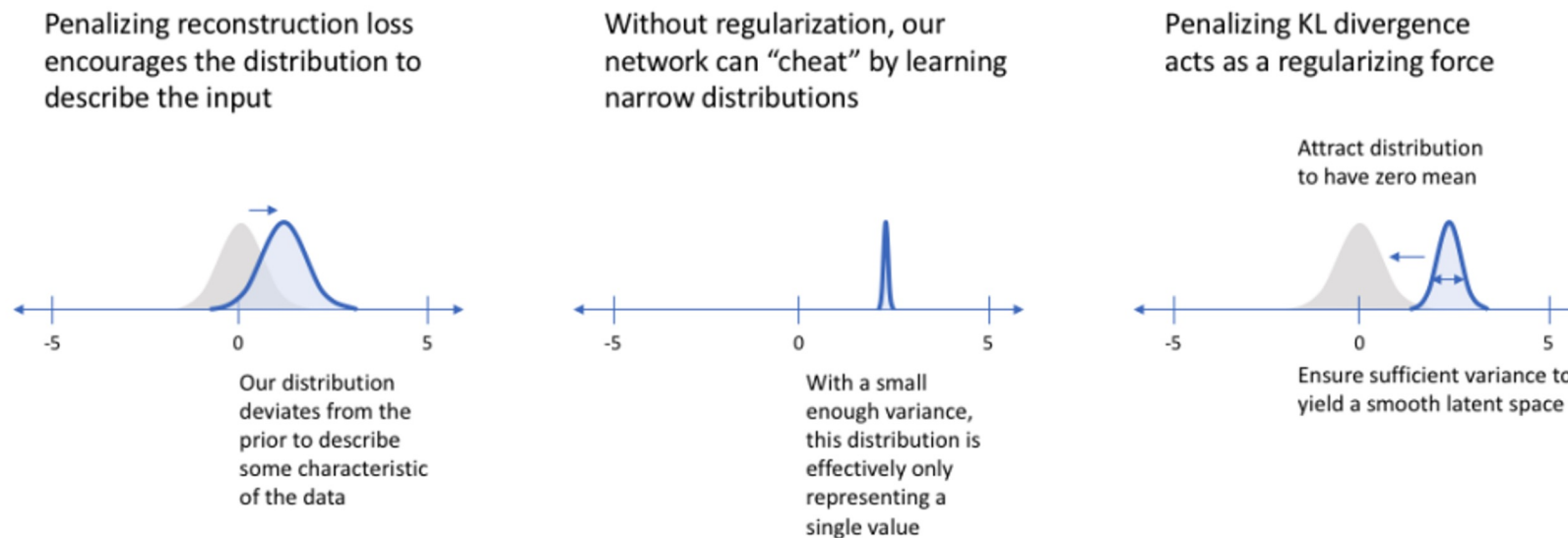
Variational Auto Encoder (VAE): Prior distribution

- Encourage encodings to be evenly distributed around the center of the latent space
- Penalize the network when it tries to memorize data

What does the VAE actually do



What does the VAE actually do



The VAE objectives arranges data on a compact manifold (we can sample from) in a continuous smooth way.

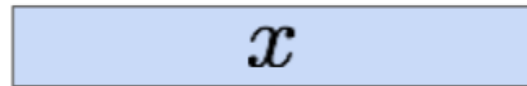
Theory of VAE

Variational Autoencoders

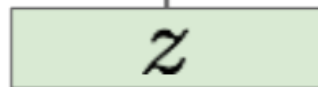
- Probabilistic spin on autoencoders
 - will let us sample from the model to generate data!

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

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



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



Intuition (remember from autoencoders!):
 $\mathbf{x}$ is an image, $\mathbf{z}$ is latent factors used to
generate $\mathbf{x}$: attributes, orientation, etc.

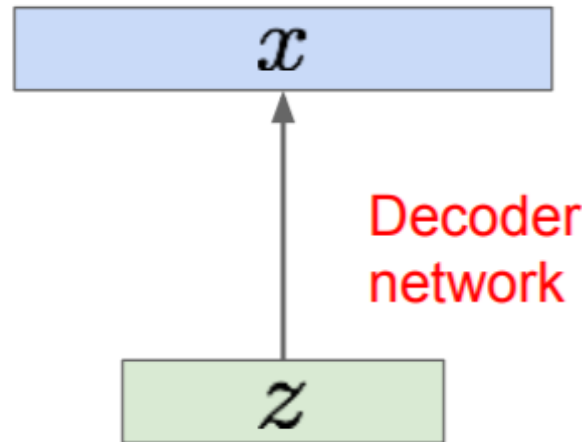
Variational Autoencoders

Sample from
true conditional

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

Sample from
true prior

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



We want to estimate the true parameters θ^* of this generative model.

How should we represent this model?

Choose prior $p(z)$ to be simple, e.g. Gaussian.

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

Variational Autoencoders

We want to optimize the true parameters θ^* of this generative model

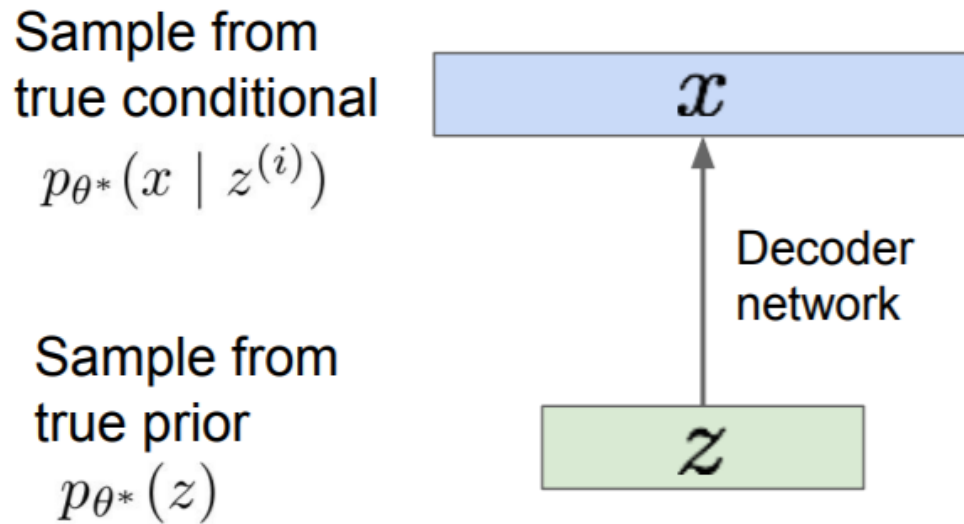
How to train the model?

Learn model parameters to maximize likelihood of training data

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

Q: What is the problem with this?

Intractable!



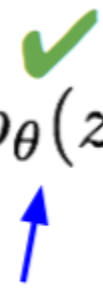
Variational Autoencoders: Intractability

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

Variational Autoencoders: Intractability

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

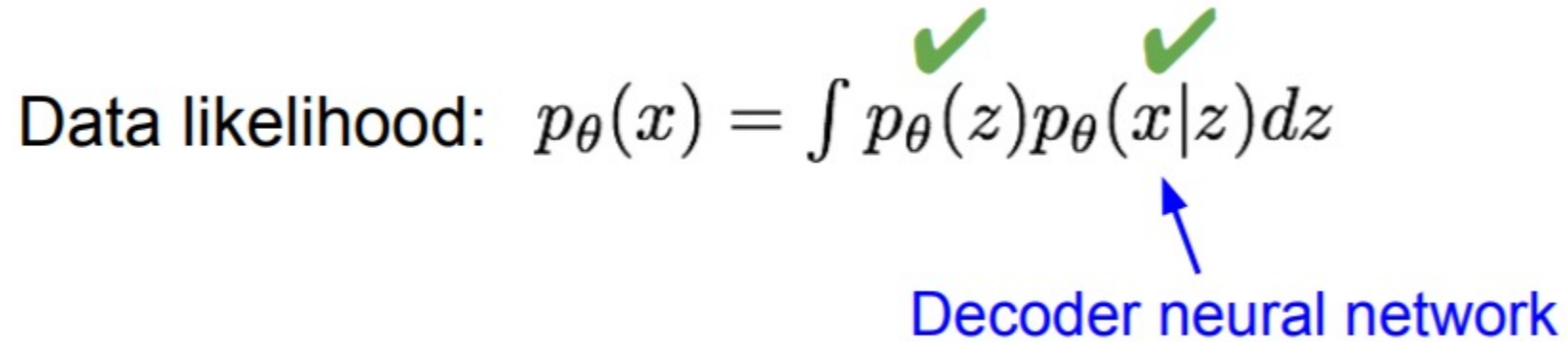
Simple Gaussian prior



Variational Autoencoders: Intractability

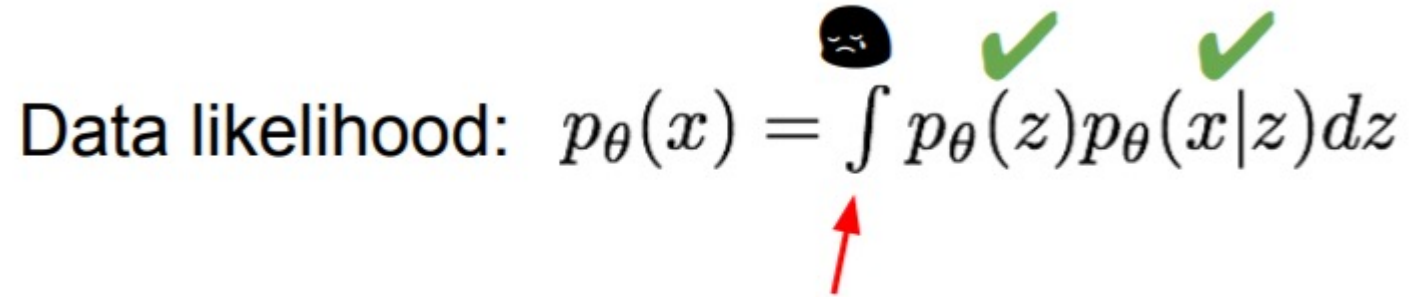
Data likelihood: $p_{\theta}(x) = \int \overset{\checkmark}{p_{\theta}(z)} \overset{\checkmark}{p_{\theta}(x|z)} dz$

Decoder neural network



Variational Autoencoders: Intractability

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



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

Variational Inference

$$\begin{aligned}\log p_{\theta}(x) &= \log \sum_z p_{\theta}(x, z) \\ &= \log \sum_z q(z) \frac{p_{\theta}(x, z)}{q(z)} \\ &\geq \sum_z q(z) \log \frac{p_{\theta}(x, z)}{q(z)} \quad \text{Jensen Inequality} \\ &= \sum_z q(z) \log \frac{p_{\theta}(x|z)p_{\theta}(z)}{q(z)} = \underbrace{E_{q(z)}[\log p_{\theta}(x|z)]}_{\mathcal{L}_q(x, \theta)} - \underbrace{KL(q(z)||p_{\theta}(z))}_{\text{Evidence Lower Bound (ELBO)}}\end{aligned}$$

When is the ELBO gap minimized?

- We want to maximize the log-likelihood
- What is the gap between ELBO and likelihood?

$$\log p_{\theta}(x) = \mathcal{L}_q(x, \theta) + KL(q(z)||p_{\theta}(z|x))$$

- To minimize the ELBO gap, we need $q(z) = p_{\theta}(z|x)$
- However $p_{\theta}(z|x)$ is intractable

Why is posterior intractable?

- For many models with latent variables $q(z) = p_{\theta}(Z|X)$ is intractable

$$p_{\theta}(Z|X) = \frac{p_{\theta}(Z)p_{\theta}(X|Z)}{p_{\theta}(X)}$$

- Since we need integral over z that is intractable

$$p_{\theta}(X) = \int p_{\theta}(X|Z)p_{\theta}(Z)dZ$$

Lower bound for log likelihood

$$\begin{aligned}\log p(x) &= E_{z \sim q(z|x)} [\log p(x)] && (p(x) \text{ does not depend on } z) \\ &= E_{z \sim q(z|x)} \left[\log \frac{p(x|z)p(z)}{p(z|x)} \right] && (\text{Bayes' rule}) \\ &= E_{z \sim q(z|x)} \left[\log \left(\frac{p(x|z)p(z)}{p(z|x)} \times \frac{q(z|x)}{q(z|x)} \right) \right] && (\text{Multiply by constant}) \\ &= E_{z \sim q(z|x)} [\log p(x|z)] - E_{z \sim q(z|x)} \left[\log \left(\frac{q(z|x)}{p(z)} \right) \right] + E_{z \sim q(z|x)} \left[\log \left(\frac{q(z|x)}{p(z|x)} \right) \right] \\ &= E_{z \sim q(z|x)} [\log p(x|z)] - D_{KL}(q(z|x) || p(z)) + D_{KL}(q(z|x) || p(z|x))\end{aligned}$$

Lower bound for log likelihood

$$\begin{aligned}\log p(x) &= E_{z \sim q(z|x)} [\log p(x)] && (p(x) \text{ does not depend on } z) \\ &= E_{z \sim q(z|x)} \left[\log \frac{p(x|z)p(z)}{p(z|x)} \right] && (\text{Bayes' rule}) \\ &= E_{z \sim q(z|x)} \left[\log \left(\frac{p(x|z)p(z)}{p(z|x)} \times \frac{q(z|x)}{q(z|x)} \right) \right] && (\text{Multiply by constant}) \\ &= E_{z \sim q(z|x)} [\log p(x|z)] - E_{z \sim q(z|x)} \left[\log \left(\frac{q(z|x)}{p(z)} \right) \right] + E_{z \sim q(z|x)} \left[\log \left(\frac{q(z|x)}{p(z|x)} \right) \right] \\ &= E_{z \sim q(z|x)} [\log p(x|z)] - D_{KL}(q(z|x) || p(z)) + D_{KL}(q(z|x) || p(z|x))\end{aligned}$$

↑
Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

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

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

Lower bound for log likelihood

$$\begin{aligned}\log p(x) &= E_{z \sim q(z|x)} [\log p(x)] && (p(x) \text{ does not depend on } z) \\ &= E_{z \sim q(z|x)} \left[\log \frac{p(x|z)p(z)}{p(z|x)} \right] && (\text{Bayes' rule}) \\ &= E_{z \sim q(z|x)} \left[\log \left(\frac{p(x|z)p(z)}{p(z|x)} \times \frac{q(z|x)}{q(z|x)} \right) \right] && (\text{Multiply by constant}) \\ &= E_{z \sim q(z|x)} [\log p(x|z)] - E_{z \sim q(z|x)} \left[\log \left(\frac{q(z|x)}{p(z)} \right) \right] + E_{z \sim q(z|x)} \left[\log \left(\frac{q(z|x)}{p(z|x)} \right) \right] \\ &= \boxed{E_{z \sim q(z|x)} [\log p(x|z)] - D_{KL}(q(z|x) || p(z))} + \boxed{D_{KL}(q(z|x) || p(z|x))}\end{aligned}$$

Tractable lower bound which we
can take gradient of and optimize!

≥ 0

Lower bound for log likelihood

$$\begin{aligned}\log p_{\theta}(x) &= E_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x)] && (p(x) \text{ does not depend on } z) \\&= E_{z \sim q_{\phi}(z|x)} \left[\log \frac{p_{\theta}(x|z)p_{\theta}(z)}{p_{\theta}(z|x)} \right] && (\text{Bayes' rule}) \\&= E_{z \sim q_{\phi}(z|x)} \left[\log \left(\frac{p_{\theta}(x|z)p_{\theta}(z)}{p_{\theta}(z|x)} \times \frac{q_{\phi}(z|x)}{q_{\phi}(z|x)} \right) \right] && (\text{Multiply by constant}) \\&= E_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)] - E_{z \sim q_{\phi}(z|x)} \left[\log \left(\frac{q_{\phi}(z|x)}{p_{\theta}(z)} \right) \right] + E_{z \sim q_{\phi}(z|x)} \left[\log \left(\frac{q_{\phi}(z|x)}{p_{\theta}(z|x)} \right) \right] \\&= \boxed{E_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)] - D_{KL} \left(q_{\phi}(z|x) || p_{\theta}(z) \right)} + \boxed{D_{KL} \left(q_{\phi}(z|x) || p_{\theta}(z|x) \right)}\end{aligned}$$

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

≥ 0

Lower bound for log likelihood

$$\begin{aligned}\log p_{\theta}(x) &= E_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x)] && (p(x) \text{ does not depend on } z) \\&= E_{z \sim q_{\phi}(z|x)} \left[\log \frac{p_{\theta}(x|z)p_{\theta}(z)}{p_{\theta}(z|x)} \right] && (\text{Bayes' rule}) \\&= E_{z \sim q_{\phi}(z|x)} \left[\log \left(\frac{p_{\theta}(x|z)p_{\theta}(z)}{p_{\theta}(z|x)} \times \frac{q_{\phi}(z|x)}{q_{\phi}(z|x)} \right) \right] && (\text{Multiply by constant}) \\&= E_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)] - E_{z \sim q_{\phi}(z|x)} \left[\log \left(\frac{q_{\phi}(z|x)}{p_{\theta}(z)} \right) \right] + E_{z \sim q_{\phi}(z|x)} \left[\log \left(\frac{q_{\phi}(z|x)}{p_{\theta}(z|x)} \right) \right] \\&= \underbrace{E_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)] - D_{KL} \left(q_{\phi}(z|x) || p_{\theta}(z) \right)}_{\mathcal{L}(x, \theta, \phi)} + \underbrace{D_{KL} \left(q_{\phi}(z|x) || p_{\theta}(z|x) \right)}_{\geq 0}\end{aligned}$$

$$\log p_{\theta}(x) \geq \mathcal{L}(x, \theta, \phi)$$

Evidence Lower Bound (ELBO)

Training: Maximize lower bound

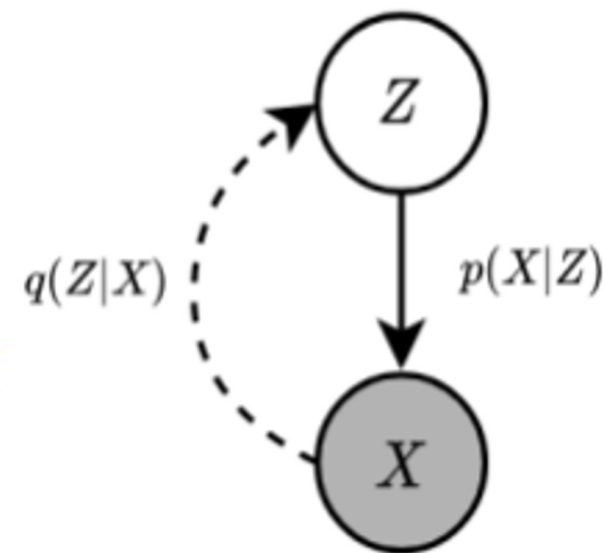
$$\theta^*, \phi^* = \operatorname{argmax}_{\theta, \phi} \sum_{i=1}^n \mathcal{L}(x^{(i)}, \theta, \phi)$$

General solution

- Introduce inference distribution $q_{\phi}(Z|X)$
- To minimize the ELBO gap we choose $q(Z) = q_{\phi}(Z|X)$.

Solution: In addition to decoder network modeling $p_{\theta}(x|z)$, define additional encoder network $q_{\phi}(z|x)$ that approximates $p_{\theta}(z|x)$

Will see that this allows us to derive a lower bound on the data likelihood that is tractable, which we can optimize



ELBO using the decoder network

- The ELBO consists of two terms: reconstruction and a KL regularization:

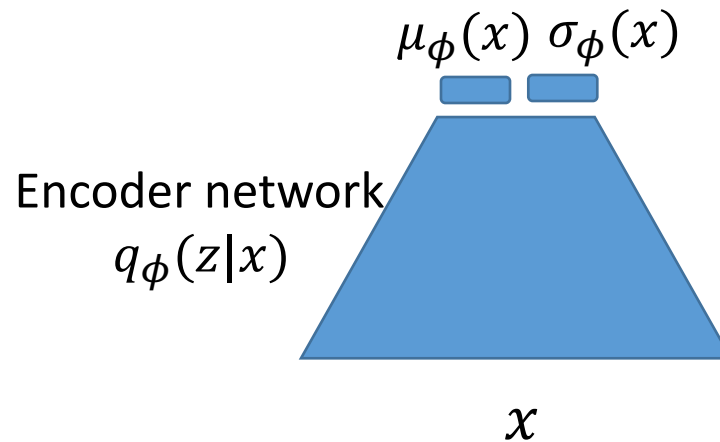
$$\mathcal{L}_{\phi, \theta}(X) = \underbrace{\mathbb{E}_{q_{\phi}(Z|X)}[p_{\theta}(X|Z)]}_{\text{Reconstruction}} - \underbrace{KL(q_{\phi}(Z|X) || P(Z))}_{\text{Regularization}}$$

- $q_{\phi}(Z|X)$: multivariate normal distribution parameterized by a neural network

$$q_{\phi}(Z|X) = N(Z | \mu_{\phi}(X), \sigma_{\phi}(X))$$

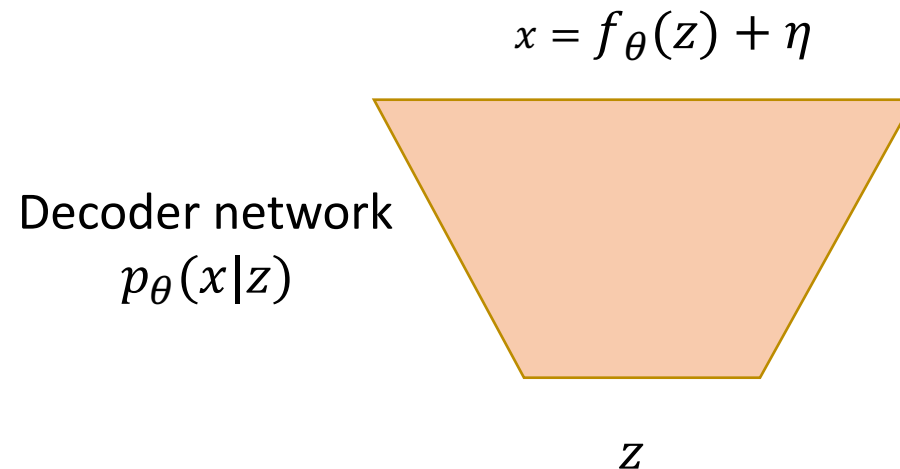
Encoder network

- Model $q_{\phi}(z|x)$ with a neural network (called encoder network)
- Assume $q_{\phi}(z|x)$ to be Gaussian
- Neural network outputs the mean vector $\mu_{\phi}(x)$ and standard deviation vector $\sigma_{\phi}(x)$
- Called recognition/inference network

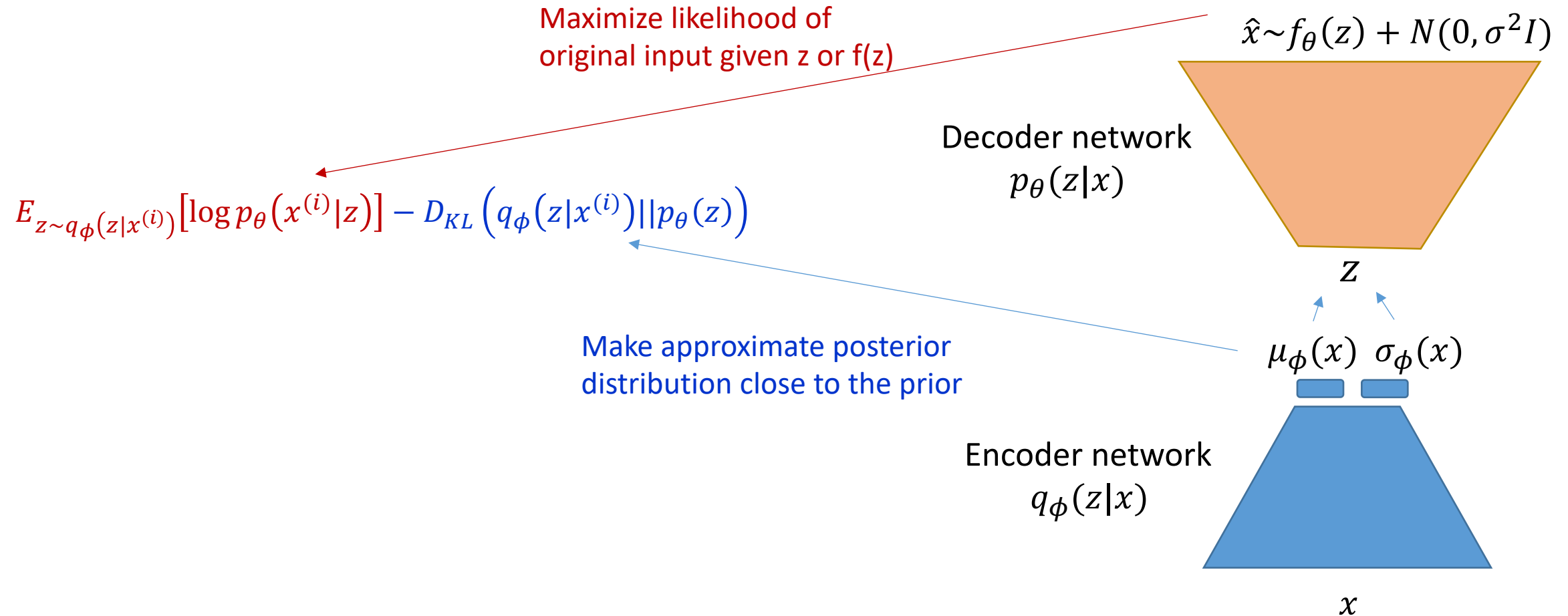


Decoder network

- Model $p_{\theta}(x|z)$ with a neural network (called decoder network)
 - let $f_{\theta}(z)$ be the network output.
 - Assume $p_{\theta}(x|z)$ to be i.i.d. Gaussian, e.g.,
 - $x = f_{\theta}(z) + \eta$, where $\eta \sim N(0, \sigma^2 I)$
- Called also generation network

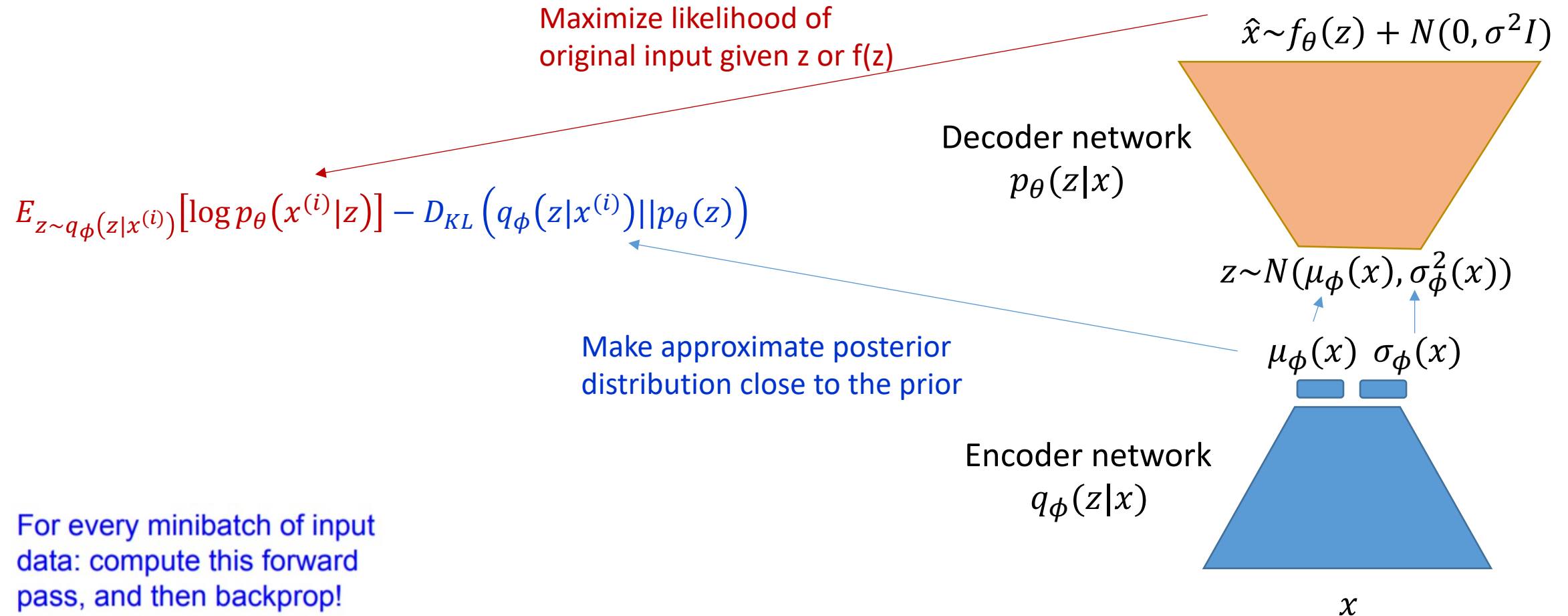


Variational Autoencoders



Variational Autoencoders

- Putting altogether=maximizing likelihood lower bound



Optimizing the ELBO

Need to compute

$$\nabla_{\phi, \theta} \mathbb{E}_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)] \approx \nabla_{\phi, \theta} \hat{\mathbb{E}}_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)]$$

Cost function

If we consider:

$$p_{\theta}(x|z) = N(f_{\theta}(z), \sigma^2 I) \quad p_{\theta}(x|z) = \frac{1}{(\sqrt{2\pi}\sigma)^D} e^{-\frac{1}{2\sigma^2}\|x-f_{\theta}(z)\|^2}$$

We have:

$$\log p_{\theta}(x|z) = C - \frac{1}{2\sigma^2} \|x - f_{\theta}(z)\|^2$$

- Putting it all together (given a (x, z) pair where $z \sim q_{\phi}(z|x)$):

$$L = \frac{1}{2\sigma^2} \|x - f_{\theta}(z)\|^2 + D_{KL}[q_{\phi}(z|x) || p(z)]$$

Corresponding to $E_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)]$
term since $z \sim q_{\phi}(z|x)$ in the training data

$$p(z) = N(0, I)$$

Maximizing ELBO

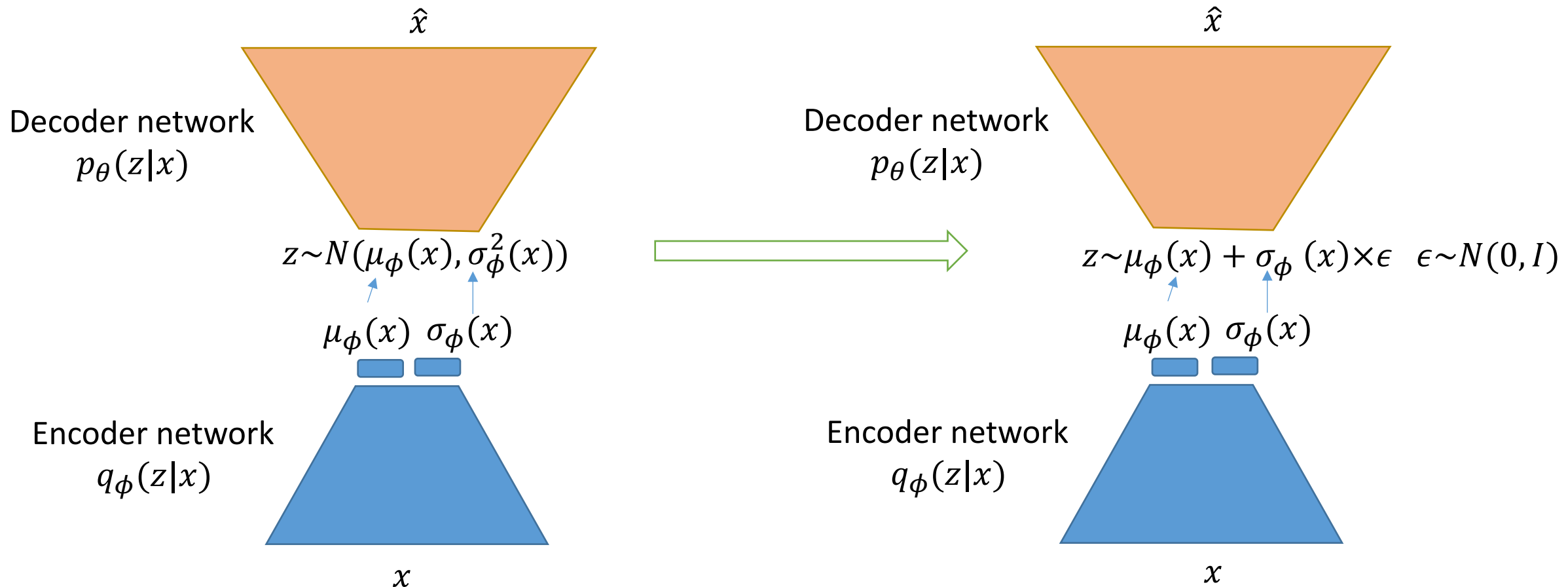
- Maximizing ELBO composed of:
 - Reconstruction cost: The expected log-likelihood measures how well samples from $q(z|x)$ are able to explain the data x .
 - Penalty: Ensures that the explanation of the data $q(z|x)$ doesn't deviate too far from your beliefs $p(z)$.
- When approximate posterior distribution $q(z|x)$ is the best match to true posterior $p(z|x)$ this lower bound gets tight

$$\begin{aligned} & \operatorname{argmax}_{\theta, \phi} \sum_{i=1}^n \mathcal{L}(x^{(i)}, \theta, \phi) \\ &= \boxed{\sum_{i=1}^n E_{z \sim q_{\phi}(z|x^{(i)})} [\log p_{\theta}(x^{(i)}|z)]} - \boxed{\sum_{i=1}^n D_{KL} \left(q_{\phi}(z|x^{(i)}) || p_{\theta}(z) \right)} \end{aligned}$$

Reconstruct the input

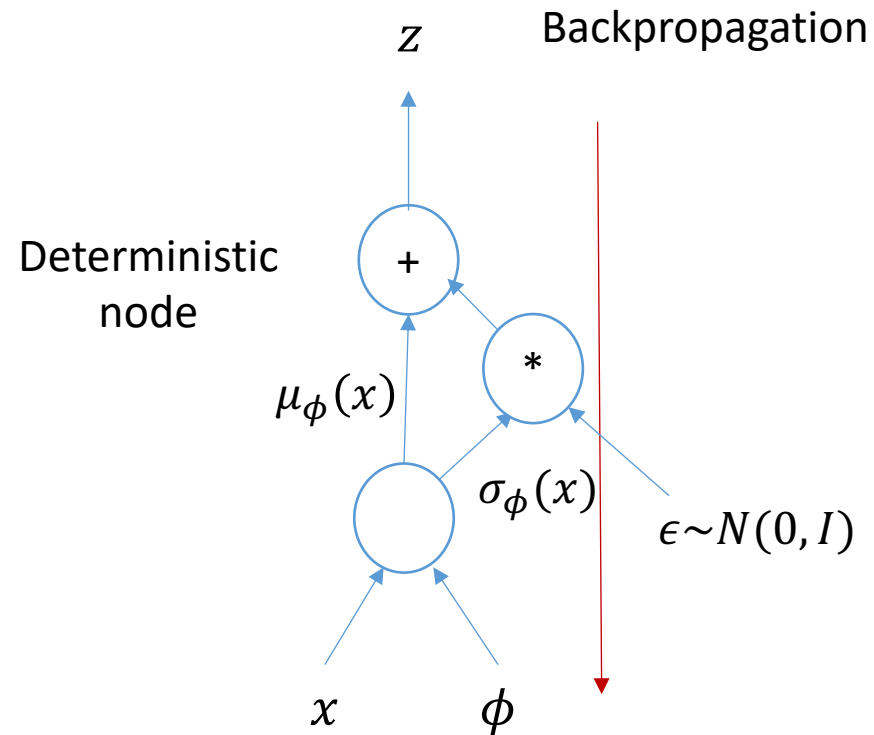
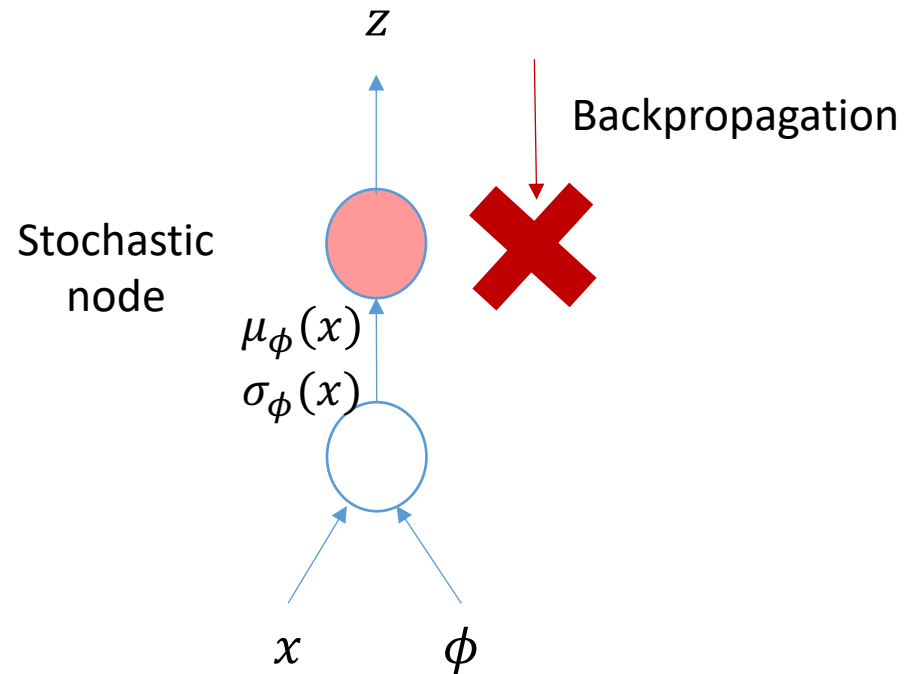
Make the posterior close to prior

Reparameterizing the sampling layer



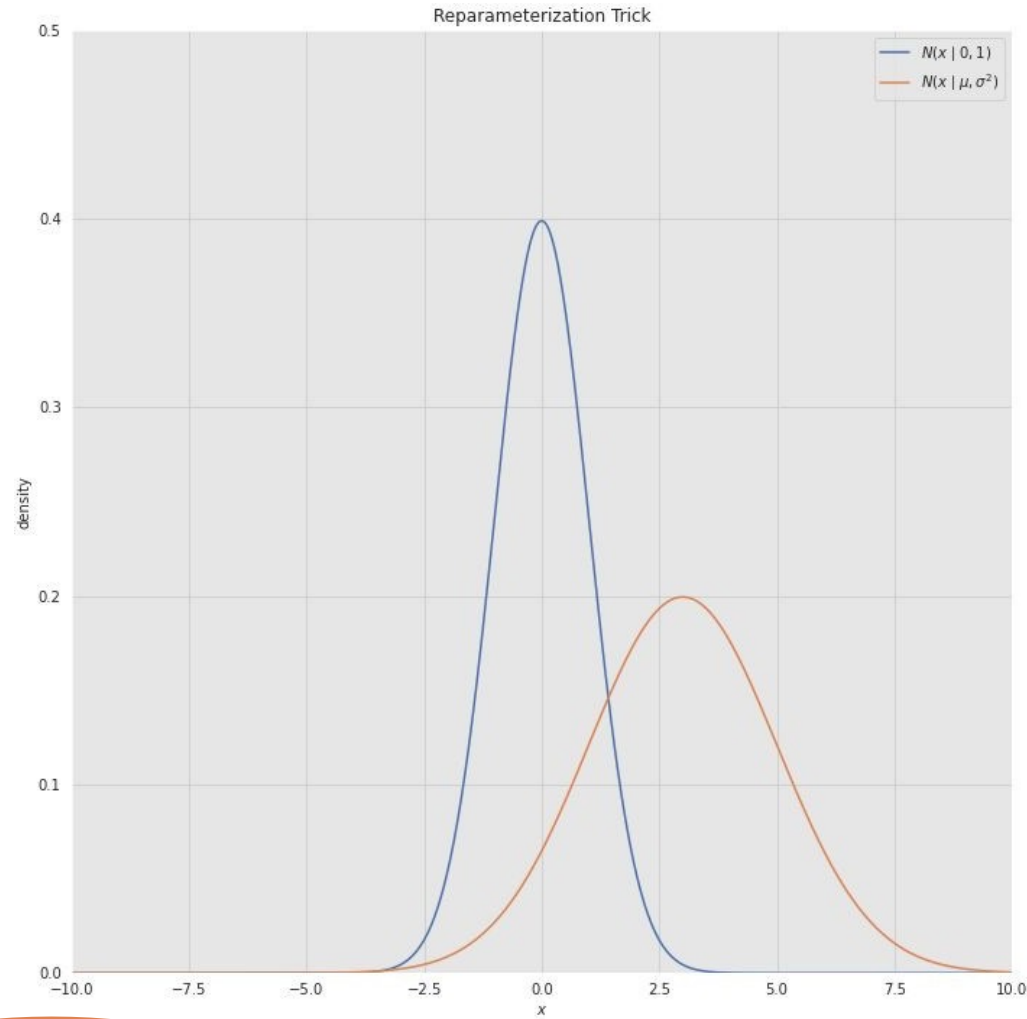
Reparameterization trick

Reparameterizing the sampling layer



Backpropagation through samples is now possible

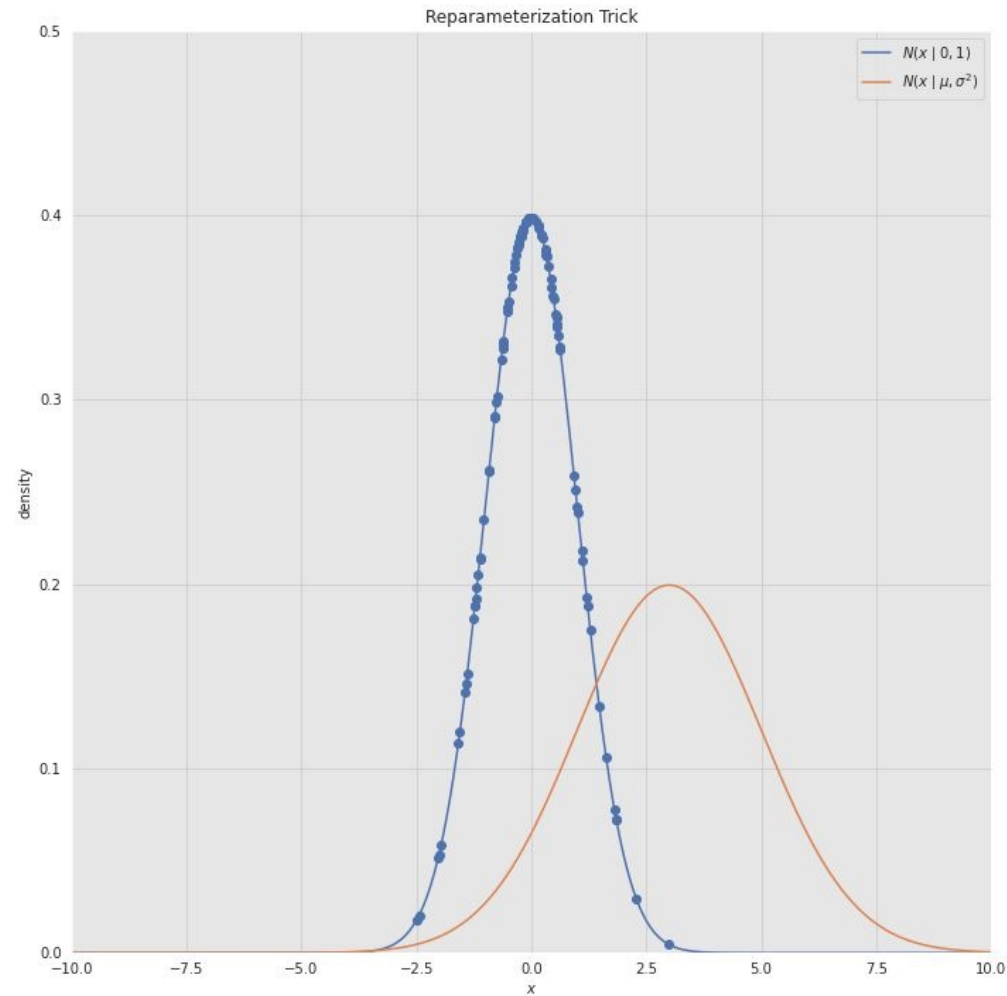
Reparameterization Trick



For $x \sim \mathcal{N}(x | \mu, \sigma^2)$

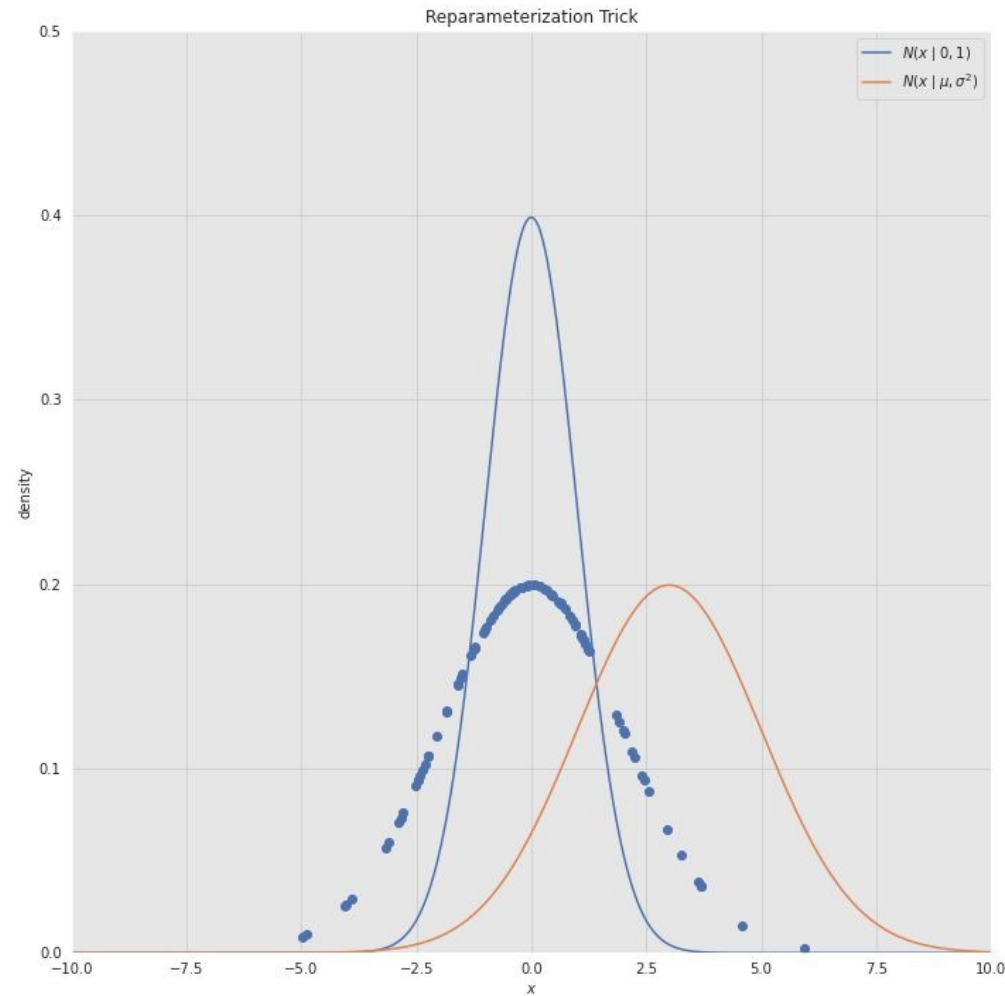
$\epsilon \sim \mathcal{N}(x | 0, I)$

Reparameterization Trick



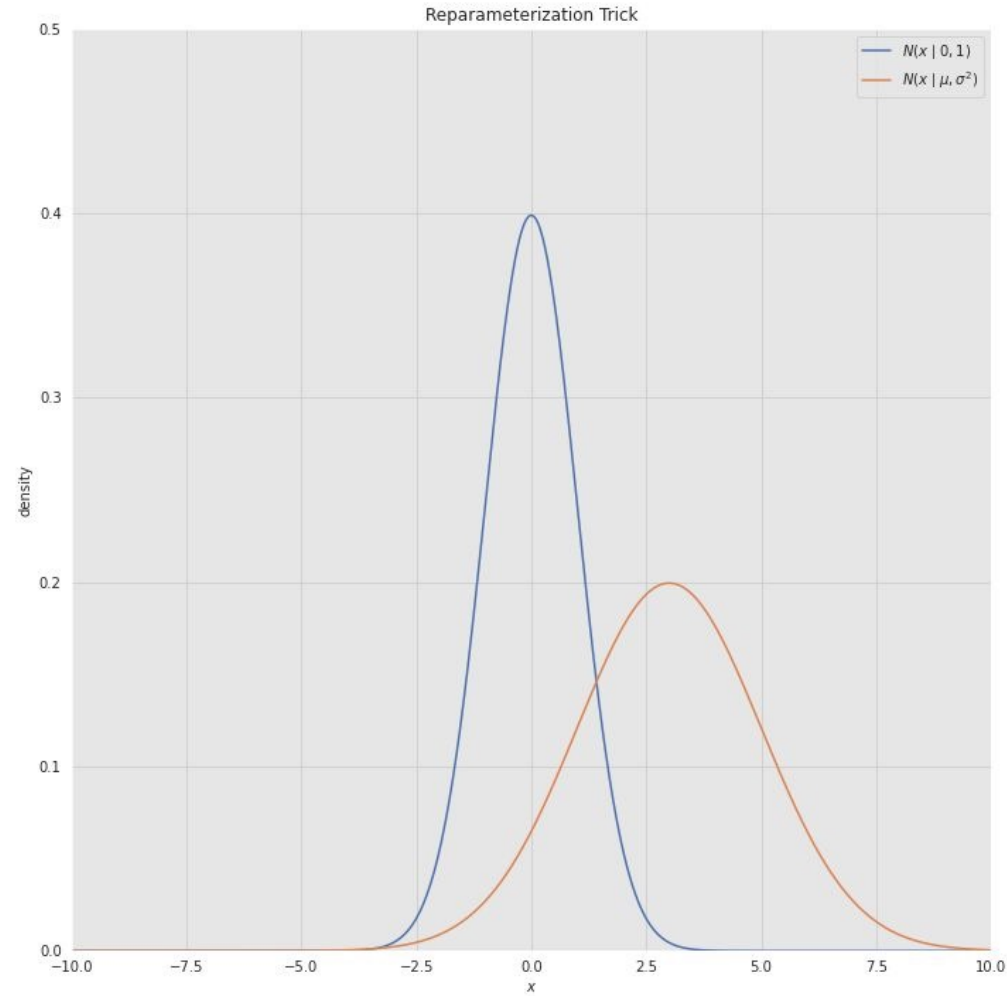
For $x \sim \mathcal{N}(x \mid \mu, \sigma^2)$, $x =$ ϵ , where $\epsilon \sim \mathcal{N}(x \mid 0, I)$

Reparameterization Trick



For $x \sim \mathcal{N}(x | \mu, \sigma^2)$, $x =$ $\sigma \epsilon$, where $\epsilon \sim \mathcal{N}(x | 0, I)$

Reparameterization Trick



For $x \sim \mathcal{N}(x | \mu, \sigma^2)$, $x = \mu + \sigma\epsilon$, where $\epsilon \sim \mathcal{N}(x | 0, \mathbf{I})$

Optimizing the ELBO

Need to compute

$$\nabla_{\phi, \theta} \mathbb{E}_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)] \approx \nabla_{\phi, \theta} \hat{\mathbb{E}}_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)]$$

Reparameterization Trick:

$$\tilde{z} = \mu_{\phi}(x) + \epsilon \sigma_{\phi}(x), \text{ where } \epsilon \sim \mathcal{N}(0, I)$$

Optimizing the ELBO

Need to compute

$$\nabla_{\phi, \theta} \mathbb{E}_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)] \approx \nabla_{\phi, \theta} \hat{\mathbb{E}}_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)]$$

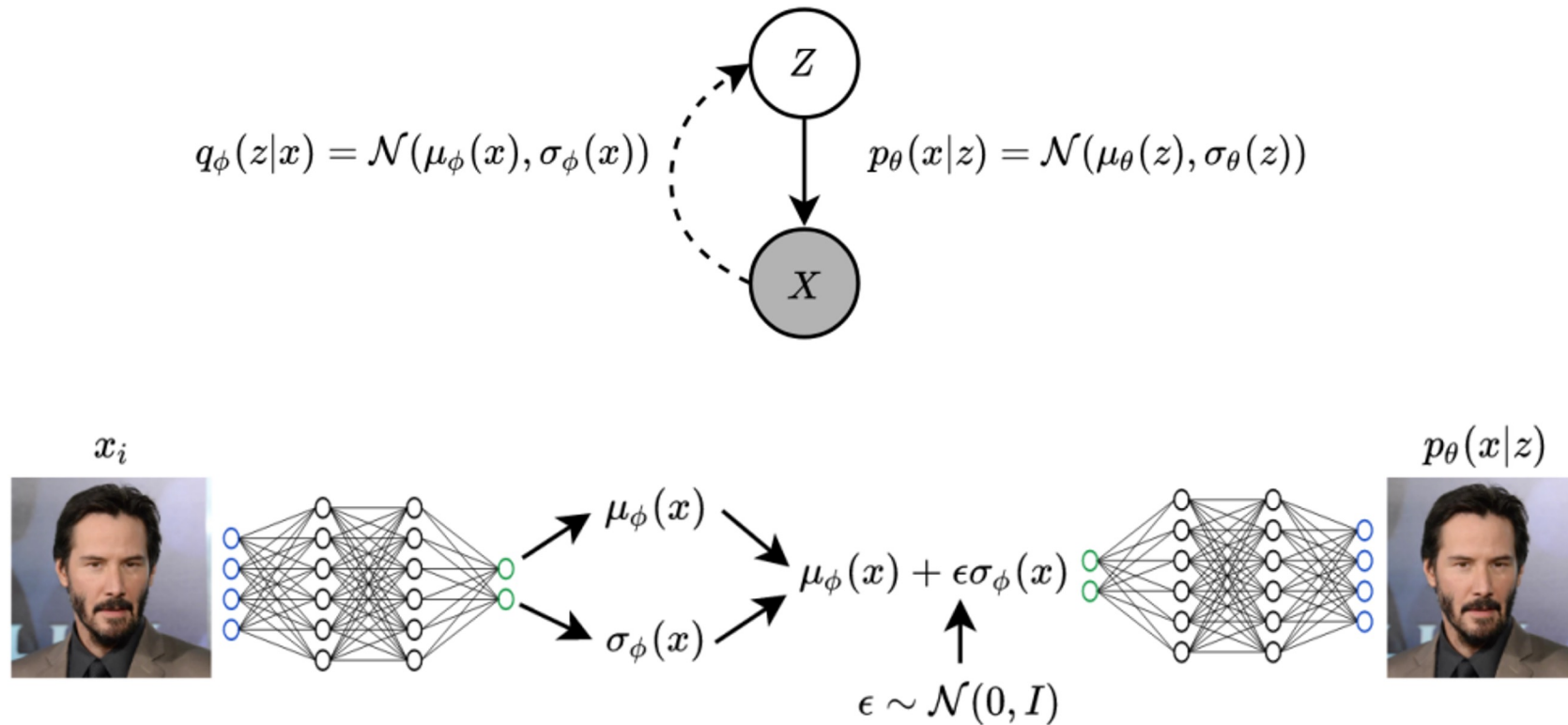
Reparameterization Trick:

$$\tilde{z} = \mu_{\phi}(x) + \epsilon \sigma_{\phi}(x), \text{ where } \epsilon \sim \mathcal{N}(0, I)$$

$$\nabla_{\phi, \theta} \mathbb{E}_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)] \approx \nabla_{\phi, \theta} \frac{1}{M} \sum_{j=1}^M \log p_{\theta}(x | \underbrace{\mu_{\phi}(x) + \epsilon^{(j)} \sigma_{\phi}(x)}_{\tilde{z}^{(j)} \sim \mathcal{N}(\mu_{\phi}(x), \sigma_{\phi}(x))})$$

In practice usually $M = 1$.

Variational Auto Encoder (VAE)

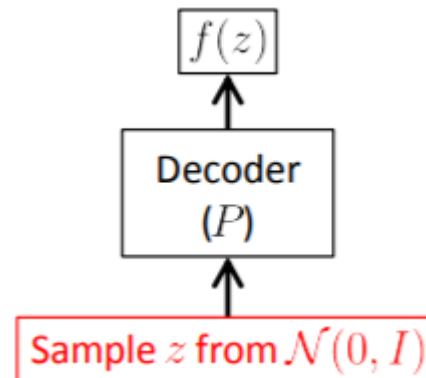


$$\max_{\phi, \theta} \frac{1}{N} \sum_{i=1}^N \log p_\theta \left(x^{(i)} \mid \mu_\phi(x^{(i)}) + \epsilon \sigma_\phi(x^{(i)}) \right) - D_{KL} \left(N \left(\mu_\phi(x^{(i)}), \sigma_\phi^2(x^{(i)}) \right) \parallel N(0, I) \right)$$

This slide has been adapted from Chelsea Finn, CS330

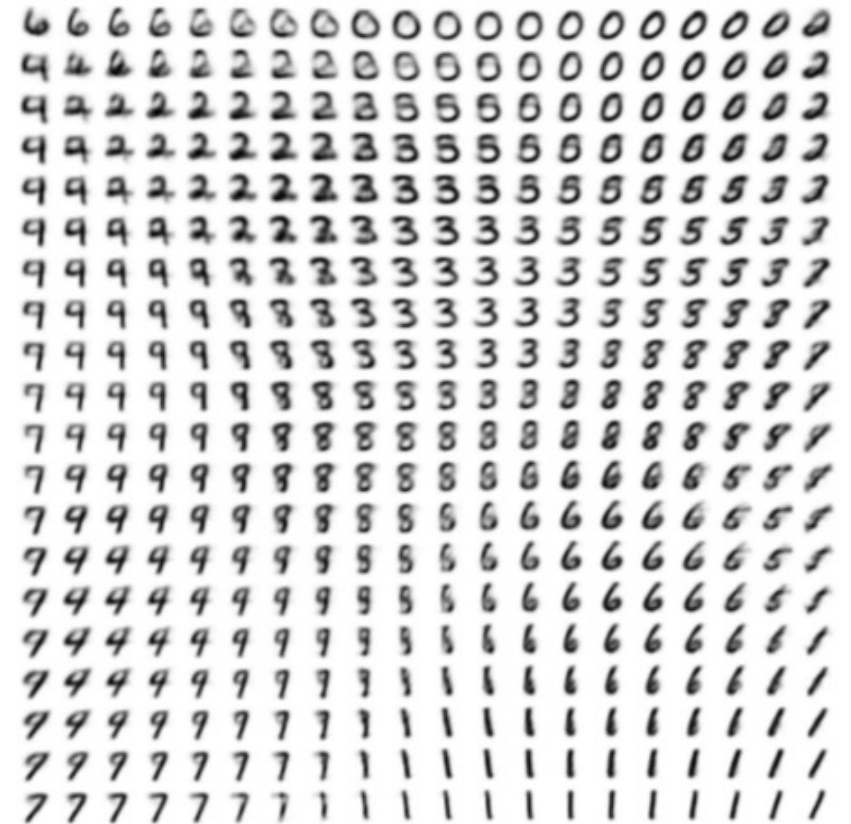
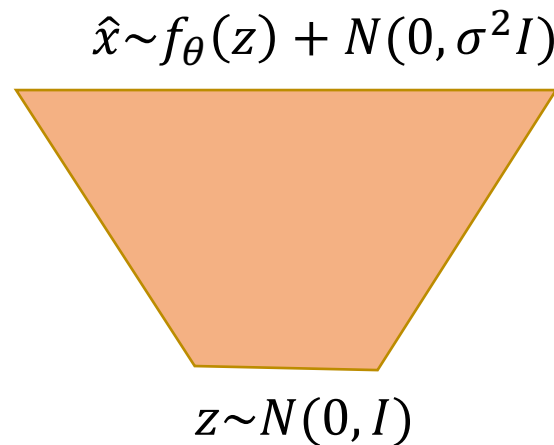
Generate Image: At Test Time

- Remove the Encoder
- Sample $z \sim \mathcal{N}(0, I)$ and pass it through the Decoder.
- No good quantitative metric, relies on visual inspection

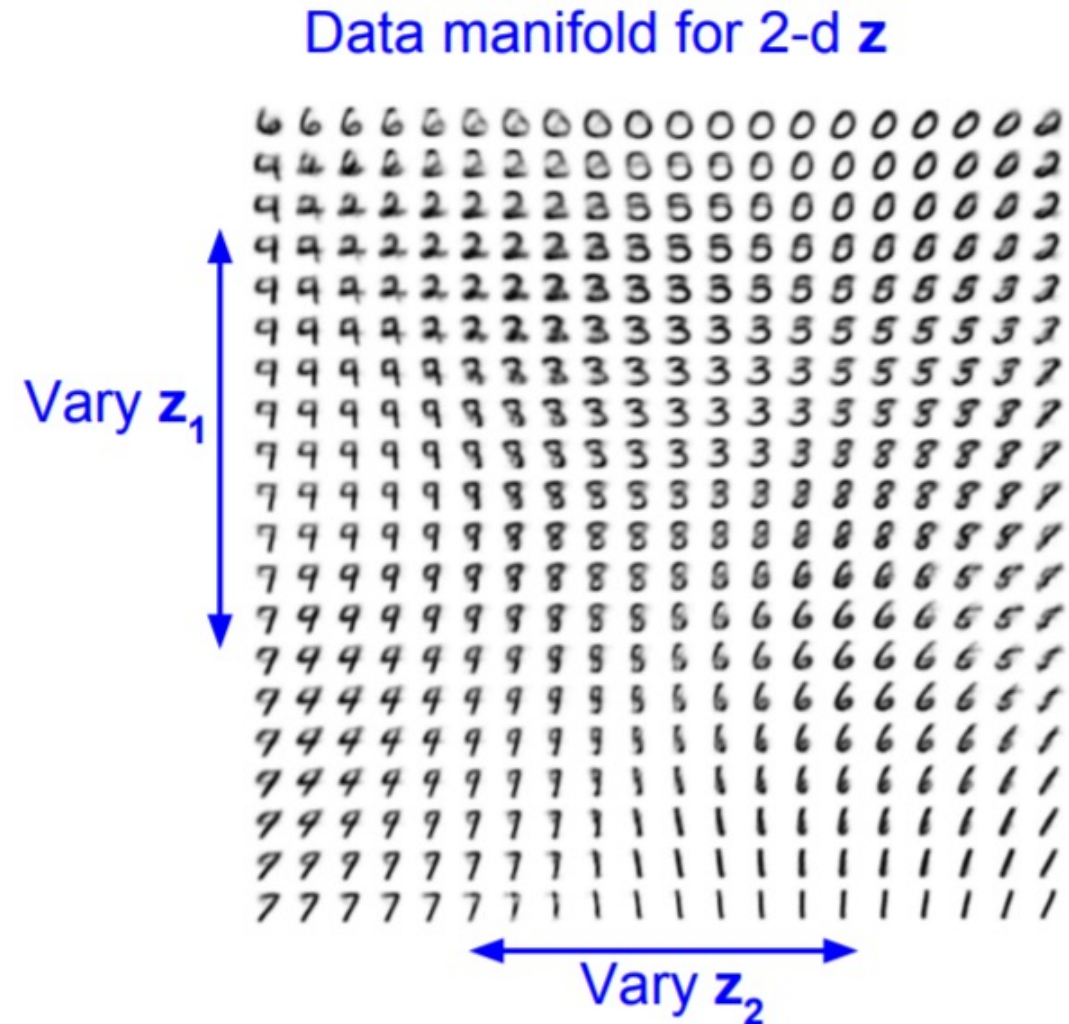


Variational Autoencoders: Generating Data!

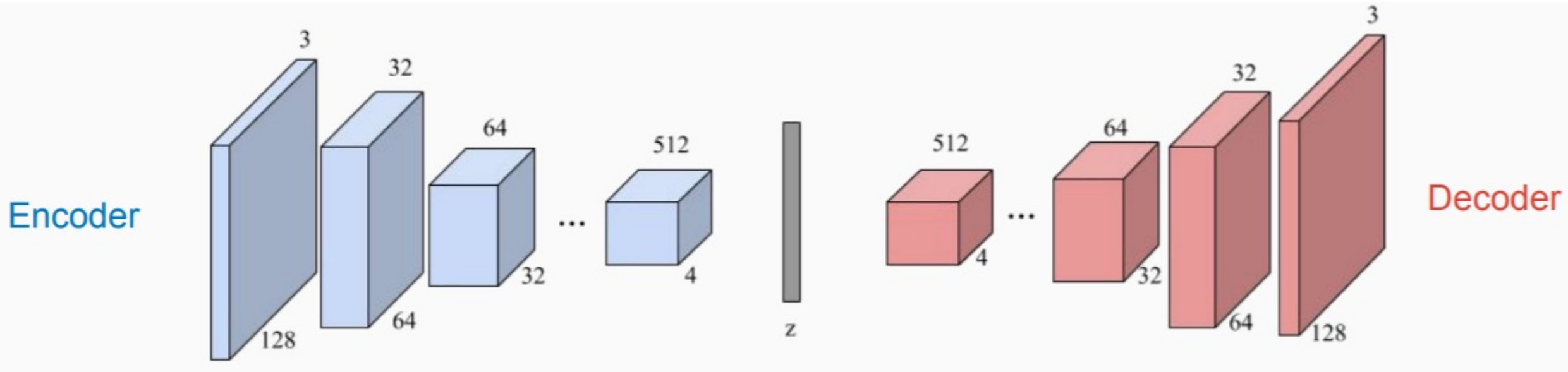
- Sample from the prior and use the decoder to generate an image



Variational Autoencoders: Generating Data!



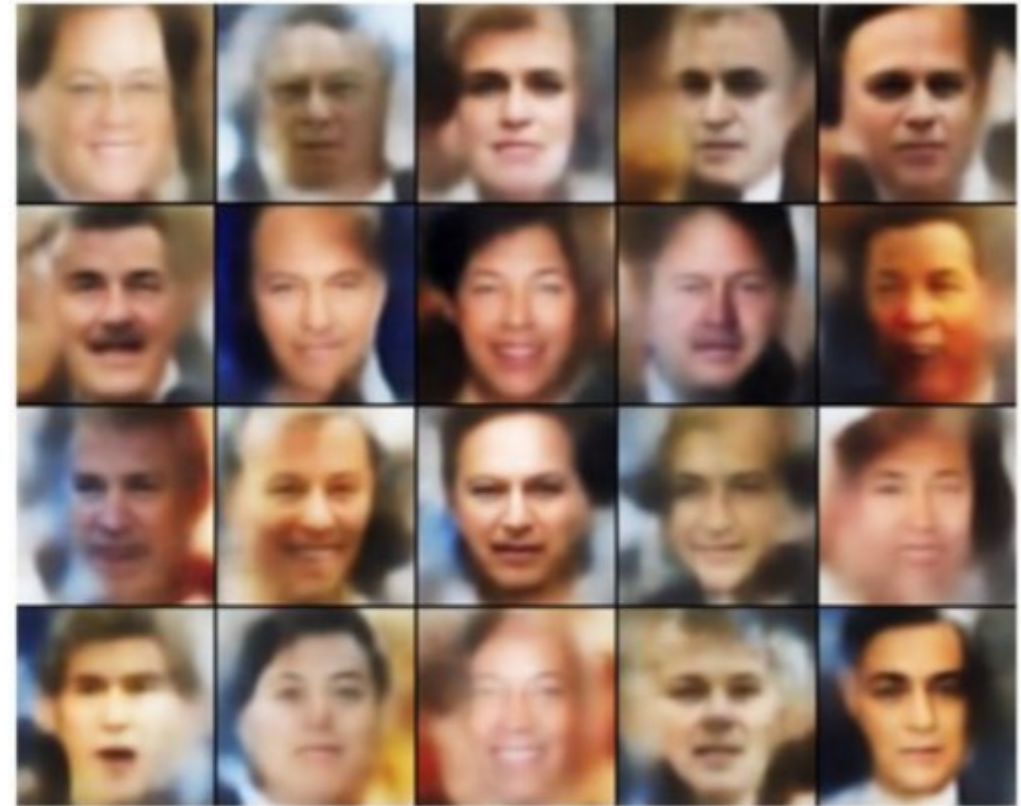
Common VAE Architecture for Image Generation



Variational Autoencoders: Generating Data!



32x32 CIFAR-10

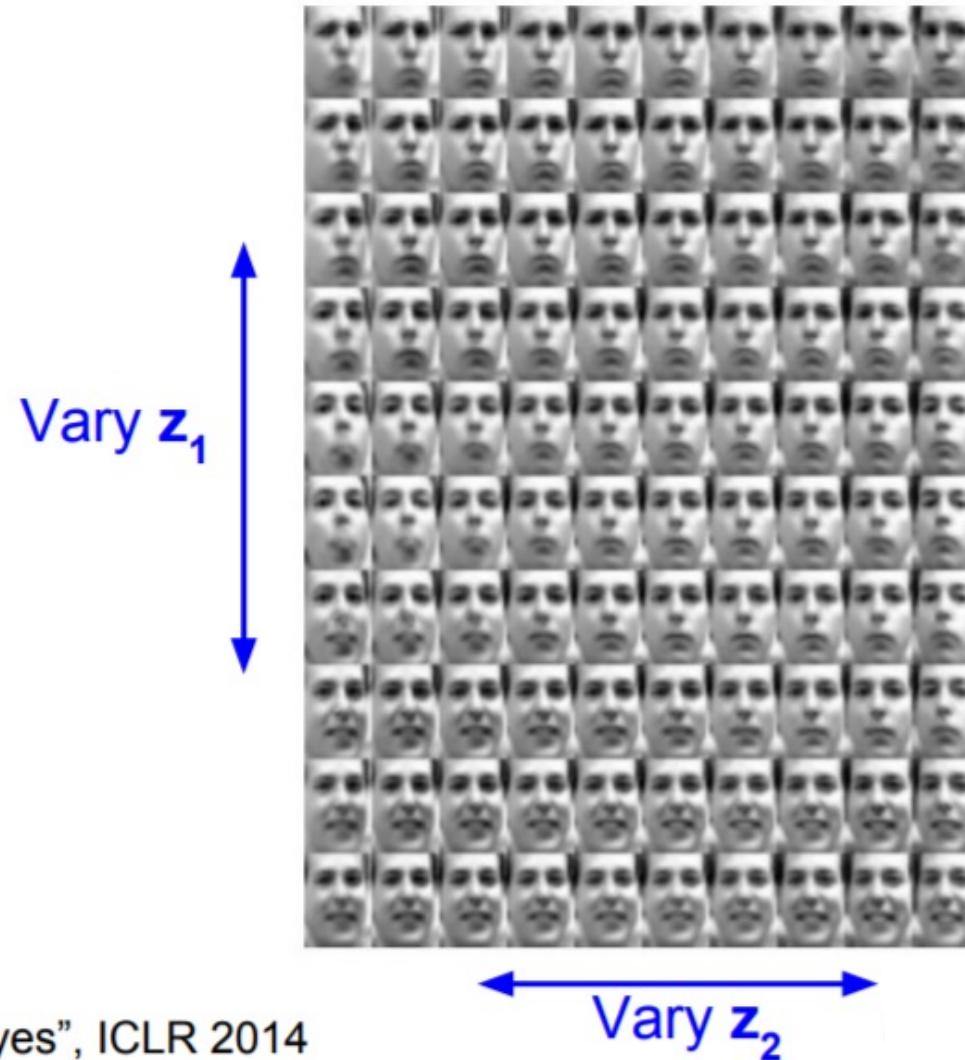


Labeled Faces in the Wild

VAE: Factors of variations

- When different dimensions of latent space encode different interpretable latent variables, it is pleasant
- Latent perturbation: slowly change a latent variable while keeping fixed others

Variational Autoencoders: Generating Data!



Variational Autoencoders: Generating Data!

Diagonal prior on $\mathbf{z}$
=> independent
latent variables

Different
dimensions of $\mathbf{z}$
encode
interpretable factors
of variation

Degree of smile

Vary z_1



Vary z_2

Head pose

Disentanglement

- Disentanglement: Ideally, we want independent latent variables
- Enforce the diagonal prior on the latent variables to encourage independence



Beta-VAE

- Beta-VAE loss: $\beta > 1$

$$\sum_{i=1}^n E_{z \sim q_{\phi}(z|x^{(i)})} [\log p_{\theta}(x^{(i)}|z)] - \beta \sum_{i=1}^n D_{KL} (q_{\phi}(z|x^{(i)}) || p_{\theta}(z))$$

Beta-VAE

- $\beta=250$



CVAE

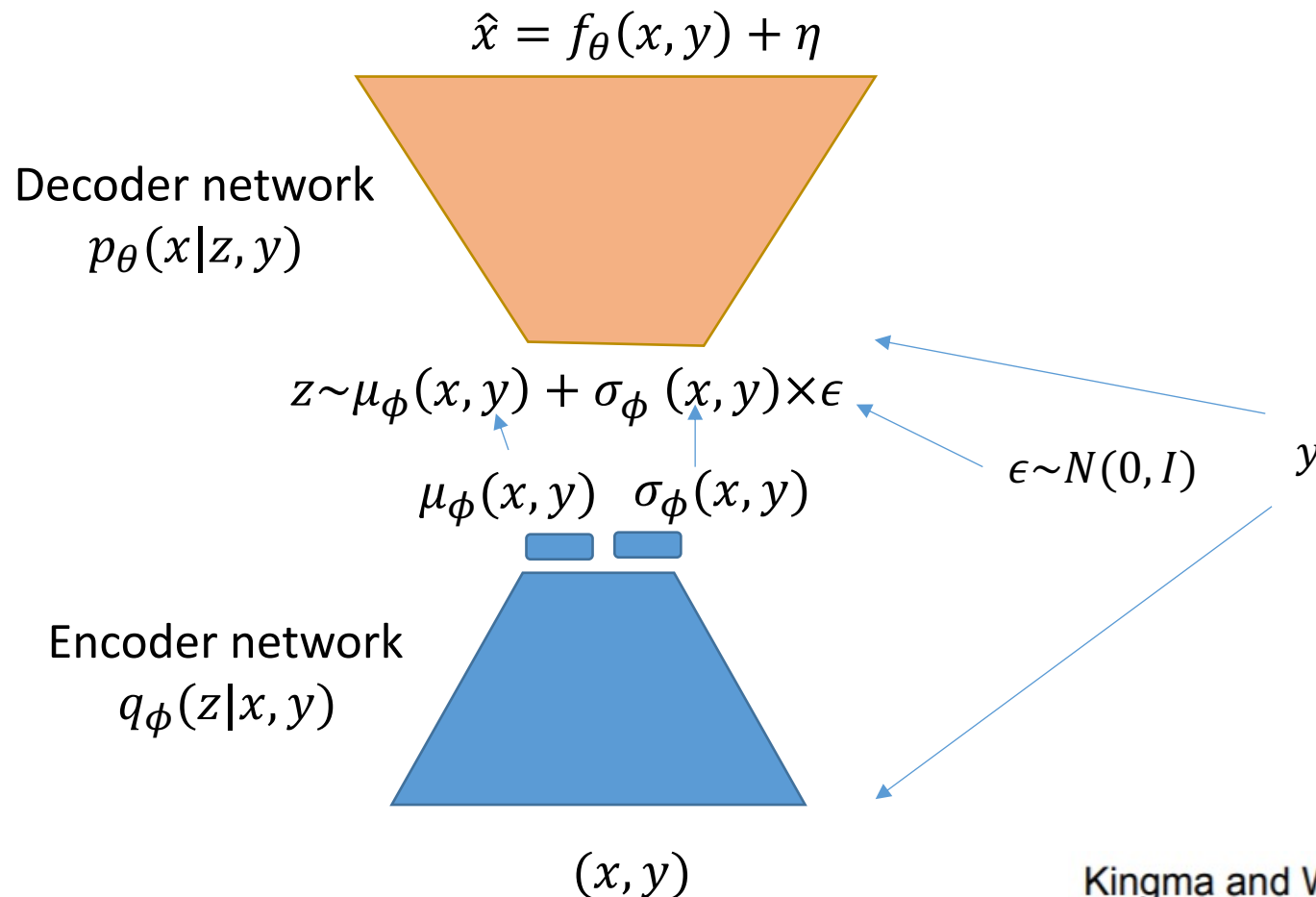
Go through the same KL divergence procedure, to get the same lower bound.

Lower bound of log conditional likelihood:

$$\begin{aligned} \log p(x|y) - D_{KL}(q(z|y, x) || p(z|y, x)) \\ = E_{z \sim q(z|x, y)} [\log p(x|y, z)] - D_{KL}(q(z|y, x) || p(z)) \end{aligned}$$

Conditional VAE (CVAE)

- Replace all $p(x|z)$ with $p(x|z,y)$
- Replace all $q(z|x)$ with $q(z|x,y)$.



Variational Autoencoders

- Probabilistic spin to traditional autoencoders => allows generating data
- Defines an intractable density => derive and optimize a (variational) lower bound
- Pros:
 - Principled approach to generative models
 - Allows inference of $q(z|x)$, can be useful feature representation for other tasks
- Cons:
 - Maximizes lower bound of likelihood: okay, but not as good evaluation as PixelRNN/PixelCNN
 - Samples blurrier and lower quality compared to state-of-the-art (GANs and diffusion models)
- Active areas of research:
 - More flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian
 - Incorporating structure in latent variables

Explicit Generative Models: So far

PixelCNNs define tractable density function, optimize likelihood of training data:

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

VAEs define intractable density function with latent $\mathbf{z}$:

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

Cannot optimize directly, derive and optimize lower bound on likelihood instead