THE UNIVERSITY OF BRITISH COLUMBIA

Topics in AI (CPSC 532S):
Multimodal Learning with Vision, Language and Sound

Lecture 16: Generative Models [part 2]

Logistics

Project Proposals Presentation Slides due **Today 11:59pm**

— Grades and comments by Monday (sorry!)

Project Proposal Document due **Tuesday, November 15th**

Assignment 4 is due today **Today 11:59pm**

Leftovers until the end of term:

— Assignment 5

— Project

— Paper presentation

PixelRNN and PixelCNN

PixelRNN

[van der Oord et al., 2016]

Explicit Density model

Use chain rule to decompose likelihood of an image $\boldsymbol{x}$ into product of (many) 1-d distributions

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

Likelihood of image $\boldsymbol{x}$

Probability of i'th pixel value given all previous pixels

then maximize likelihood of training data

PixelRNN

[van der Oord et al., 2016]

Explicit Density model

Use chain rule to decompose likelihood of an image $\boldsymbol{x}$ into product of (many) 1-d distributions

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

Likelihood of image $\boldsymbol{x}$

Probability of i'th pixel value given all previous pixels

Complex distribution over pixel values, so lets model using **neural network**

then maximize likelihood of training data

PixelRNN

[van der Oord et al., 2016]

Explicit Density model

Use chain rule to decompose likelihood of an image $\boldsymbol{x}$ into product of (many) 1-d distributions

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

Likelihood of image $\boldsymbol{x}$

Probability of i'th pixel value given all previous pixels

then maximize likelihood of training data

Complex distribution over pixel values, so lets model using **neural network**

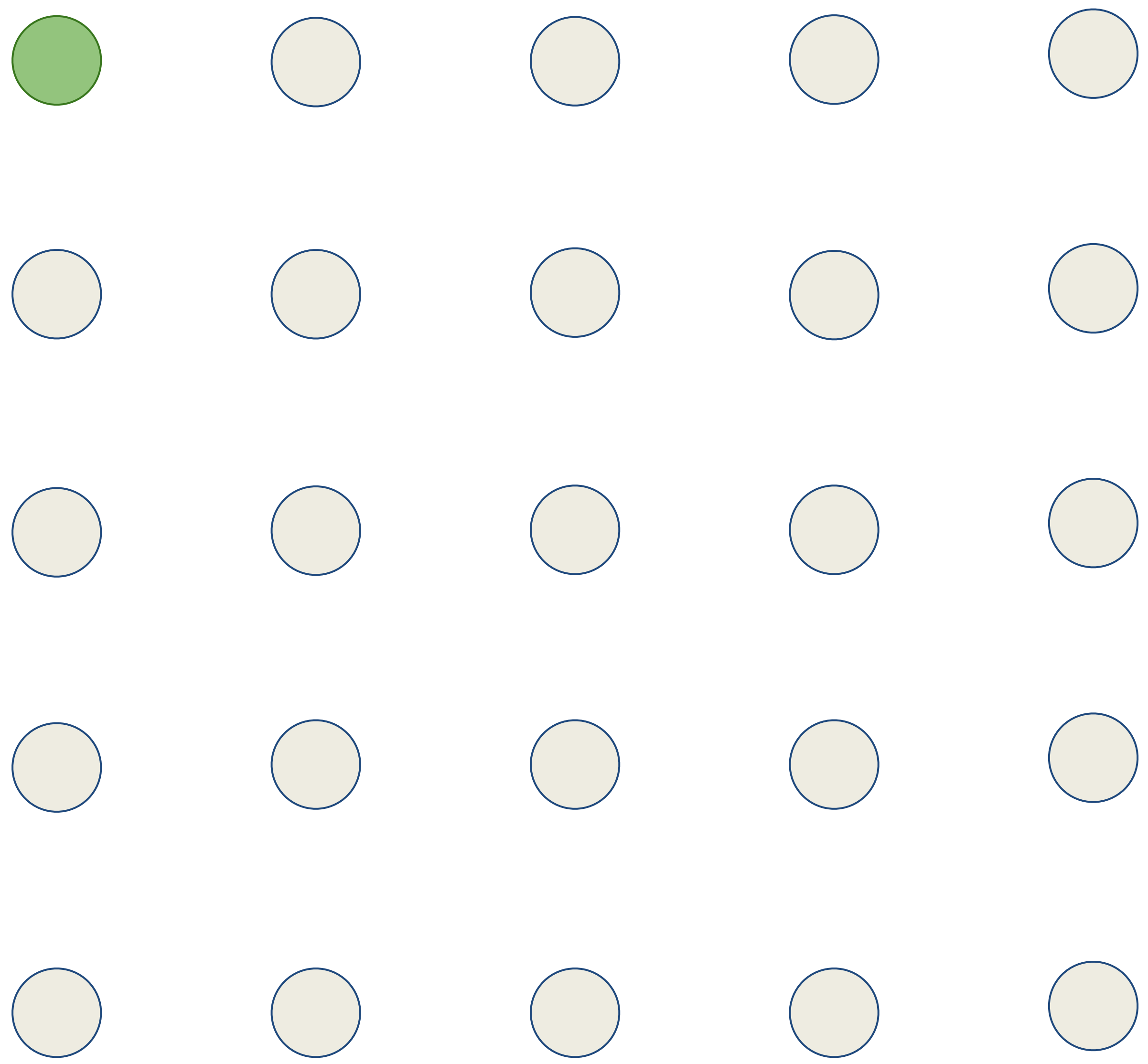
Also requires defining **ordering** of "previous pixels"

PixelRNN

[van der Oord et al., 2016]

Generate image pixels starting from the corner

Dependency on previous pixels model using an RNN (LSTM)



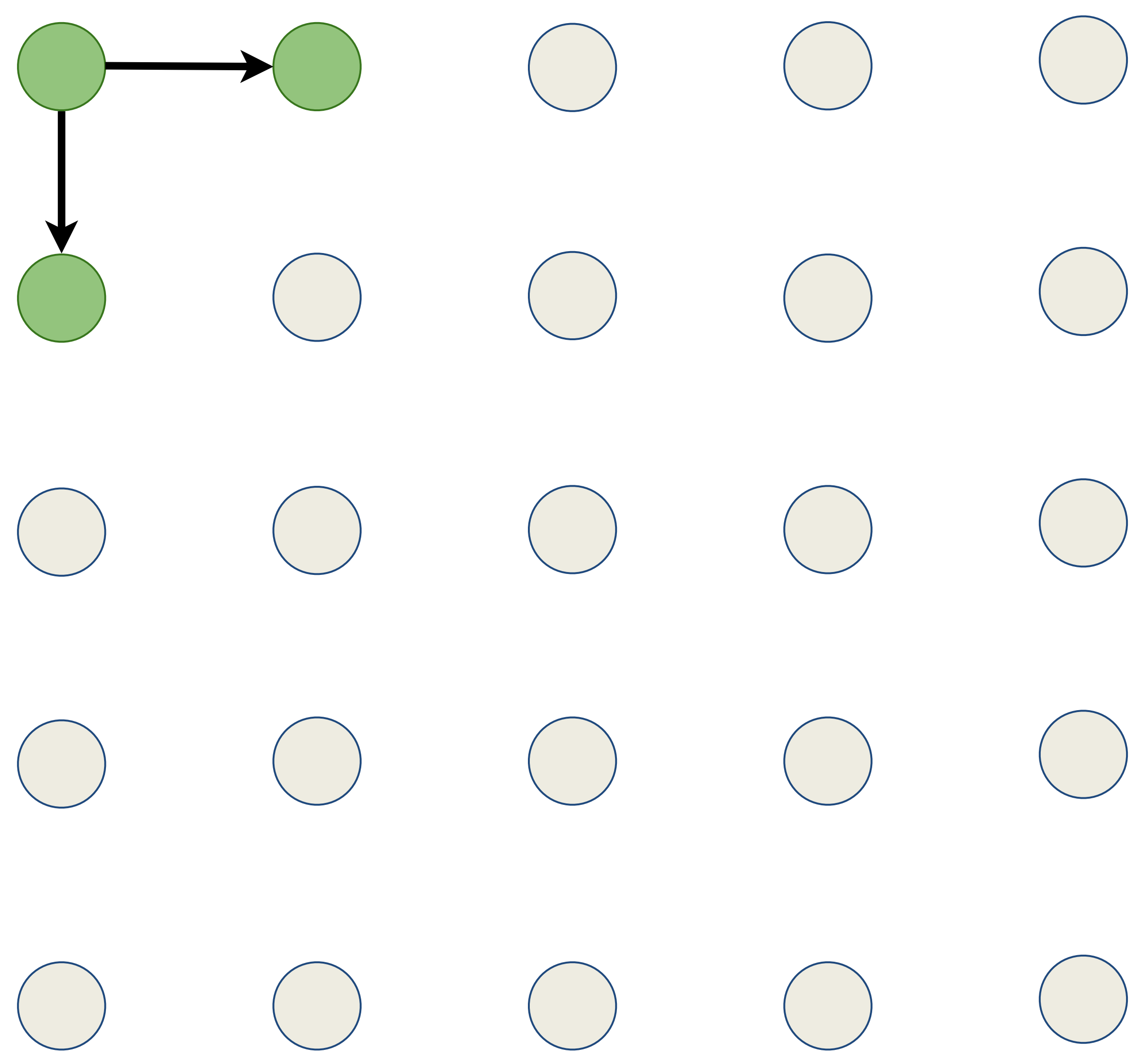
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

PixelRNN

[van der Oord et al., 2016]

Generate image pixels starting from the corner

Dependency on previous pixels model using an RNN (LSTM)



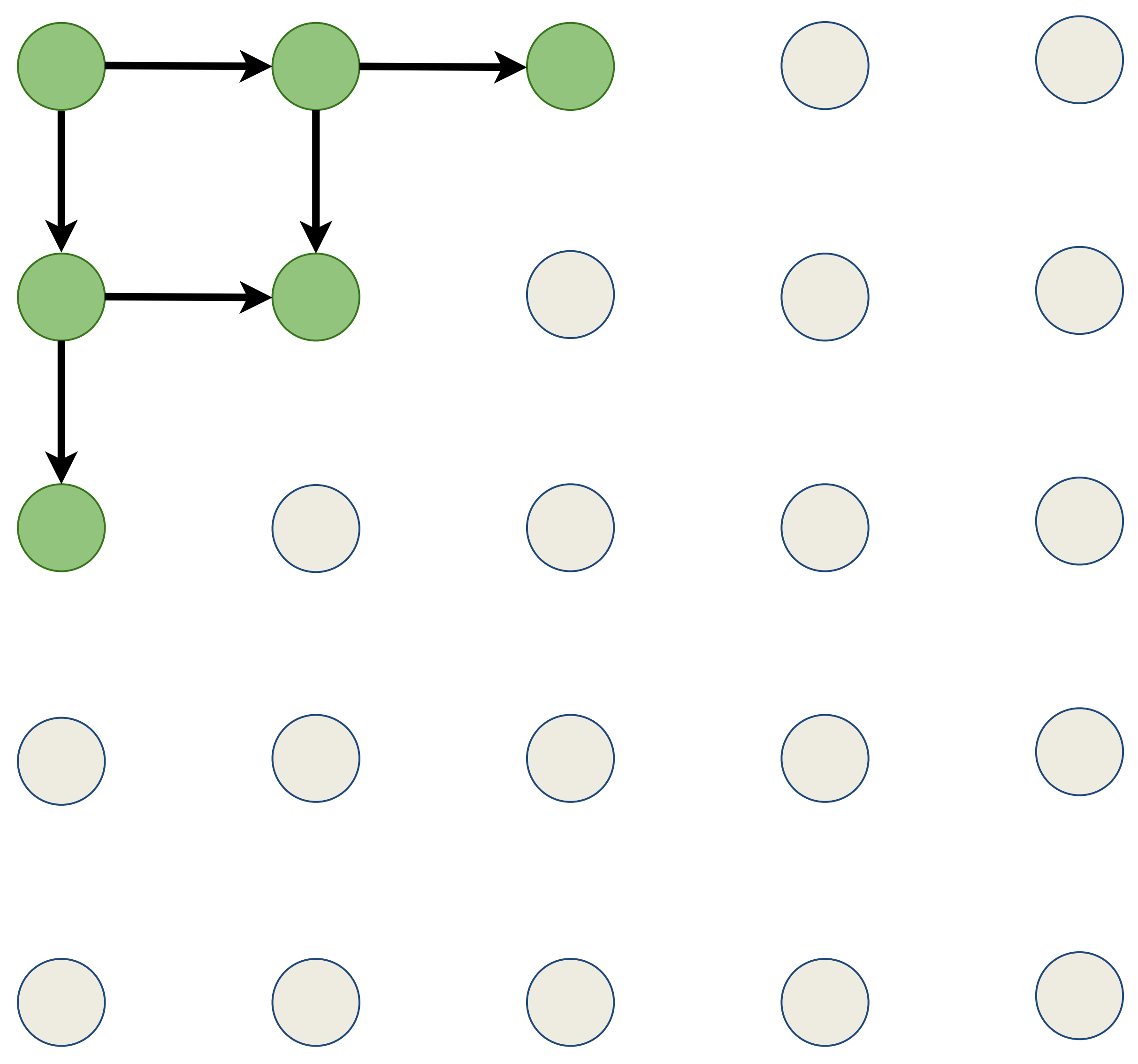
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

PixelRNN

[van der Oord et al., 2016]

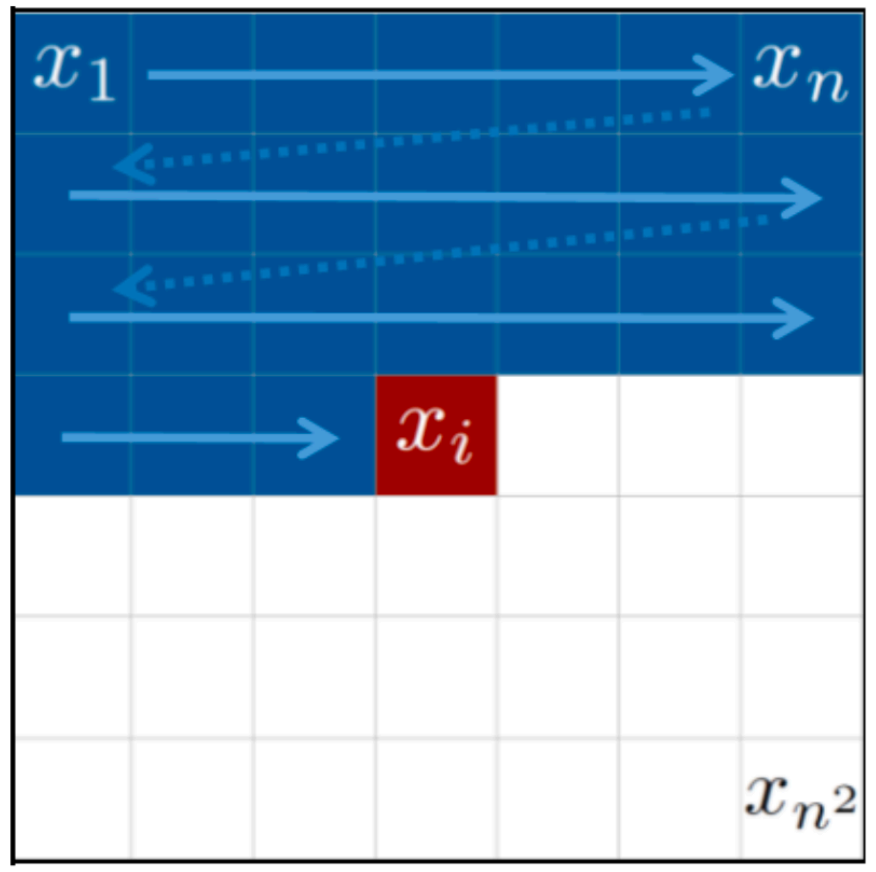
Generate image pixels starting from the corner

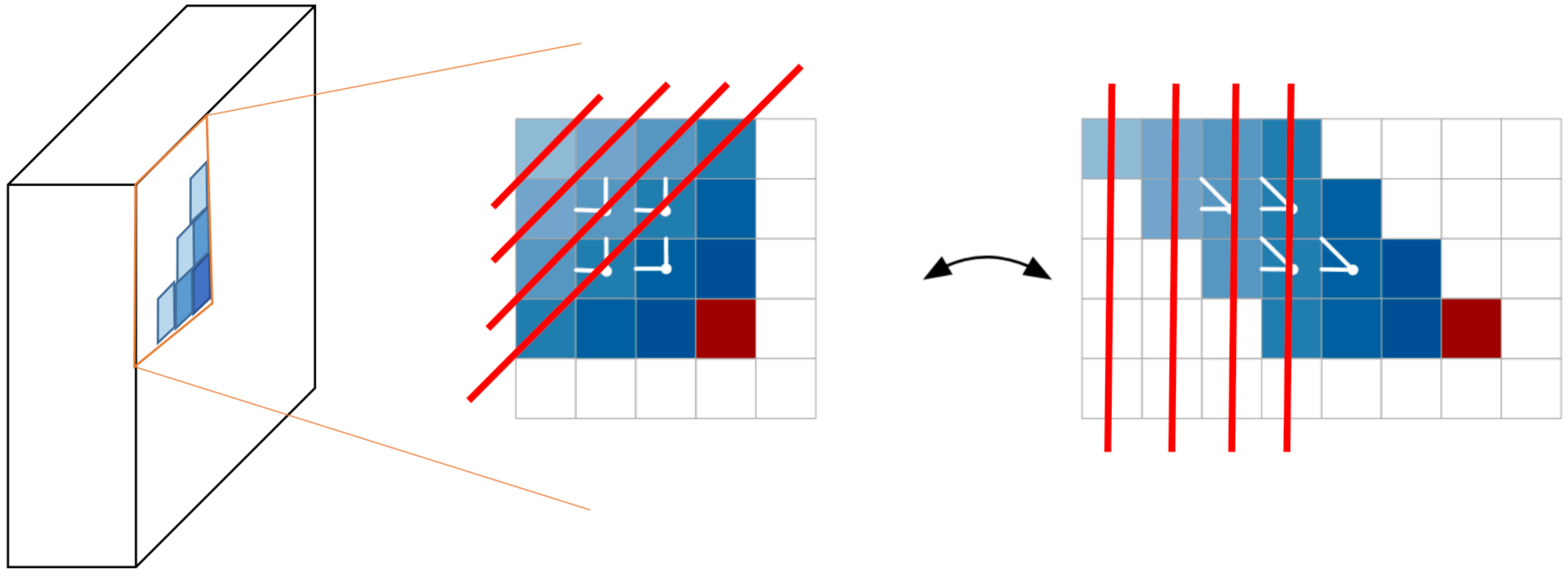
Dependency on previous pixels model using an RNN (LSTM)



PixelRNN

[van der Oord et al., 2016]





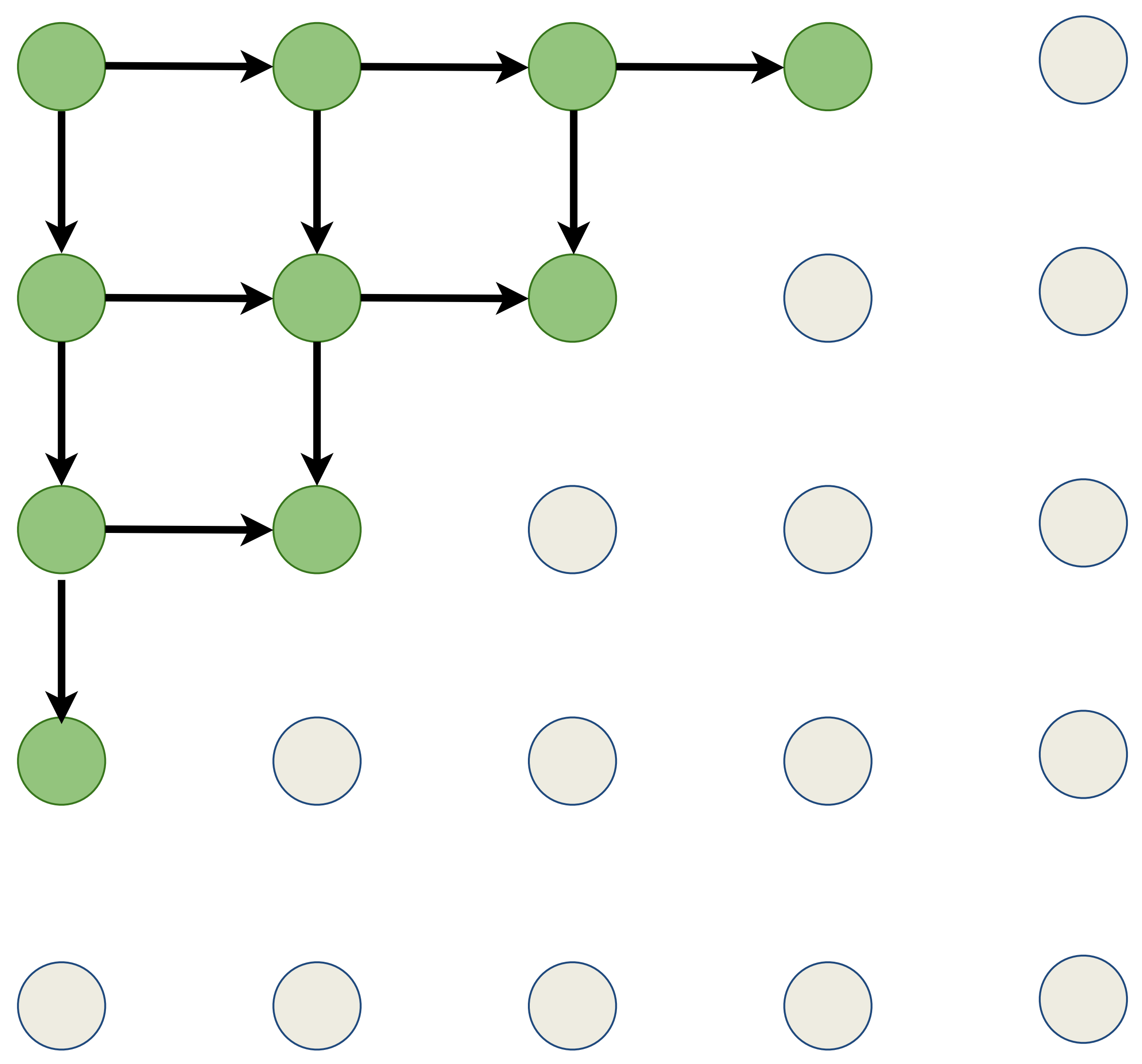
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

PixelRNN

[van der Oord et al., 2016]

Generate image pixels starting from the corner

Dependency on previous pixels model using an RNN (LSTM)

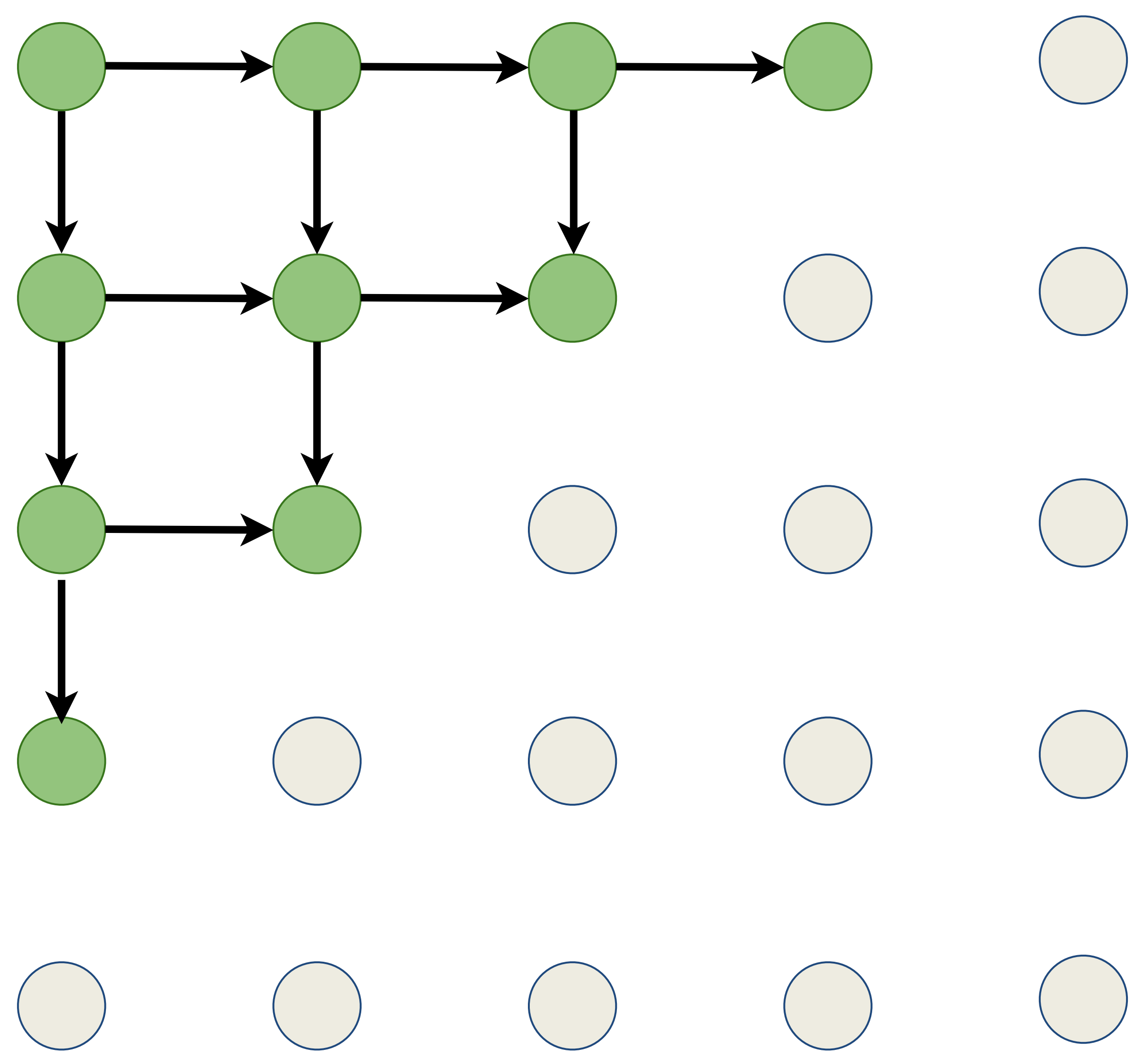


PixelRNN

[van der Oord et al., 2016]

Generate image pixels starting from the corner

Dependency on previous pixels model using an RNN (LSTM)



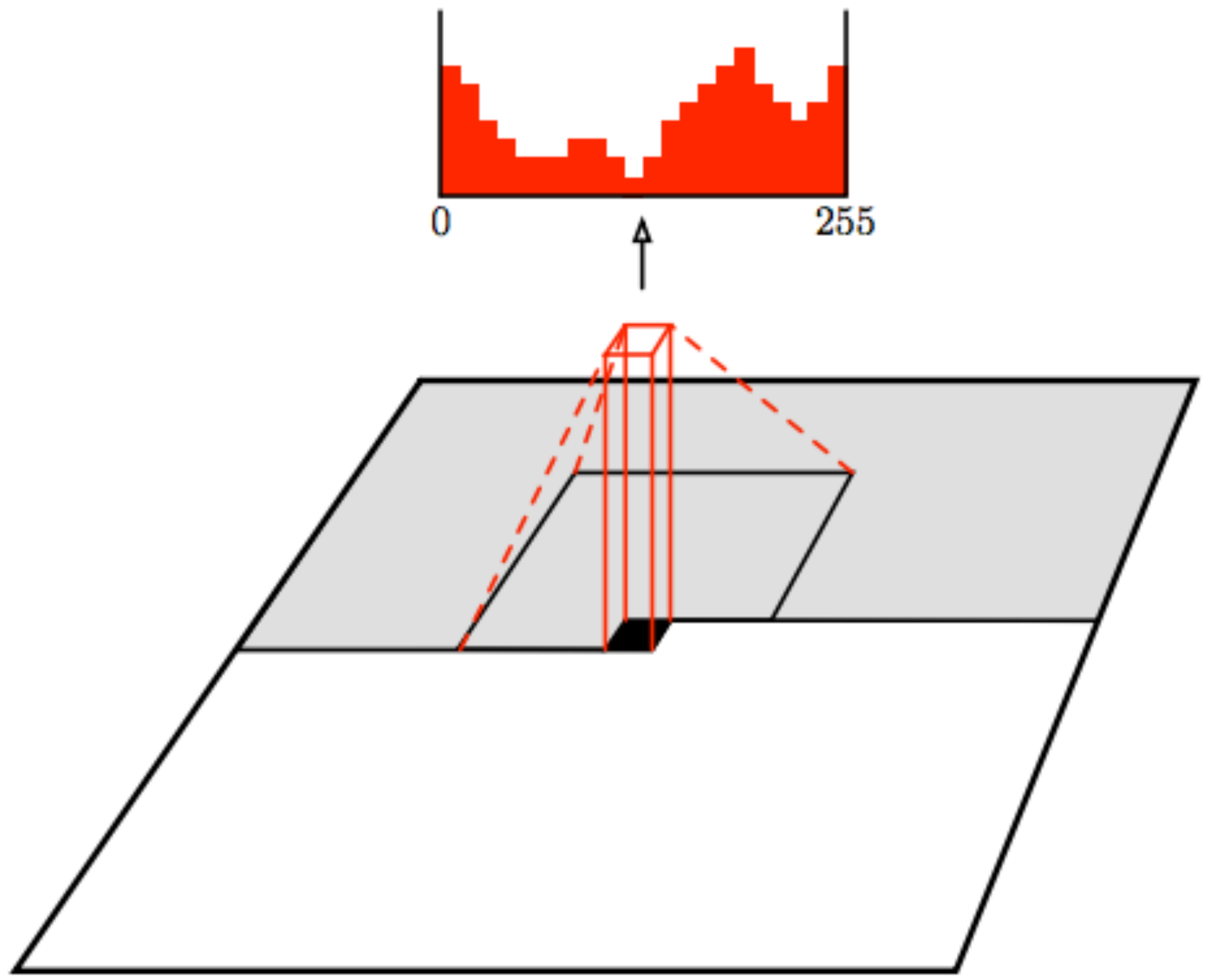
Problem: sequential generation is slow

PixelCNN

[van der Oord et al., 2016]

Still generate image pixels starting from the corner

Dependency on previous pixels now modeled using a CNN over context region



PixelCNN

[van der Oord et al., 2016]

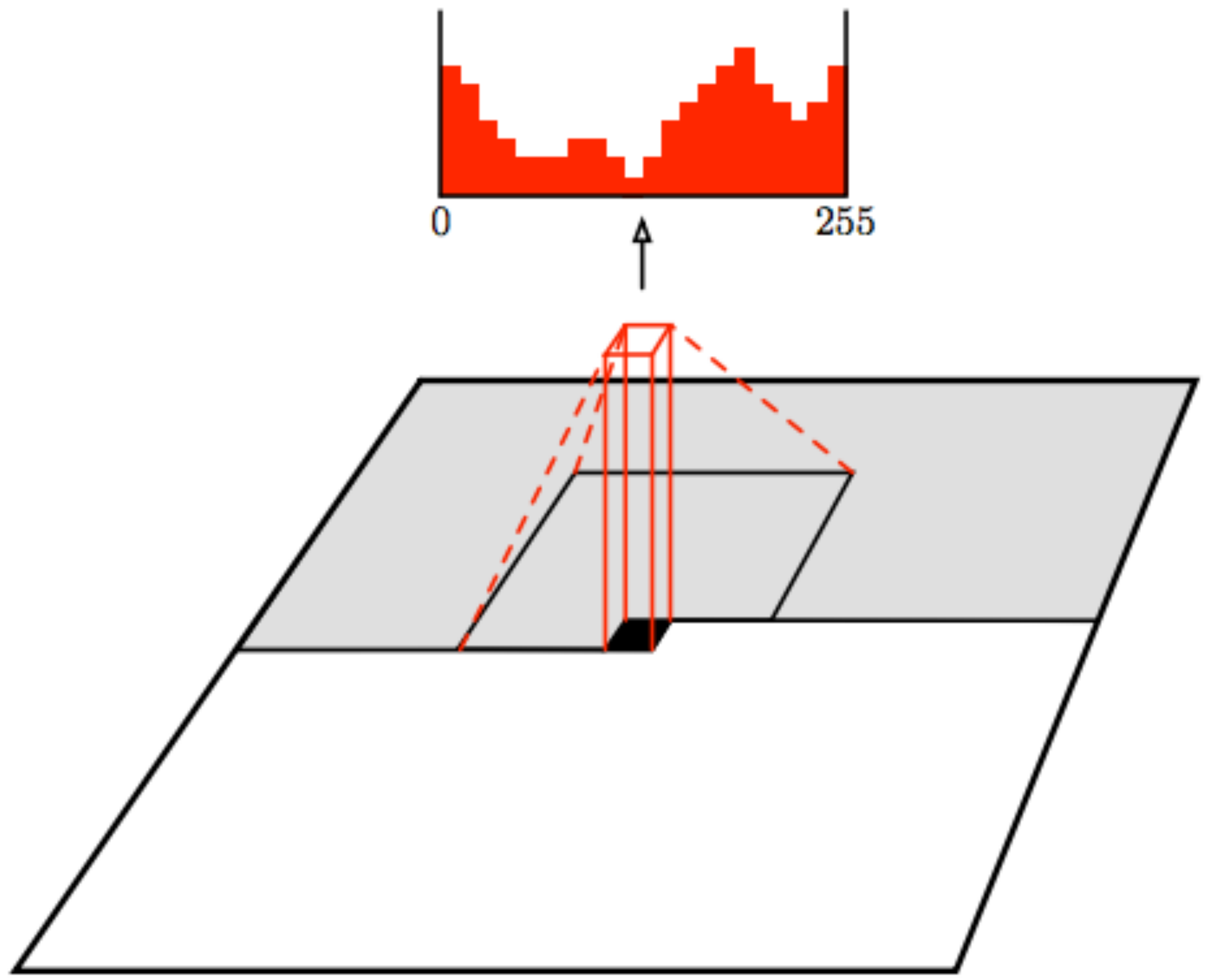
Still generate image pixels starting from the corner

Dependency on previous pixels now modeled using a CNN over context region

Training: maximize likelihood of training images

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

Softmax loss at each pixel



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…

Variational Autoencoders (VAE)

So far …

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

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

So far …

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

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

VAEs define intractable density function with latent variables z (that we need to marginalize):

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

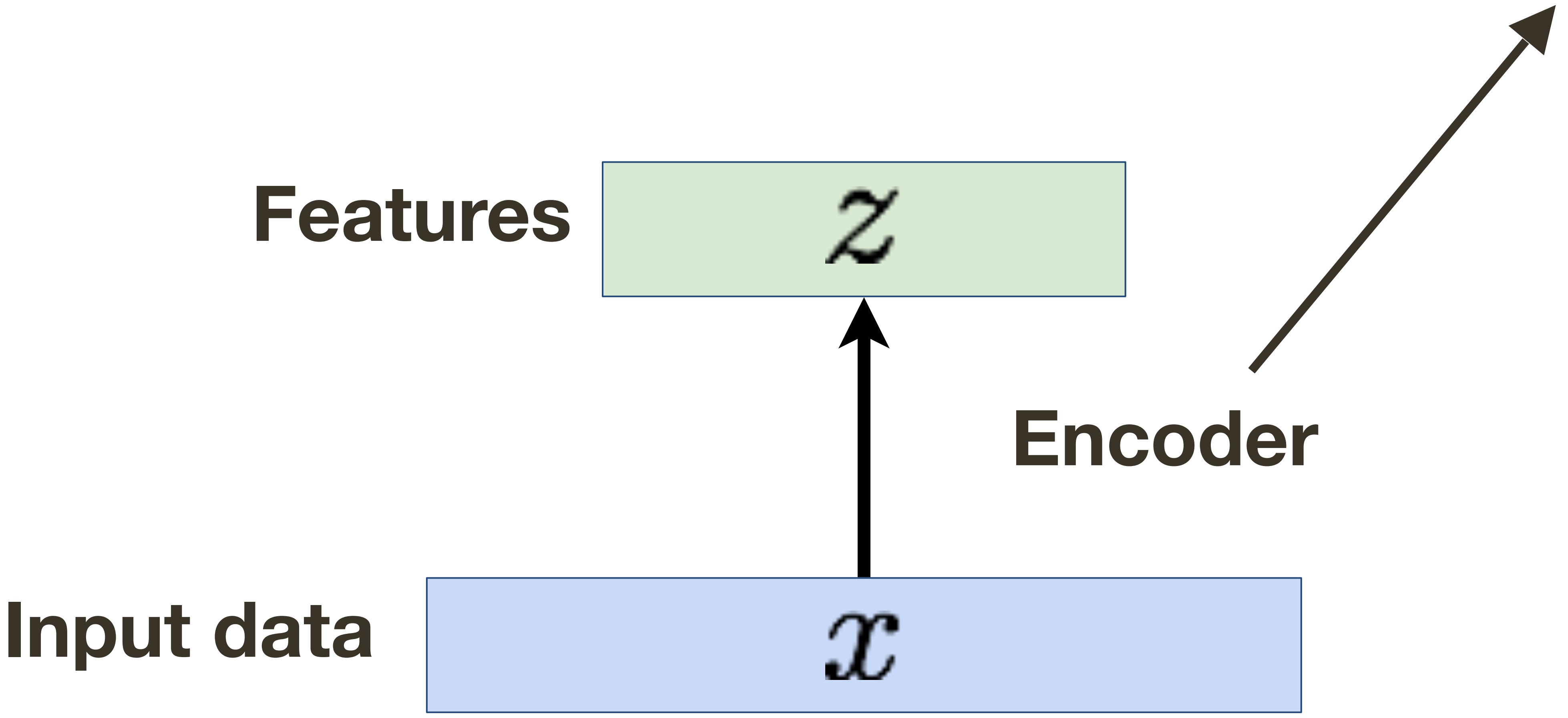
cannot optimize directly, derive and optimize lower bound of likelihood instead

* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

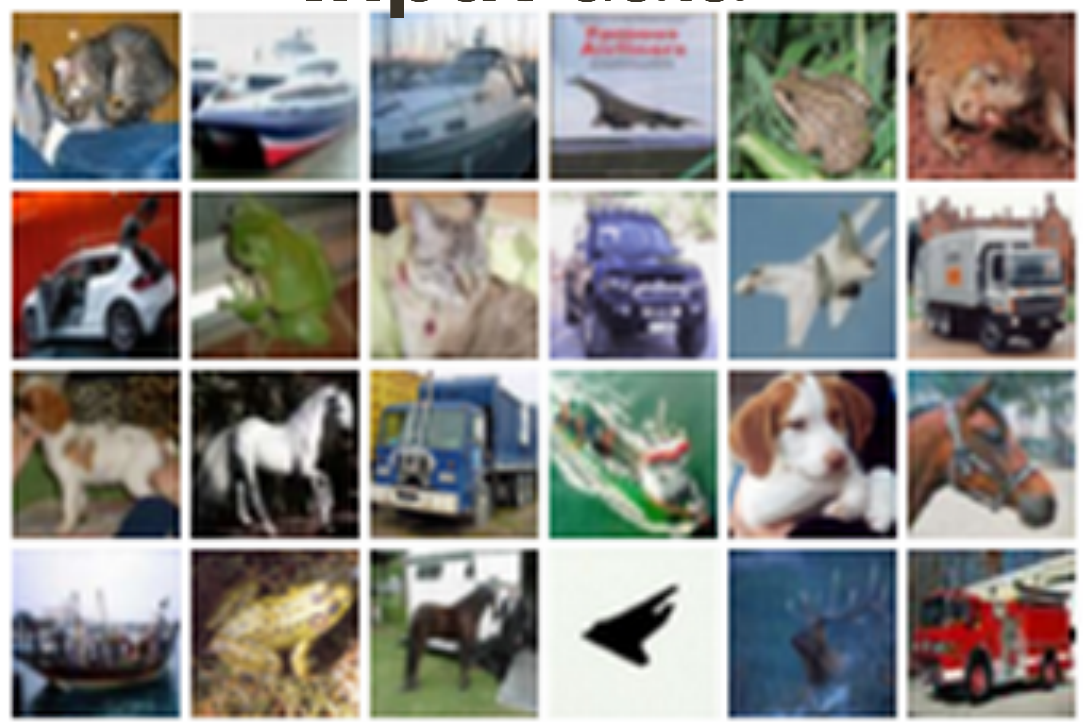
Autoencoders Reminder …

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

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



Input data



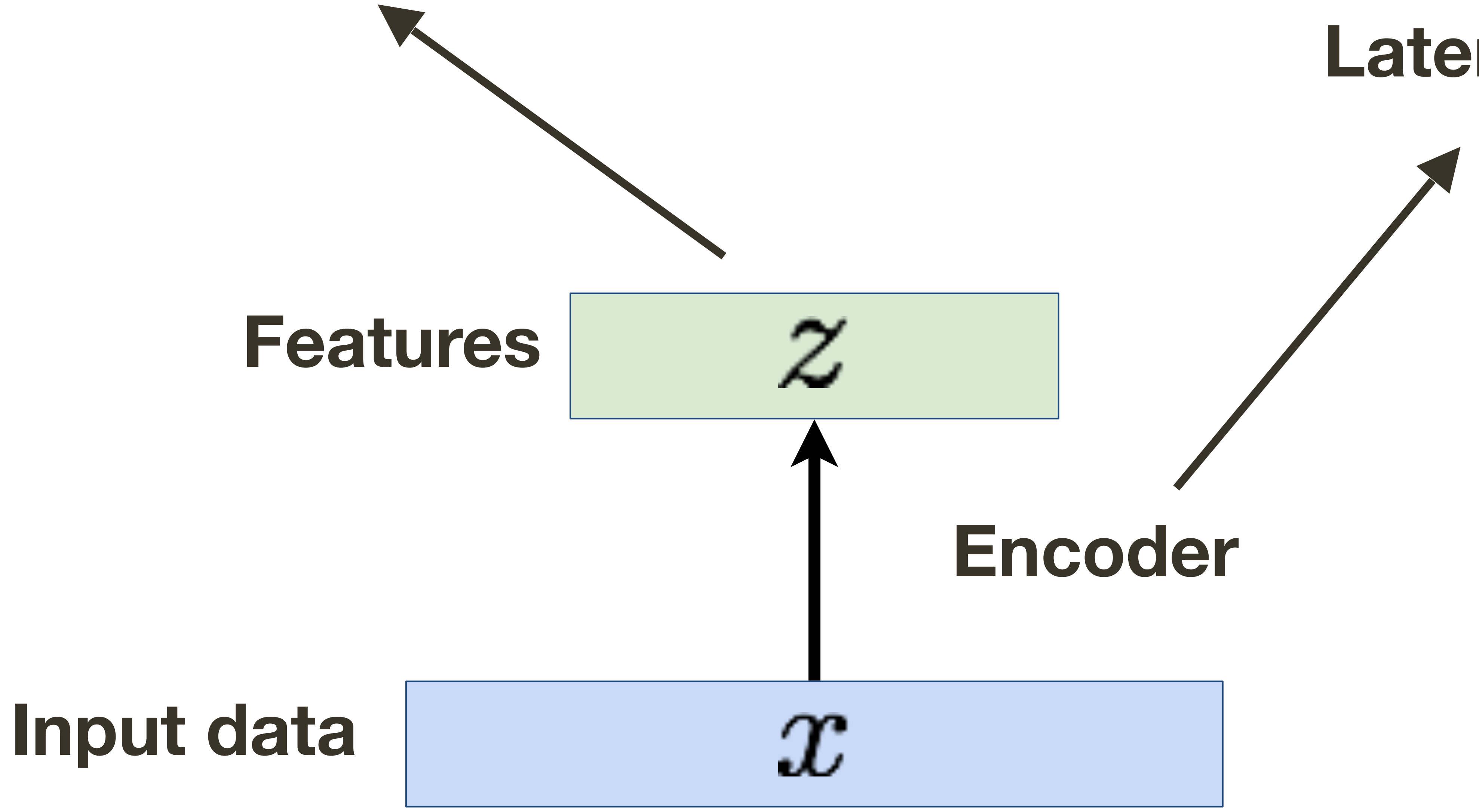
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Autoencoders Reminder …

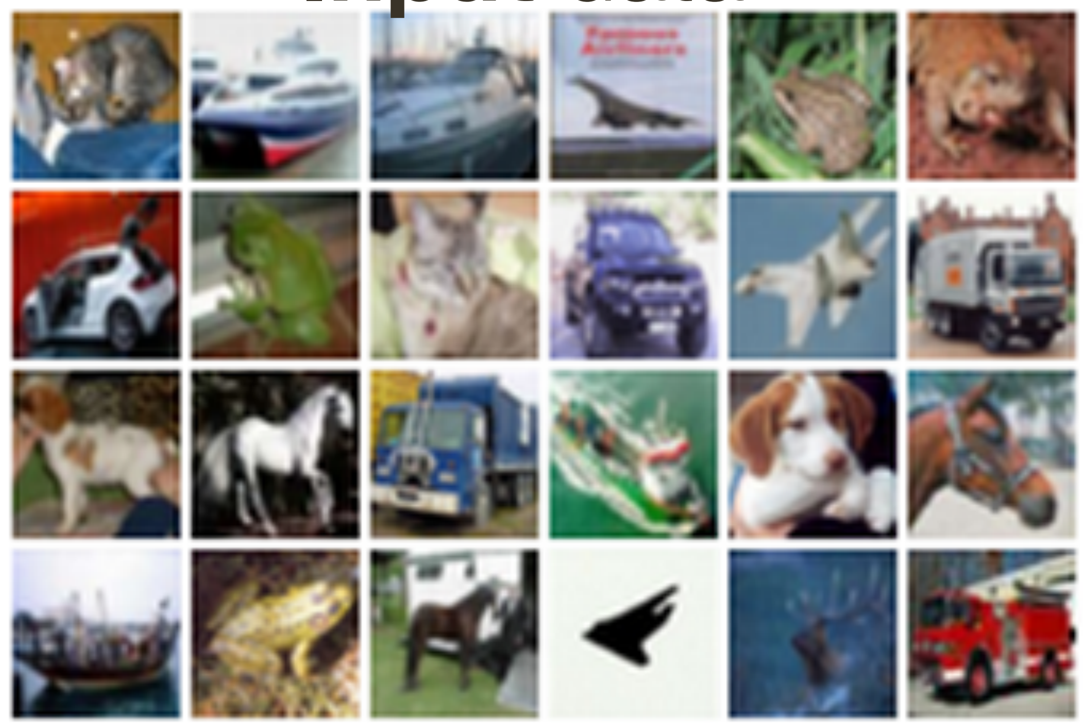
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

z usually smaller than x
(dimensionality reduction)

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



Input data



* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

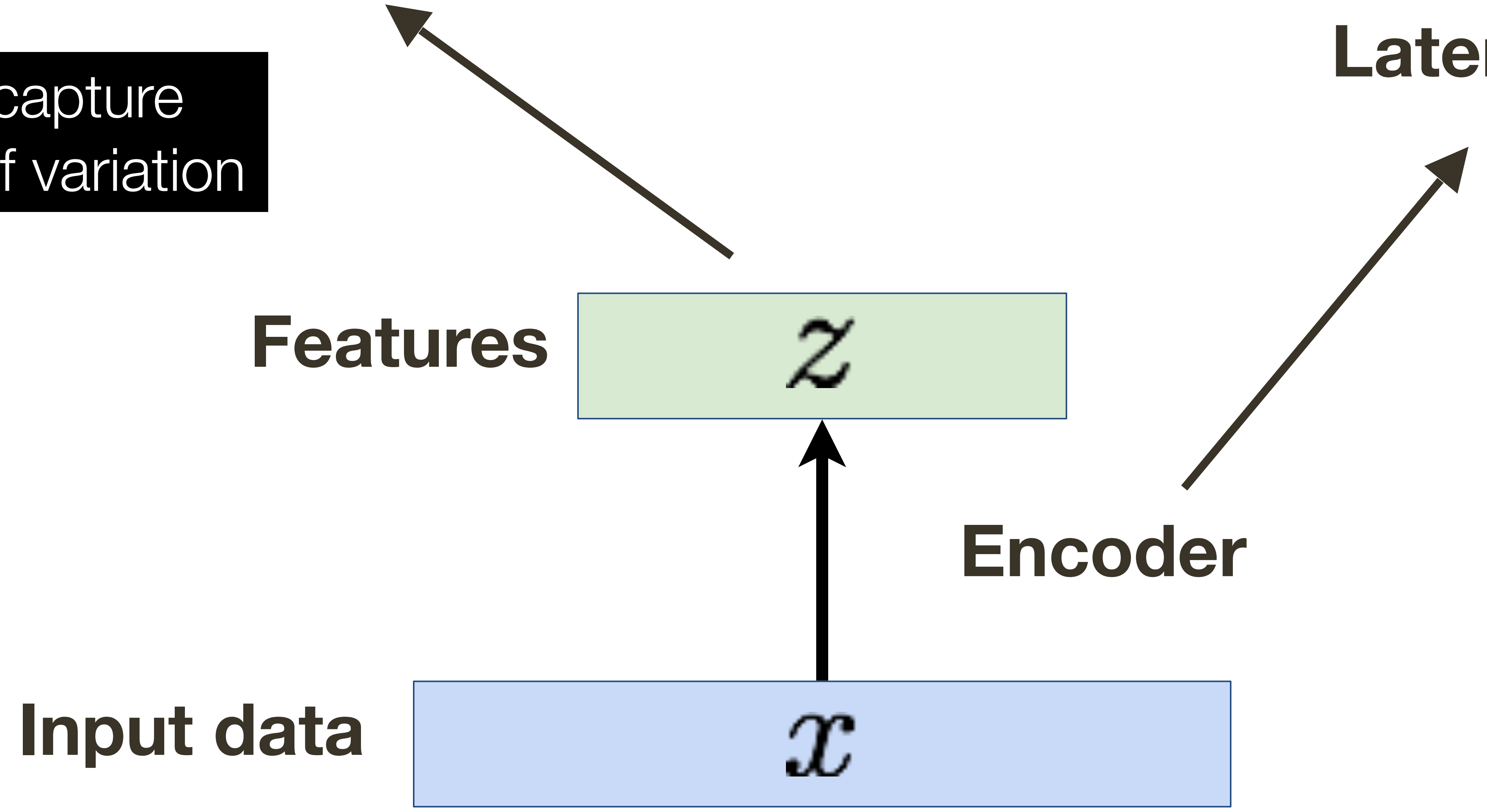
Autoencoders Reminder …

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

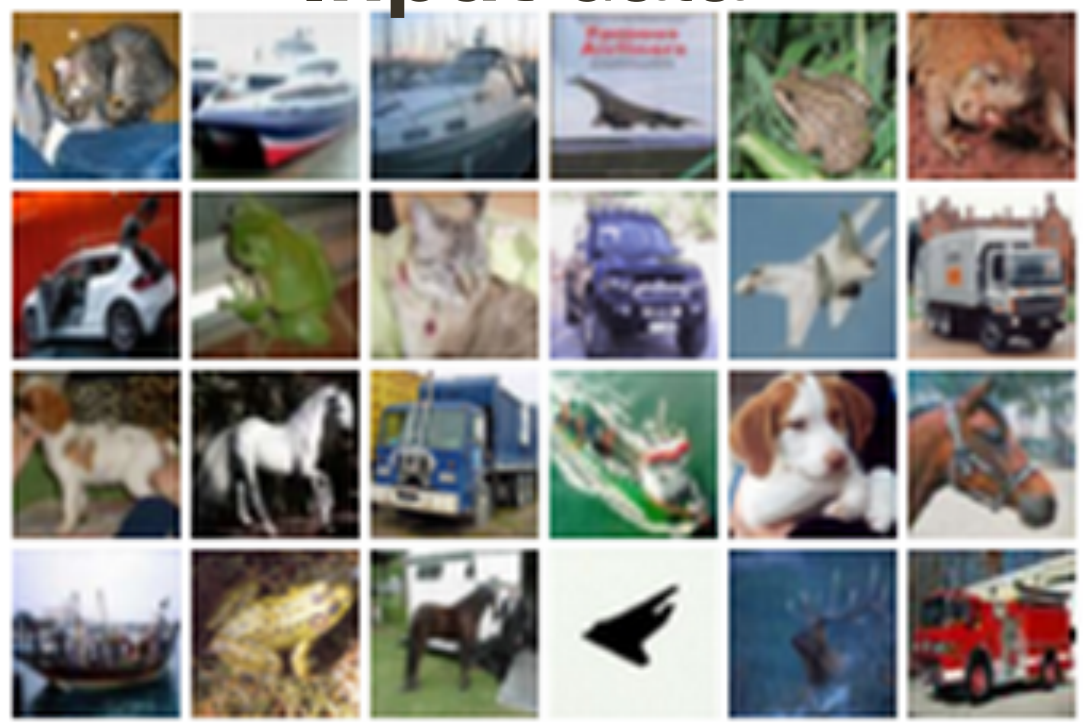
z usually smaller than x
(dimensionality reduction)

Want features that capture **meaningful** factors of variation

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



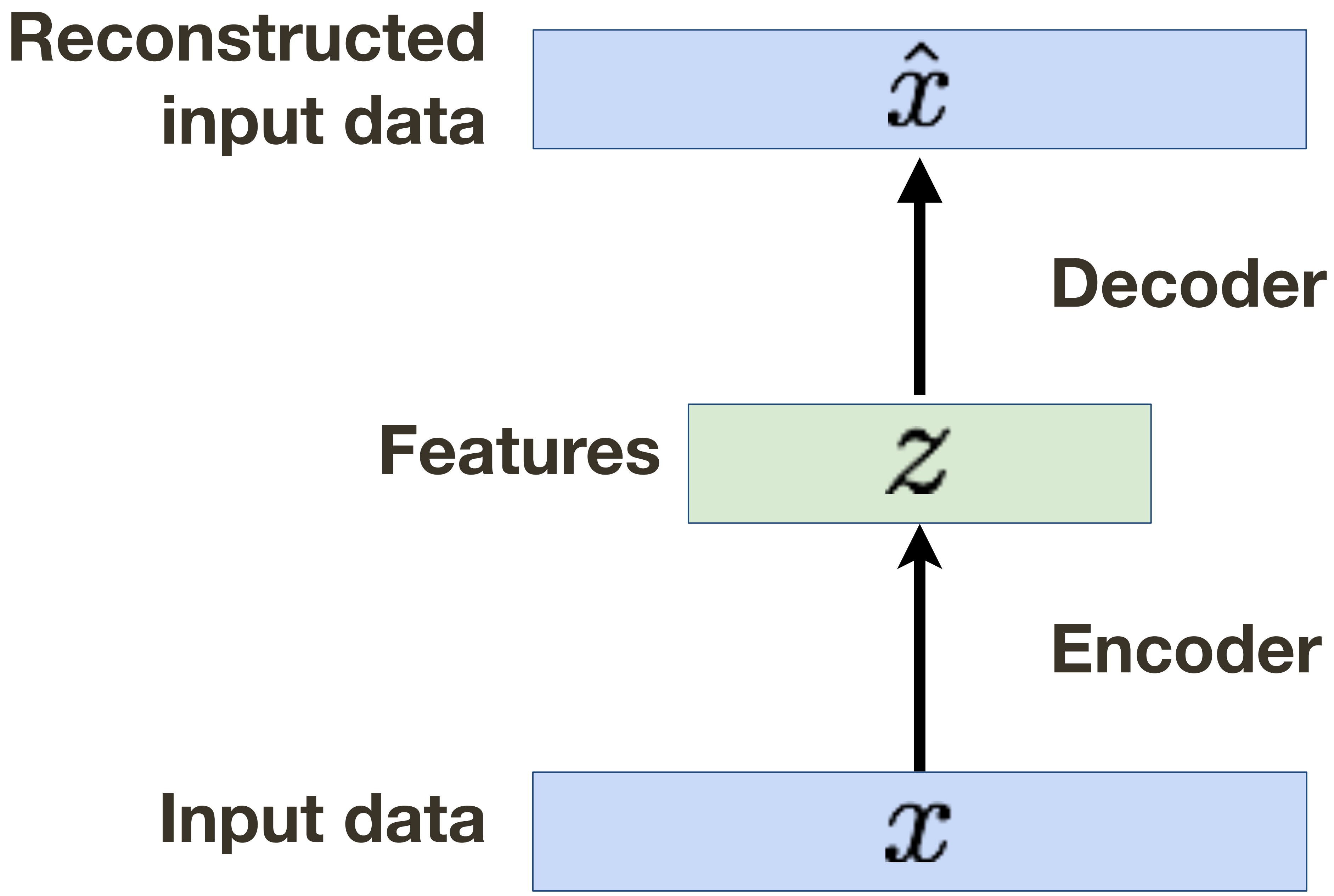
Input data



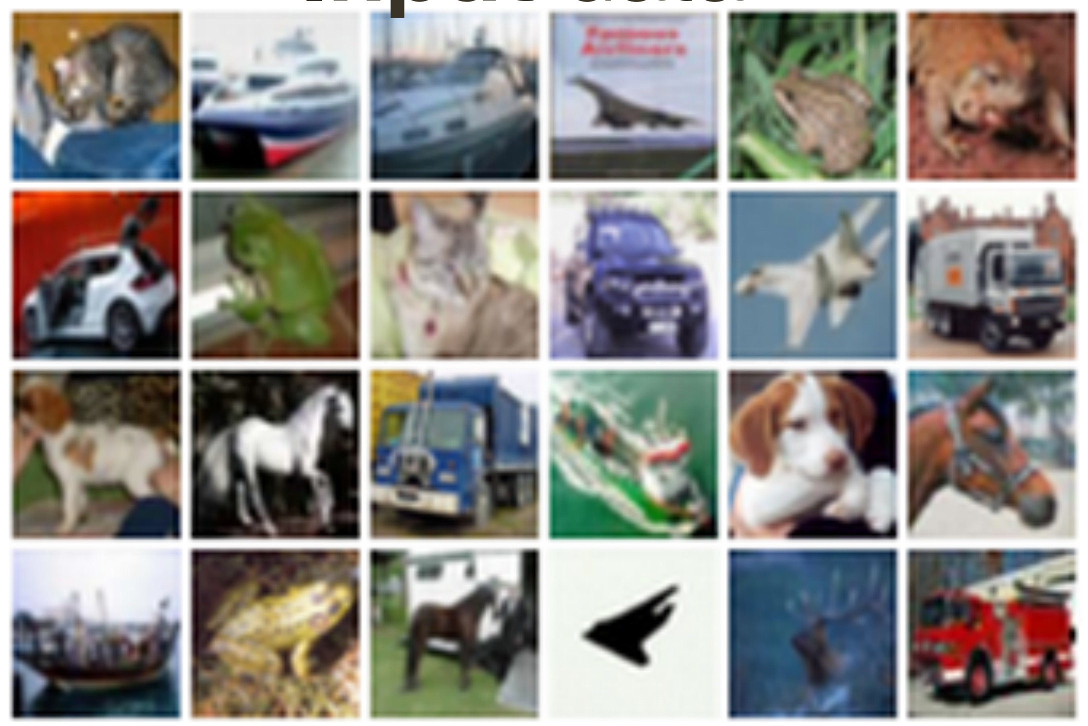
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Autoencoders Reminder …

Train such that features can reconstruct original data best they can

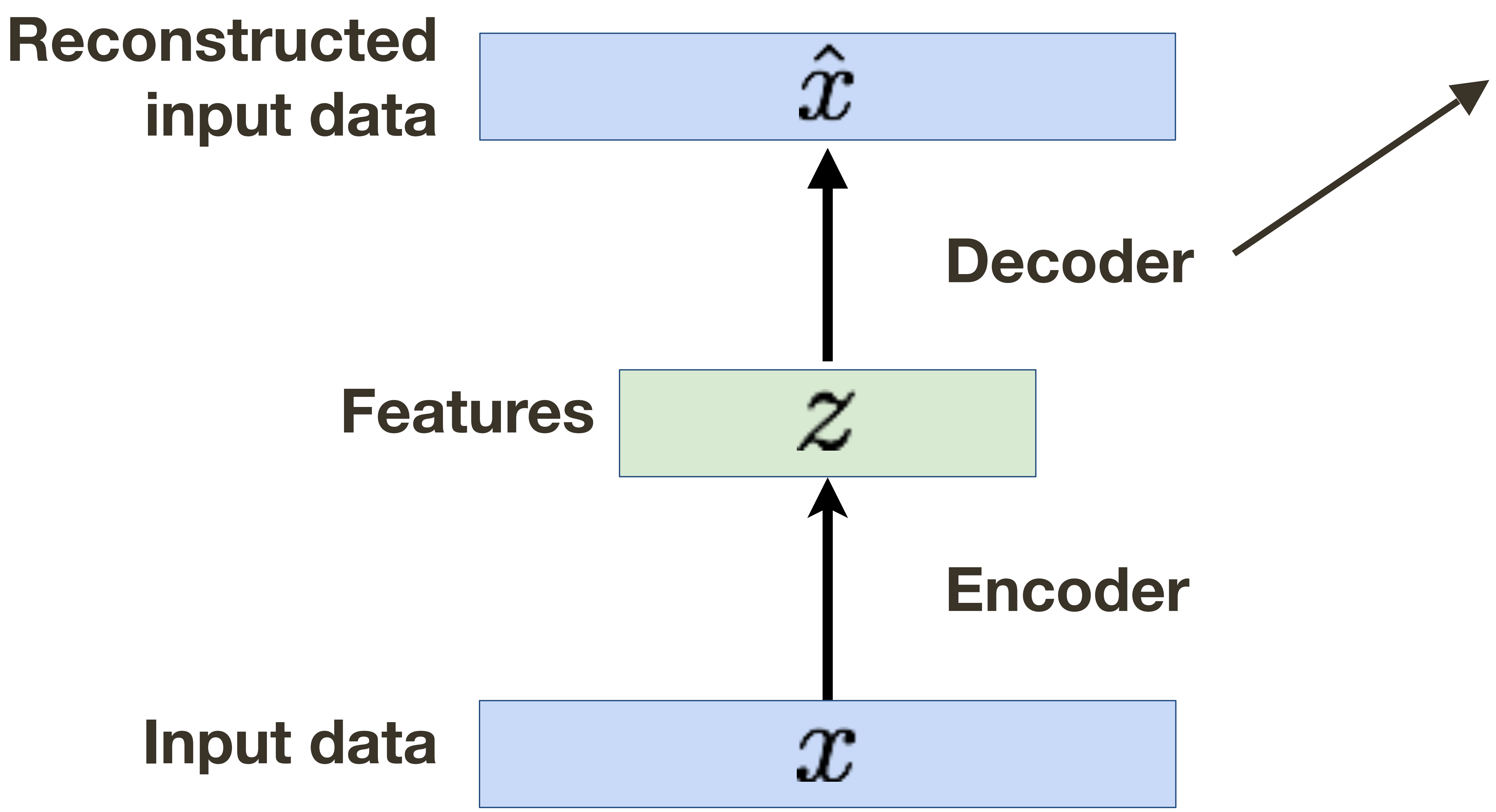


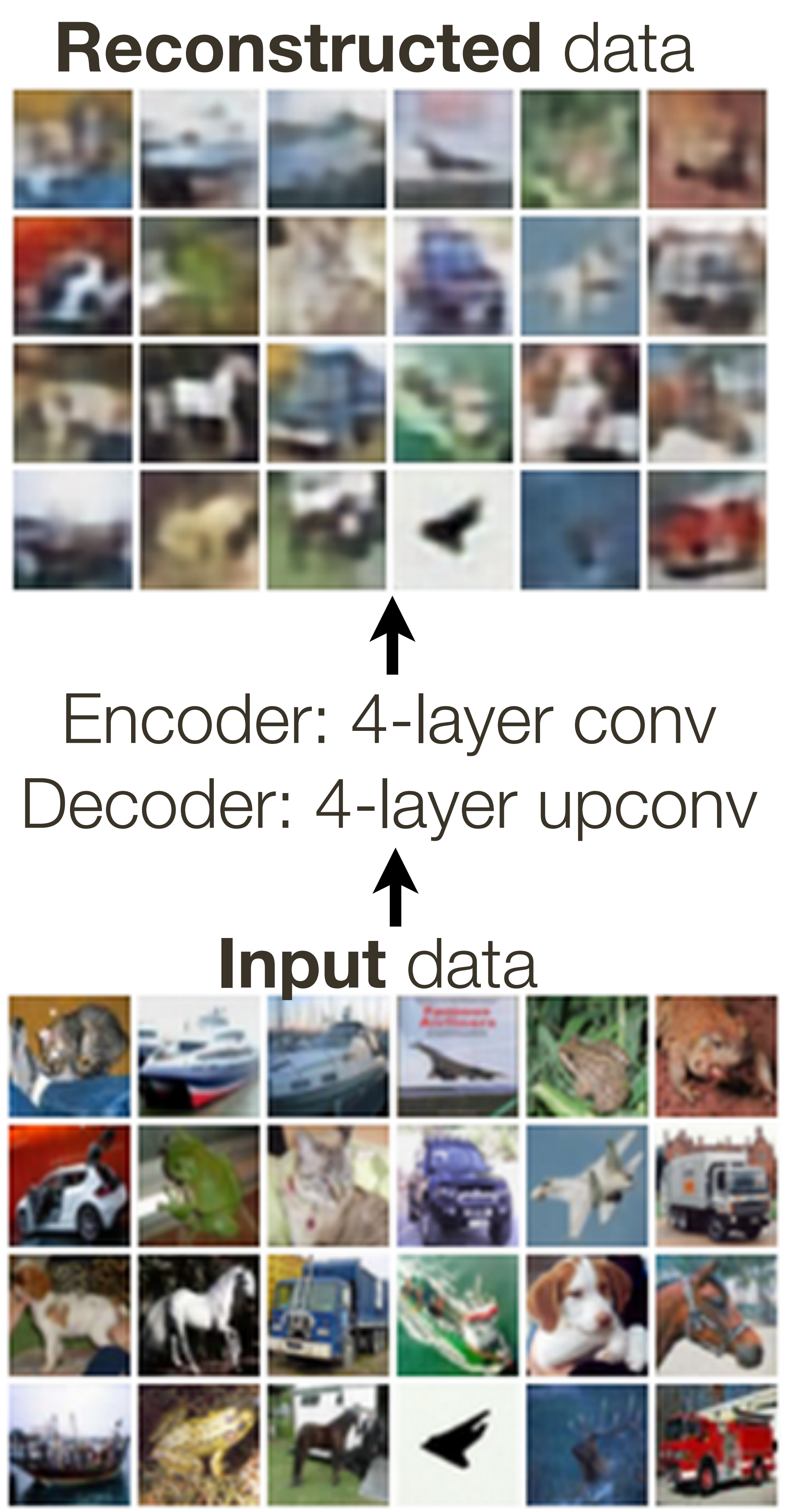
Input data



Autoencoders Reminder …

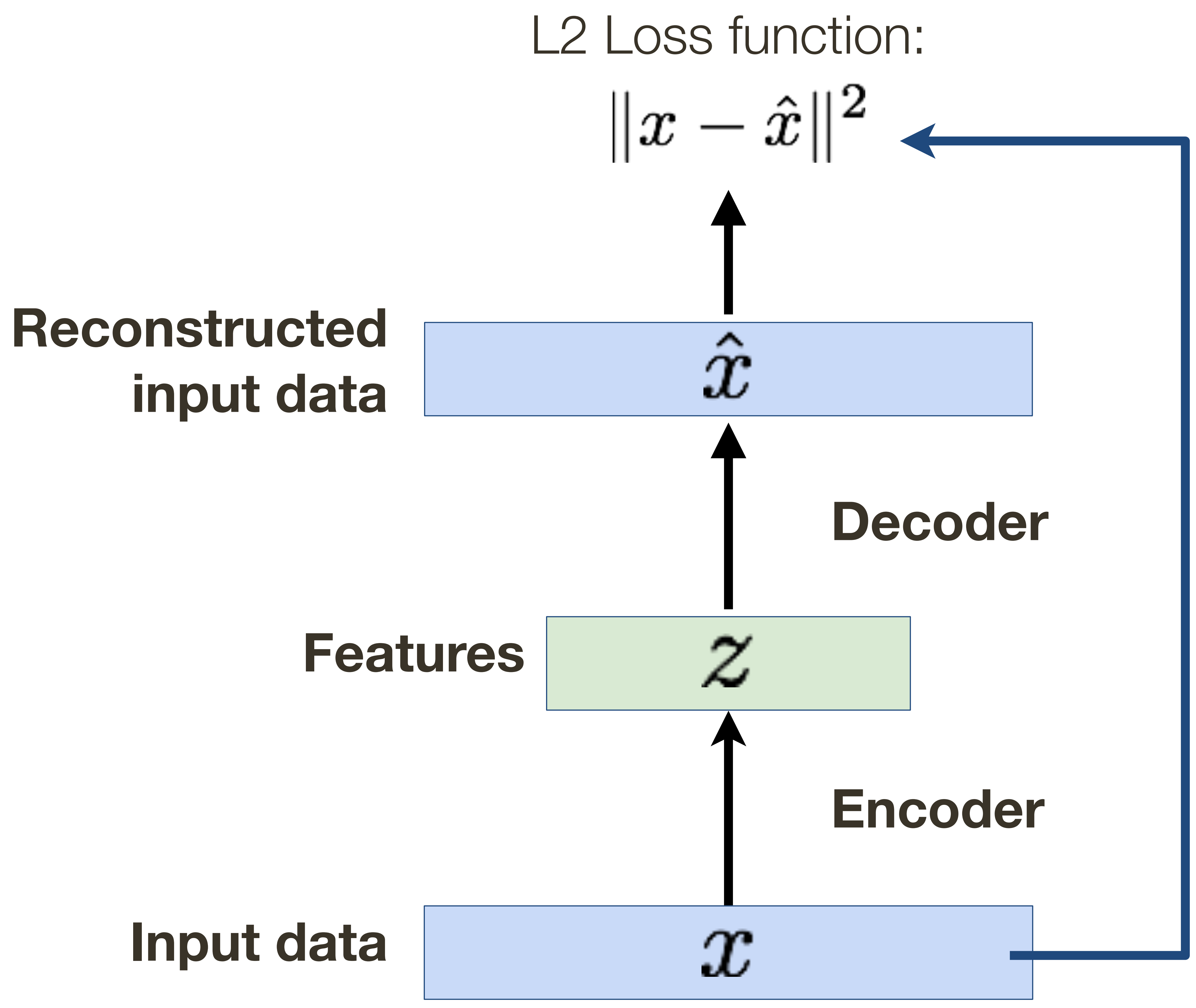
Train such that features can reconstruct original data best they can

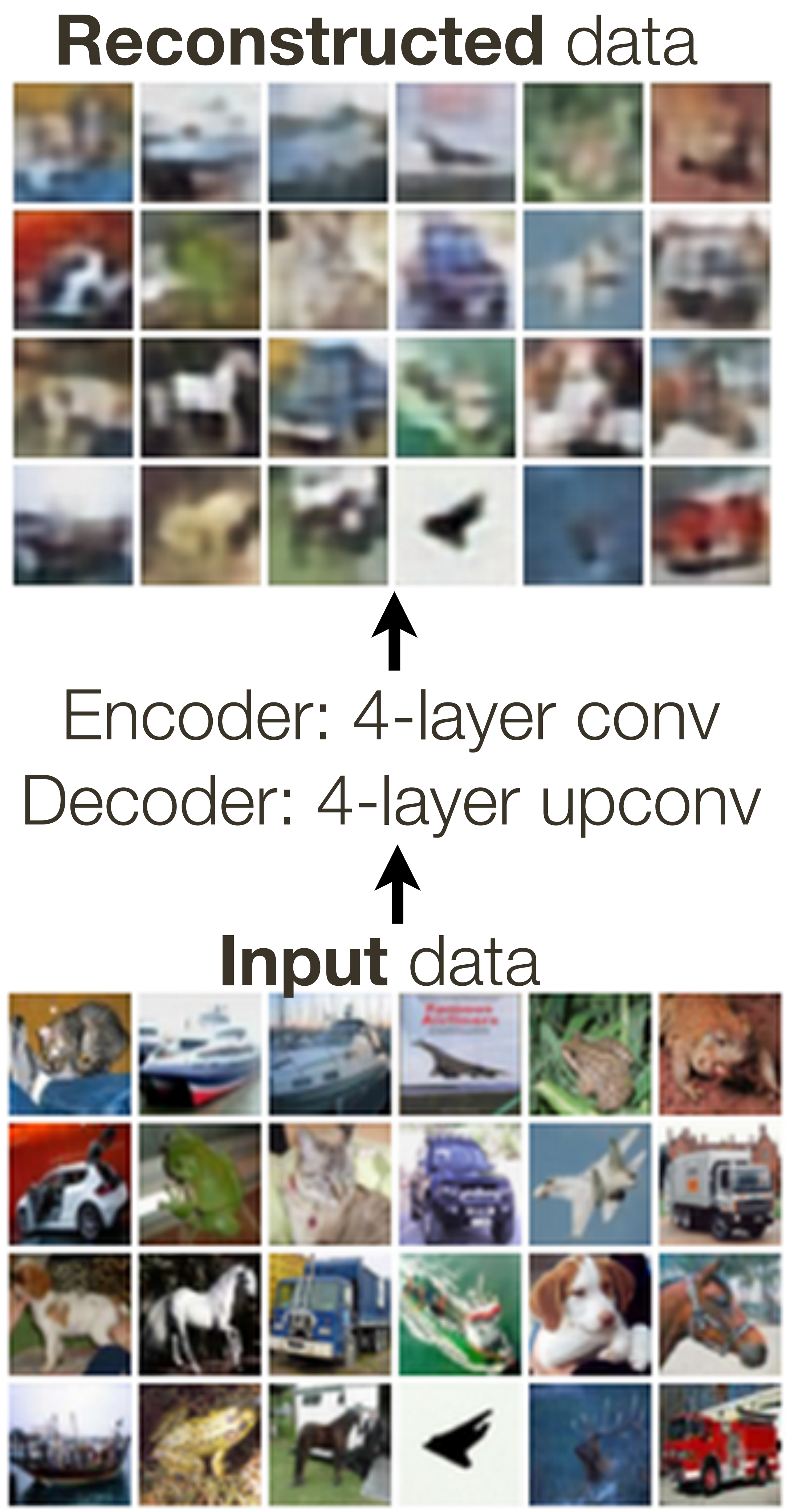




* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

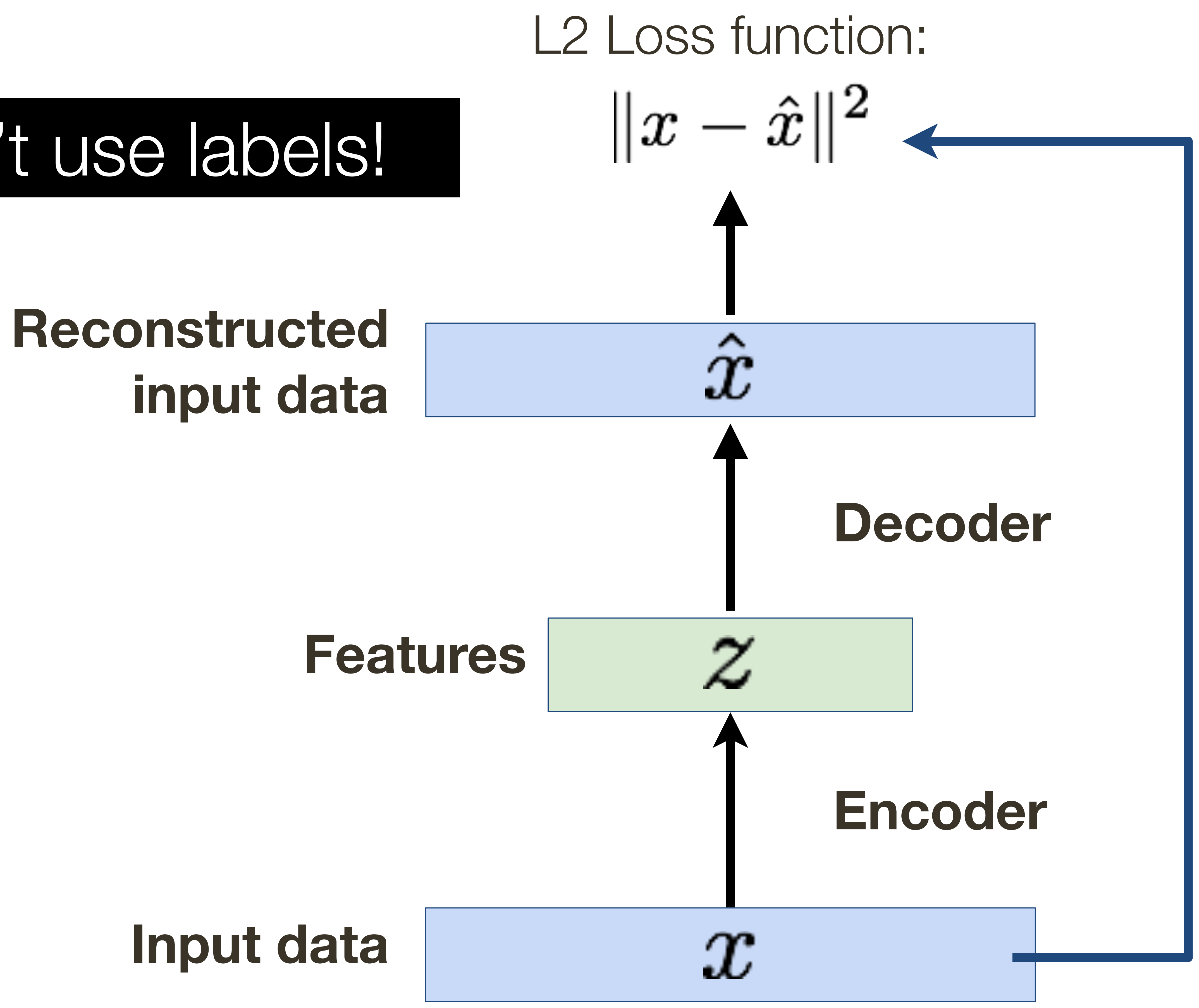
Autoencoders Reminder …

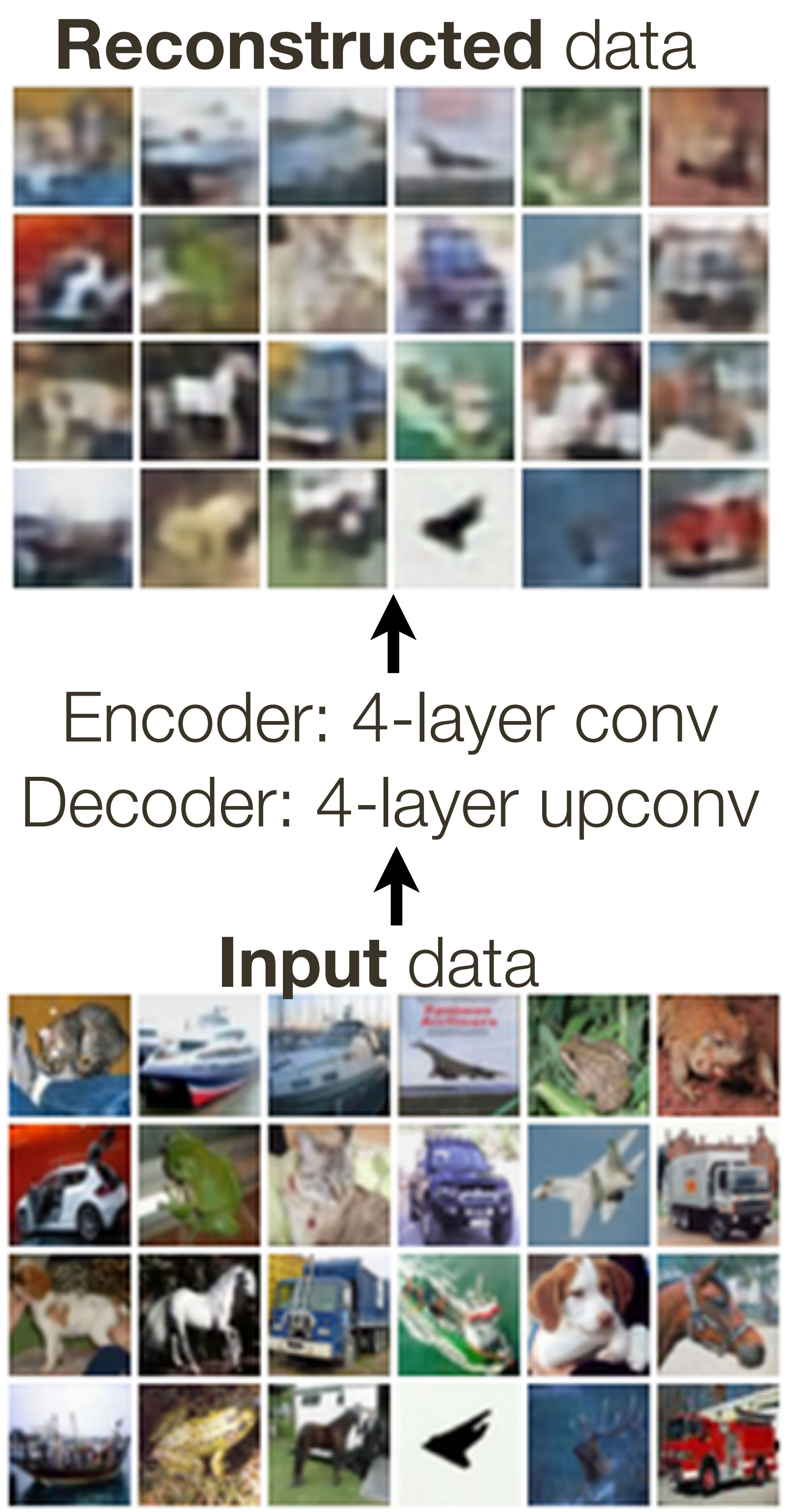




Autoencoders Reminder …

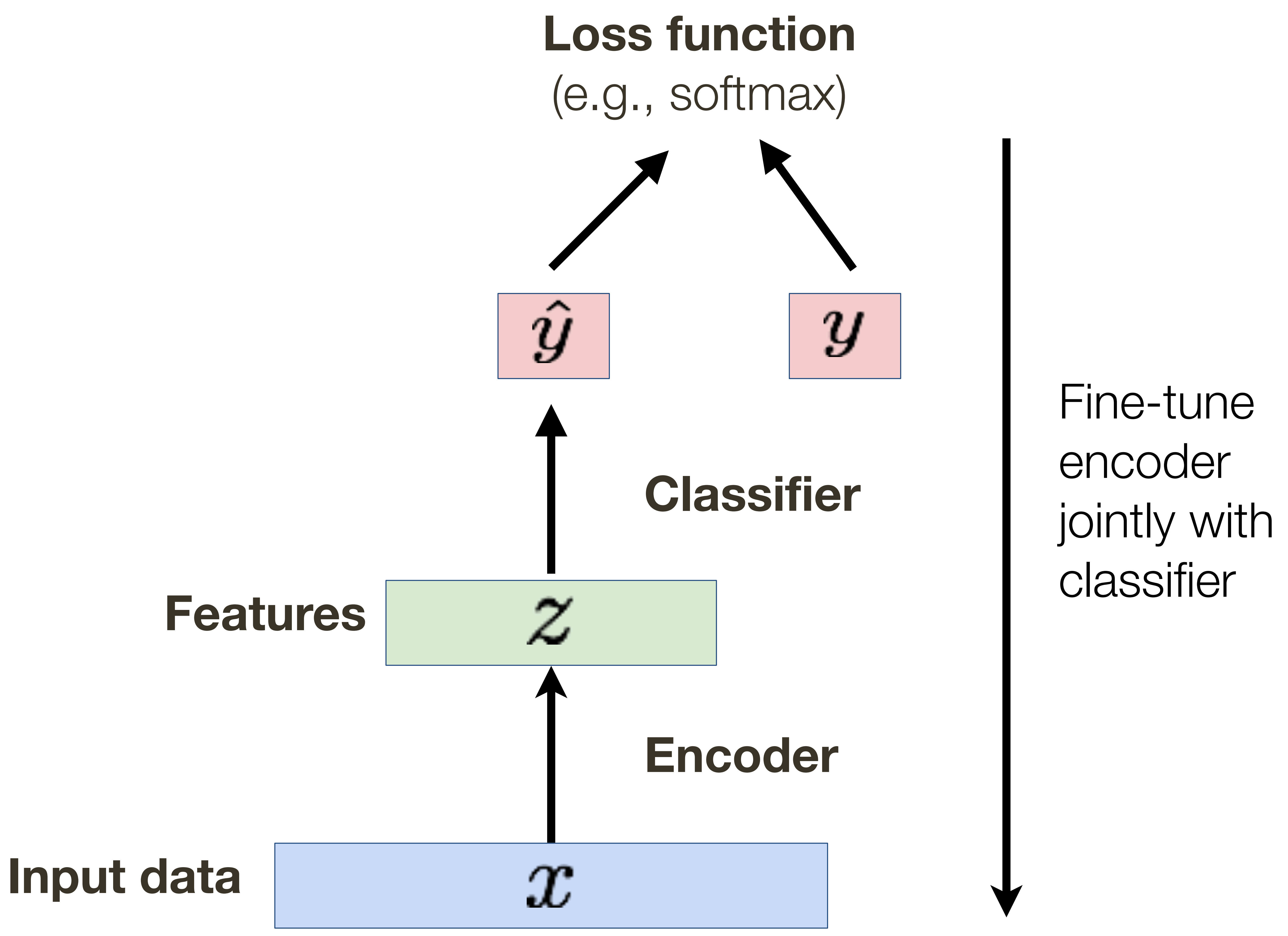
Doesn't use labels!



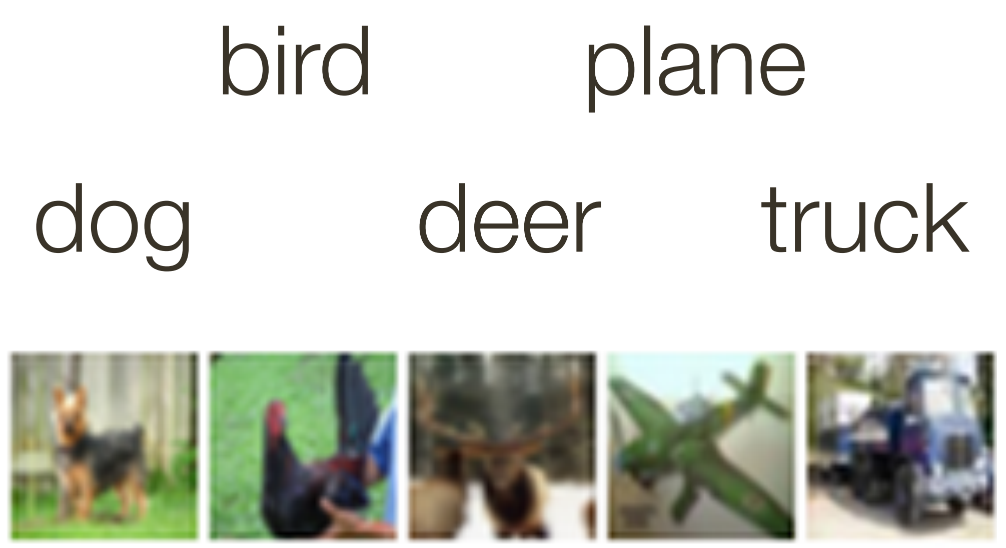


* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Autoencoders Reminder …



Train for **final task**
(sometimes with small data)



* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoders

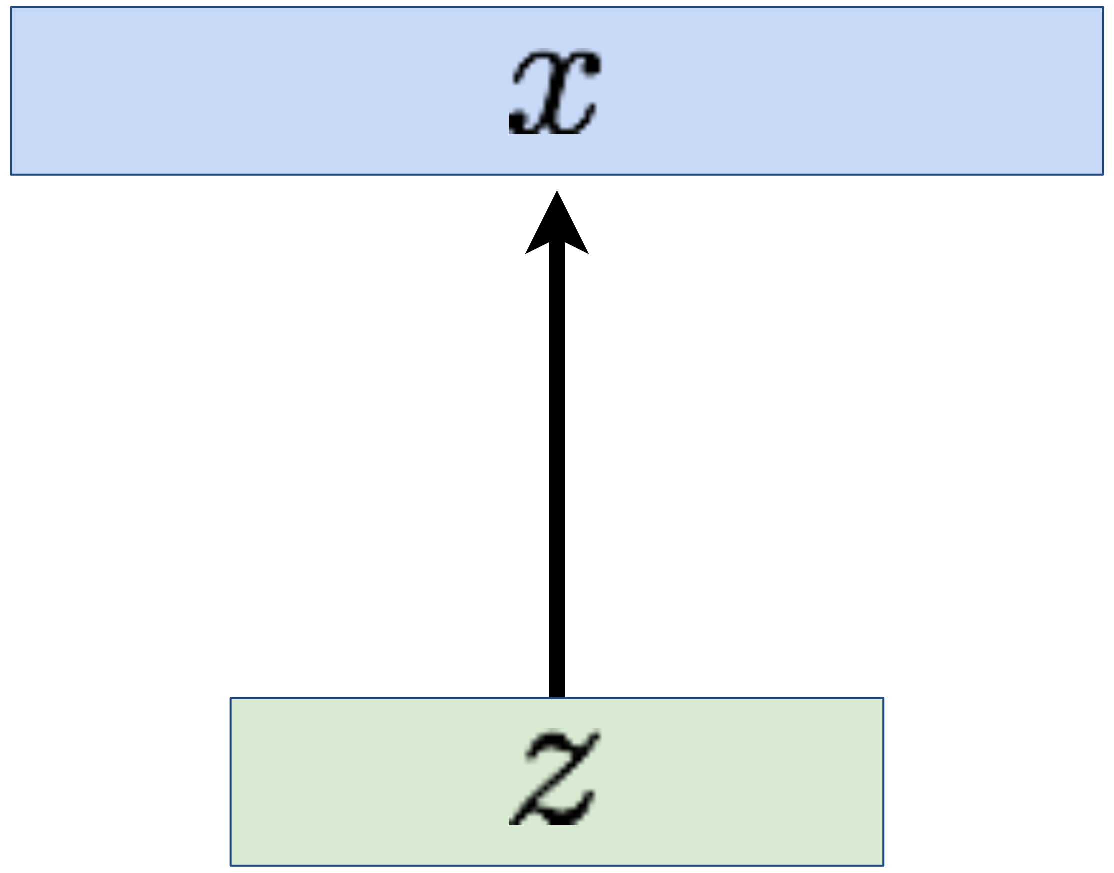
[Kingma and Welling, 2014]

Probabilistic spin on autoencoder - will let us sample from the model to generate

Assume training data is generated from underlying unobserved (latent) representation z

Sample from
true **conditional**

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



Sample from
true **prior**

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

* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoders

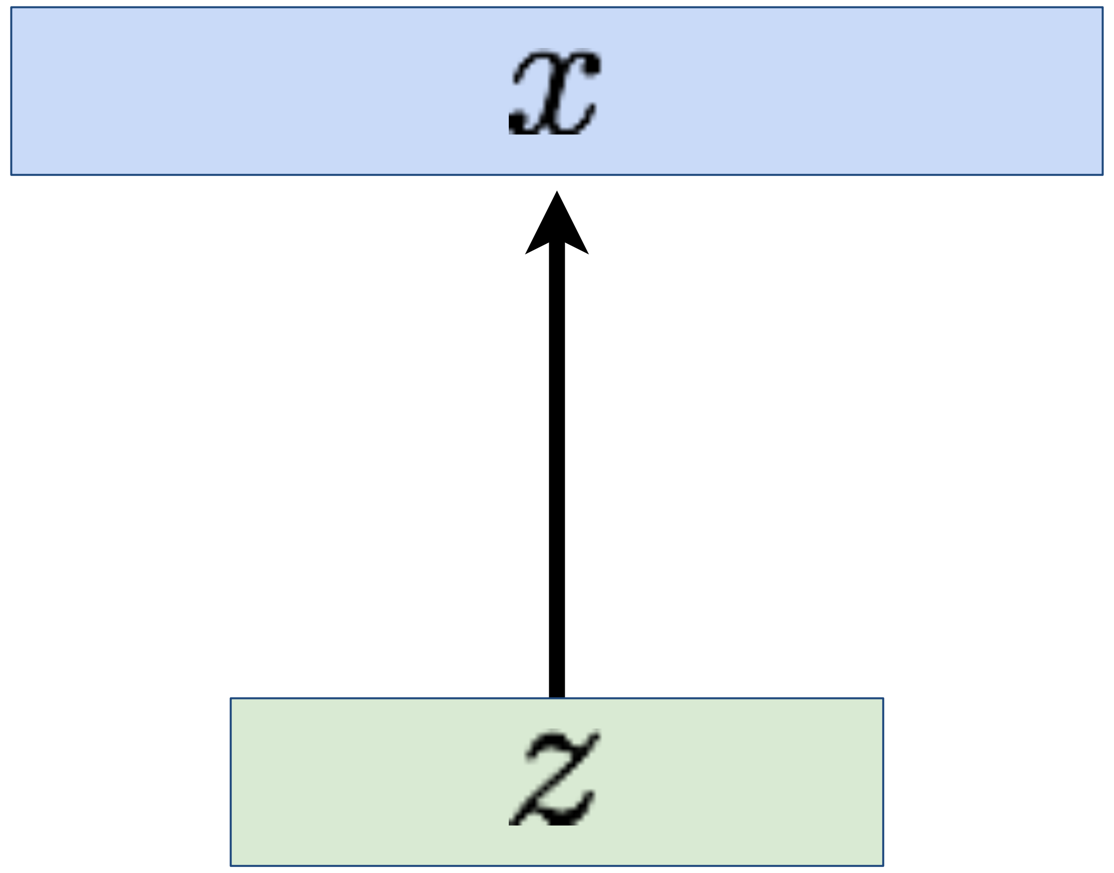
[Kingma and Welling, 2014]

Probabilistic spin on autoencoder - will let us sample from the model to generate

Assume training data is generated from underlying unobserved (latent) representation z

Sample from true **conditional**

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



Sample from true **prior**

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

Intuition: $\boldsymbol{x}$ is an image, $\boldsymbol{z}$ is latent factors used to generate $\boldsymbol{x}$ (e.g., attributes, orientation, *etc.*)

* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

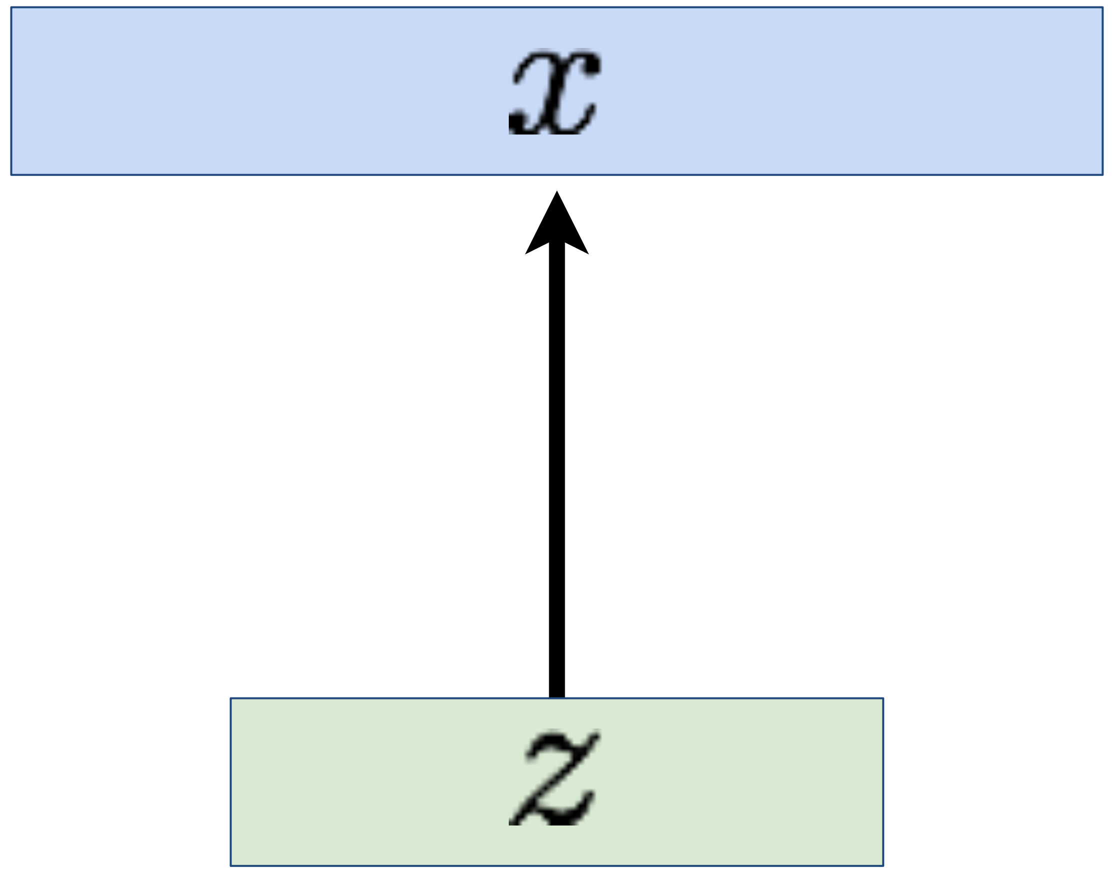
Variational Autoencoders

[Kingma and Welling, 2014]

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

Sample from true **conditional**

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



Sample from true **prior**

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

* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoders

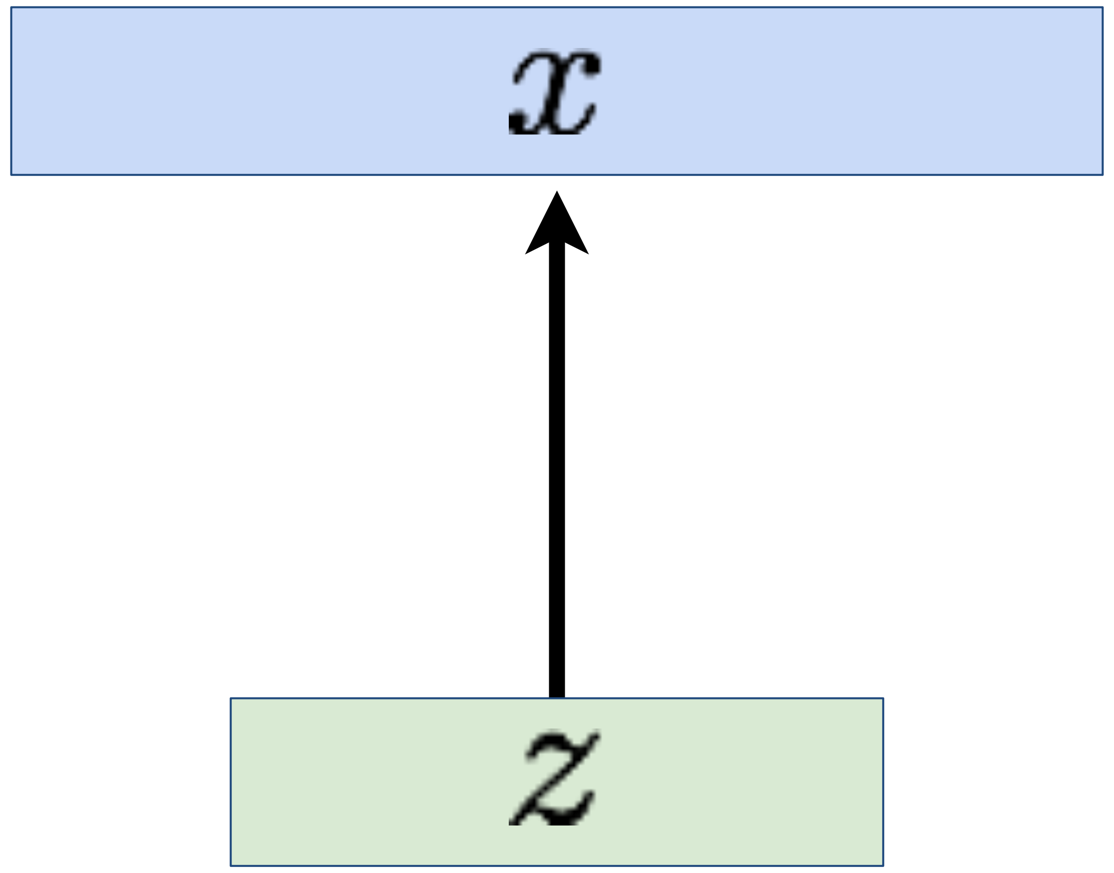
[Kingma and Welling, 2014]

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

How do we **represent** this model?

Sample from
true **conditional**

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



Sample from
true **prior**

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

Variational Autoencoders

[Kingma and Welling, 2014]

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

How do we **represent** this model?

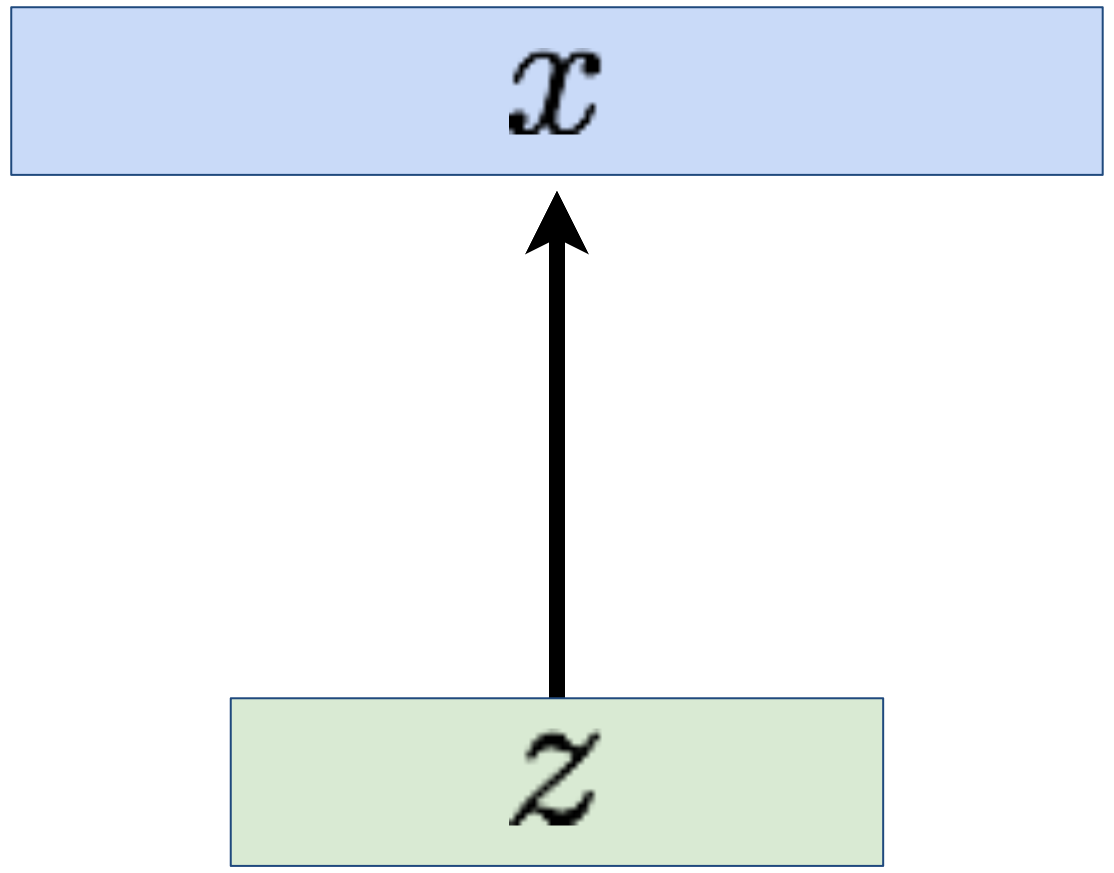
Choose prior $p(z)$ to be simple, e.g., Gaussian
Reasonable for latent attributes, e.g., pose, amount of smile

Sample from
true **conditional**

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

Sample from
true **prior**

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



Variational Autoencoders

[Kingma and Welling, 2014]

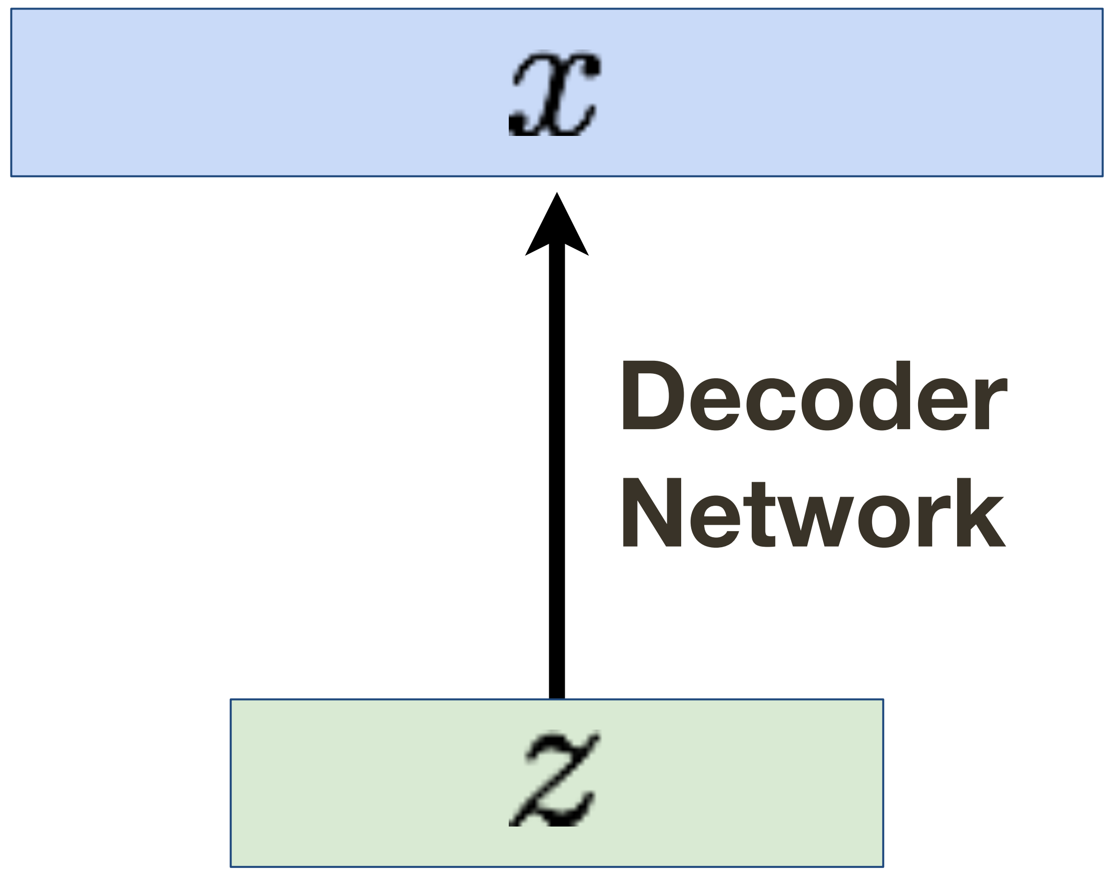
We want to **estimate the true parameters** θ^* of this generative model

Sample from
true **conditional**

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

Sample from
true **prior**

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



How do we **represent** this model?

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

Conditional $p(x|z)$ is complex (generates image)
Represent with Neural Network

Variational Autoencoders

[Kingma and Welling, 2014]

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

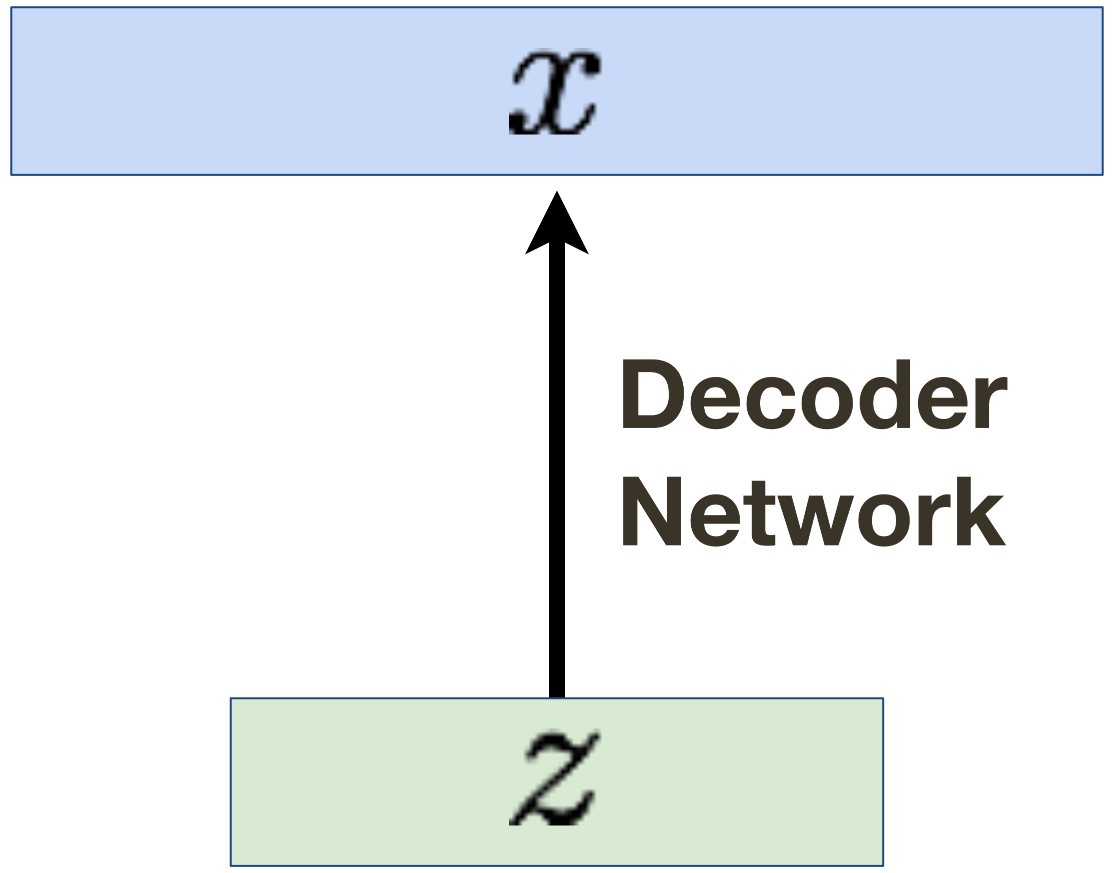
How do we **train** this model?

Sample from
true **conditional**

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

Sample from
true **prior**

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



* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoders

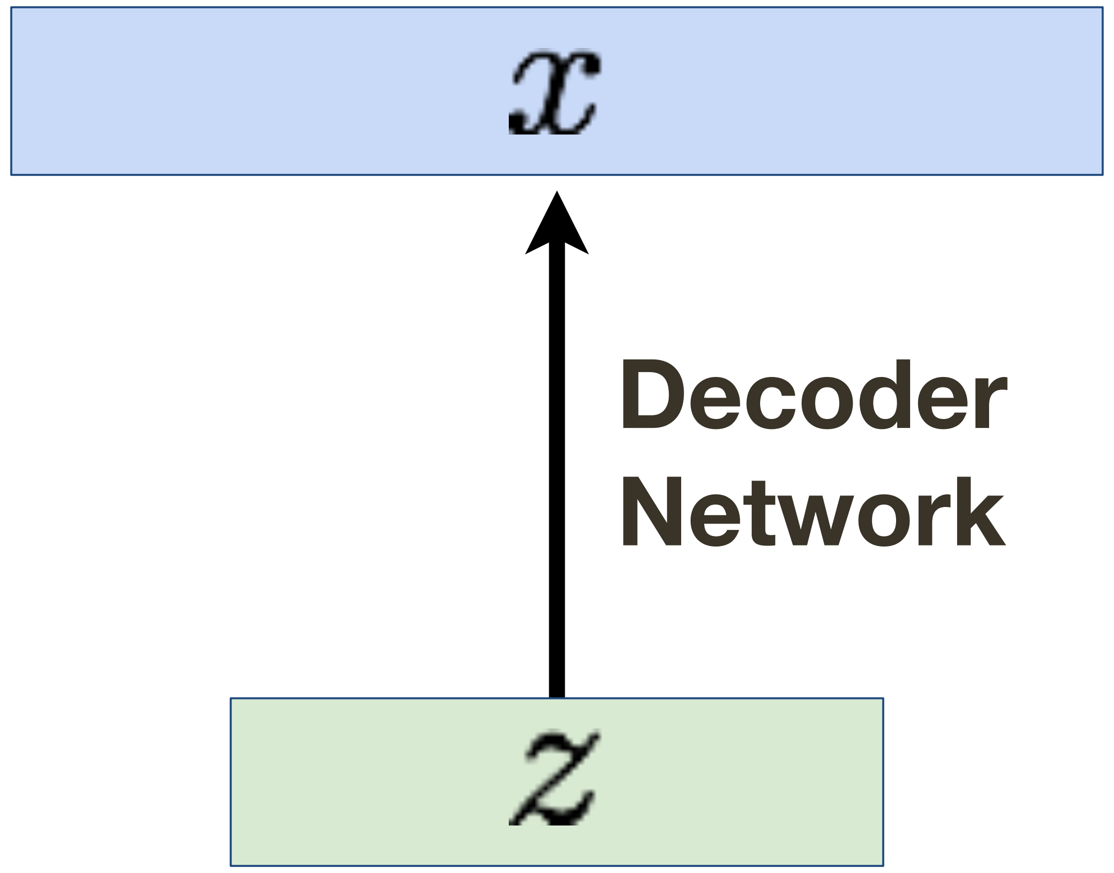
[Kingma and Welling, 2014]

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

How do we **train** this model?

Sample from
true **conditional**

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



Sample from
true **prior**

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

Remember the strategy from earlier — learn model parameters to maximize likelihood of training data

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

(now with latent z that we need to marginalize)

* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoders

[Kingma and Welling, 2014]

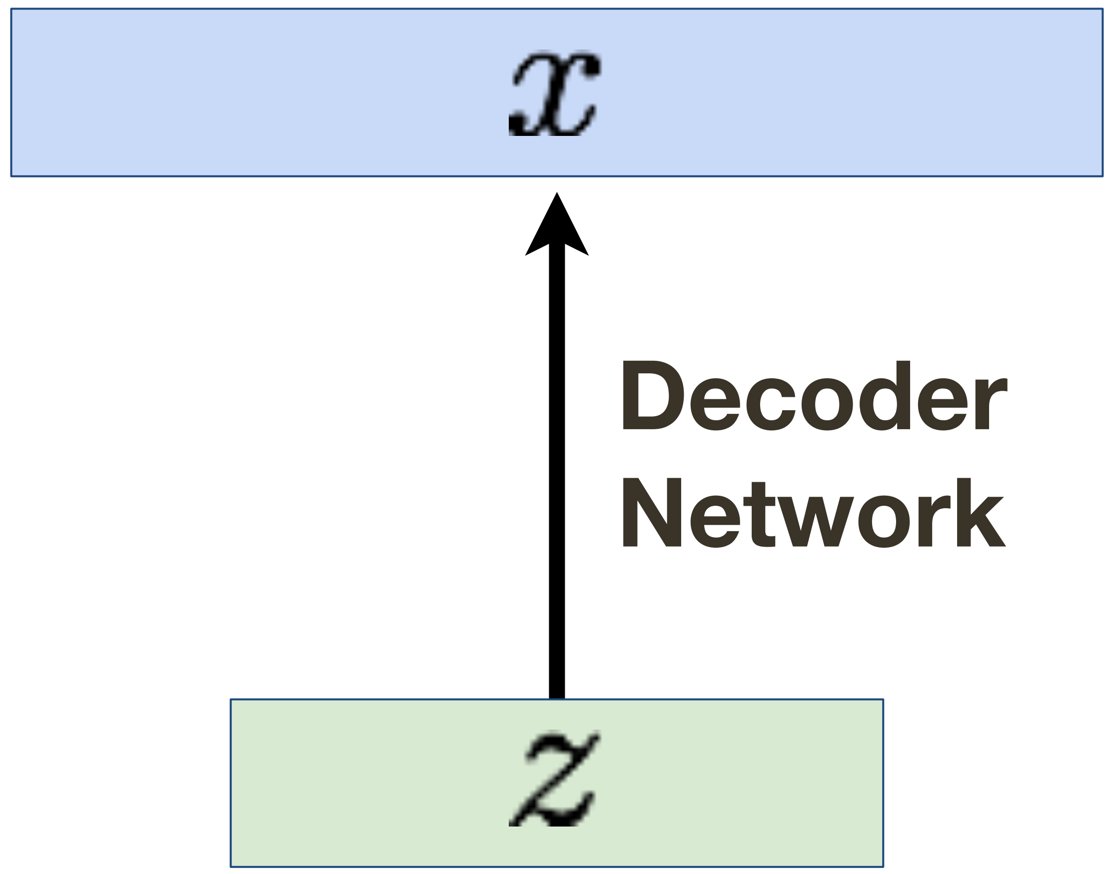
We want to **estimate the true parameters** θ^* of this generative model

Sample from
true **conditional**

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

Sample from
true **prior**

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



How do we **train** this model?

Remember the strategy from earlier — learn model parameters to maximize likelihood of training data

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

(now with latent z that we need to marginalize)

What is the problem with this?

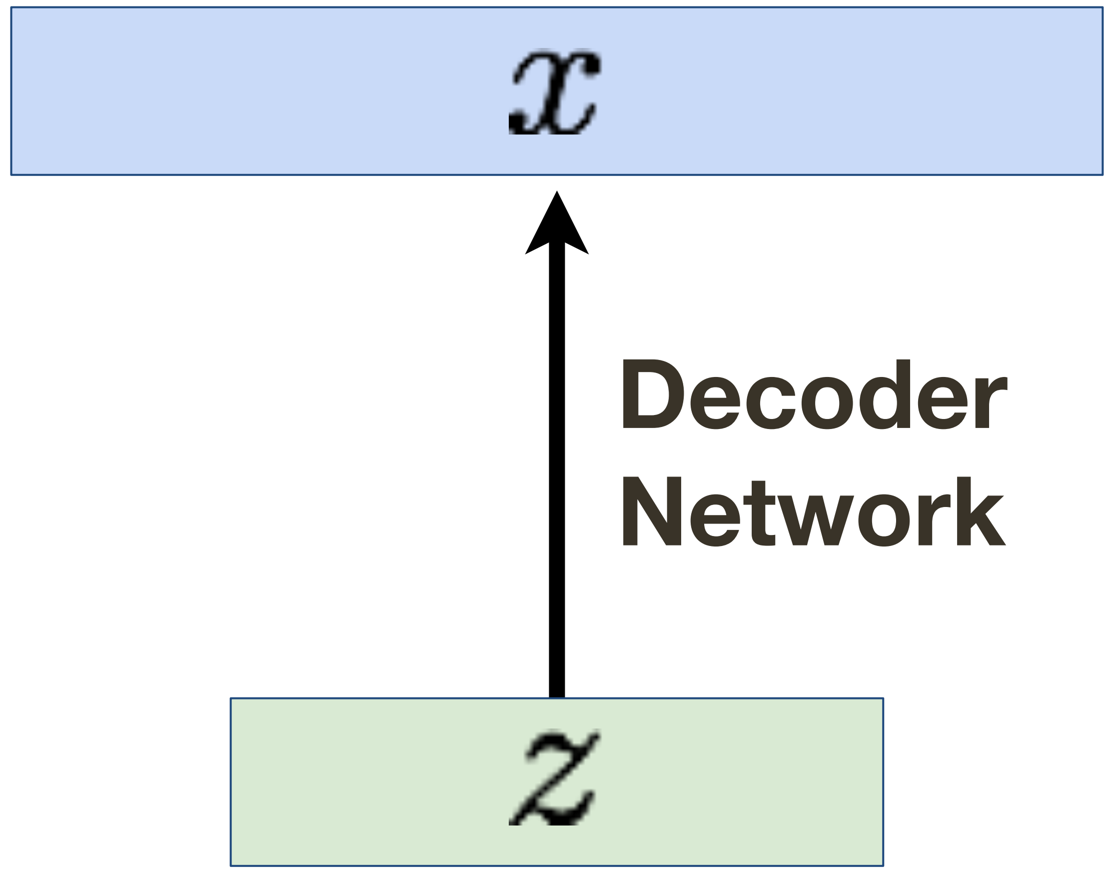
Variational Autoencoders

[Kingma and Welling, 2014]

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

Sample from true **conditional**

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



Sample from true **prior**

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

How do we **train** this model?

Remember the strategy from earlier — learn model parameters to maximize likelihood of training data

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

(now with latent z that we need to marginalize)

Intractable !

* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Intractability in Variational Autoencoder

[Kingma and Welling, 2014]

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

Intractability in Variational Autoencoder

[Kingma and Welling, 2014]

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

🙂

Simple **Gaussian** Prior

* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Intractability in Variational Autoencoder

[Kingma and Welling, 2014]

Decoder Neural Network

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

Simple **Gaussian** Prior

Intractability in Variational Autoencoder

[Kingma and Welling, 2014]

Intractable to compute for every z

Decoder Neural Network

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

Simple **Gaussian** Prior

Intractability in Variational Autoencoder

[Kingma and Welling, 2014]

Intractable to compute for every z ☹️

Decoder Neural Network 🙂

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

🙂 Simple **Gaussian** Prior

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

Intractability in Variational Autoencoder

[Kingma and Welling, 2014]

Intractable to compute for every z

Decoder Neural Network

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

Simple **Gaussian** Prior

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

Intractability in Variational Autoencoder

[Kingma and Welling, 2014]

Intractable to compute for every z ☹️

Decoder Neural Network 🙂

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

🙂 Simple **Gaussian** Prior

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

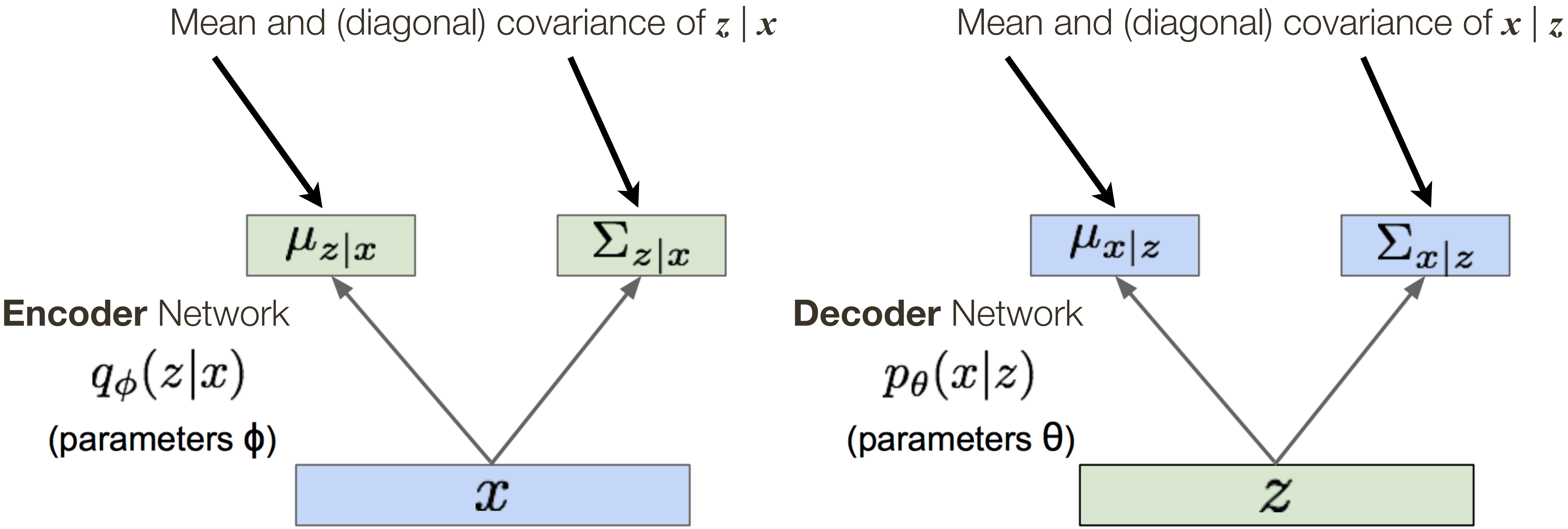
Solution: In addition to decoder network modeling $p_\theta(x|z)$, define additional encoder network $q_\phi(z|x)$ that approximates $p_\theta(z|x)$

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

Variational Autoencoder

[Kingma and Welling, 2014]

Since we are modeling probabilistic generation of data, encoder and decoder networks are probabilistic (they model distributions)



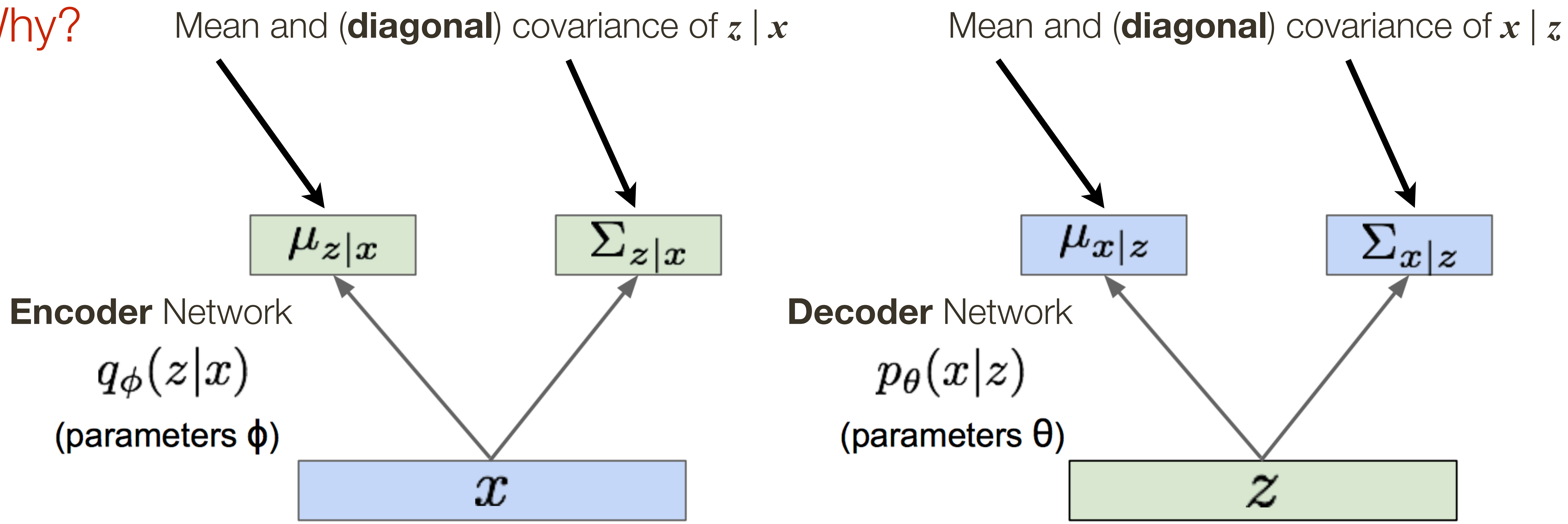
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder

[Kingma and Welling, 2014]

Since we are modeling probabilistic generation of data, encoder and decoder networks are probabilistic (they model distributions)

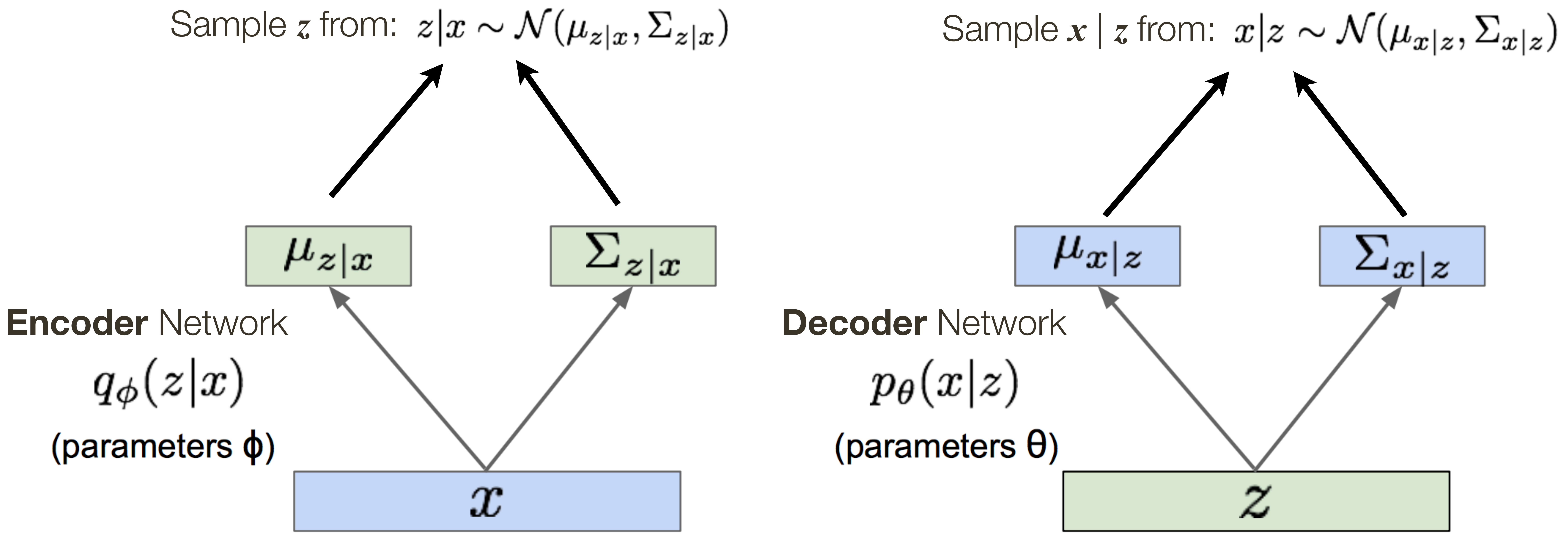
Why?



Variational Autoencoder

[Kingma and Welling, 2014]

Since we are modeling probabilistic generation of data, encoder and decoder networks are probabilistic (they model distributions)



* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder

[Kingma and Welling, 2014]

Derivation of lower bound of the data likelihood

Now equipped with **encoder** and **decoder** networks, let's see (log) data likelihood:

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

Taking expectation with respect to z (using encoder network) will come in handy later

Variational Autoencoder

[Kingma and Welling, 2014]

Derivation of lower bound of the data likelihood

Now equipped with **encoder** and **decoder** networks, let's see (log) data likelihood:

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

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

Variational Autoencoder

[Kingma and Welling, 2014]

Derivation of lower bound of the data likelihood

Now equipped with **encoder** and **decoder** networks, let's see (log) data likelihood:

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

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

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

Variational Autoencoder

[Kingma and Welling, 2014]

Derivation of lower bound of the data likelihood

Now equipped with **encoder** and **decoder** networks, let's see (log) data likelihood:

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

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

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

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

Variational Autoencoder

[Kingma and Welling, 2014]

Derivation of lower bound of the data likelihood

Now equipped with **encoder** and **decoder** networks, let's see (log) data likelihood:

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

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

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

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

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

Expectation with respect to z
(using encoder network) leads to nice KL terms

* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder

[Kingma and Welling, 2014]

Derivation of lower bound of the data likelihood

Now equipped with **encoder** and **decoder** networks, let's see (log) data likelihood:

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

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

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

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

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

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.

Variational Autoencoder

[Kingma and Welling, 2014]

Derivation of lower bound of the data likelihood

Now equipped with **encoder** and **decoder** networks, let's see (log) data likelihood:

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

Tractable lower bound which we can take gradient of
and optimize! (pθ(x|z) differentiable, KL term differentiable)

Variational Autoencoder

[Kingma and Welling, 2014]

Derivation of lower bound of the data likelihood

Now equipped with **encoder** and **decoder** networks, let's see (log) data likelihood:

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

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

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

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

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

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

Variational lower bound ("**ELBO**")

Training: Maximize lower bound

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)},\theta,\phi)$$

Variational Autoencoder

[Kingma and Welling, 2014]

Derivation of lower bound of the data likelihood

Now equipped with **encoder** and **decoder** networks, let's see (log) data likelihood:

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

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

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

Reconstruct Input Data

Make approximate posterior close to the prior

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

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

Variational lower bound ("**ELBO**")

Training: Maximize lower bound

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)},\theta,\phi)$$

Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

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

Lets look at **computing the bound** (forward pass) for a given mini batch of input data

Input Data x

Variational Autoencoder: Learning

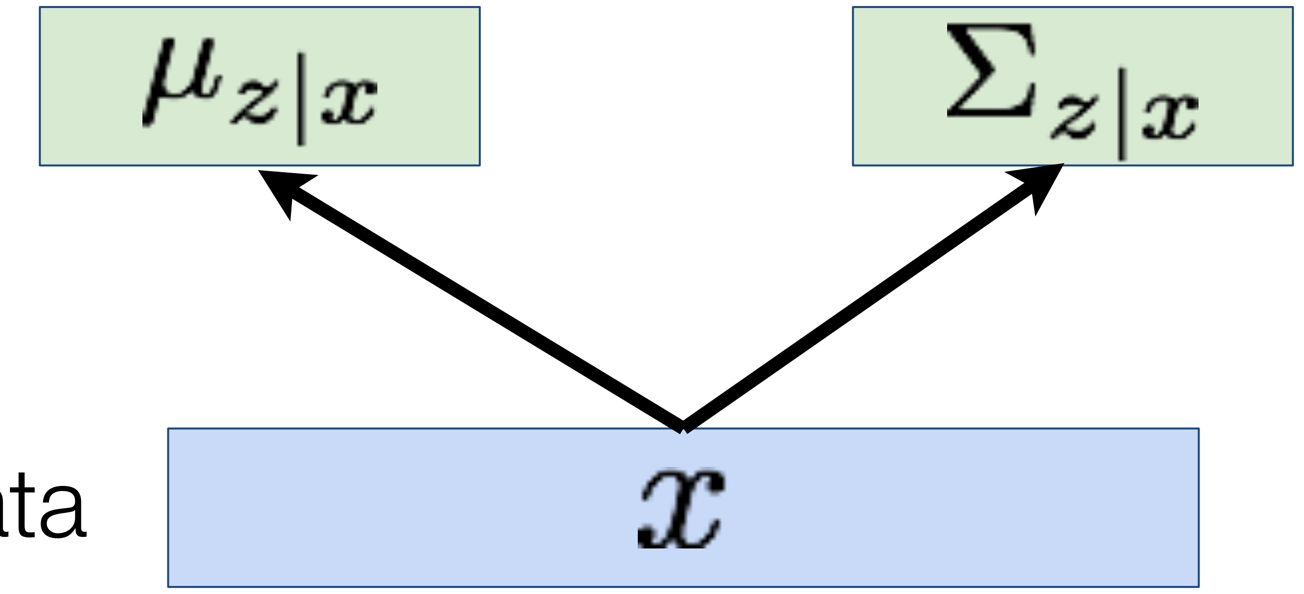
Putting it all together:
maximizing the likelihood lower bound

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

Encoder network

$q_\phi(z|x)$

Input Data



* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

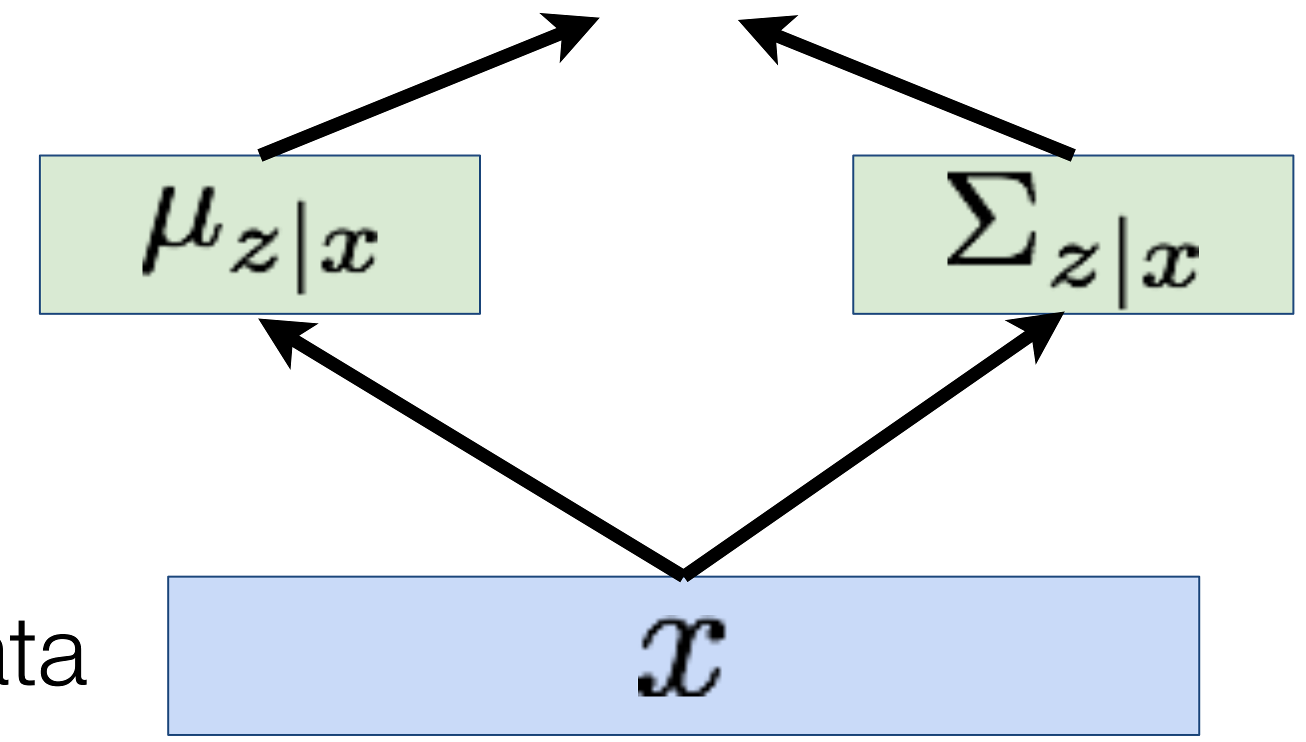
$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)}$$

Make approximate posterior distribution close to prior

Encoder network

$q_\phi(z|x)$

Input Data

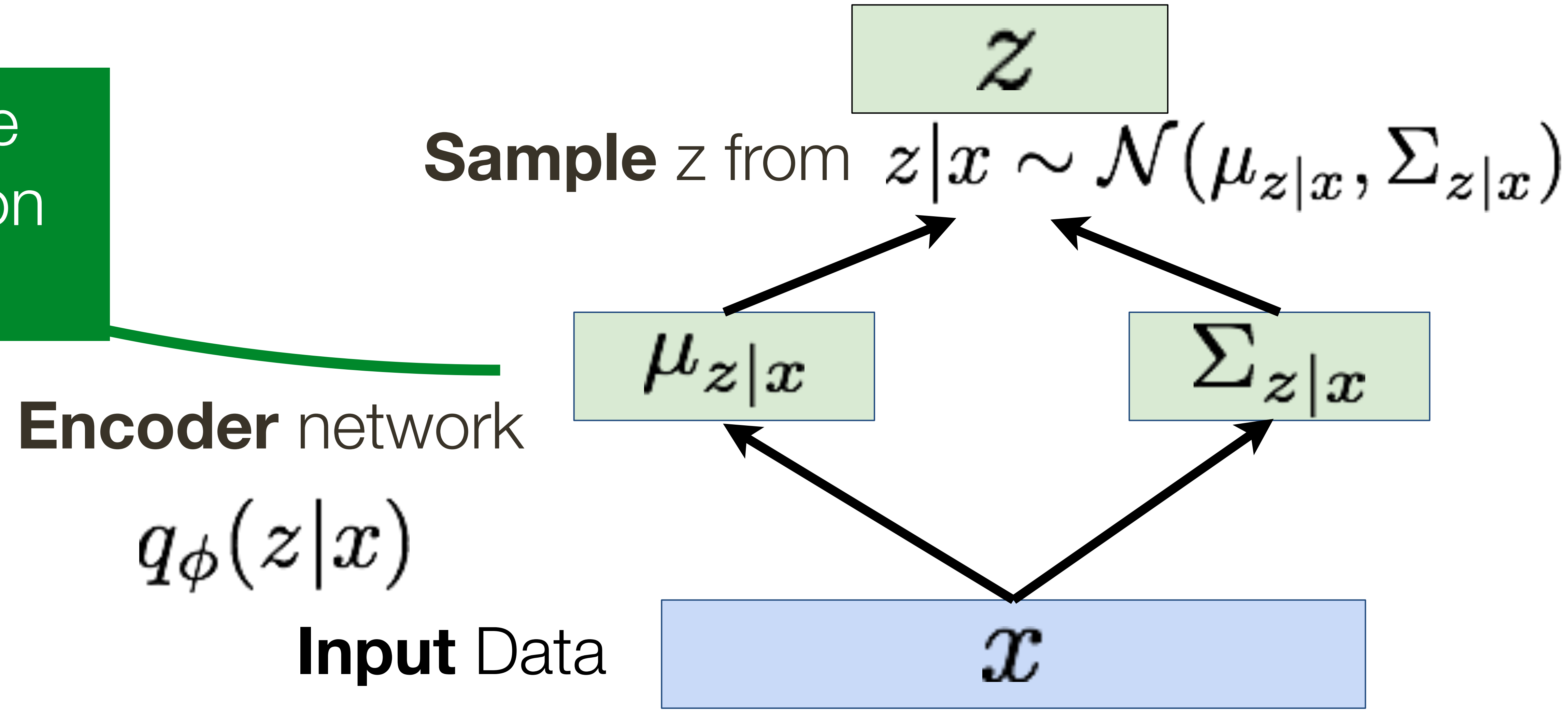


Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior



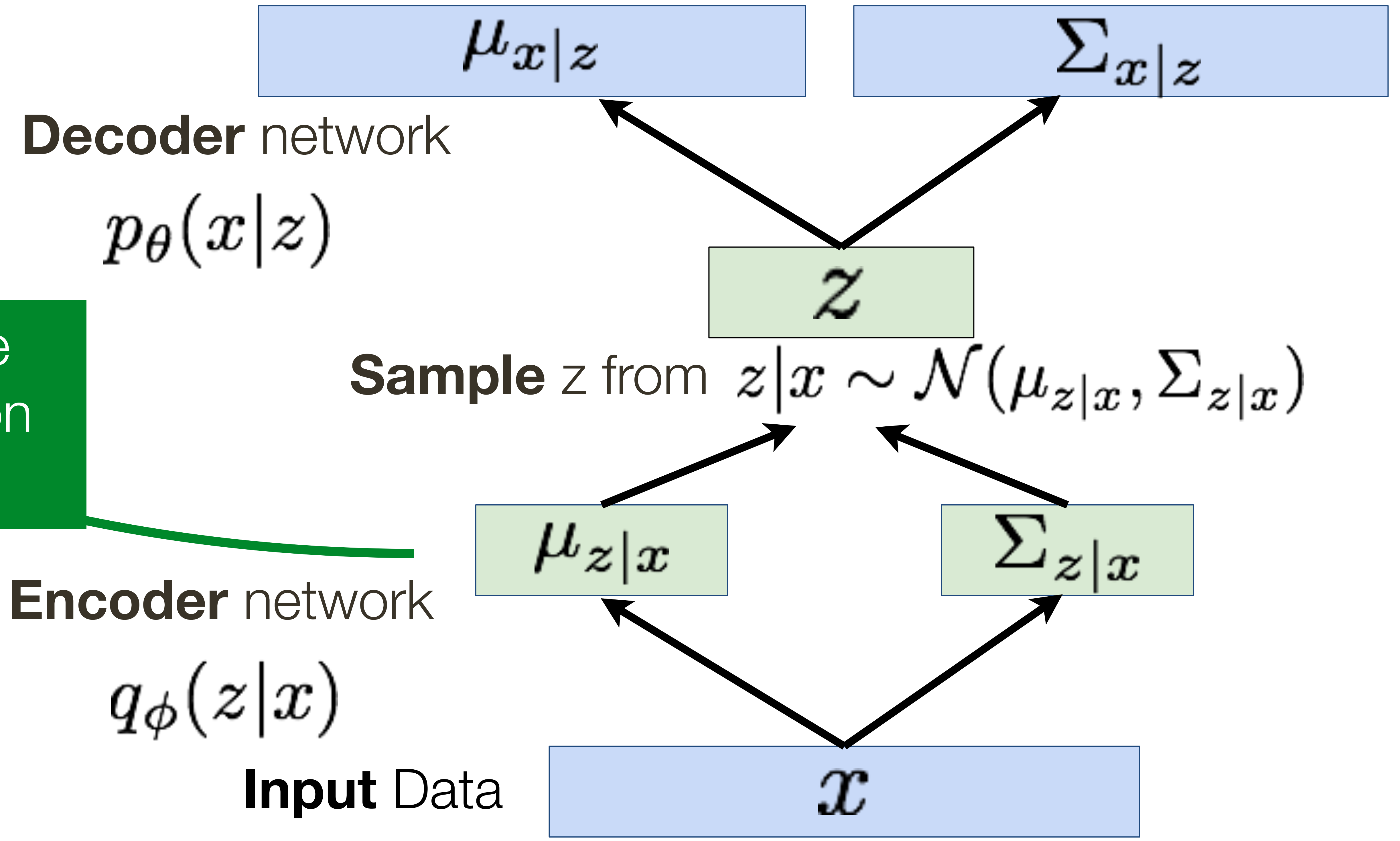
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior



* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

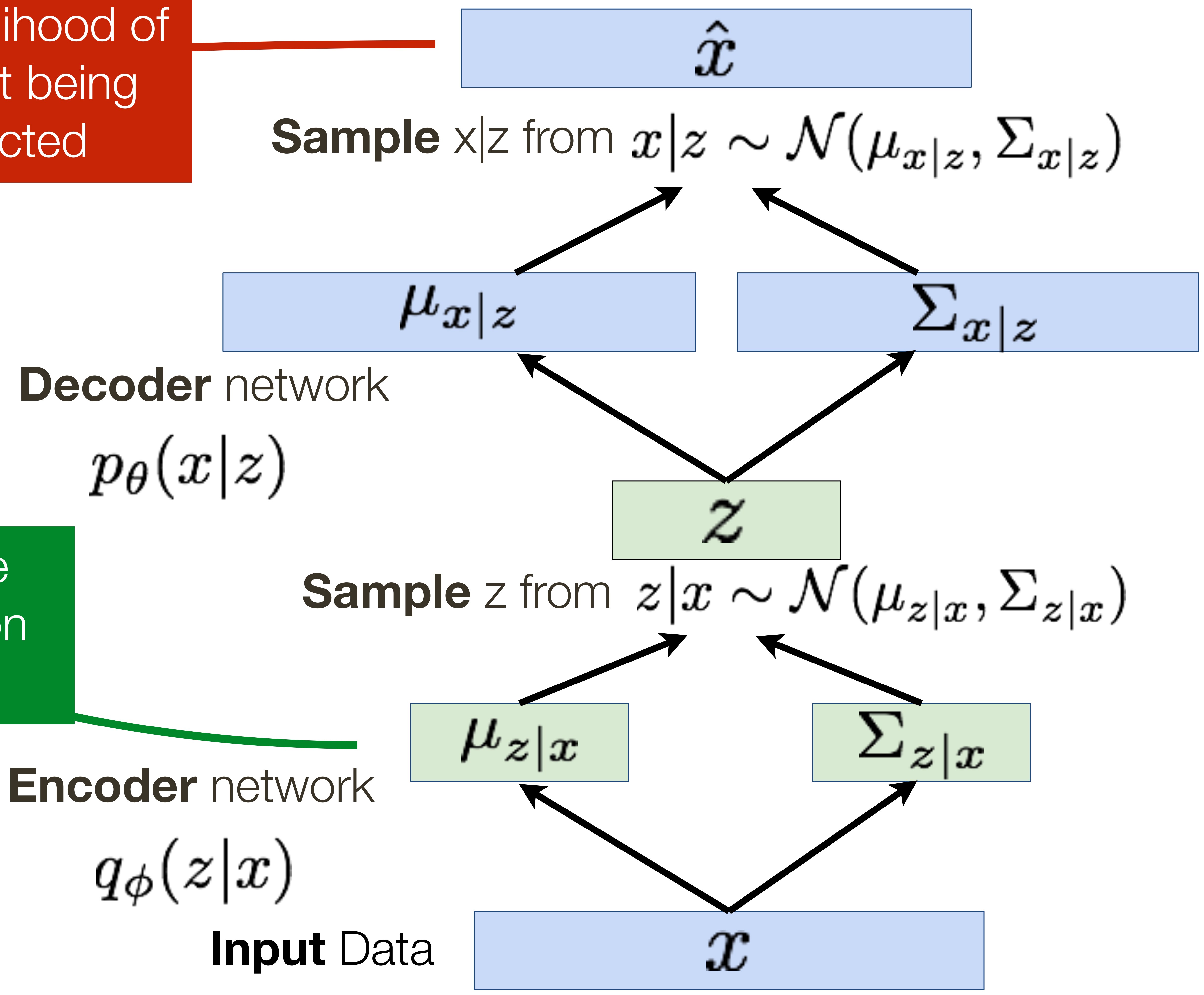
Variational Autoencoder: Learning

Putting it all together: maximizing the likelihood lower bound

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

Maximize likelihood of original input being reconstructed

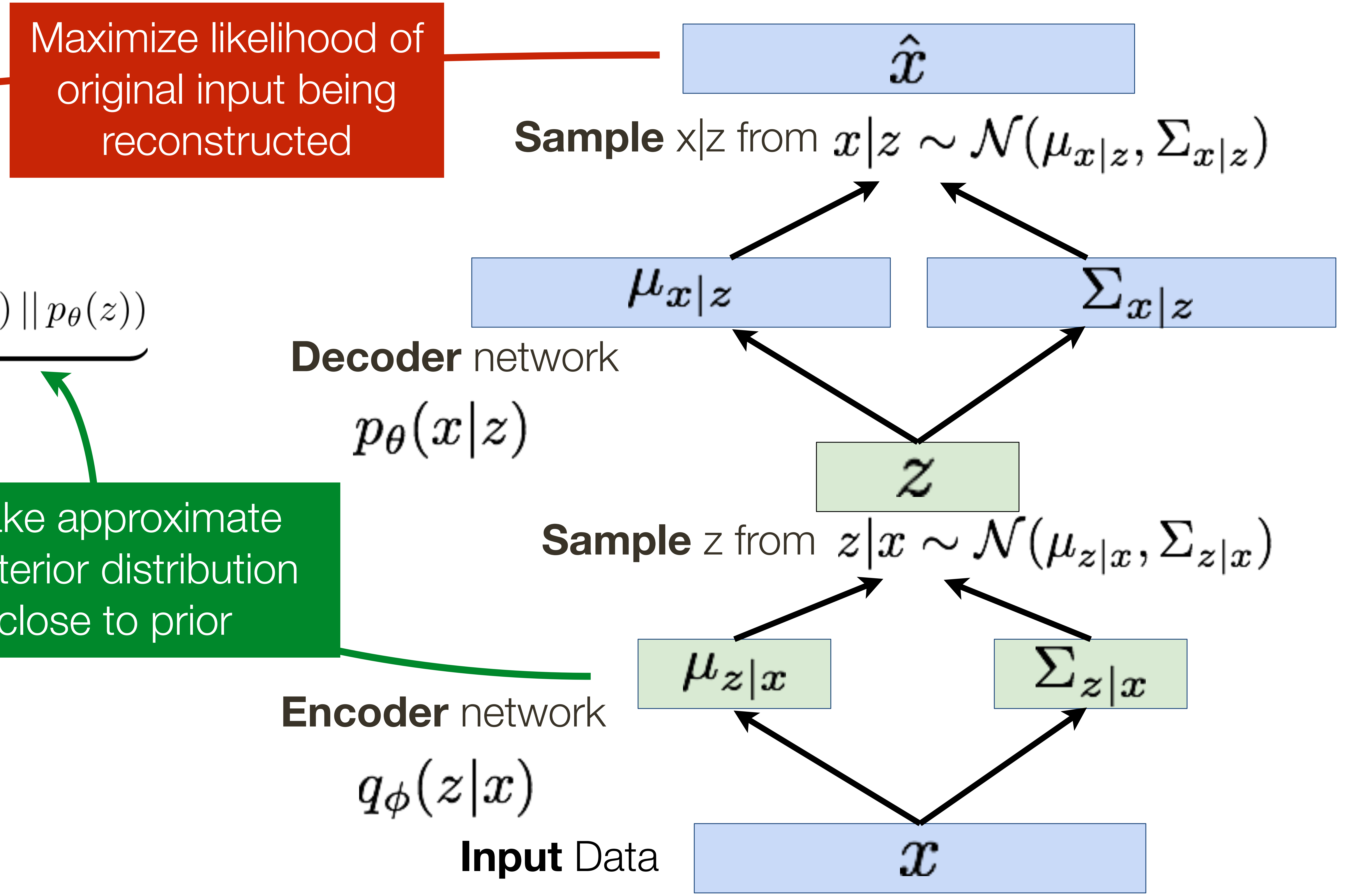
Make approximate posterior distribution close to prior



Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

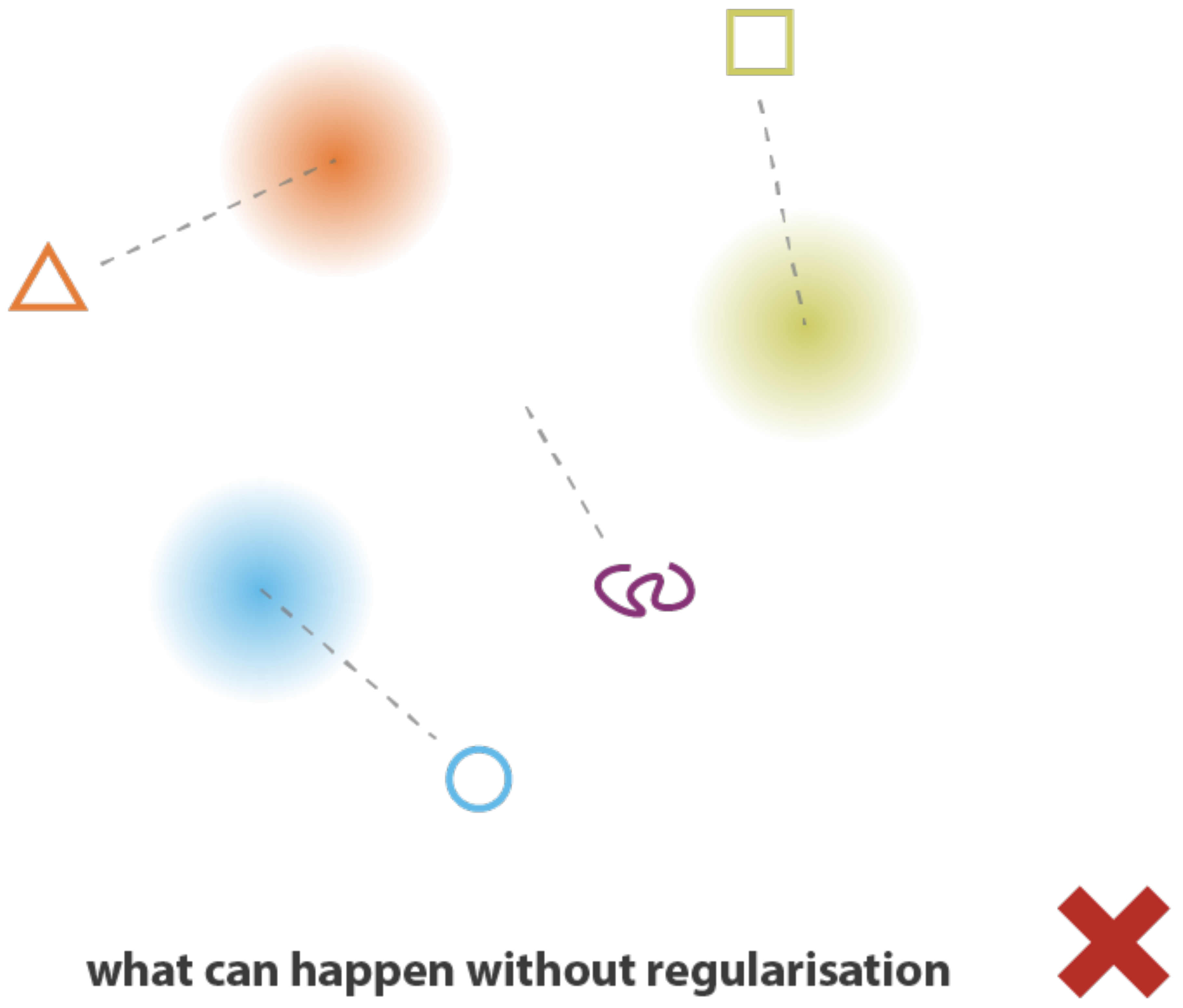
$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

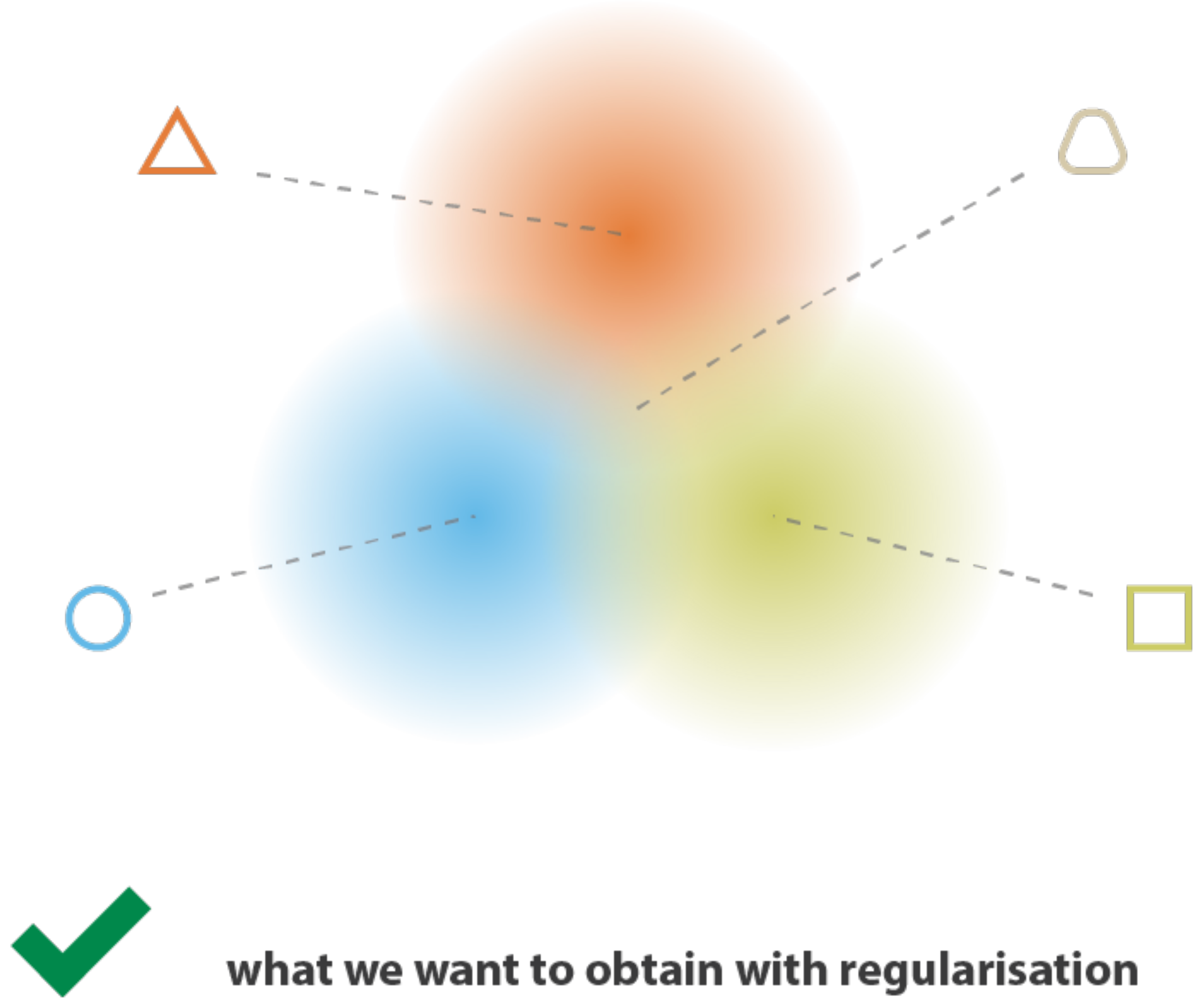


For every minibatch of input data: compute this forward pass, and then backprop!

Variational Autoencoder: Learning

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

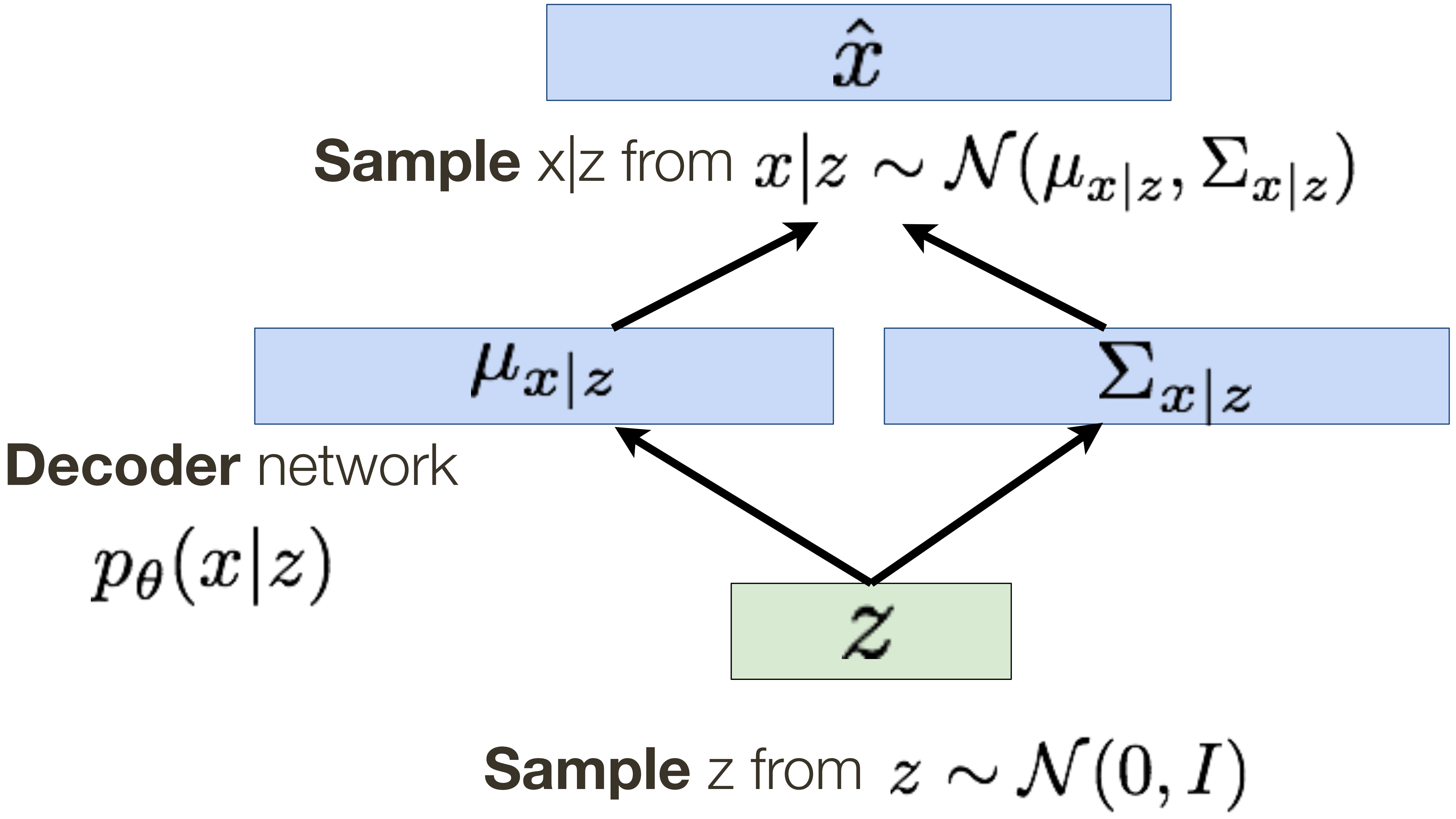




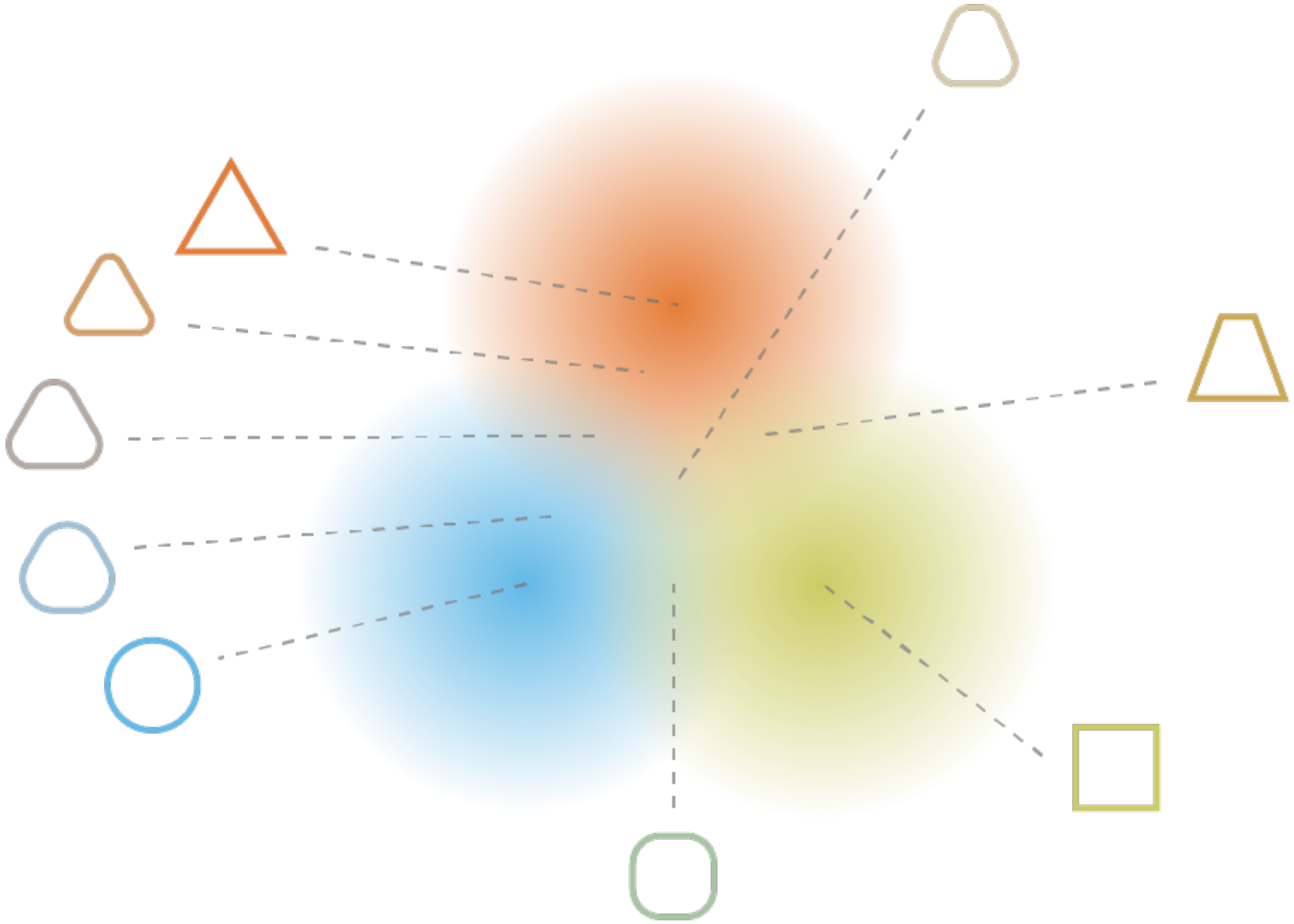
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoder: Generating Data

Use decoder network and sample z from **prior**



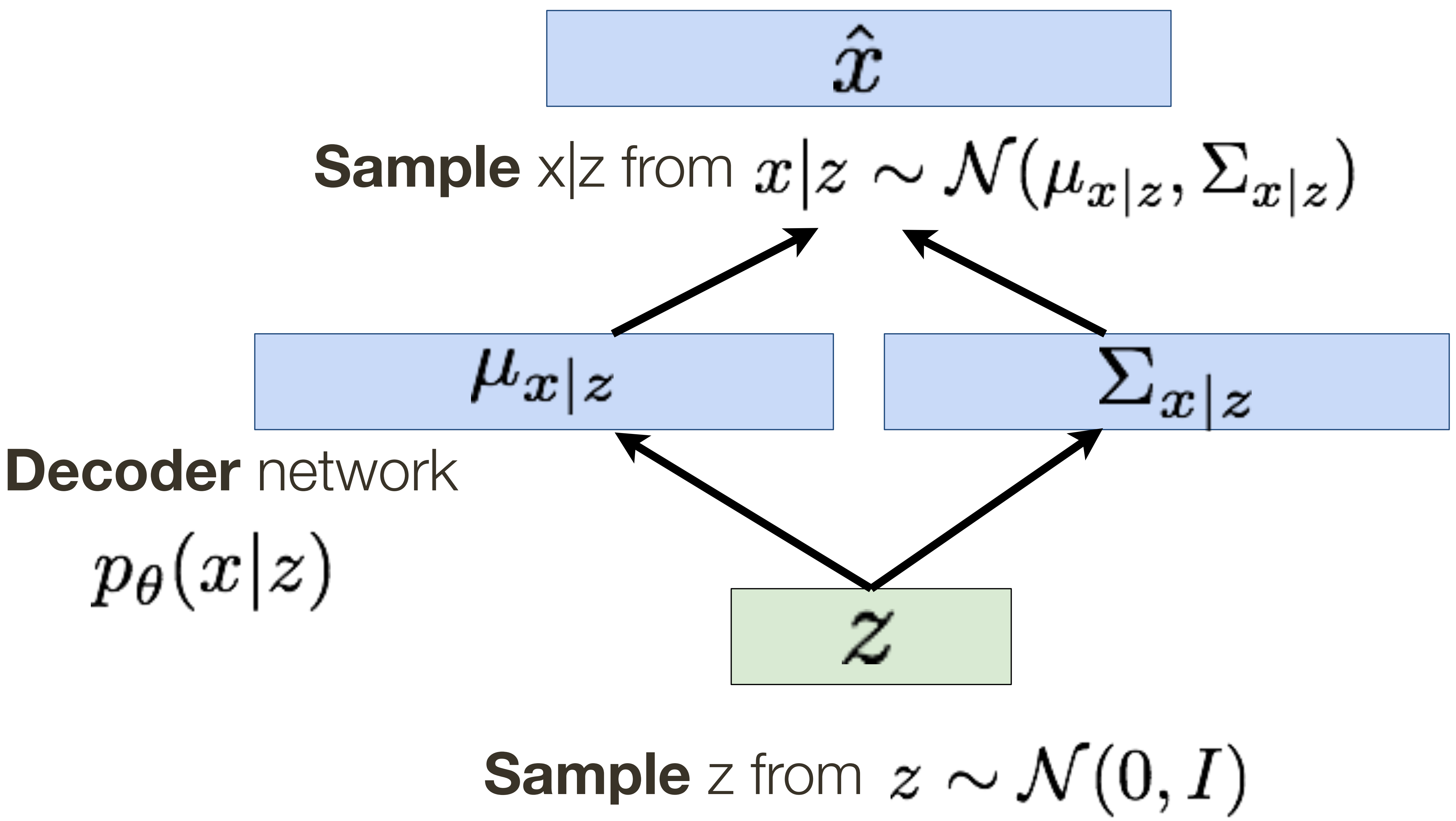
Variational Autoencoder: Generating Data



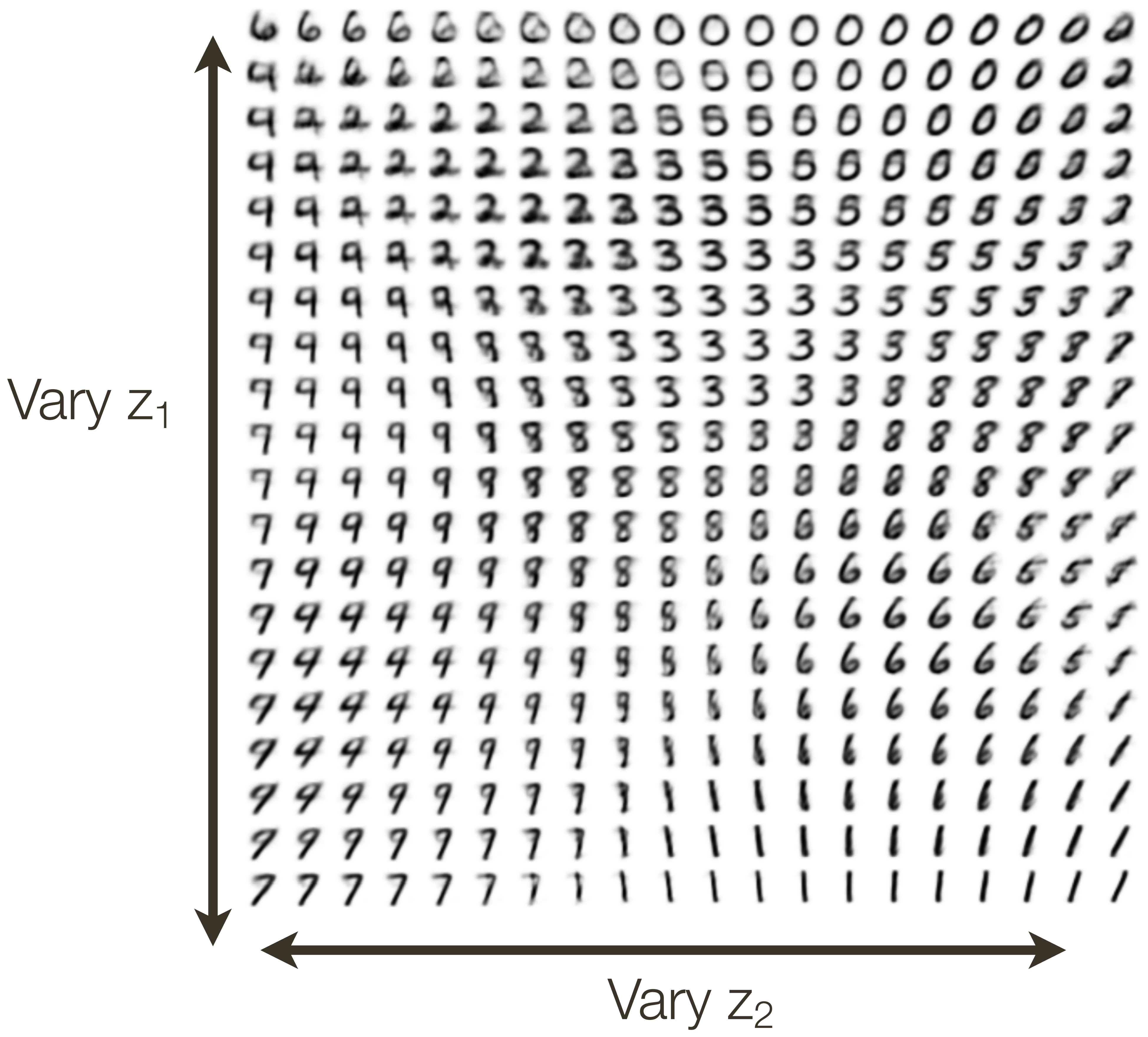
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoder: Generating Data

Use decoder network and sample z from **prior**



Data manifold for 2-d z



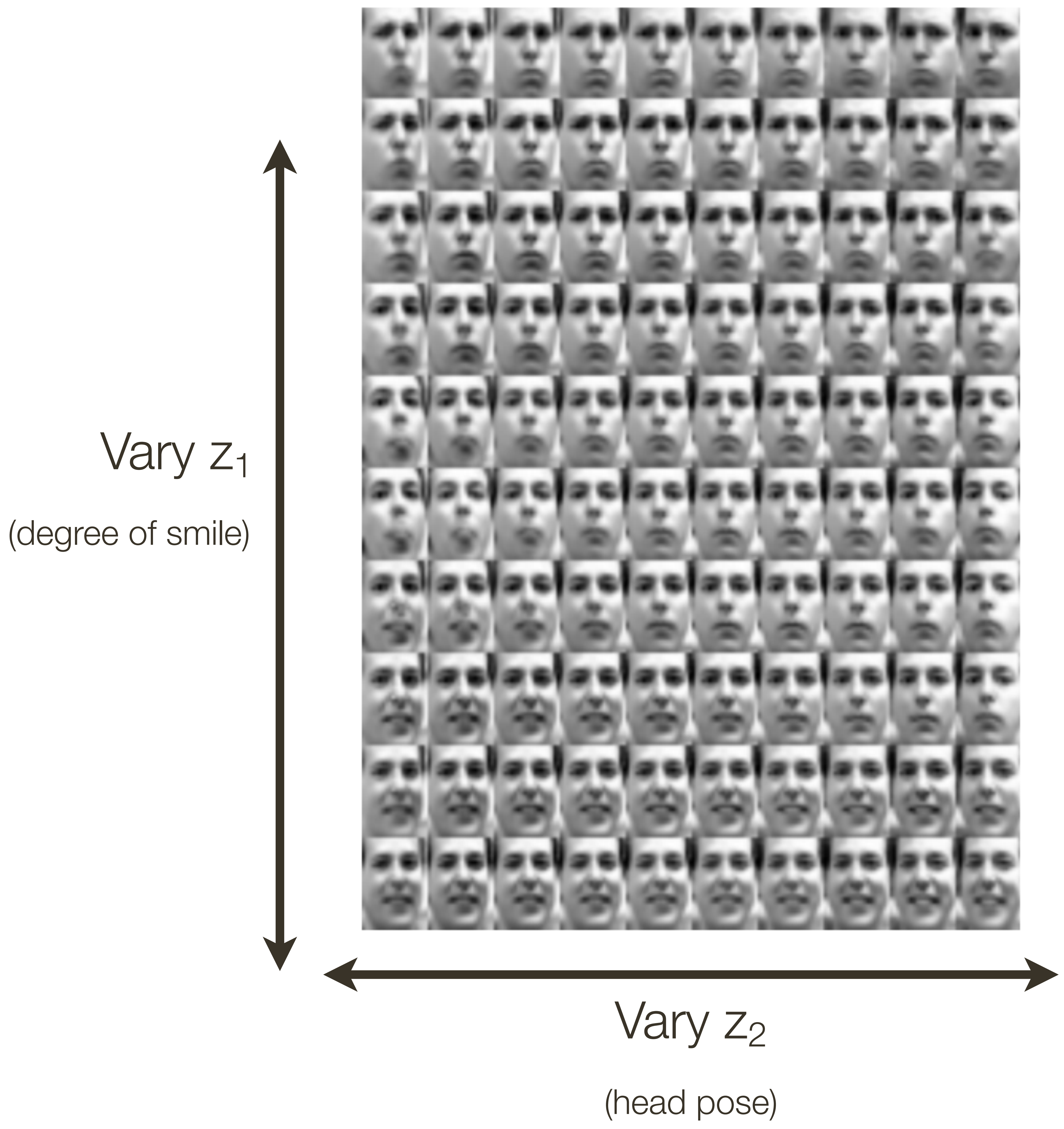
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder: Generating Data

Diagonal prior on z => independent latent variables

Different dimensions of z encode interpretable factors of variation

Data manifold for 2-d z



* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

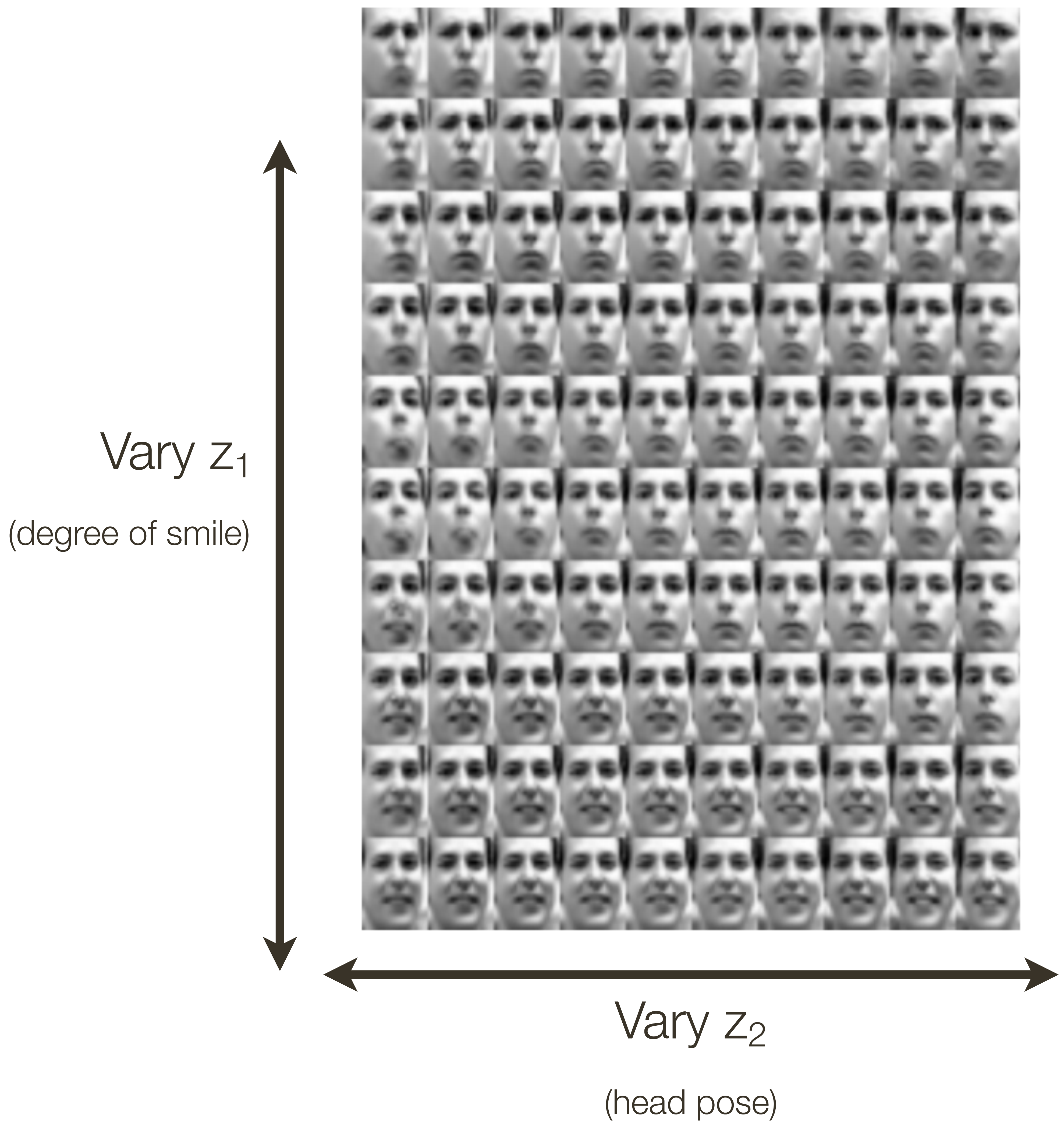
Variational Autoencoder: Generating Data

Diagonal prior on z => independent latent variables

Different dimensions of z encode interpretable factors of variation

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

Data manifold for 2-d z



* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

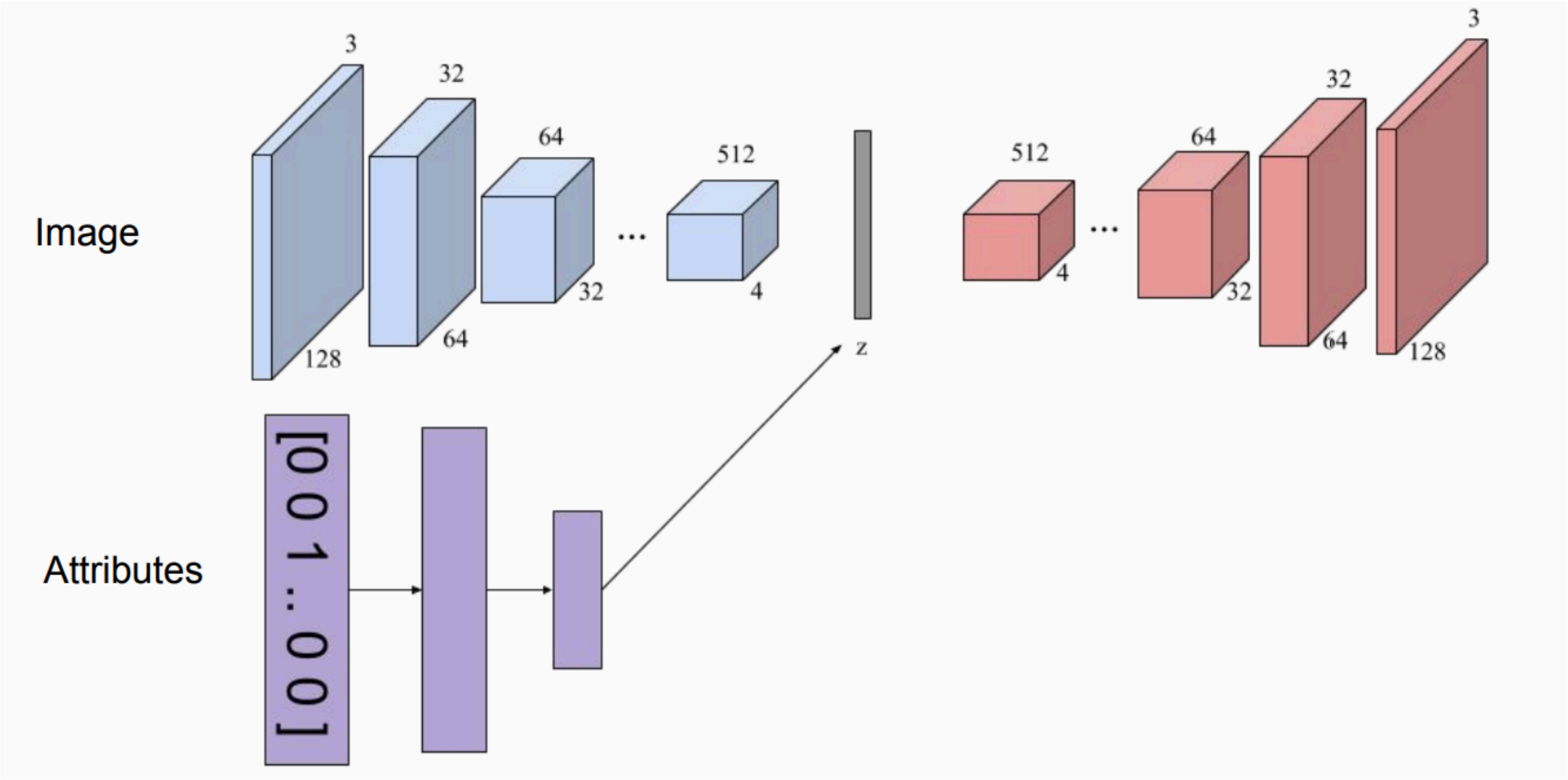
Variational Autoencoder: Generating Data

32x32 CIFAR-10

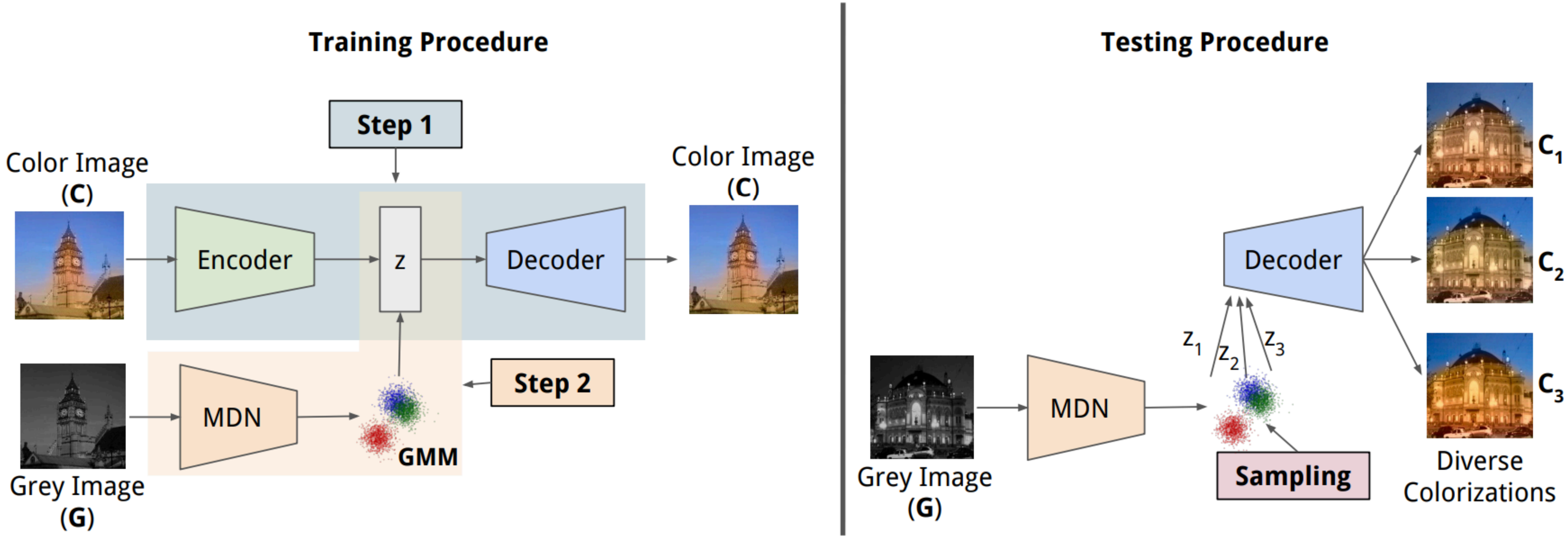
Labeled Faces in the Wild

* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Conditional VAEs



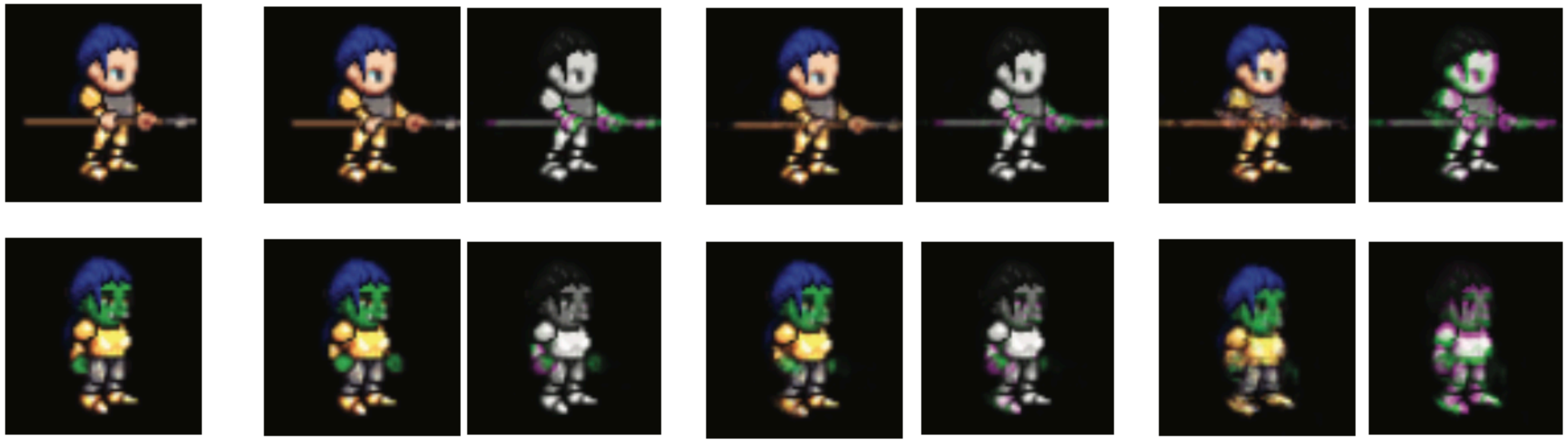
Conditional VAE: Diverse Image Colorization



[Deshpande et al., 2017]

Conditional VAE: Temporal Predictions

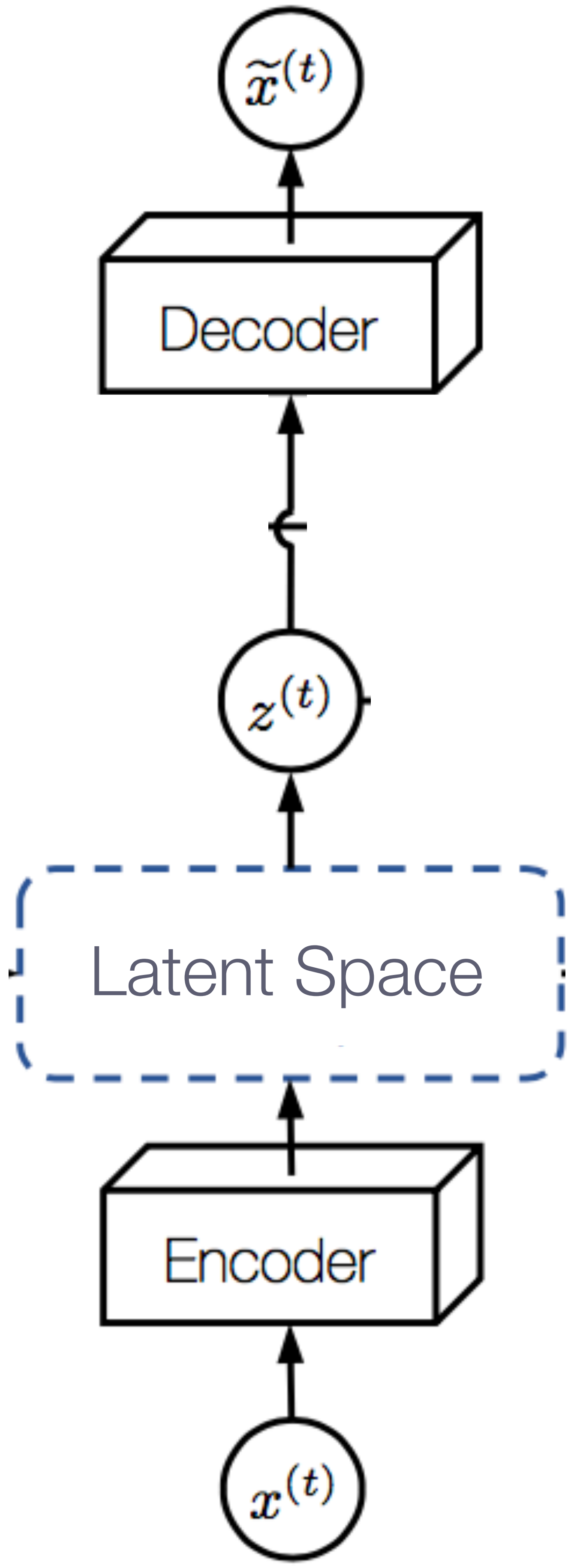
[Xue et al., 2016]



(a) Frame 1 (b) Frame 2 (ground truth) (c) Frame 2 (Sample 1) (d) Frame 2 (Sample 2)

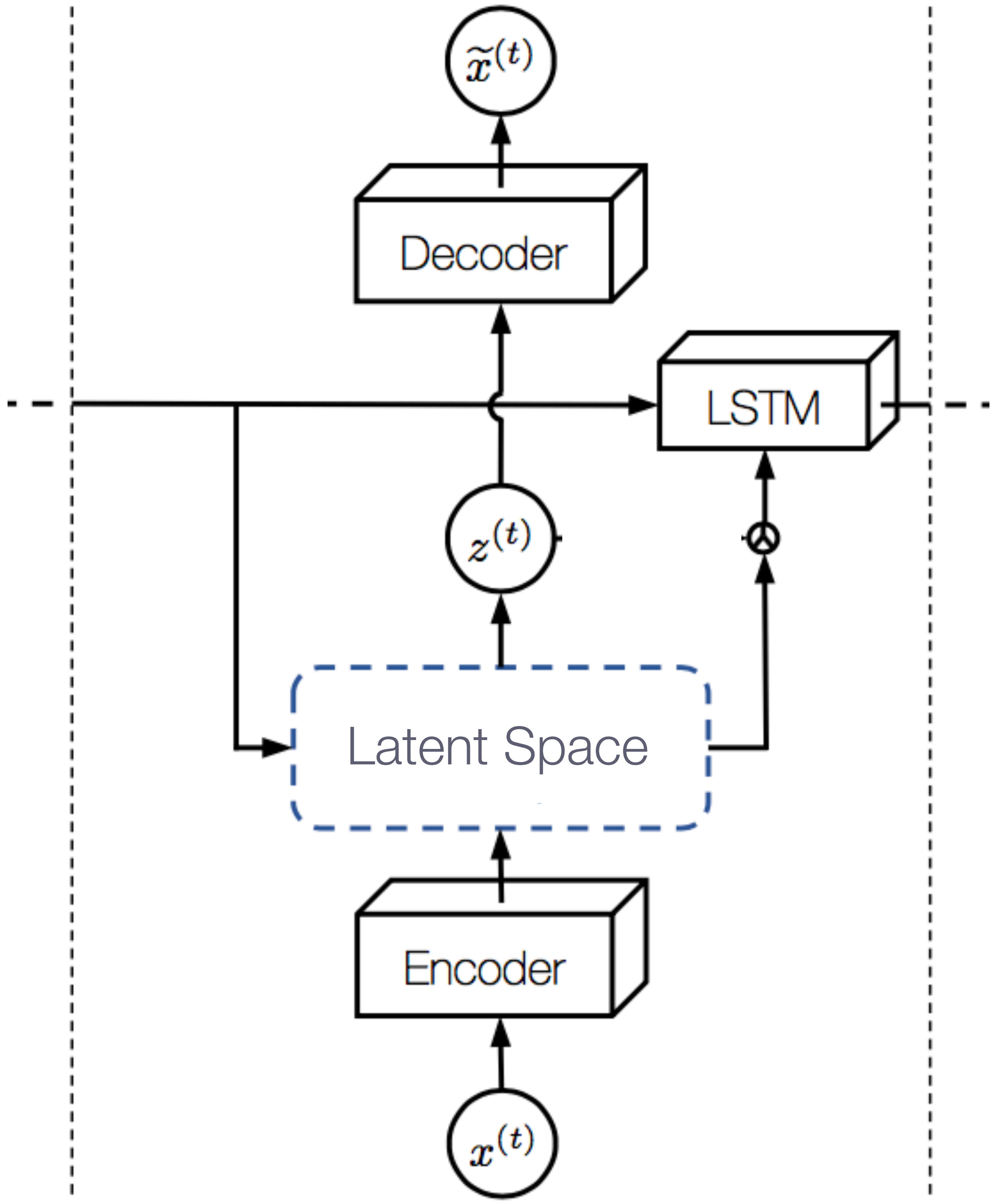
Variational Autoencoder (VAE)

[He et al., 2018]



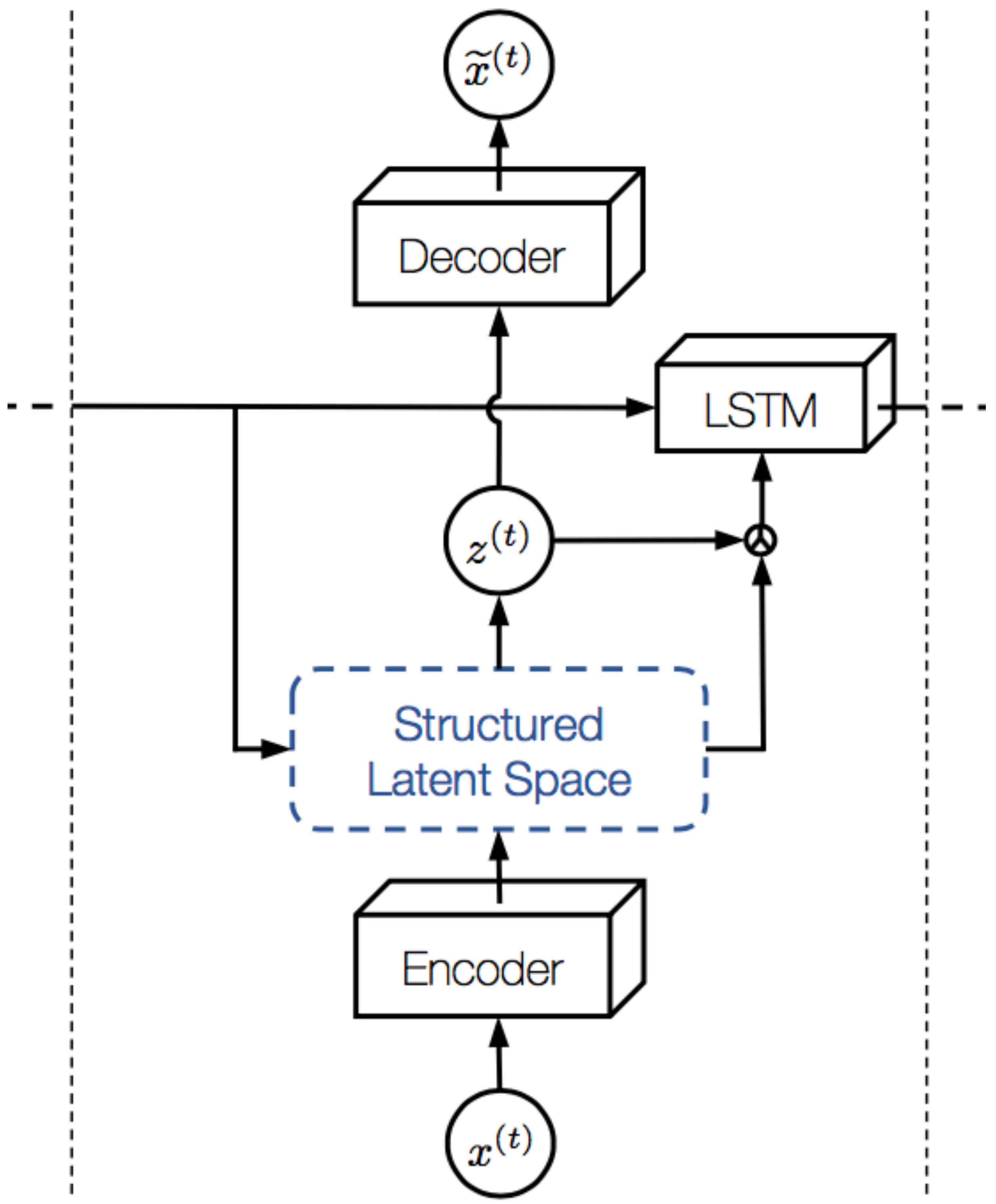
Variational Autoencoder (VAE) + **LSTM**

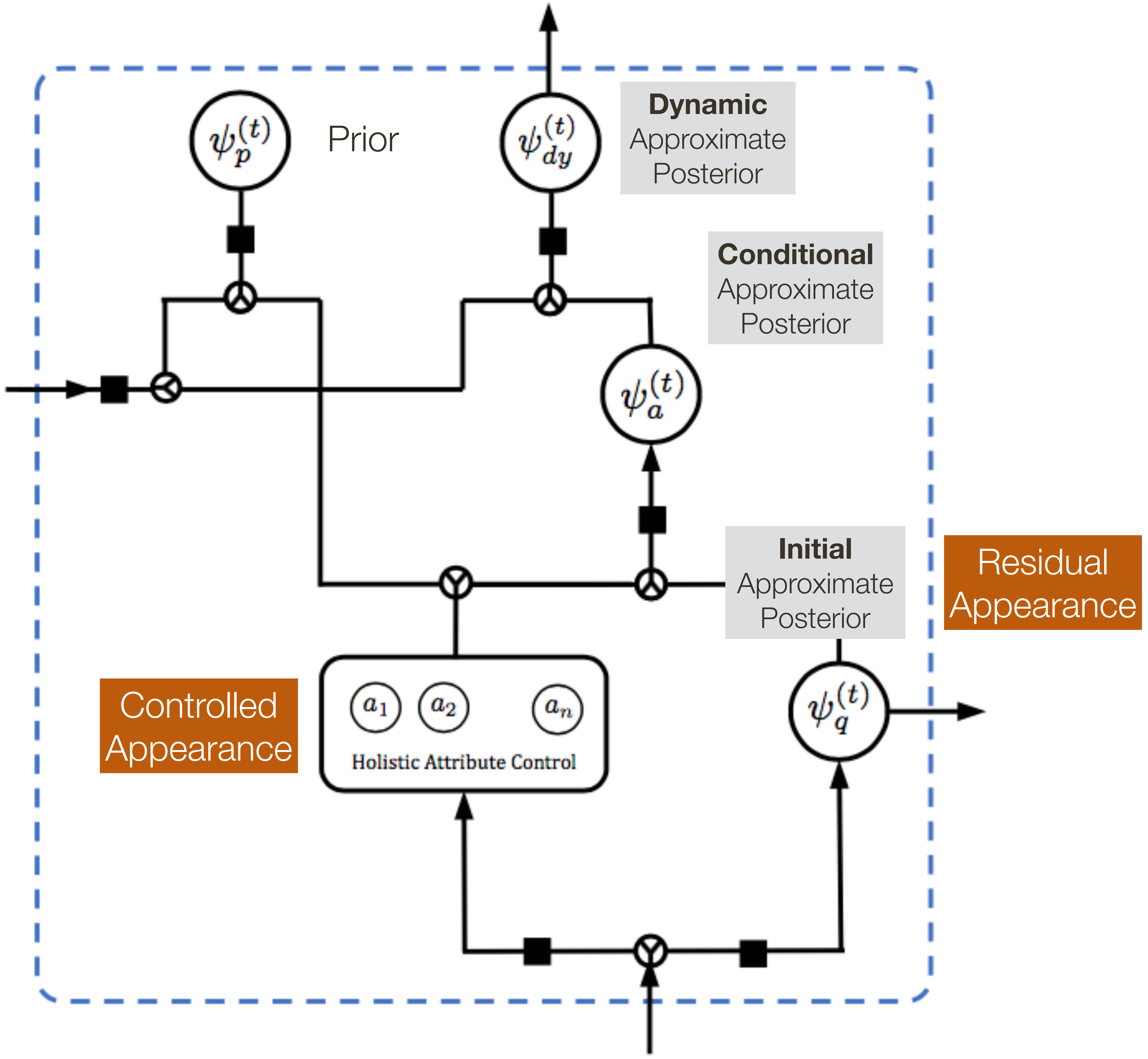
[He et al., 2018]



VAE + LSTM with Structured Latent Space

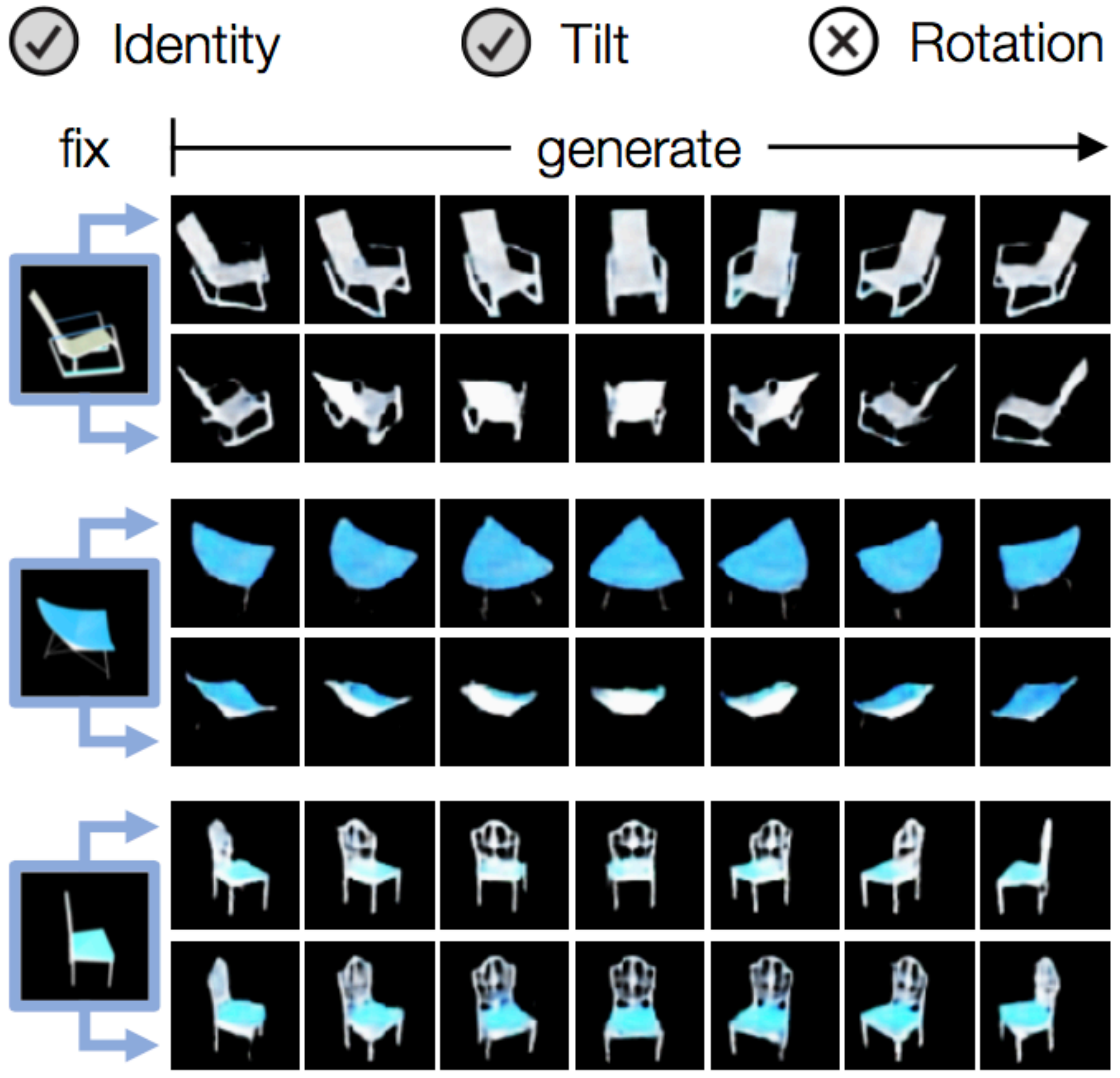
[He et al., 2018]



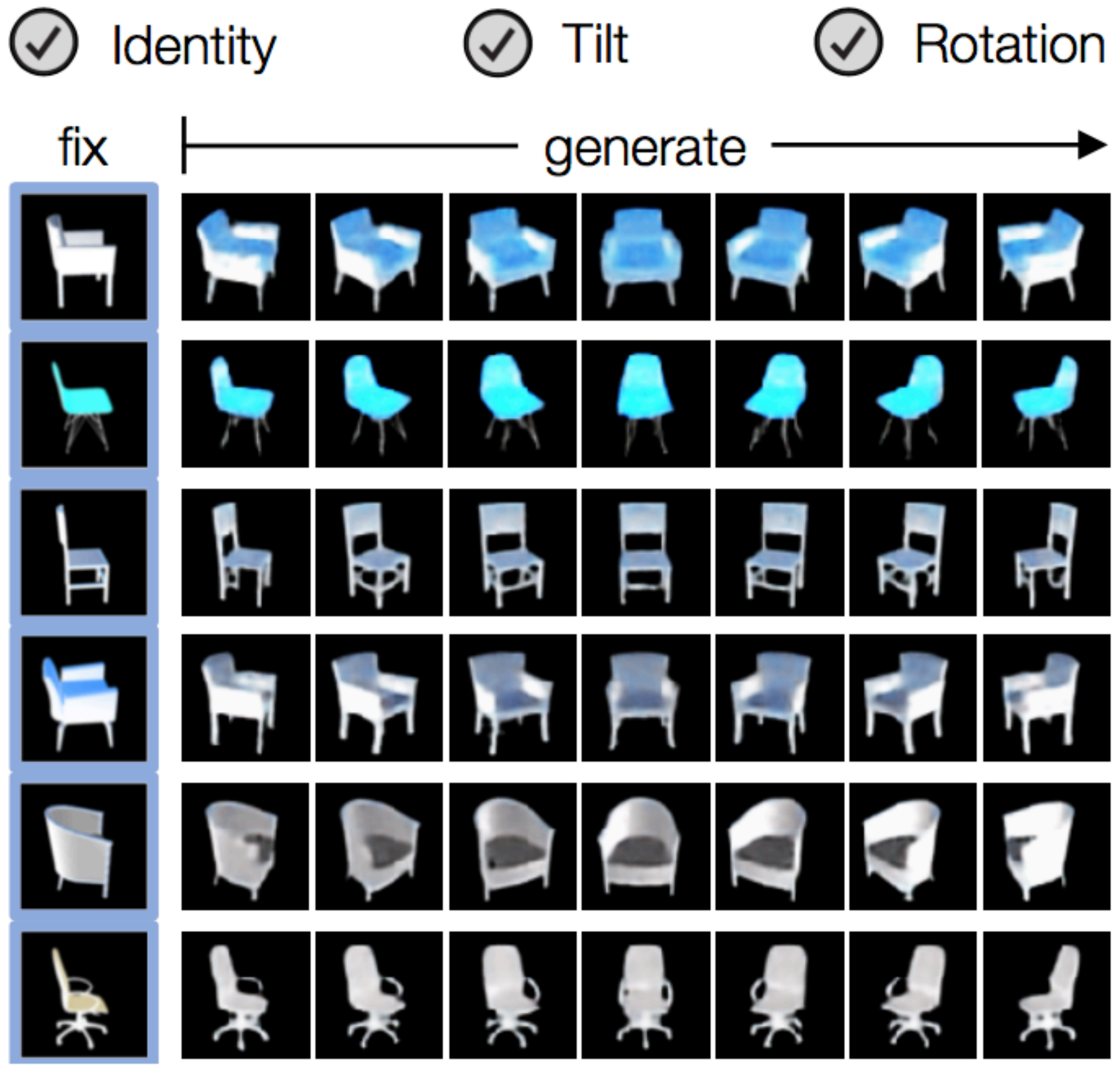


Results: Chair CAD dataset

[He et al., 2018]



(a) Partial control.



(b) Full control.

Ablation

		Bound	Static	$-C$		$+C$	
				$-S$	$+S$	$-S$	$+S$
Intra-E	↓	1.98	40.33	17.64	7.79	14.81	**5.50**
Inter-E	↑	1.39	0.42	0.73	1.35	1.02	**1.37**
I-Score	↑	4.01	1.28	1.83	3.63	2.56	**3.94**

Quantitative

		Chair CAD [1, 40]		
		Bound	Deep Rot. [40] ◉	VideoVAE (**ours**) ◉
Intra-E	↓	1.98	14.68	**5.50**
Inter-E	↑	1.39	1.34	**1.37**
I-Score	↑	4.01	3.39	**3.94**

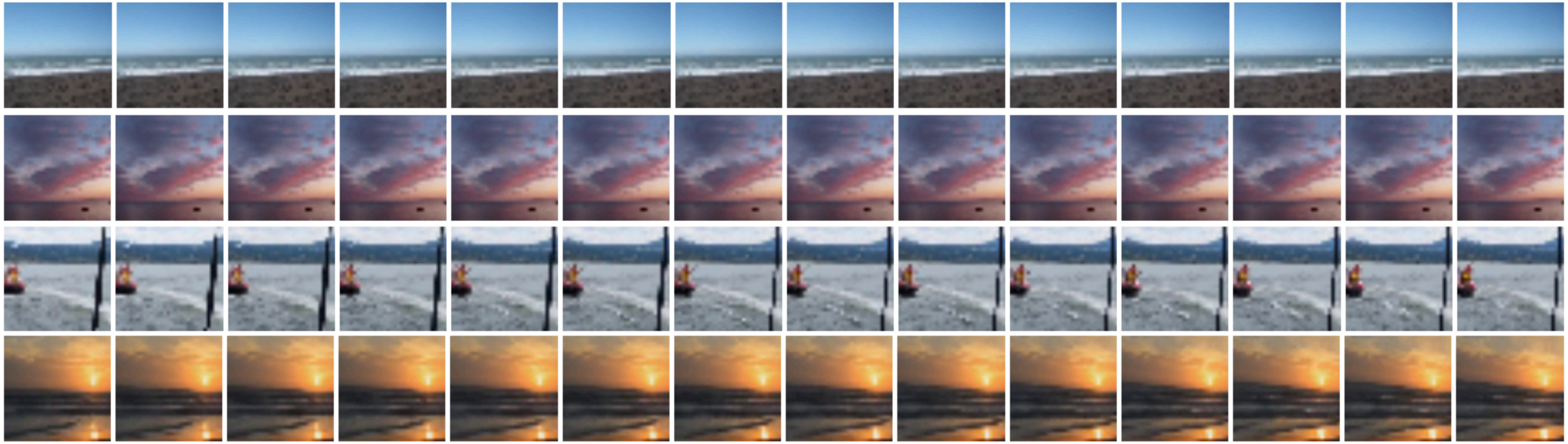
Results: Weizmann Human Action dataset

[He et al., 2018]

		Weizmann Human Action [2]			
		Bound	MoCoGAN [32]	VideoVAE (ours)	
			○	○	◉
Intra-E	↓	0.63	23.58	9.53	**9.44**
Inter-E	↑	4.49	2.91	**4.37**	**4.37**
I-Score	↑	89.12	13.87	69.55	**70.10**

Results: MIT Flickr

[He et al., 2018]



		YFCC [31] — MIT Flickr [34]			
		Bound	VGAN [34]	VideoVAE (ours)	
			○	○	◉
Intra-E	↓	30.34	46.96	44.03	**38.20**
Inter-E	↑	0.693	**0.692**	0.691	**0.692**
I-Score	↑	1.87	1.58	1.62	**1.81**

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)

Active area of research:
- More flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian
- Incorporating structure in latent variables (our submission to CVPR)

VAE /w (powerful) PixelCNN Decoder

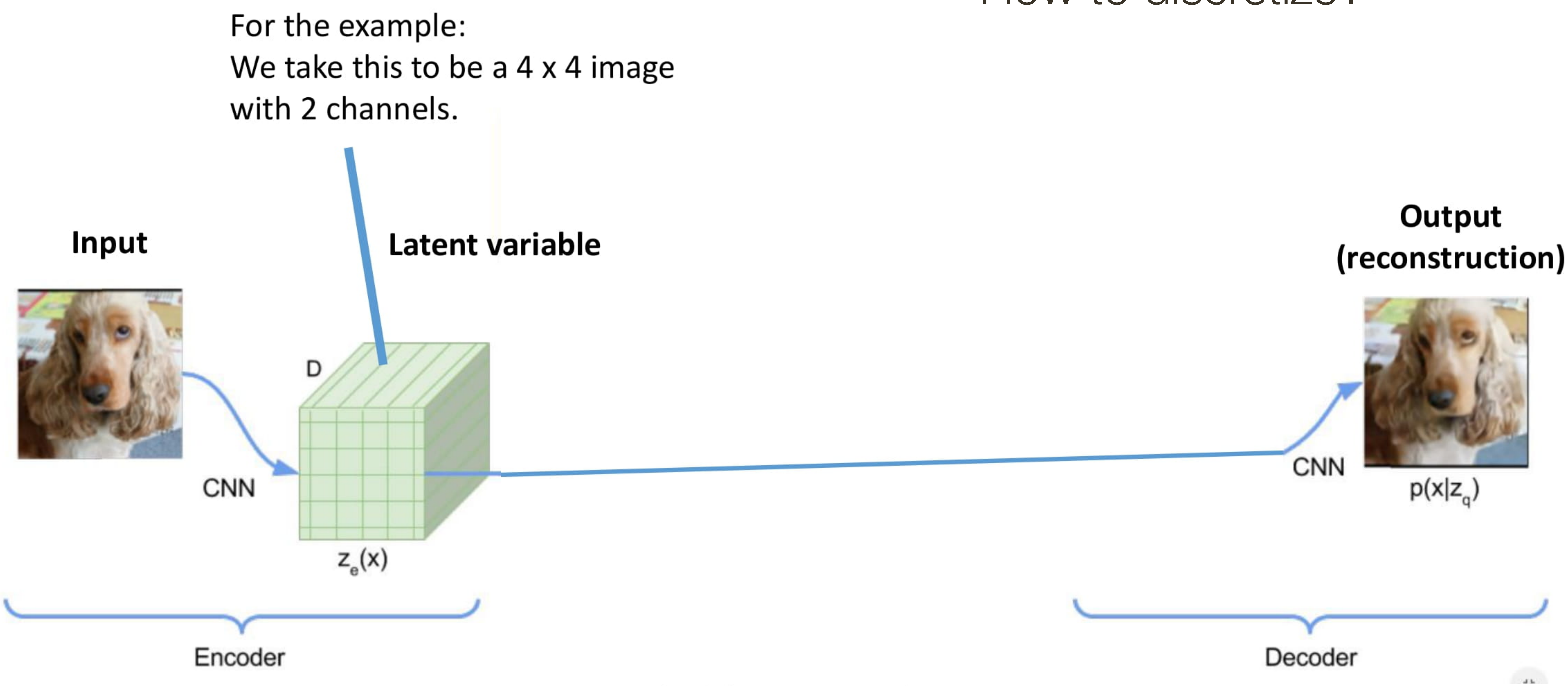
Problem: If the decoder is too powerful, it may just ignore the latent variables (i.e. posterior collapse). This happens when the decoder can make the reconstruction loss incredibly small, such that the regularization term dominates the loss function. In such a case, the encoder will learn to reduce the regularization term, and produce meaningless latents to match $p(z) = N(0, 1)$.

Vector Quantized Variational Autoencoders (VQ-VAE)

Autoencoders Reminder …

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

How to discretize?

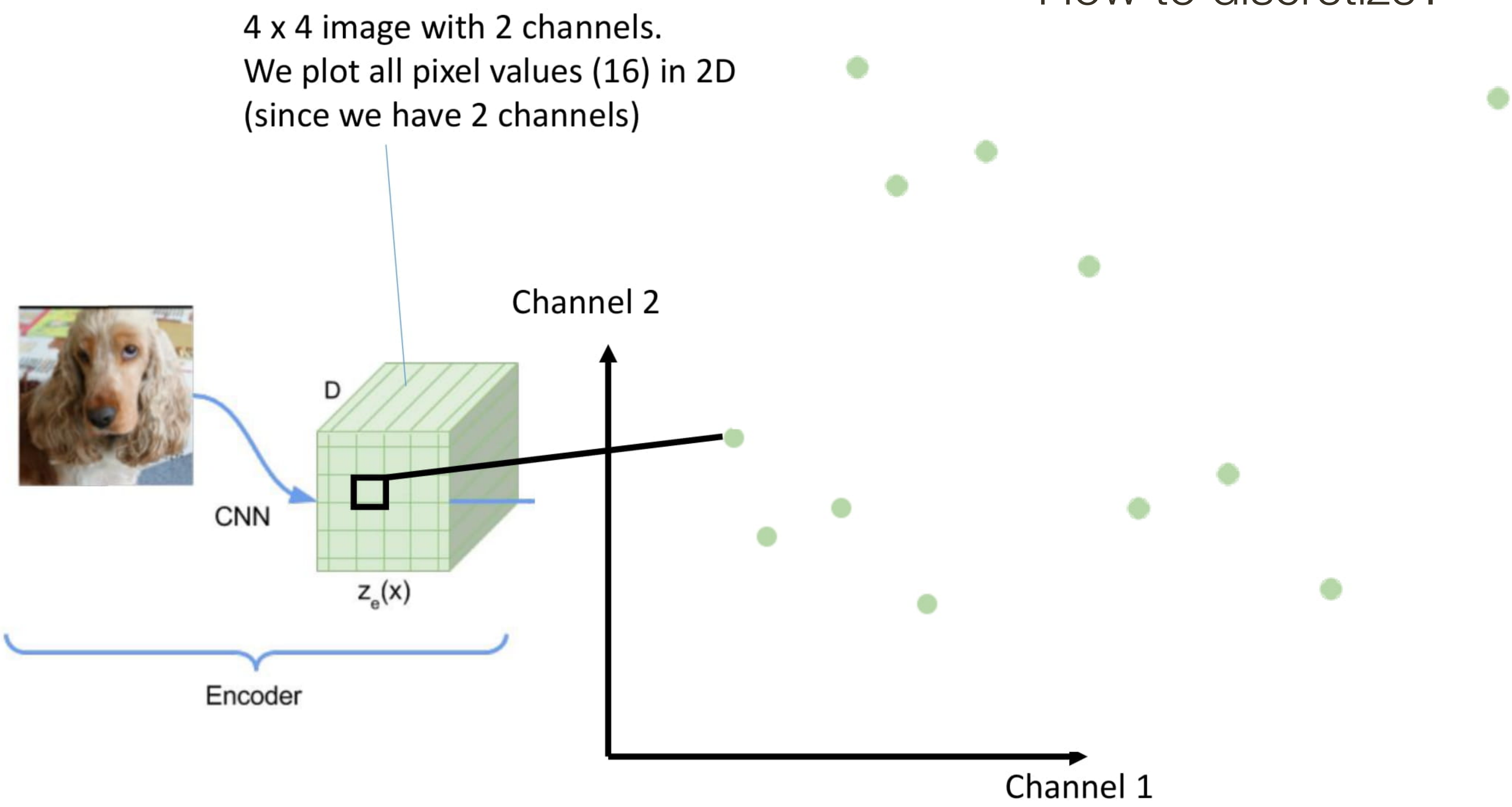


We can train this system end-to-end using MSE (reconstruction loss)

Autoencoders Reminder …

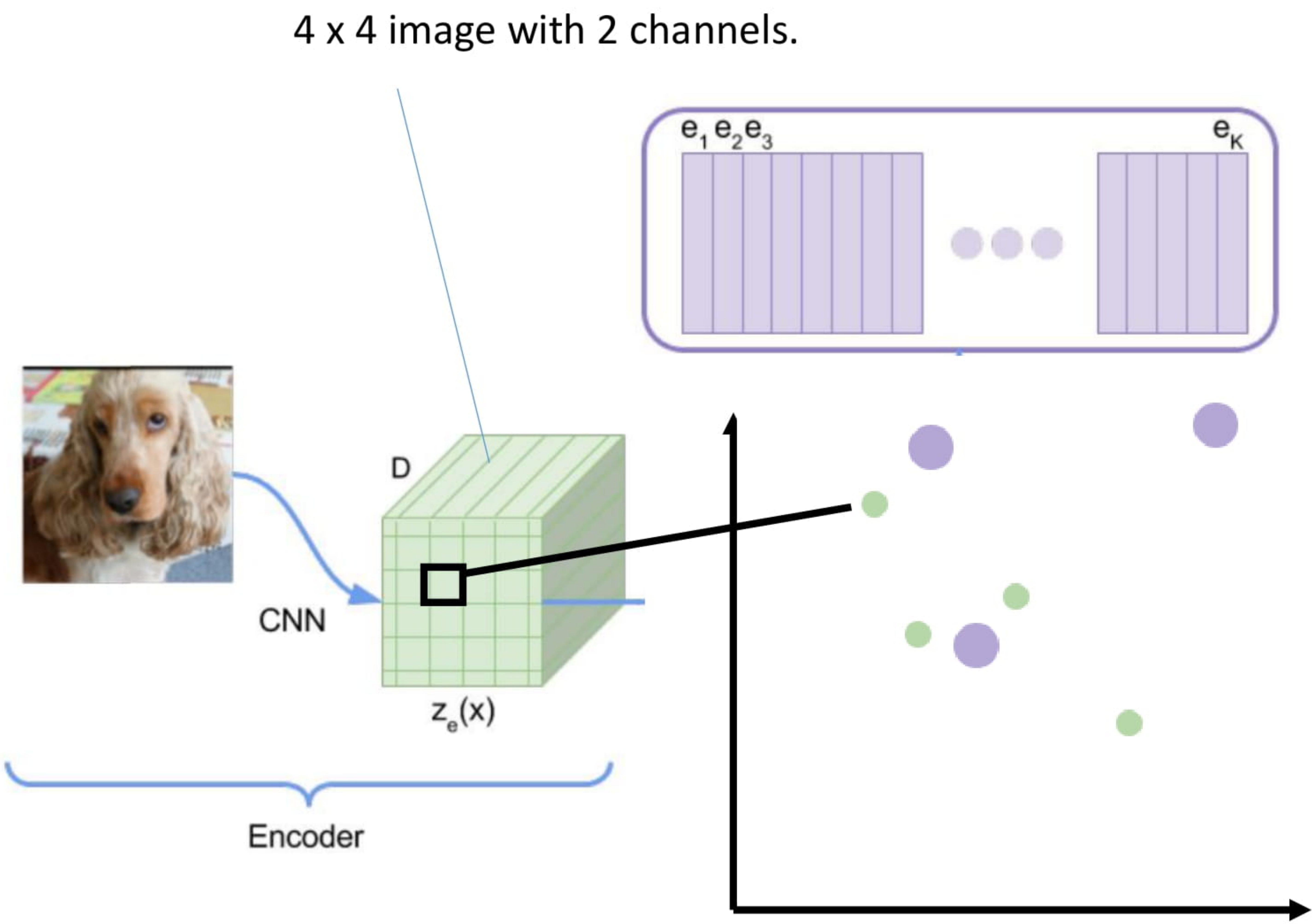
Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

How to discretize?



Vector Quantized - VAE

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

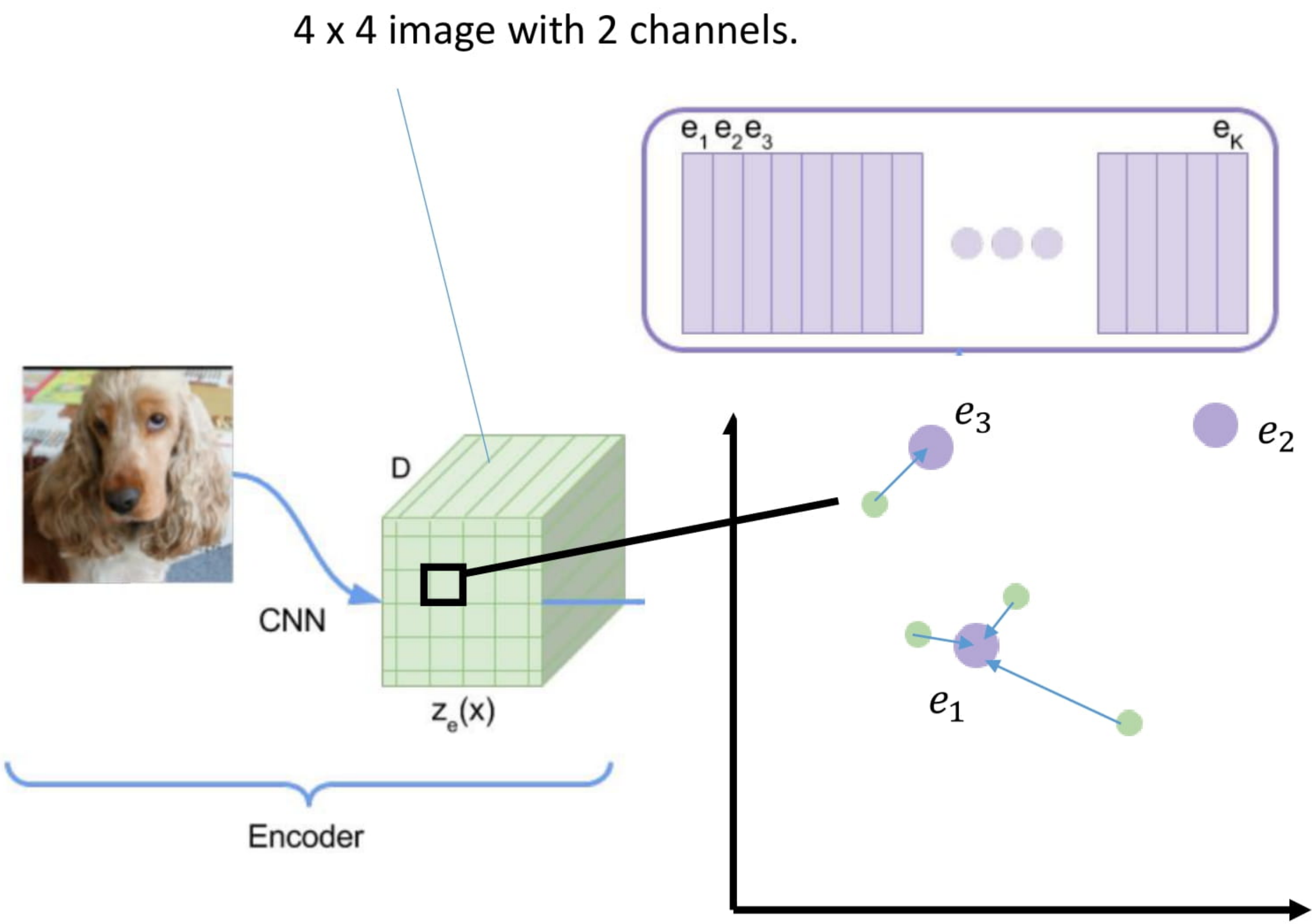


Make dictionary of vectors
$e_1, \ldots, e_K$

Each e_i has 2 dimensions.

Vector Quantized - VAE

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf



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

Each e_i has 2 dimensions.

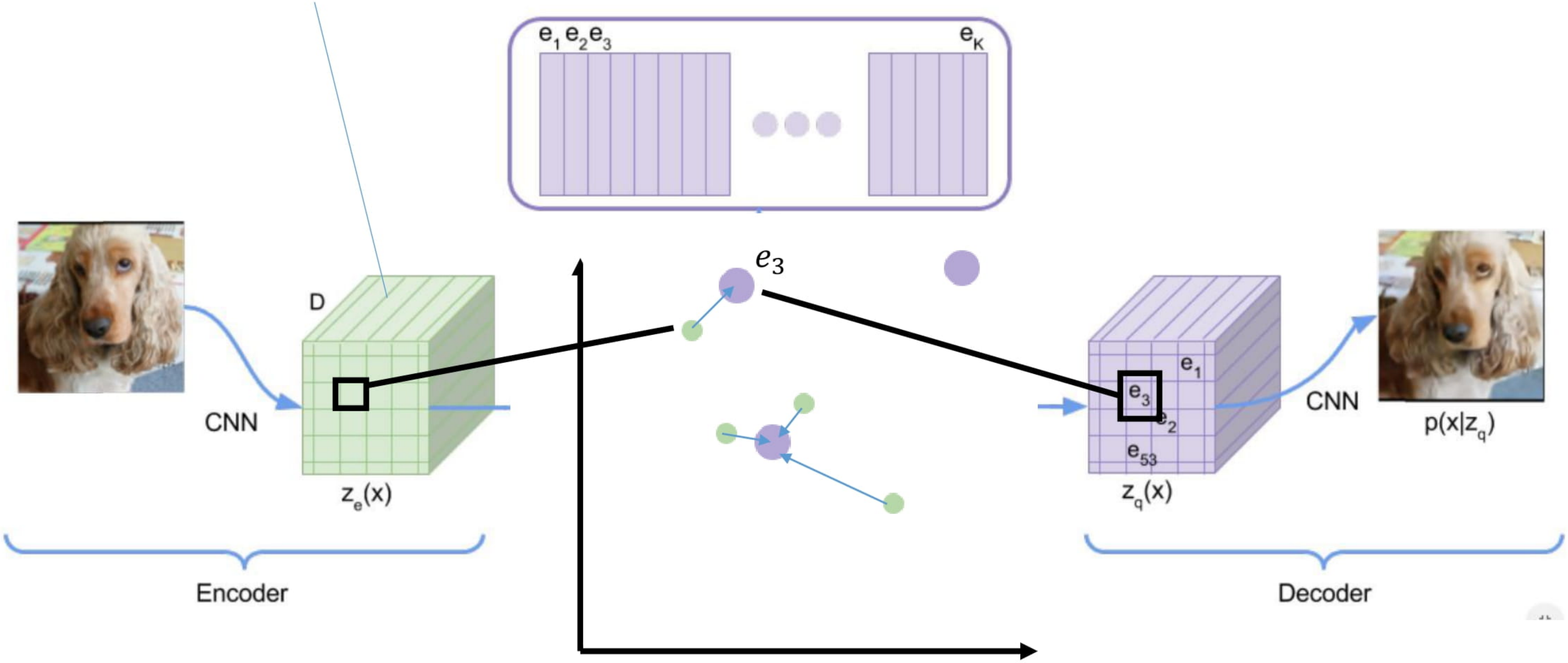
For each latent pixel, look up nearest dictionary element e

Vector Quantized - VAE

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

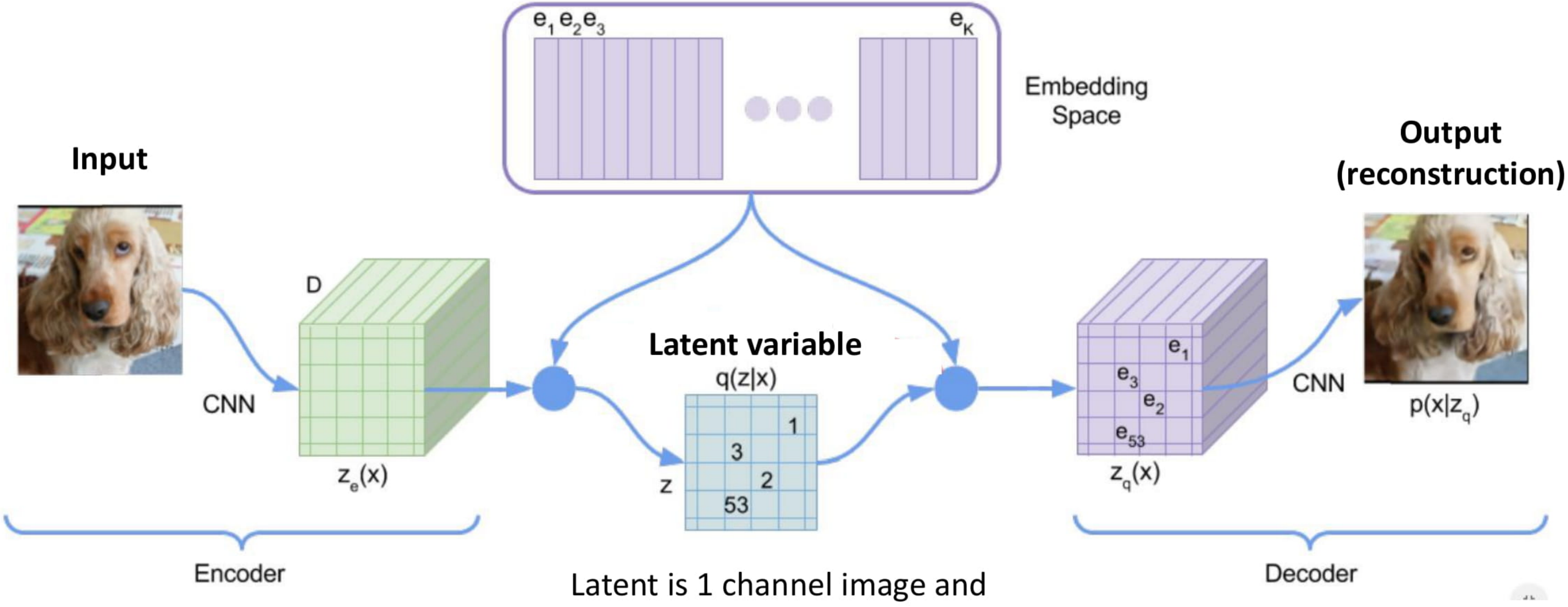
4 x 4 image with 2 channels.

Each e_i has 2 dimensions.



Vector Quantized - VAE

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

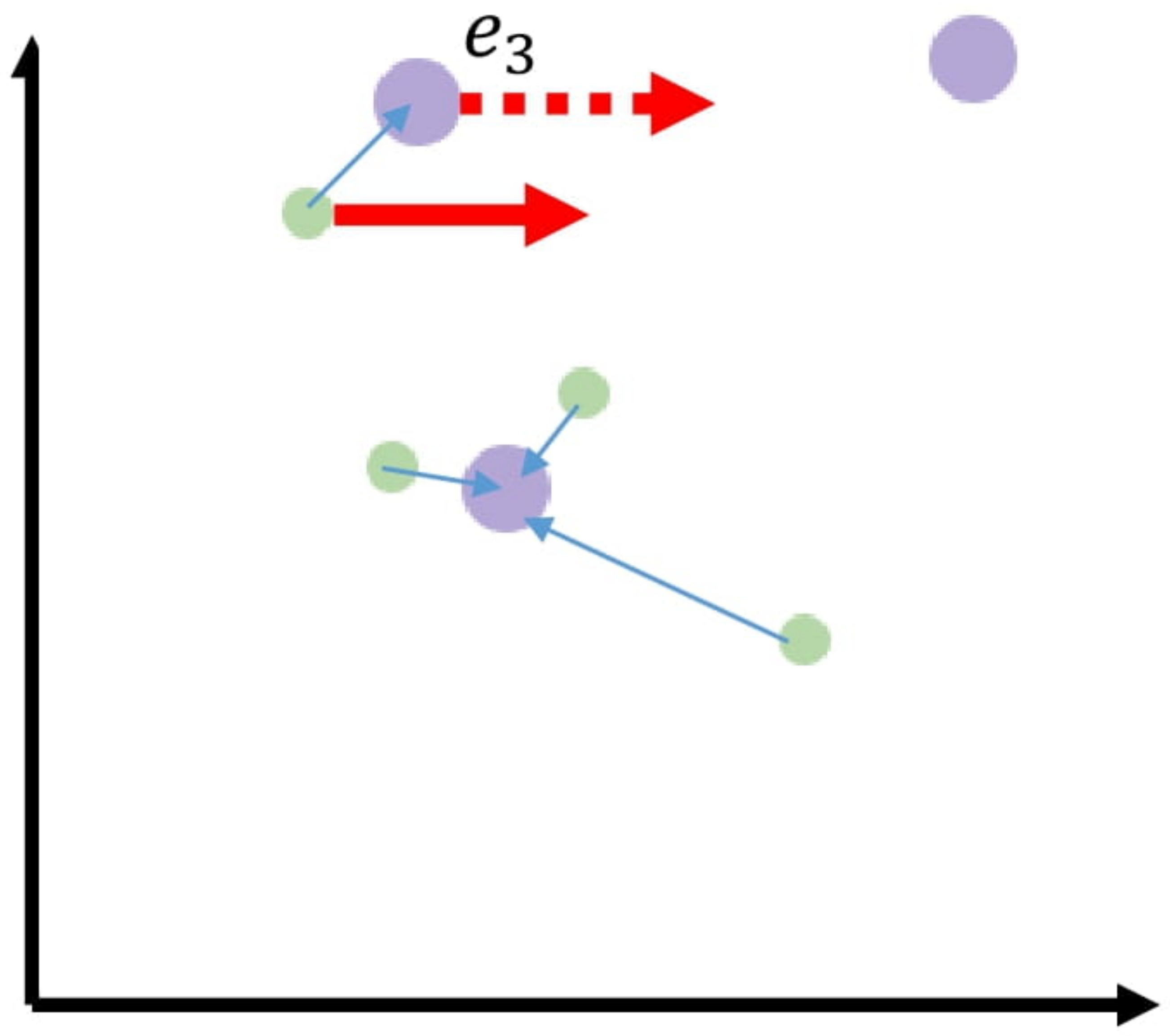


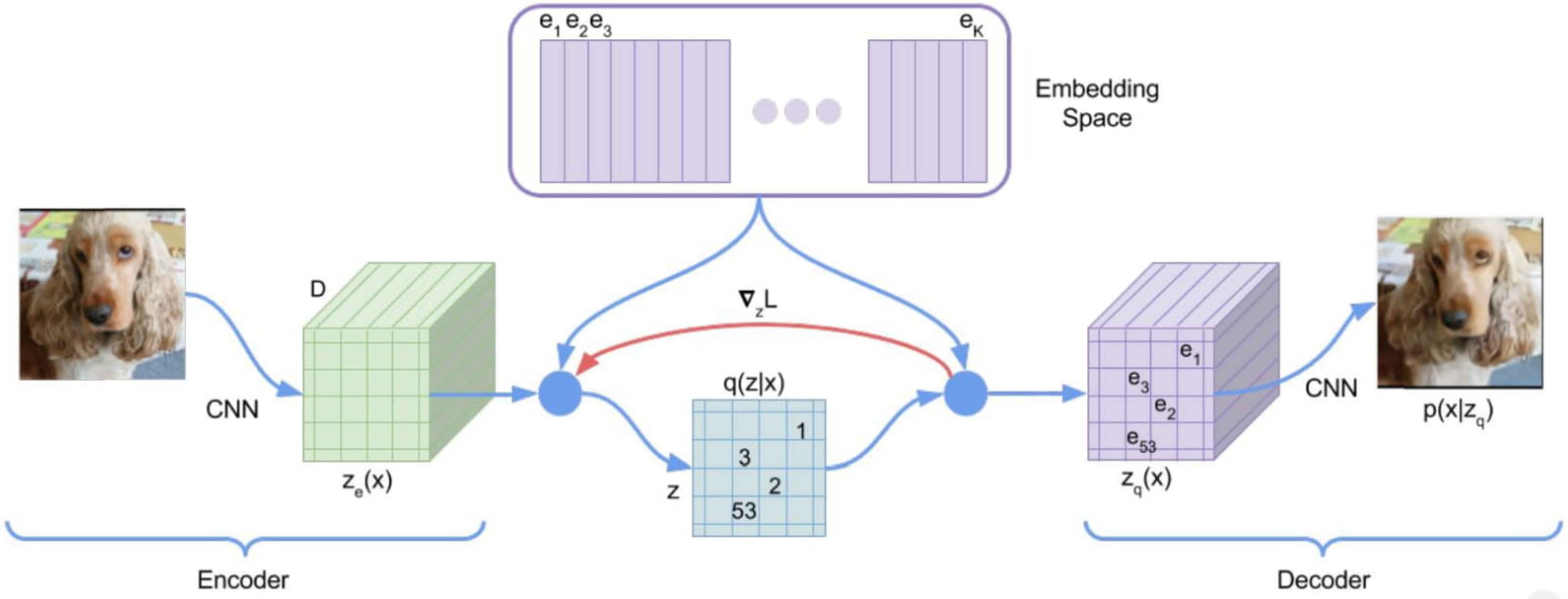
Latent is 1 channel image and contains the id of each e for each pixel (**discrete**).

VQ-VAE — **Training**

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

- 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



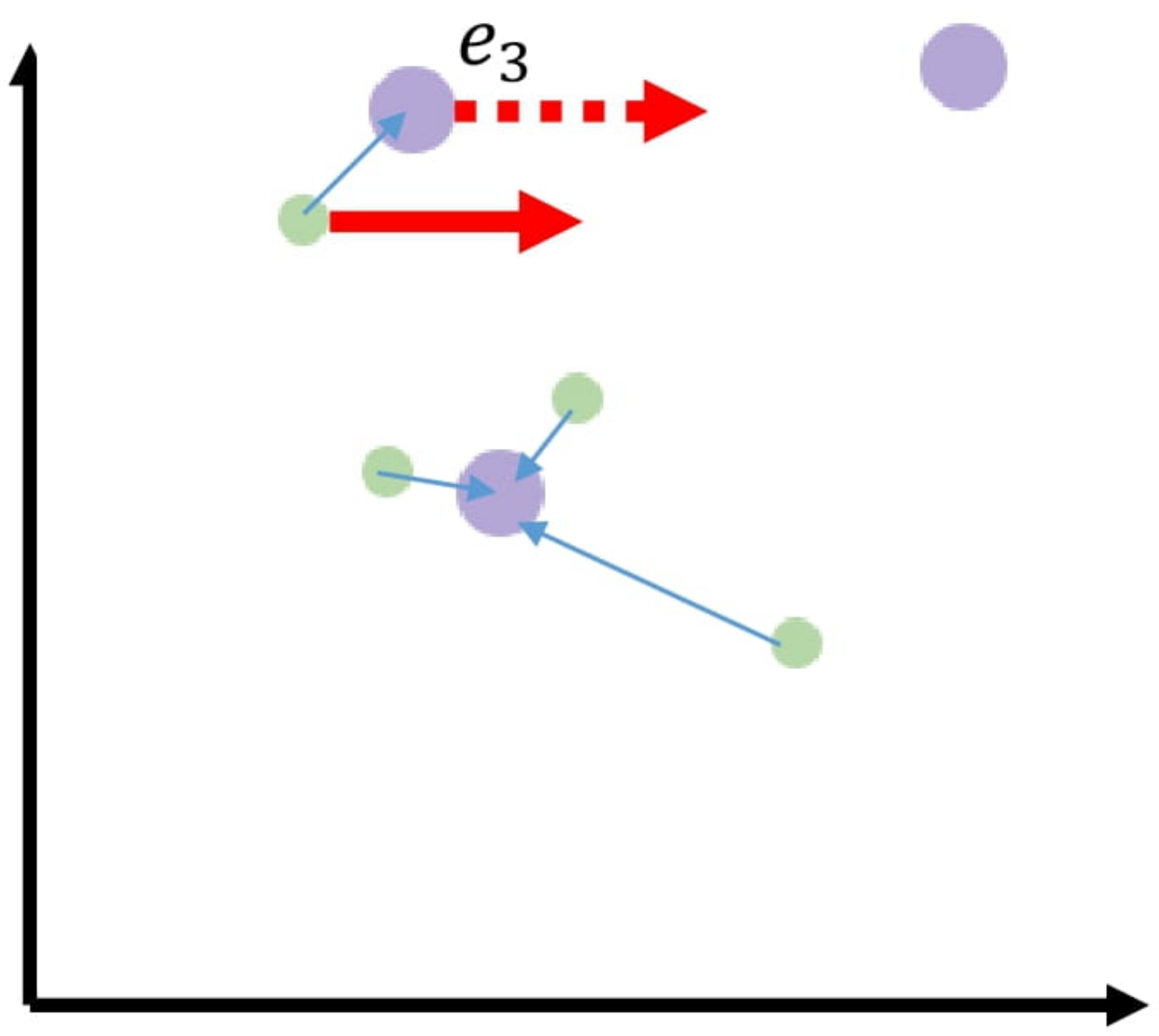


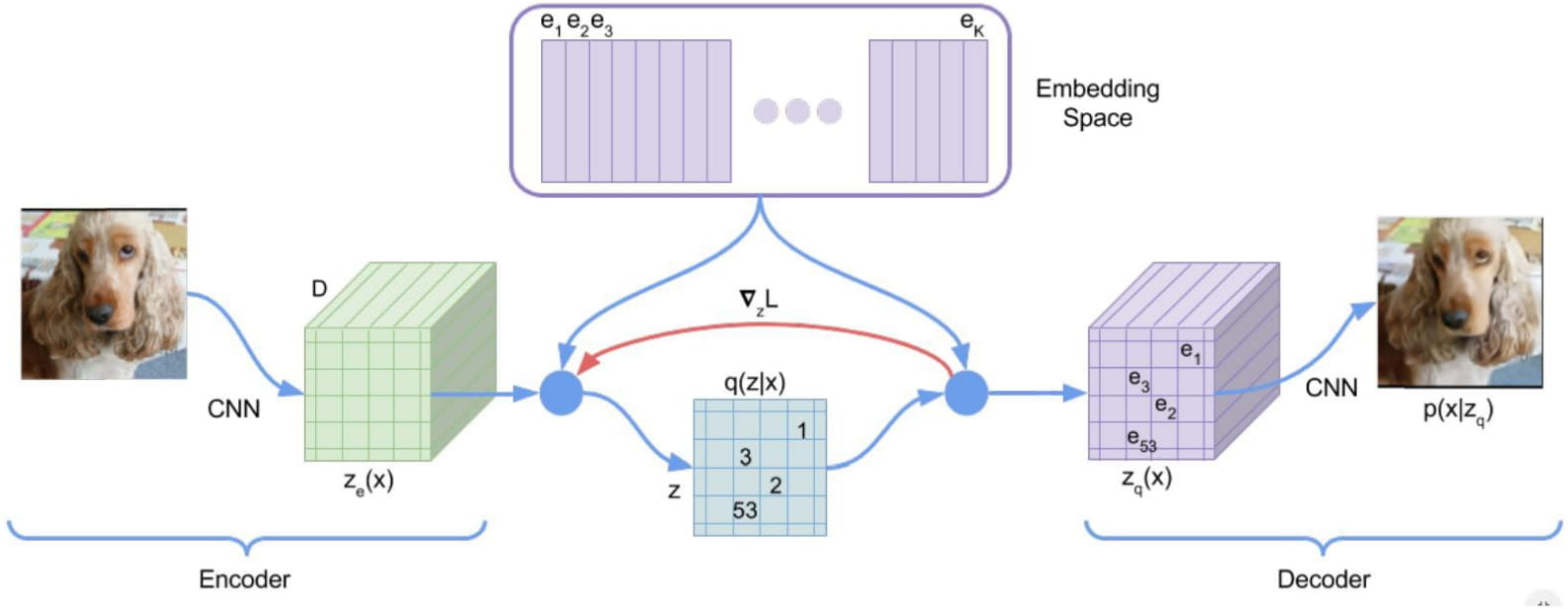
VQ-VAE — **Training**

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

$$L = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta \|z_e(x) - \text{sg}[e]\|_2^2,$$

- 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





VQ-VAE — **Training**

Source: http://www.tomviering.nl/talks/slides/2018_01_09.pdf

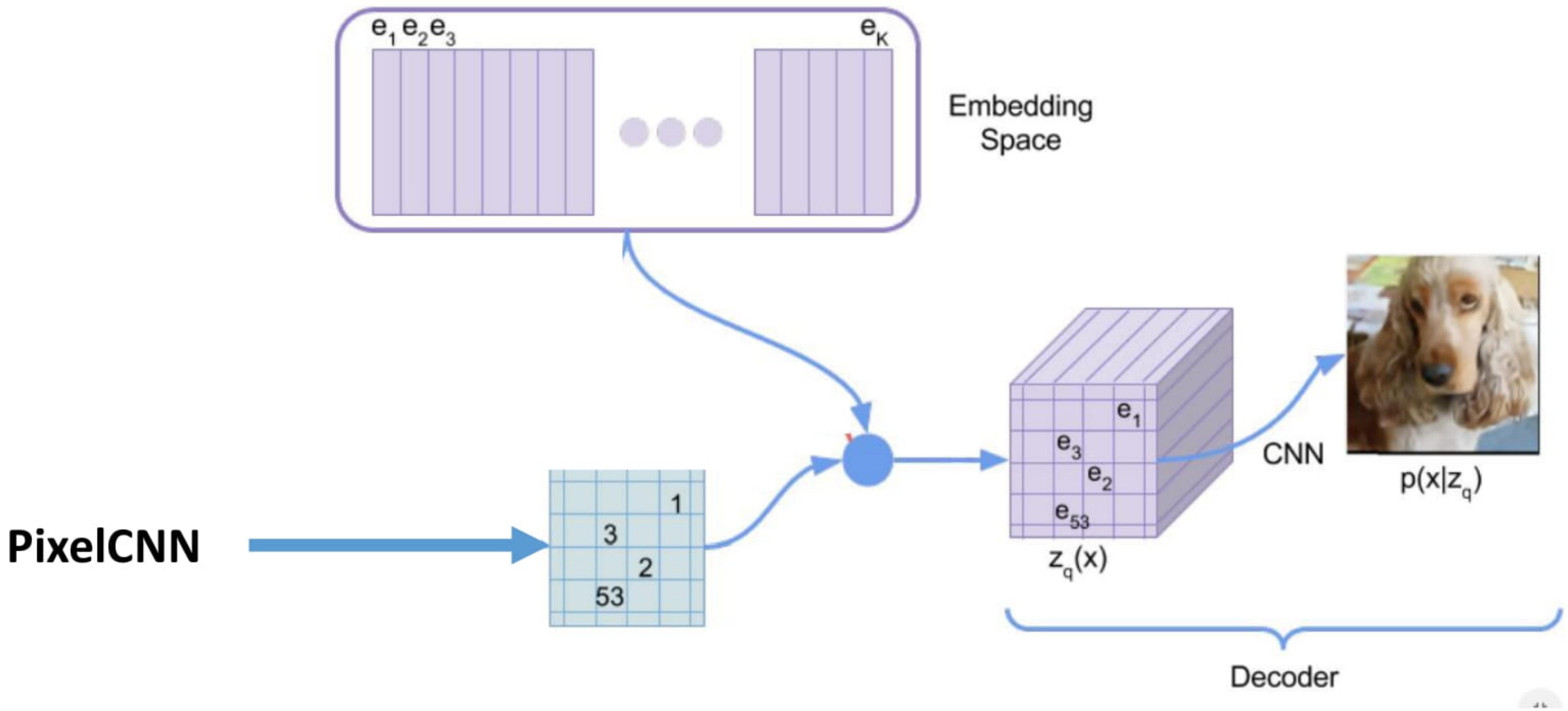
$$L = \log p(x|z_q(x)) + \|\text{sg}[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - \text{sg}[e]\|_2^2,$$

Reconstruction loss, which optimizes both the encoder and decoder

Regularization, ensures that encoder does not grow arbitrarily

VQ loss, which moves the embedding vectors towards encoder outputs

VQ-VAE — **Sampling / Generation**



Class: pickup

VQ-VAE — **Sampling / Generation**

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

Figure 2: Left: ImageNet 128x128x3 images, right: reconstructions from a VQ-VAE with a 32x32x1 latent space, with K=512.

VQ-VAE vs. GAN

Relationship of VQ-VAE to VAE

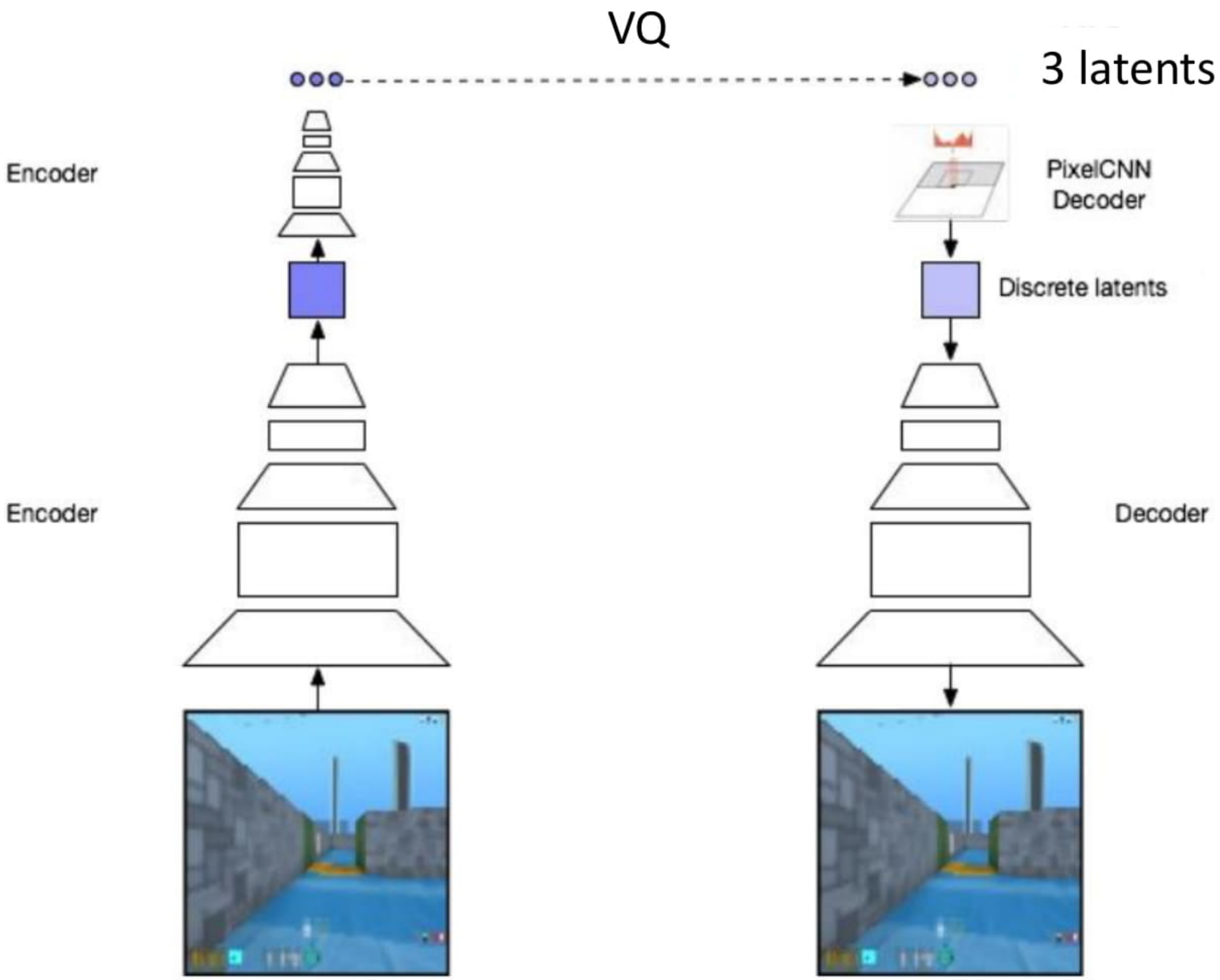
VAE: Assumes Gaussian prior over continuous latent space

VQ-VAE: Assumes uniform categorical distribution over discrete keywords (all keywords are equally likely)

Comparison

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

Multi-stage VQ-VAE



3 latents in [0,512]

21 x 21 x 1 in
[0,512]

84 x 84 x 3
In [0,256]

Before: 84 * 84 * 3 * 8 =
21168 bits = 3 Kb

After
3 * 9 = 27 bits

Reconstruction not very
accurate but powerful
representation

So far …

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

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

VAEs define intractable density function with latent variables z (that we need to marginalize):

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

cannot optimize directly, derive and optimize lower bound of likelihood instead

What if we give up on explicitly modeling density, and just want to sample?

So far …

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

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

VAEs define intractable density function with latent variables z (that we need to marginalize):

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

cannot optimize directly, derive and optimize lower bound of likelihood instead

What if we give up on explicitly modeling density, and just want to sample?

GANs: don't work with any explicit density function