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 due Monday 11:59pm

— They are graded for completeness / development of the proposal

— They are **<u>not</u>** graded for quality of the idea

I will provide feedback on the proposals, but **<u>don’t wait for it</u>** to work on the projects

I will hold additional office hours for feedback on **Friday** … tentatively **4:30-5:30pm**

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

$$\underbrace{p(x)}_{\text{Likelihood of image } \boldsymbol{x}} = \prod_{i=1}^{n} \underbrace{p(x_i | x_1, ..., x_{i-1})}_{\text{Probability of i'th pixel value given all previous pixels}}$$

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

$$\underbrace{p(x)}_{\text{Likelihood of image } \boldsymbol{x}} = \prod_{i=1}^{n} \underbrace{p(x_i|x_1, ..., x_{i-1})}_{\text{Probability of i'th pixel value given all previous pixels}}$$

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

then maximize likelihood of training data

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

Optional subtitle

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**

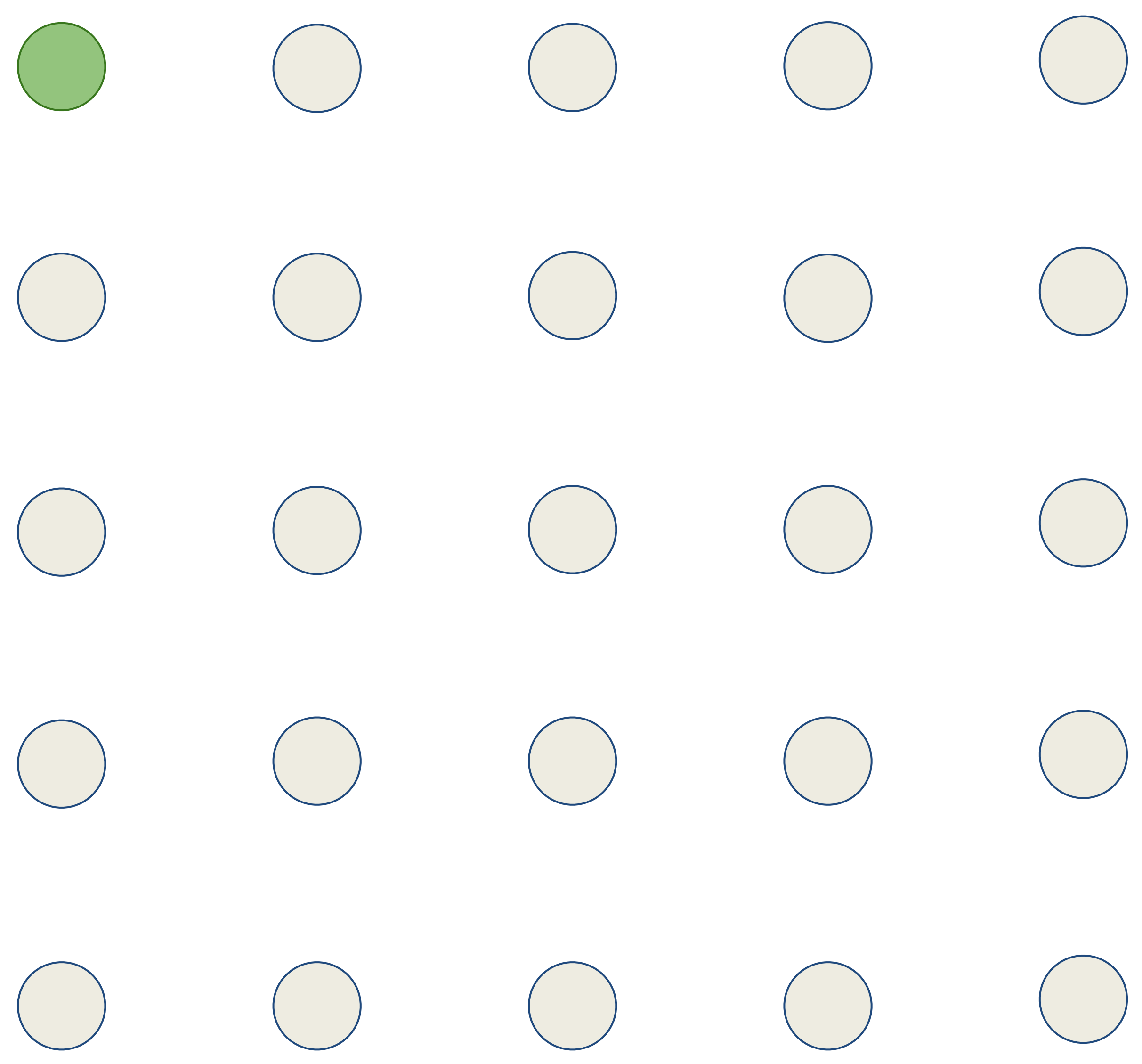
Also requires defining **ordering** of "previous pixels"

PixelRNN

[van der Oord et al., 2016]

Generate image pixels starting from the corner

Dependency on previous pixels model using an RNN (LSTM)

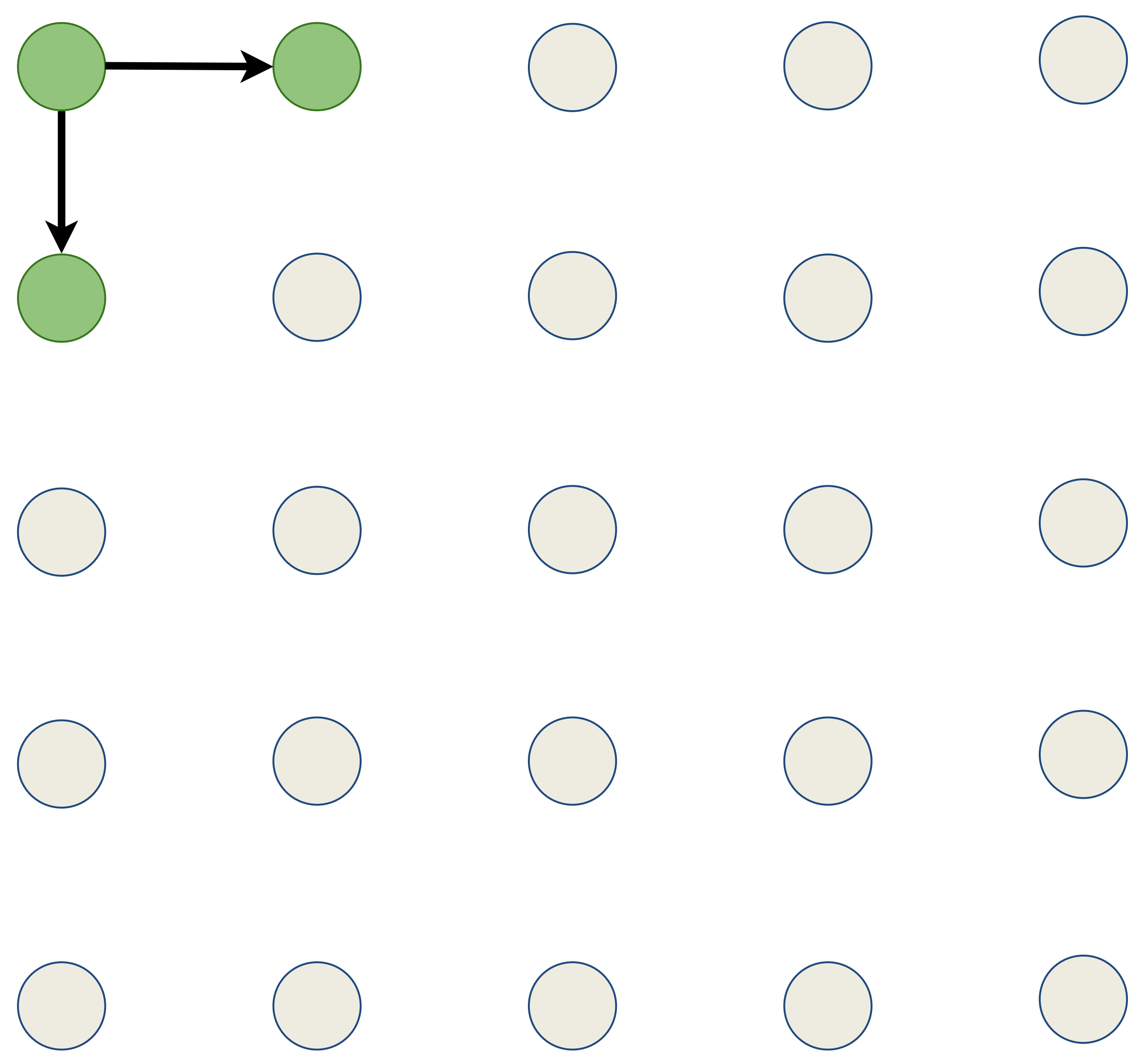


PixelRNN

[van der Oord et al., 2016]

Generate image pixels starting from the corner

Dependency on previous pixels model using an RNN (LSTM)

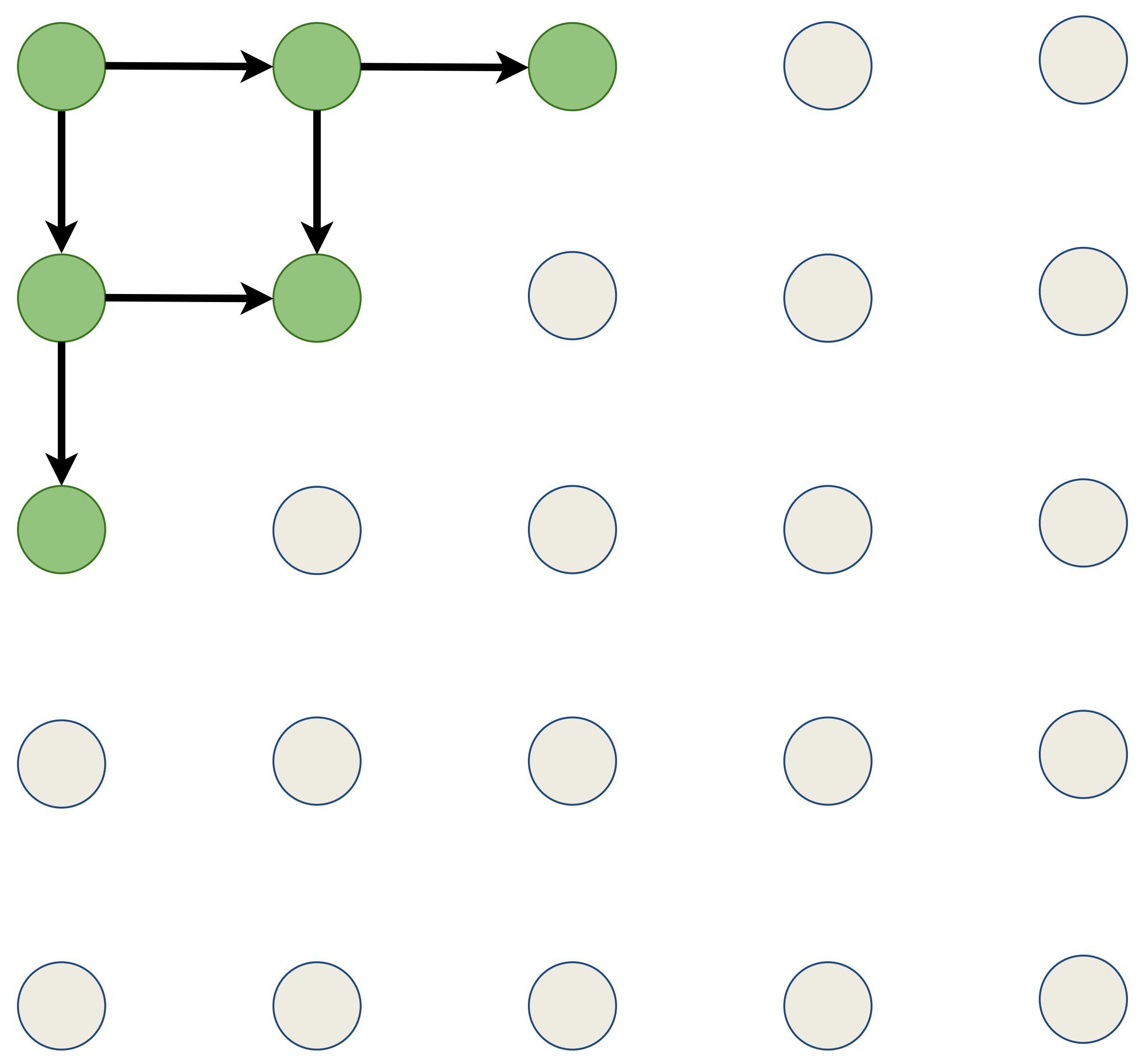


PixelRNN

[van der Oord et al., 2016]

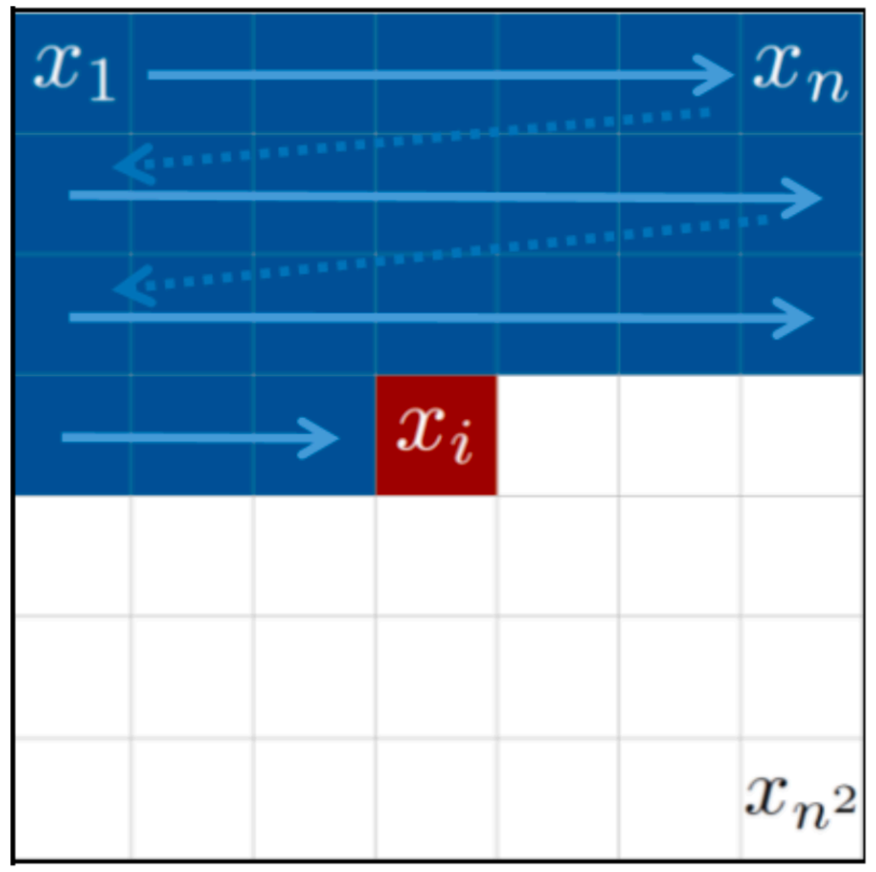
Generate image pixels starting from the corner

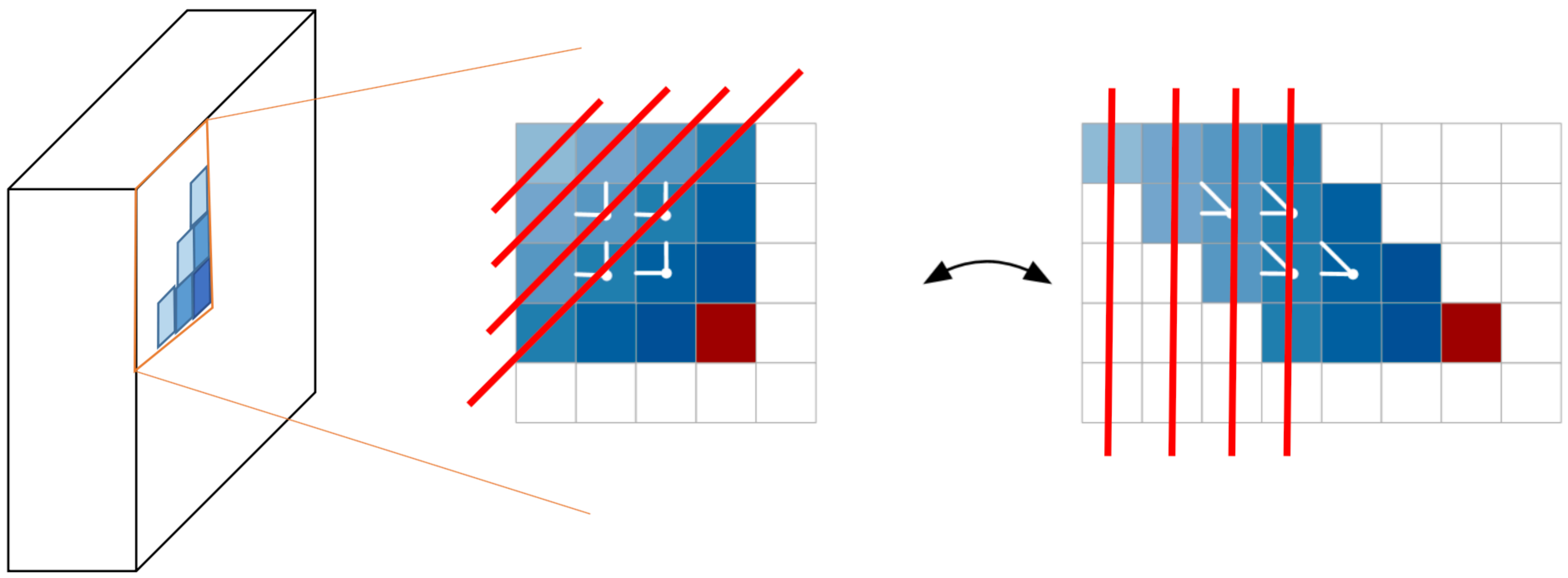
Dependency on previous pixels model using an RNN (LSTM)



PixelRNN

[van der Oord et al., 2016]



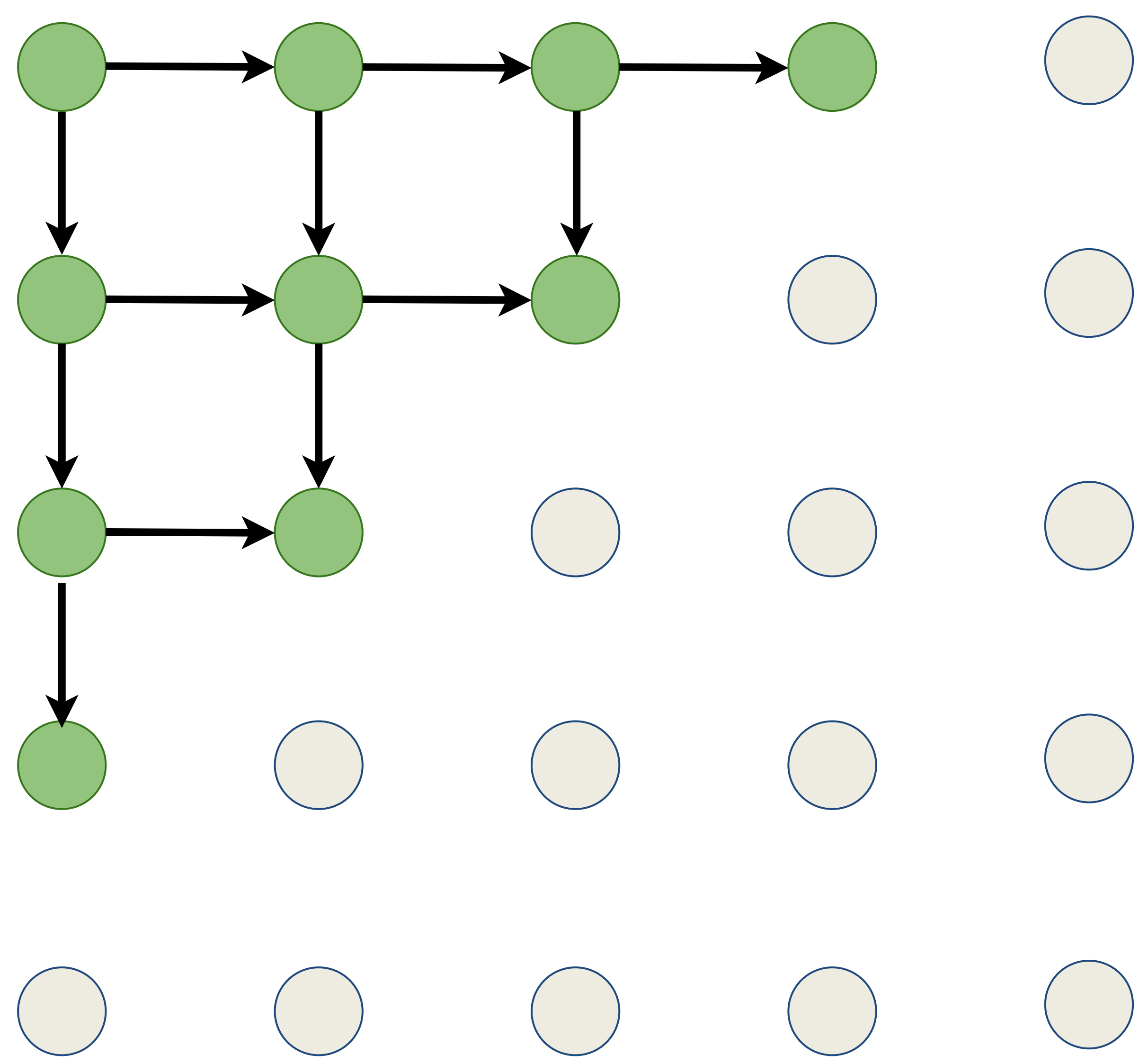


PixelRNN

[van der Oord et al., 2016]

Generate image pixels starting from the corner

Dependency on previous pixels model using an RNN (LSTM)

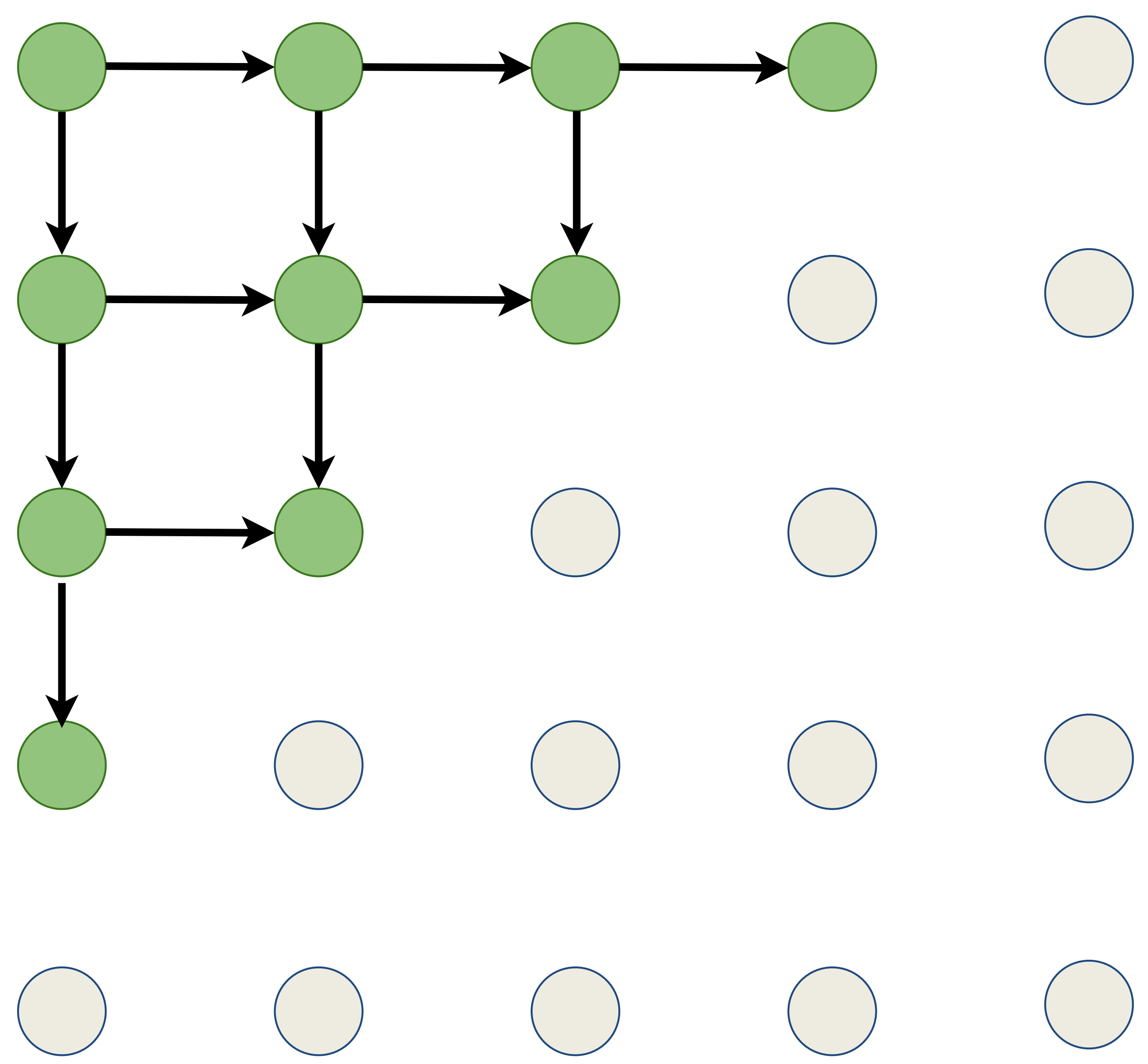


PixelRNN

[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

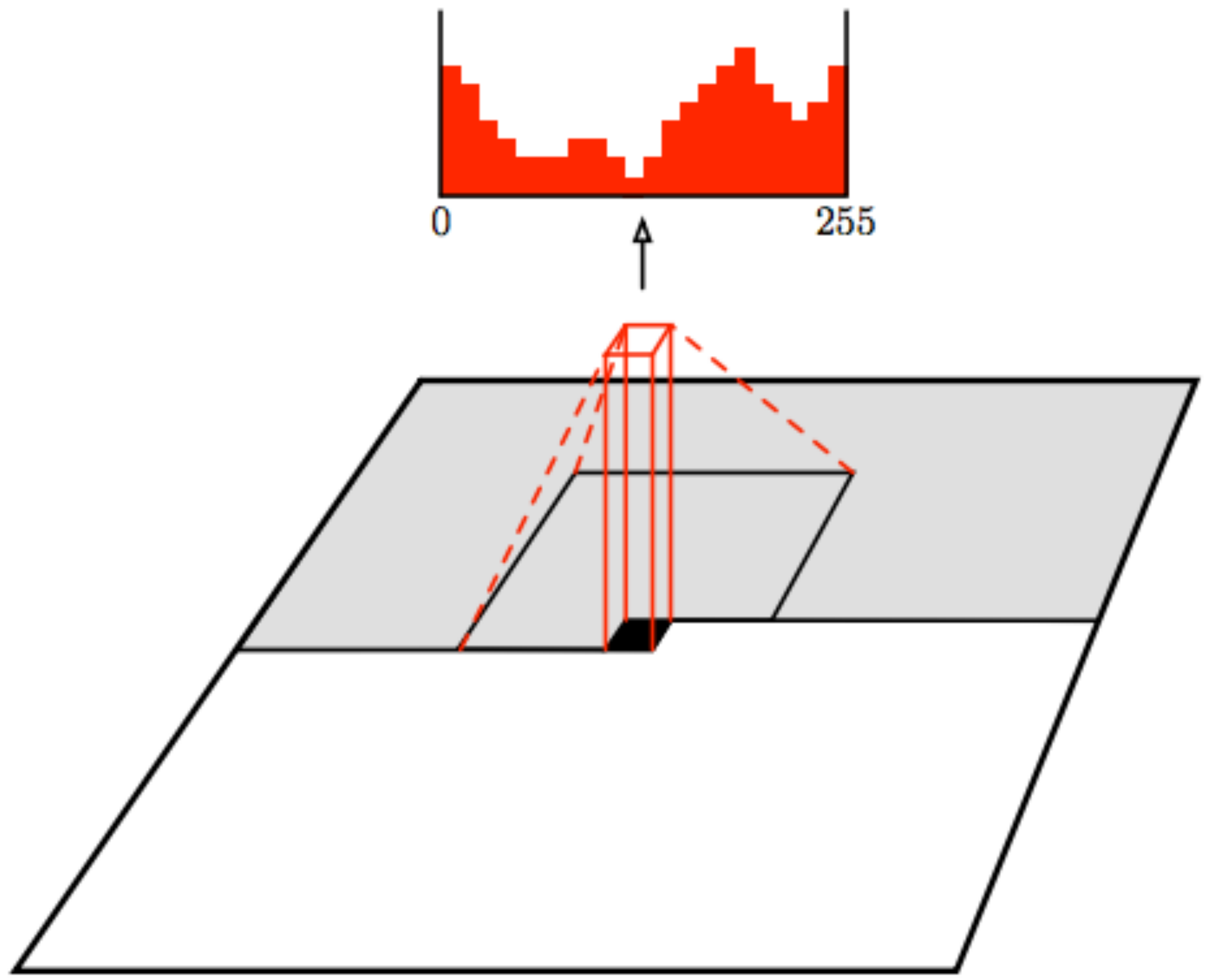
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

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]

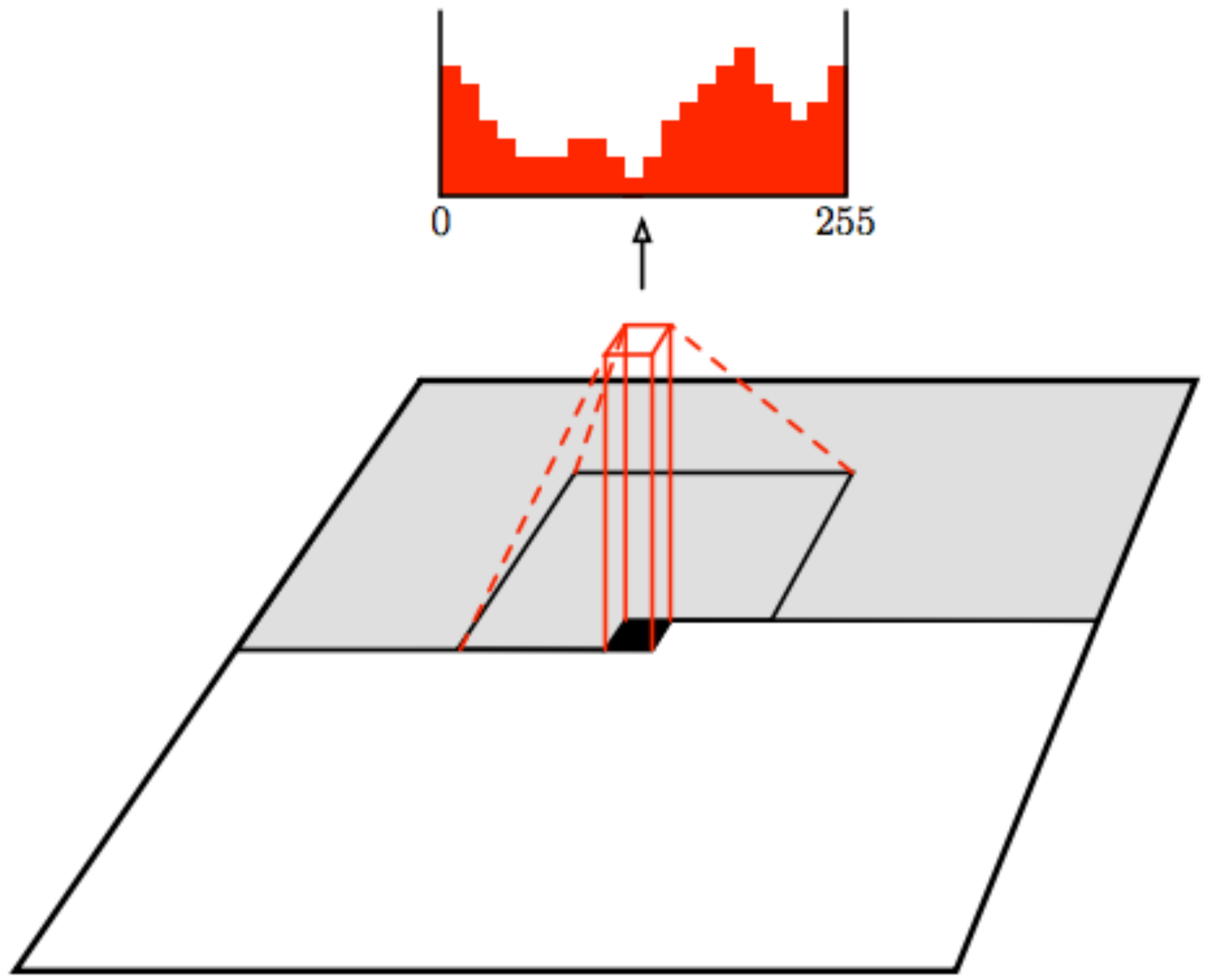
Still generate image pixels starting from the corner

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



PixelCNN

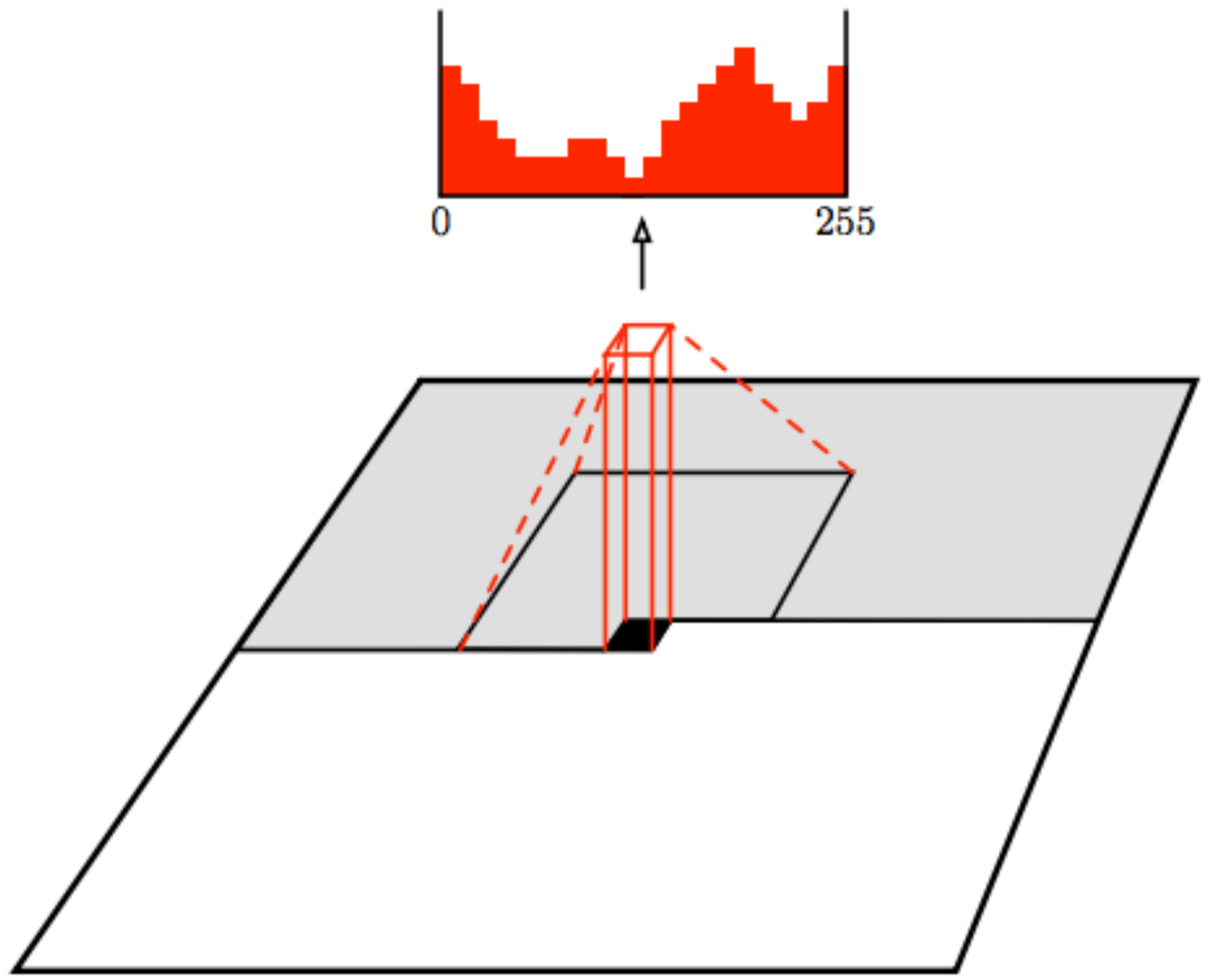
[van der Oord et al., 2016]

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

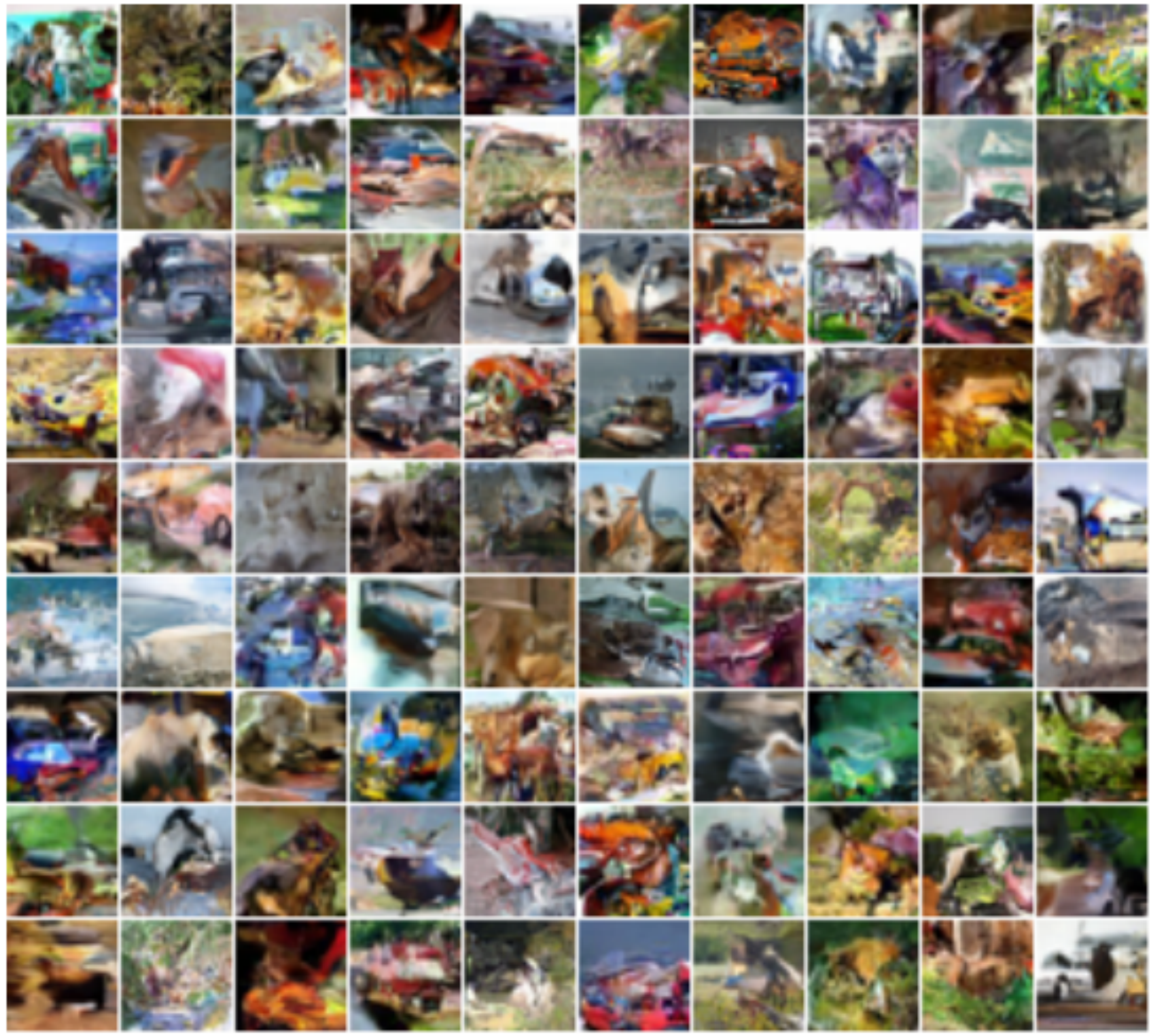
$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$



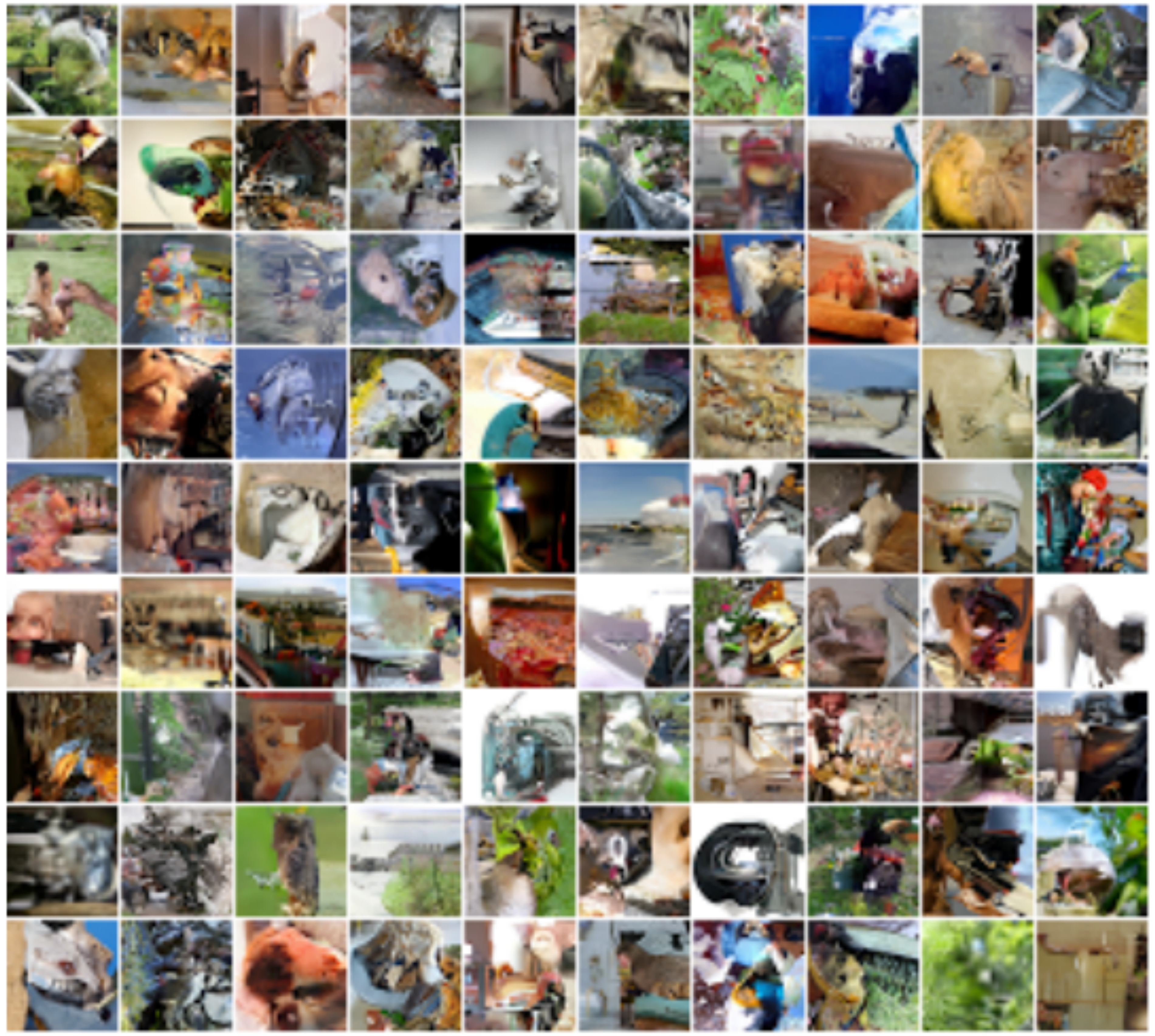
Generation is still slow (sequential), but learning is faster

Generated Samples

[van der Oord et al., 2016]



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

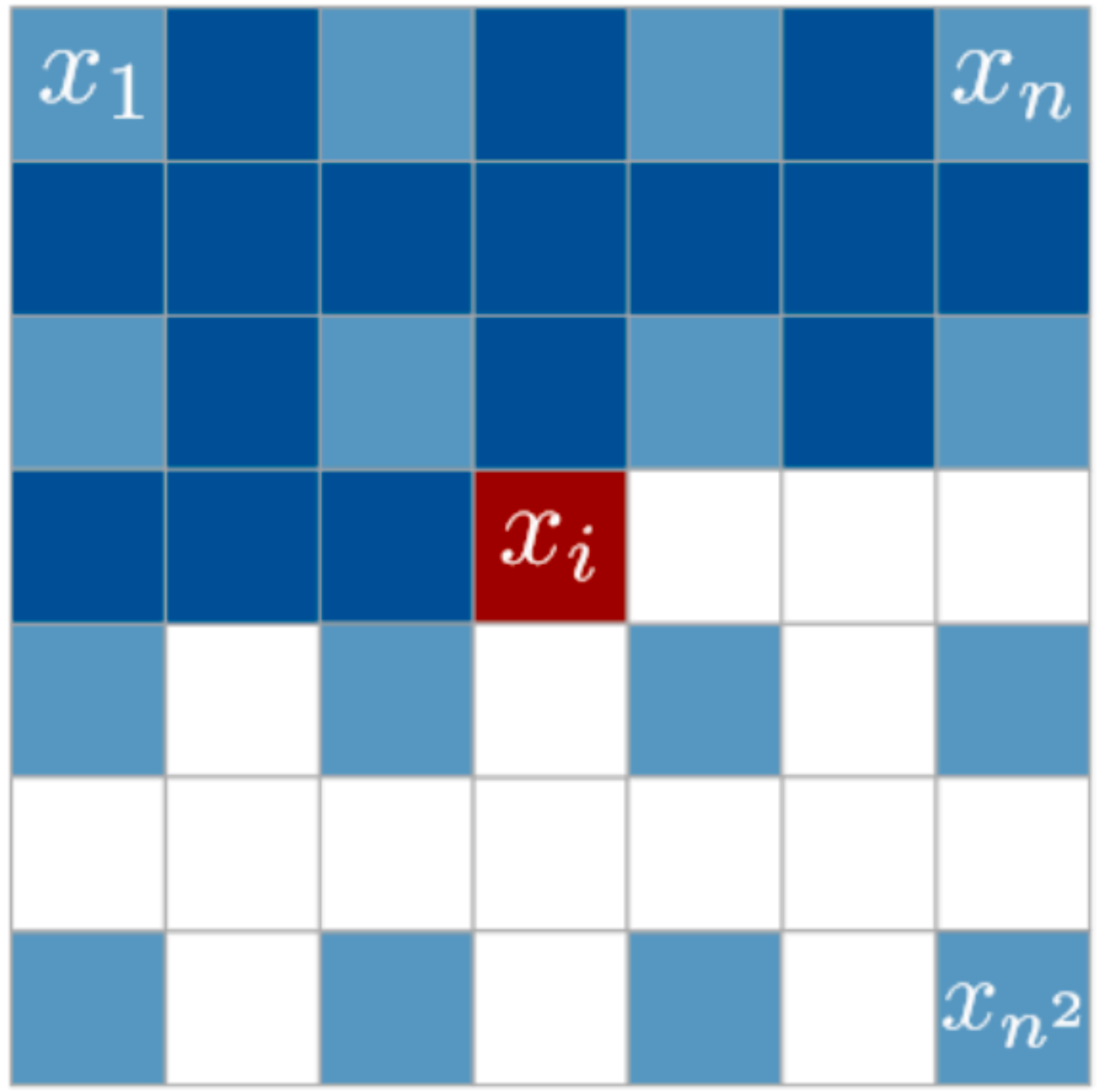
— Etc…

Multi-scale PixelRNN

[van der Oord et al., 2016]

Take sub-sampled pixels as additional input pixels

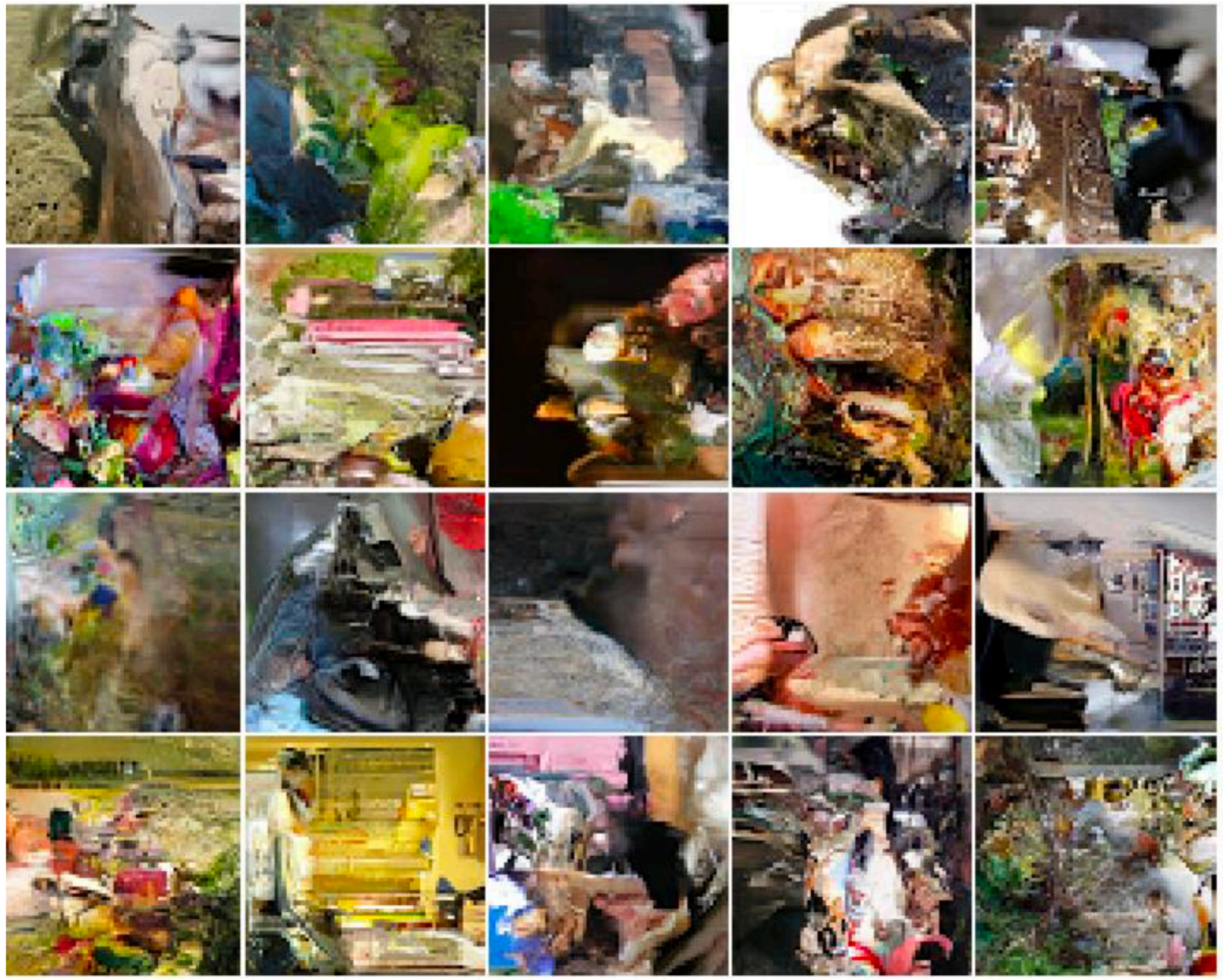
Can capture better global information (more visually coherent)

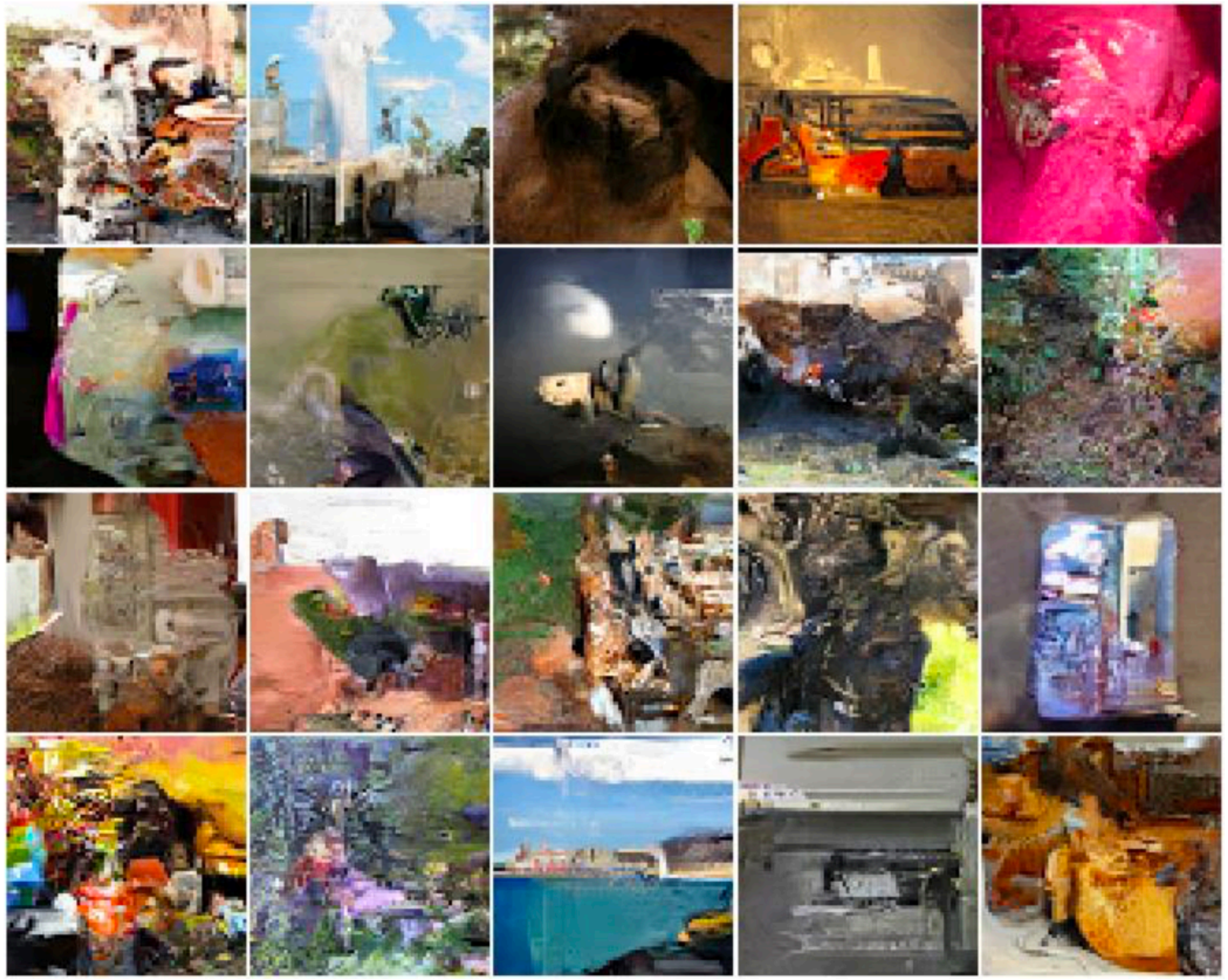


* slide from Hsiao-Ching Chang, Ameya Patil, Anand Bhattad

Multi-scale PixelRNN

[van der Oord et al., 2016]





* slide from Hsiao-Ching Chang, Ameya Patil, Anand Bhattad

Conditional Image Generation

[van der Oord et al., 2016]

Similar to PixelRNN/CNN but conditioned on a high-level image description vector $\mathbf{h}$

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

$$\downarrow$$

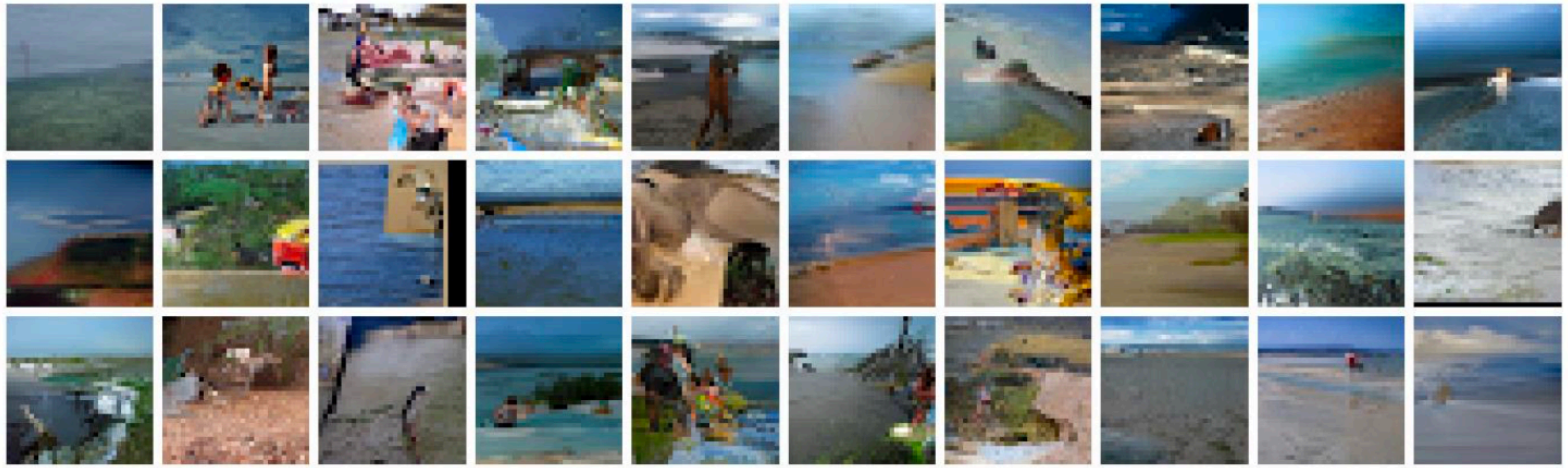
$$p(\mathbf{x}|\mathbf{h}) = p(x_1, x_2, ..., x_{n^2}|\mathbf{h})$$

* slide from Hsiao-Ching Chang, Ameya Patil, Anand Bhattad

Conditional Image Generation

[van der Oord et al., 2016]

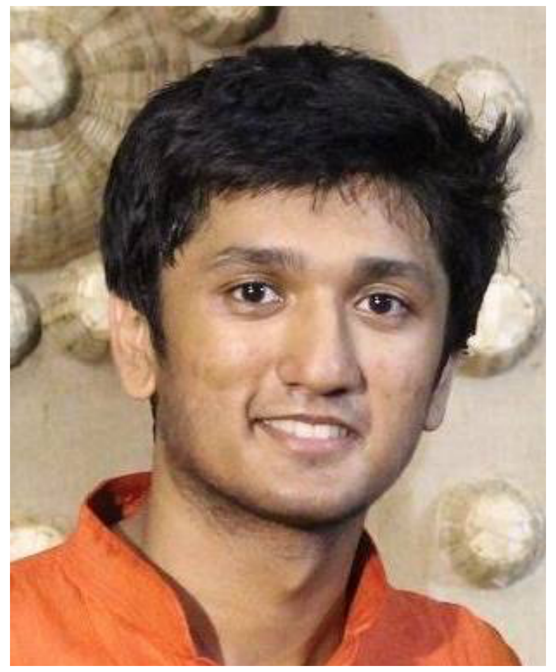
African elephant



Sandbar

* slide from Hsiao-Ching Chang, Ameya Patil, Anand Bhattad

Attention RNN: Structured Spatial Attention Mechanism



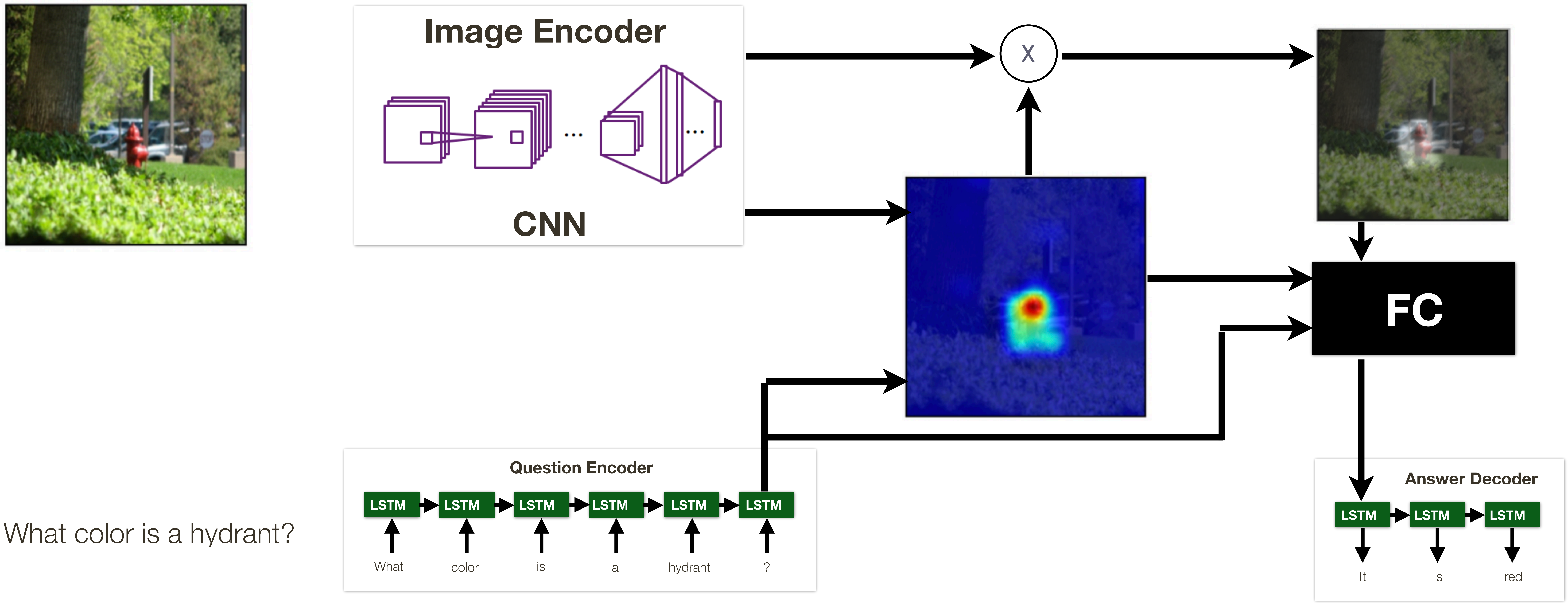

Siddhesh Khandelwal

Leonid Sigal

UBC

Motivation

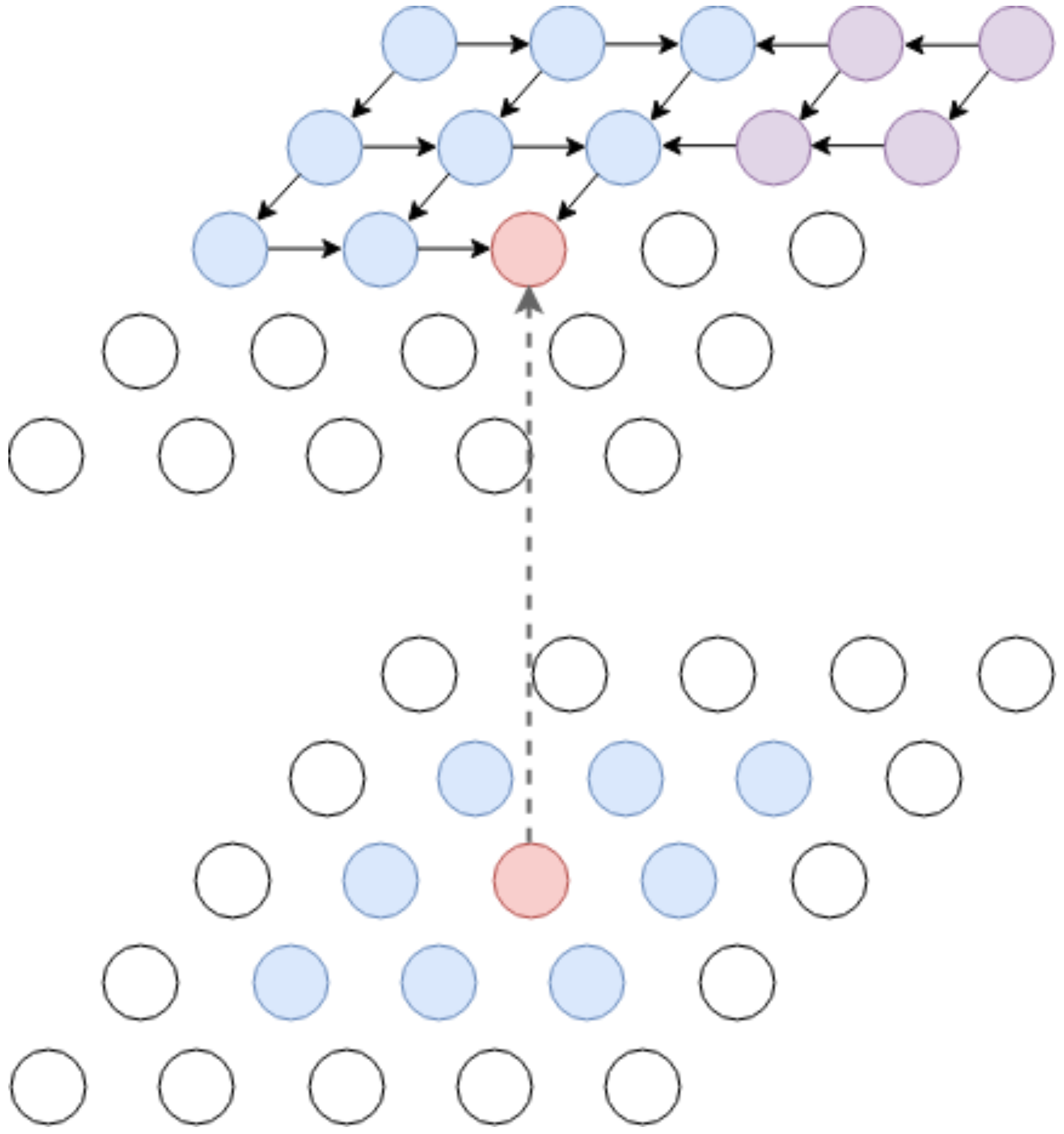
Attention is widely used in vision: helps identify relevant regions of the image



Q: What color is a hydrant?

A: It is red

AttentionRNN: Structured Spatial Attention



Novel **autoregressive attention mechanism** that can encode structural dependencies among attention values

— Inspired by diagonal Bi-LSTM architecture from PixelRNN

— Spatial attention values are generated sequentially

— Image is traversed diagonally from top-left to bottom-right

AttentionRNN: Structured Spatial Attention

Each **attention value** depends on

— Local image context

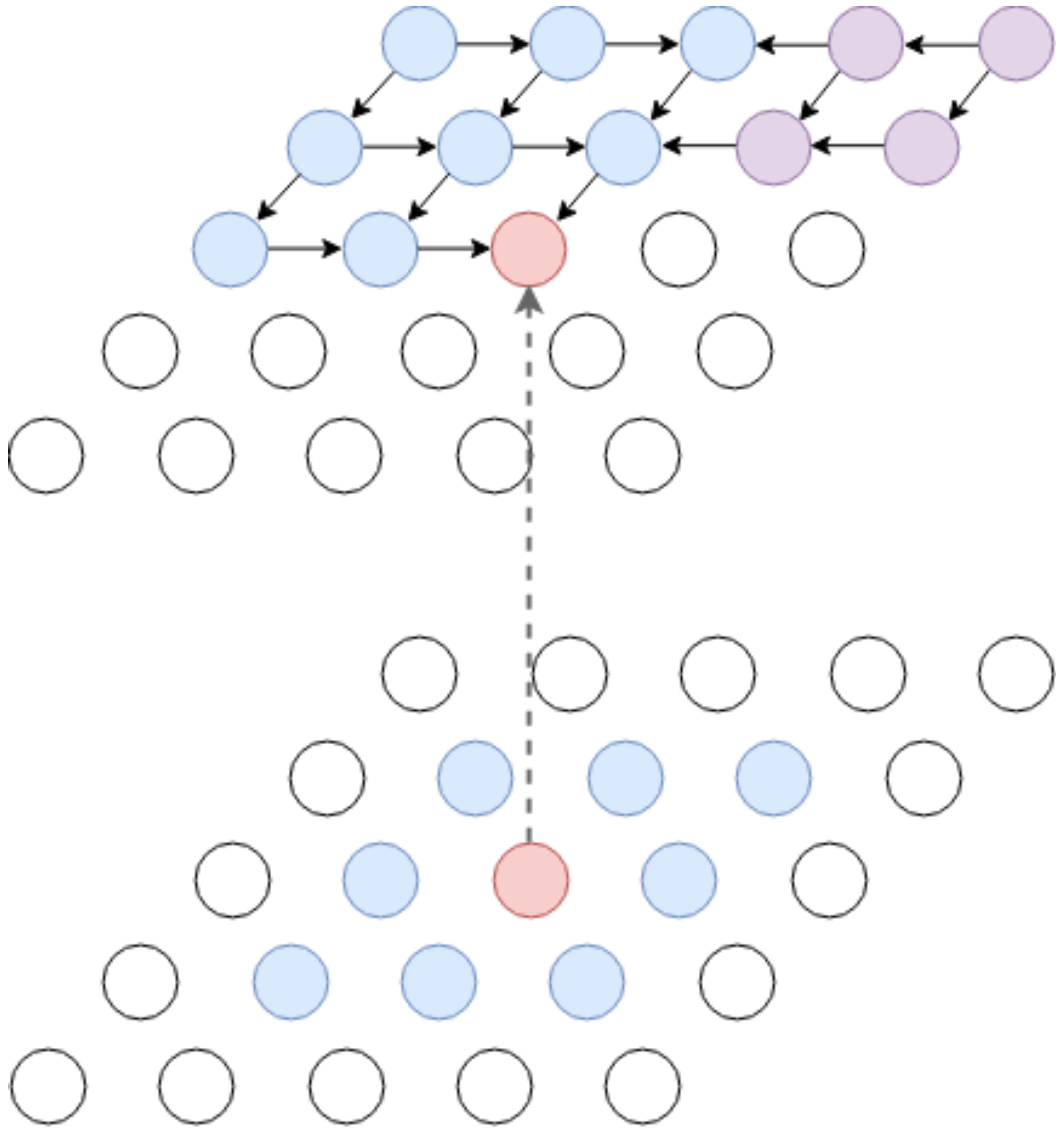
— Previously generated attention values

Novel **autoregressive attention mechanism** that can encode structural dependencies among attention values

— Inspired by diagonal Bi-LSTM architecture from PixelRNN

— Spatial attention values are generated sequentially

— Image is traversed diagonally from top-left to bottom-right

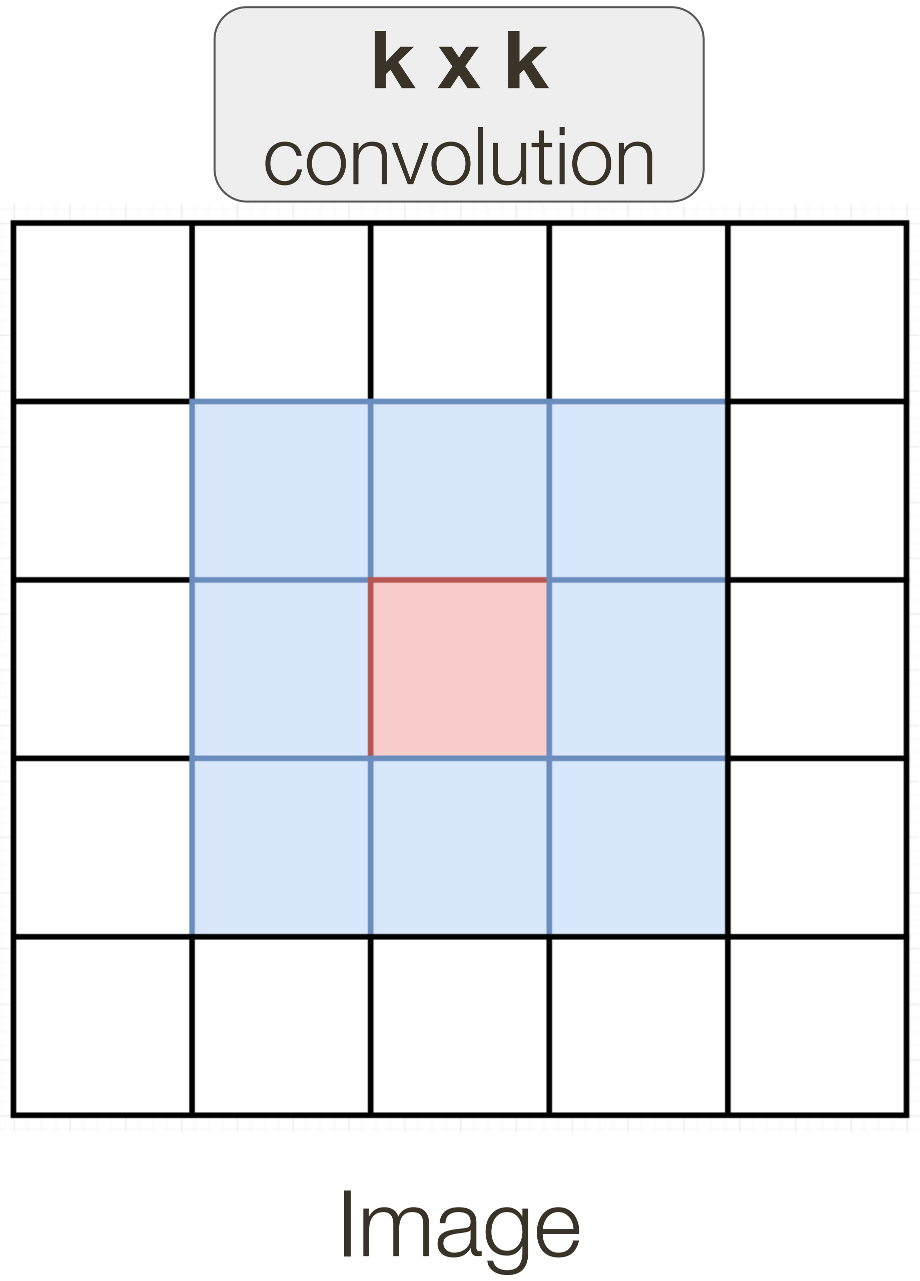


AttentionRNN: Structured Spatial Attention

Each **attention value** depends on

— **Local image context**

— Previously generated attention values

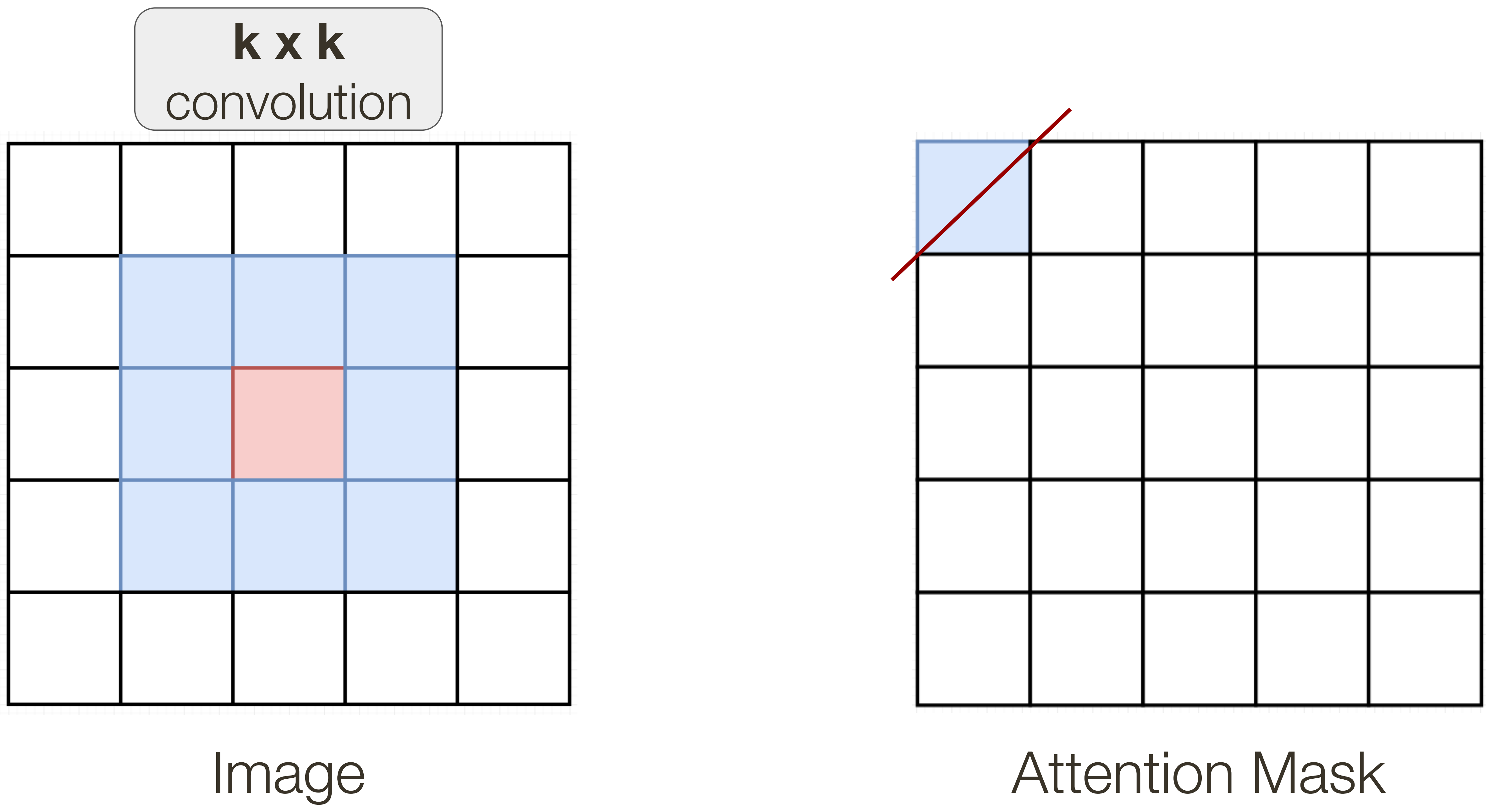


AttentionRNN: Structured Spatial Attention

Each **attention value** depends on

— Local image context

— **Previously generated attention values**

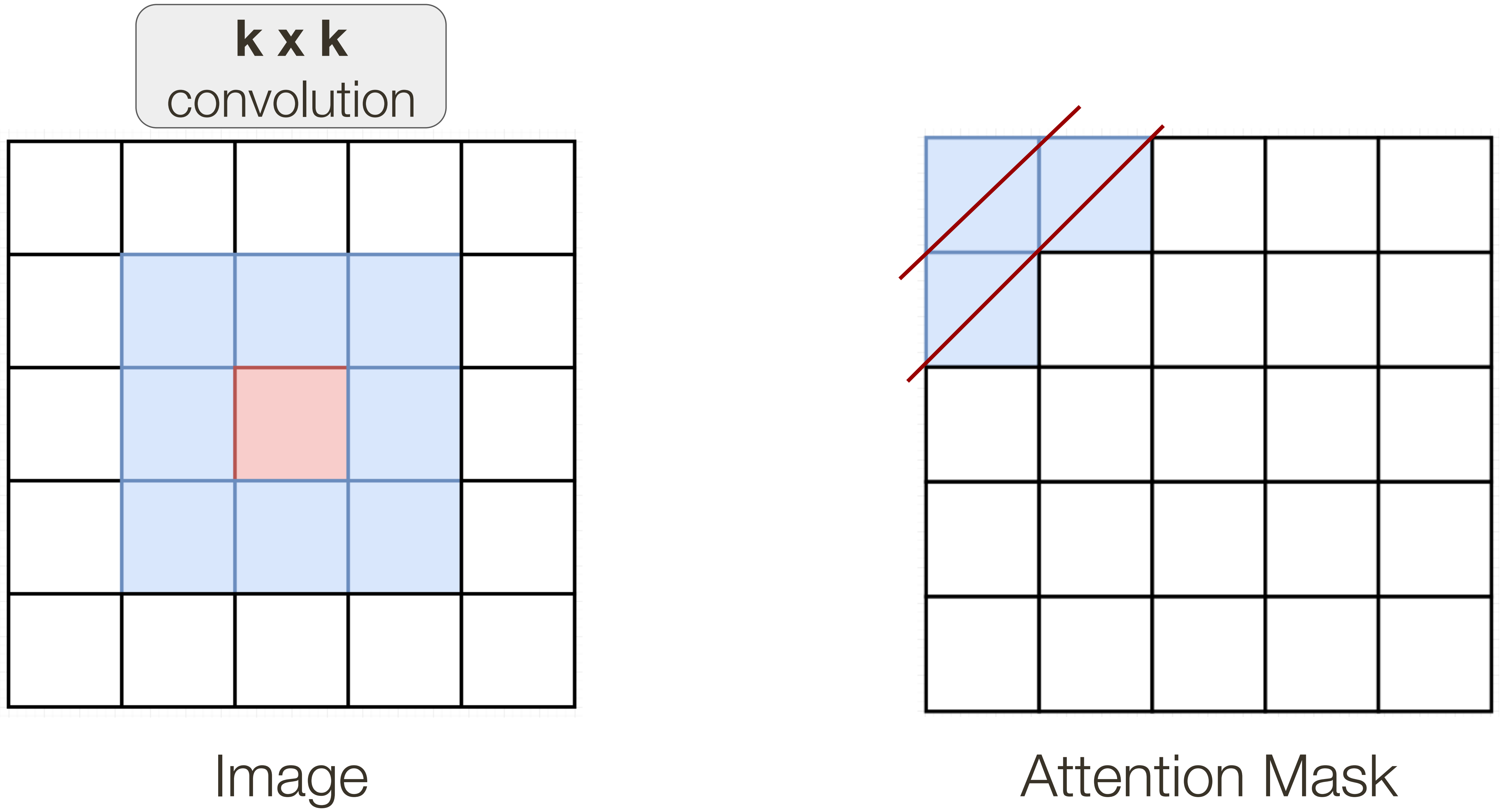


AttentionRNN: Structured Spatial Attention

Each **attention value** depends on

— Local image context

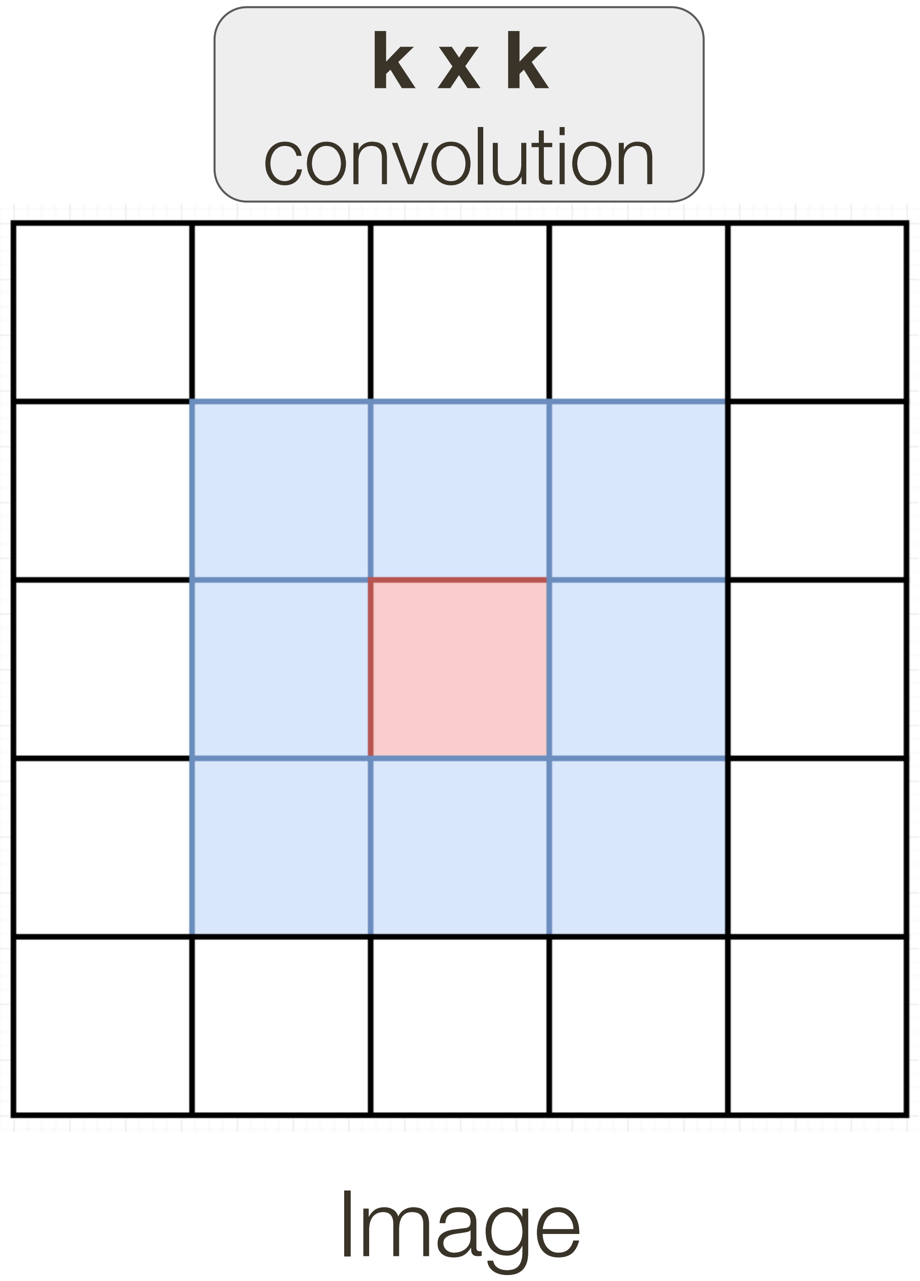
— **Previously generated attention values**

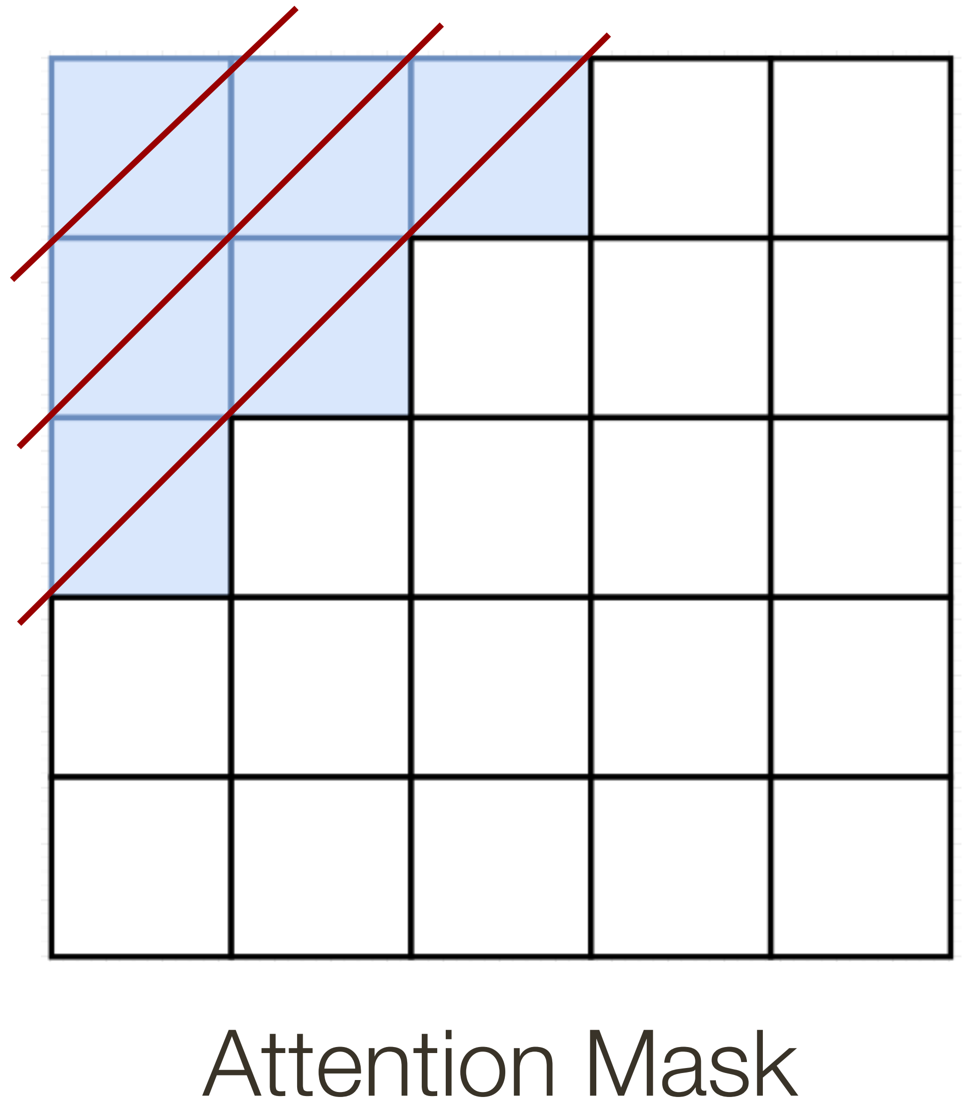


AttentionRNN: Structured Spatial Attention

Each **attention value** depends on

- Local image context
- **Previously generated attention values**



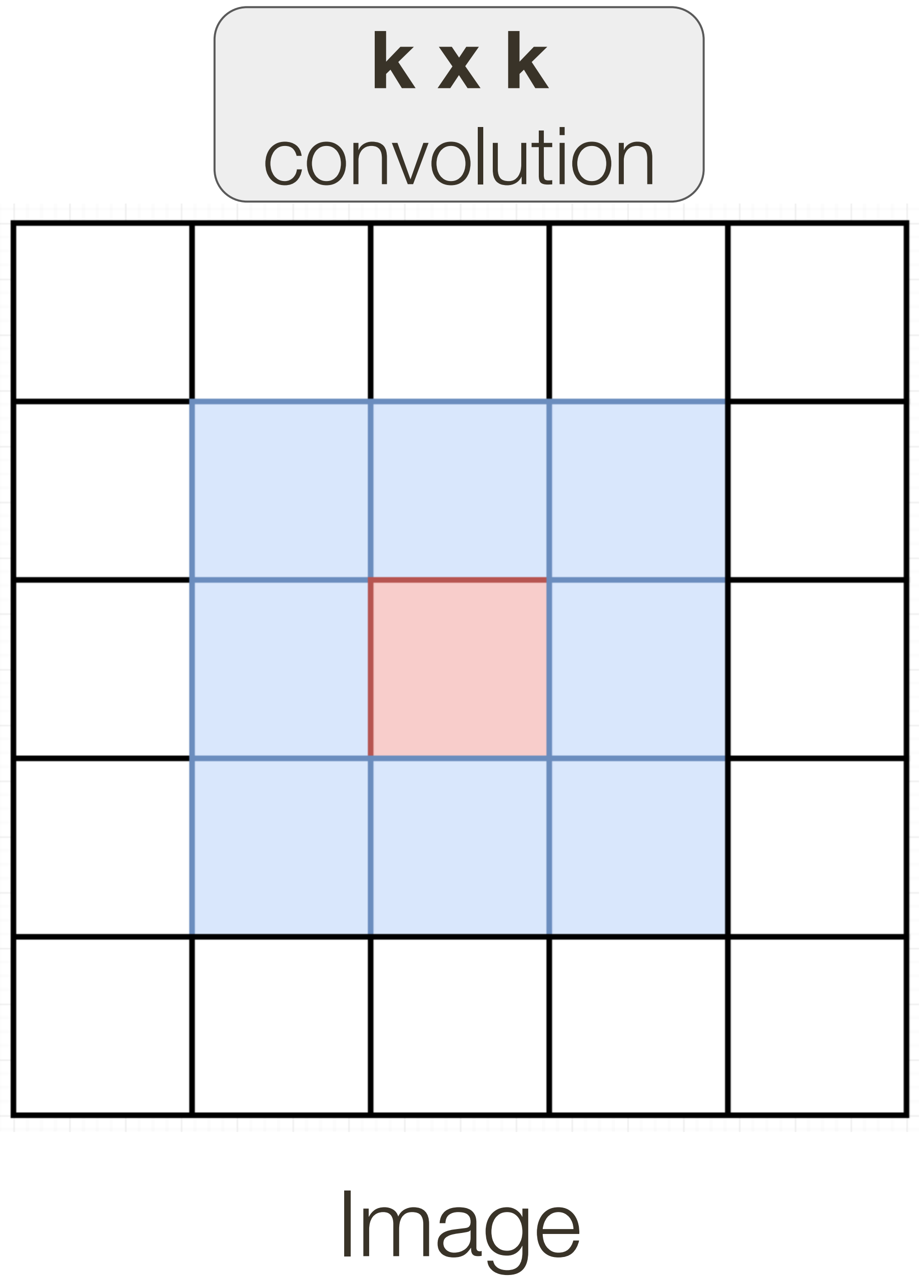


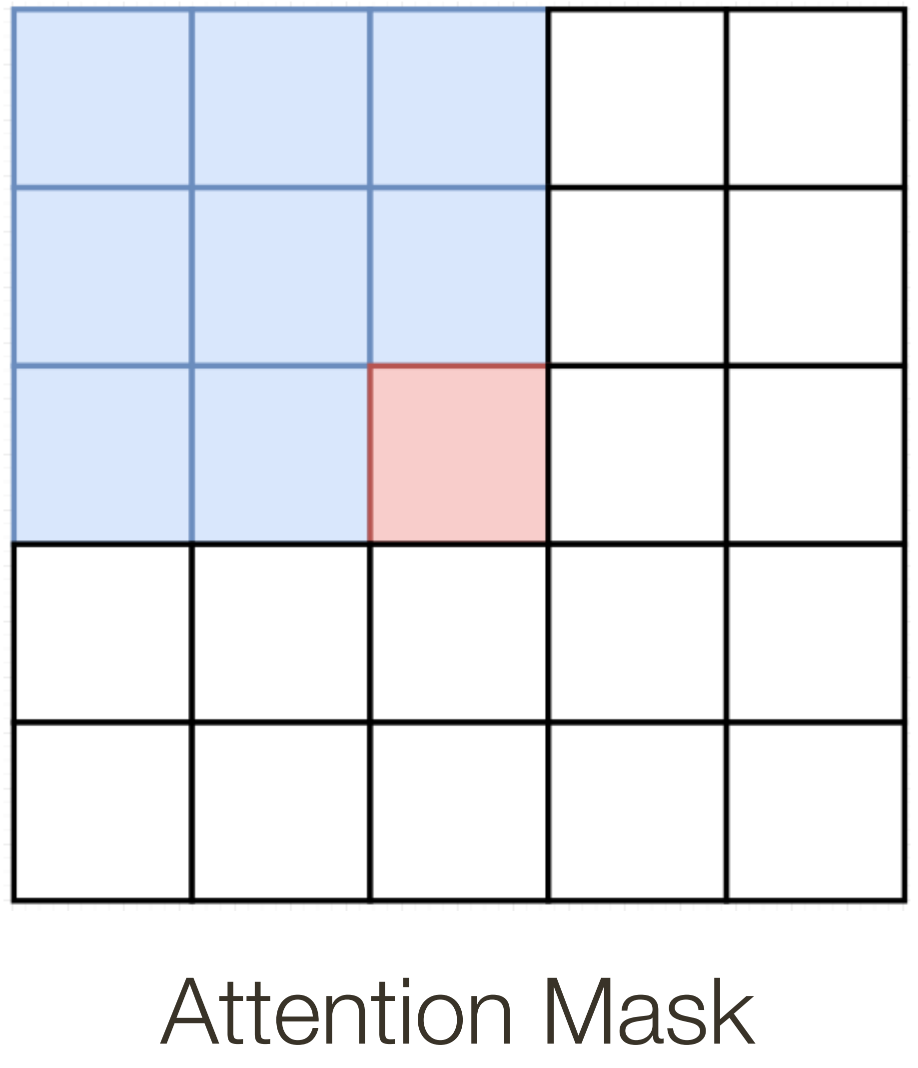
AttentionRNN: Structured Spatial Attention

Each **attention value** depends on

— Local image context

— **Previously generated attention values**



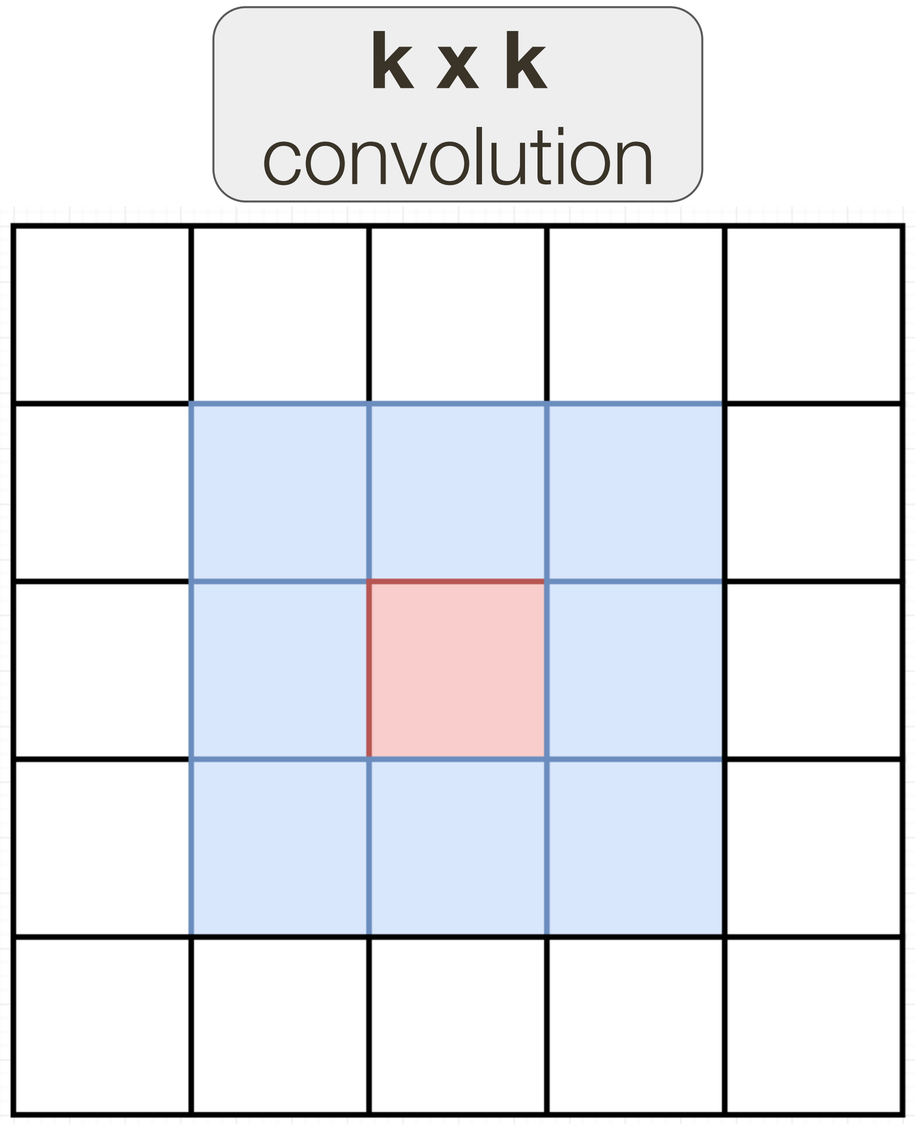


AttentionRNN: Structured Spatial Attention

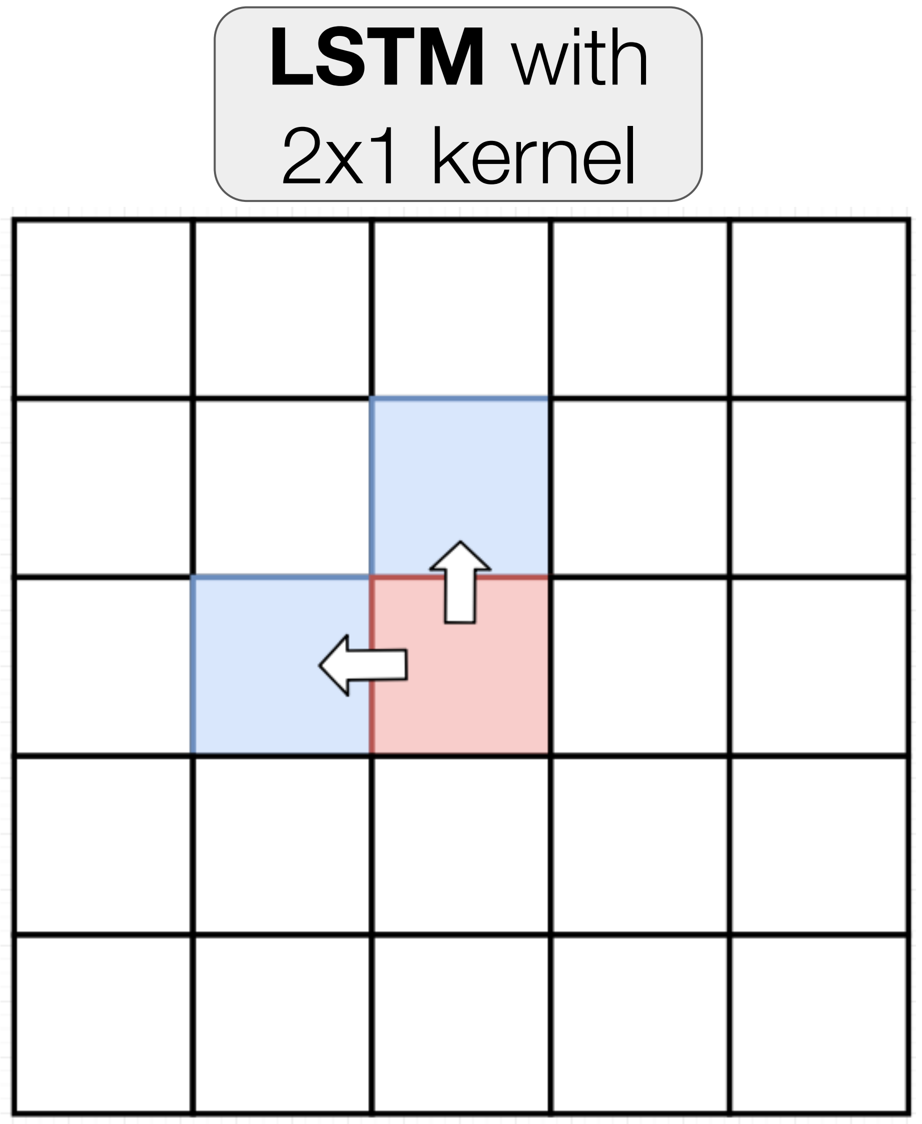
Each **attention value** depends on

— Local image context

— Previously generated attention values



Image



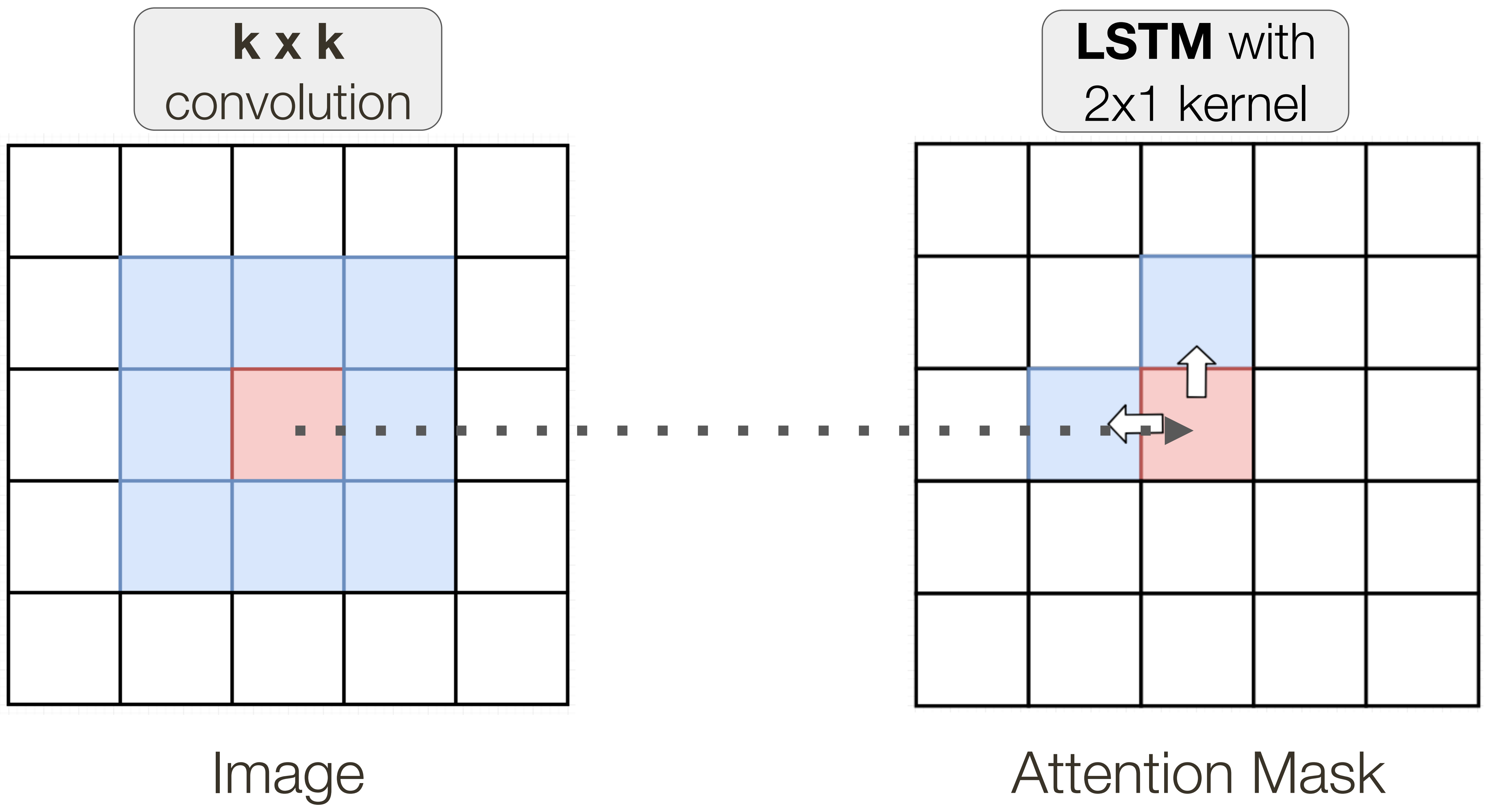
Attention Mask

AttentionRNN: Structured Spatial Attention

Each **attention value** depends on

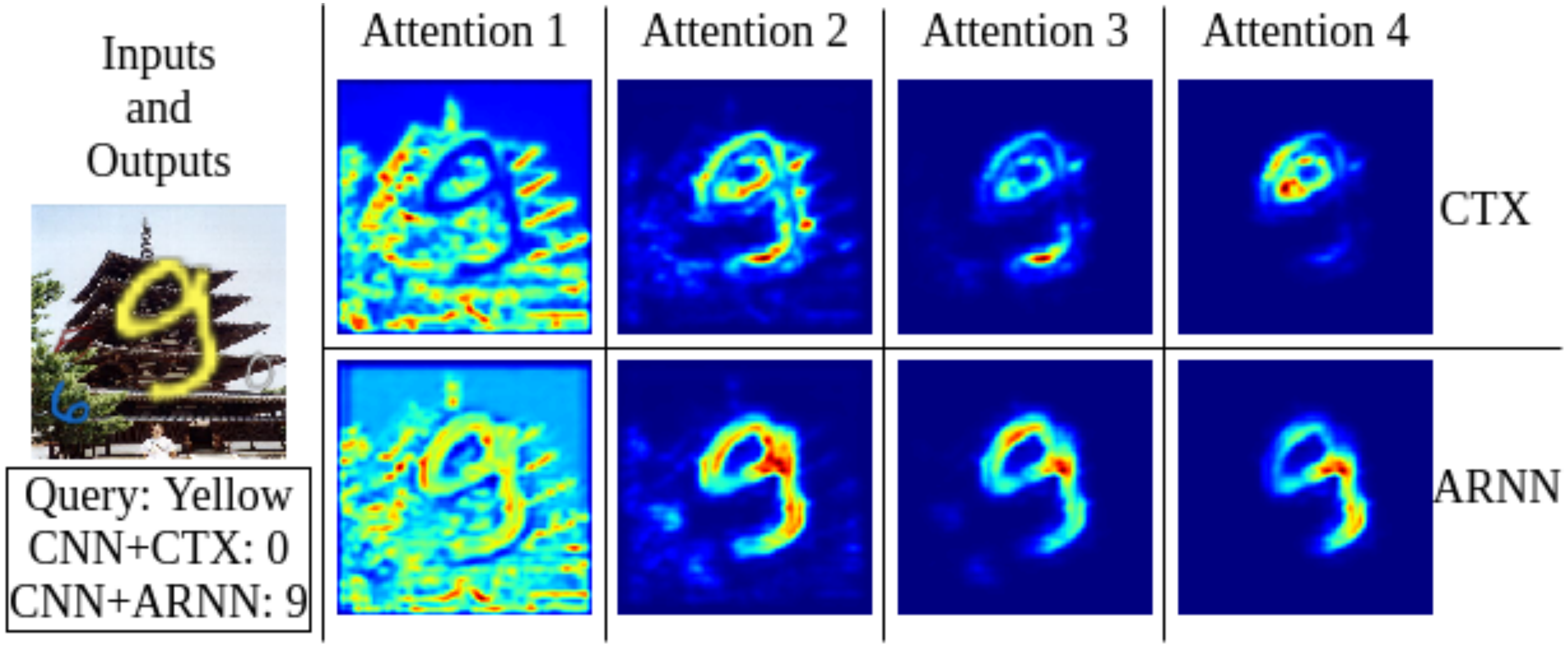
— Local image context

— Previously generated attention values



Experiments: Visual Digit Prediction

Task: Given an image, predict a digit number specified by a query color



	SAN	$\neg$ CTX	CTX	ARNN
Correctness	0.1258	0.2017	0.2835	**0.3729**

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):

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

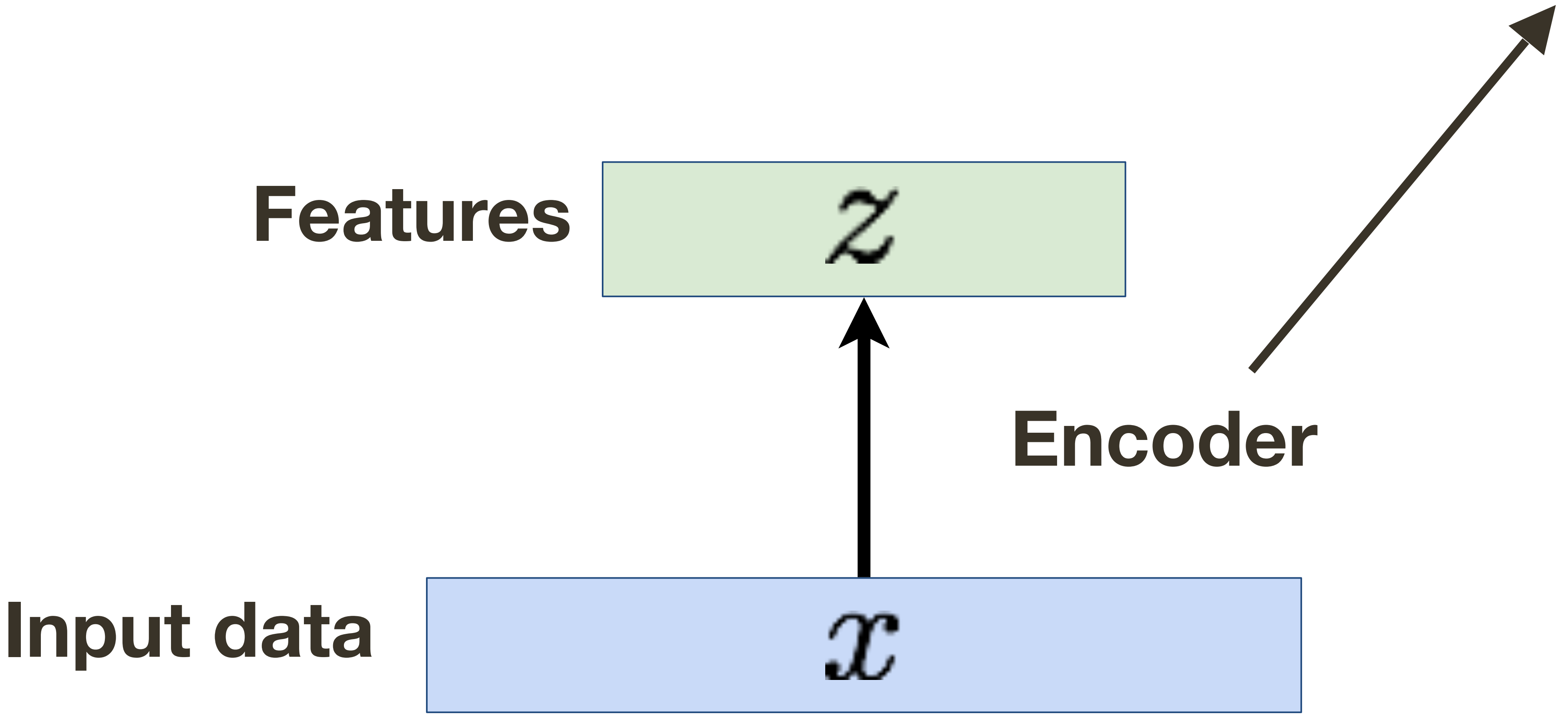
cannot optimize directly, derive and optimize lower bound of likelihood instead

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

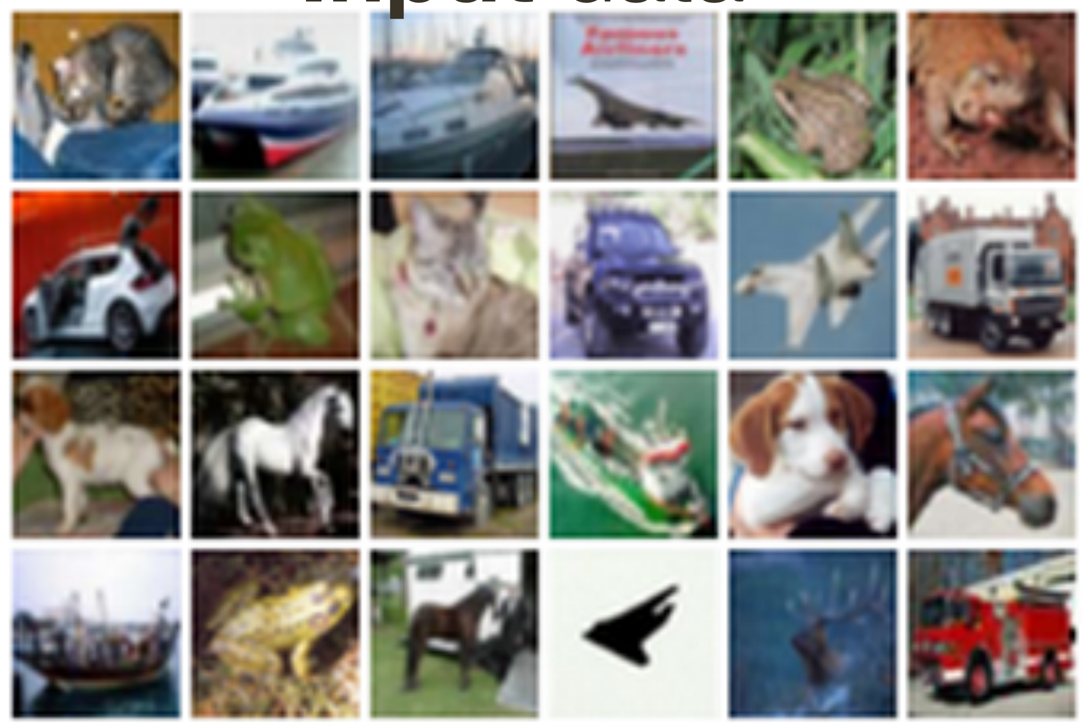
Autoencoders Reminder …

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



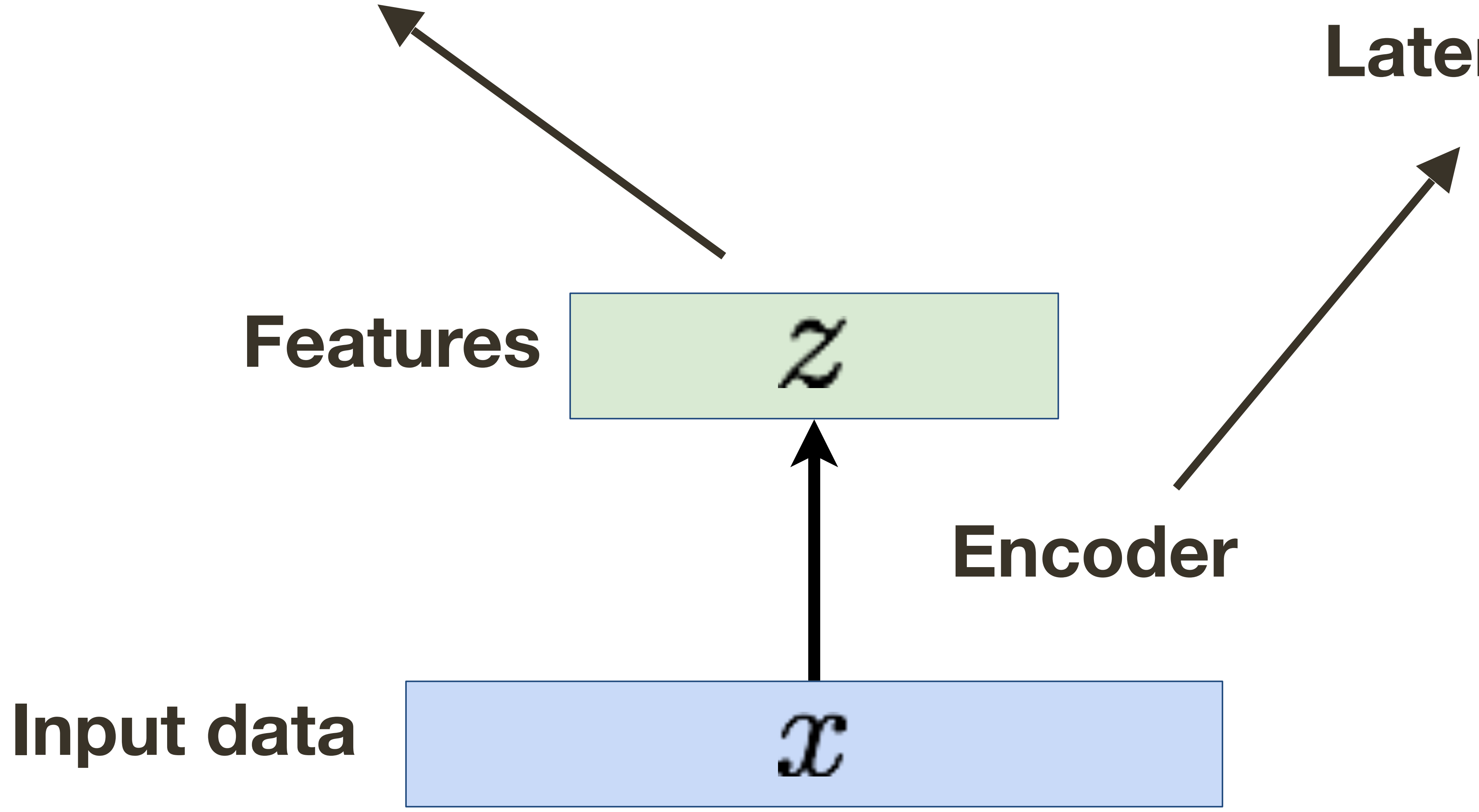
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Autoencoders Reminder …

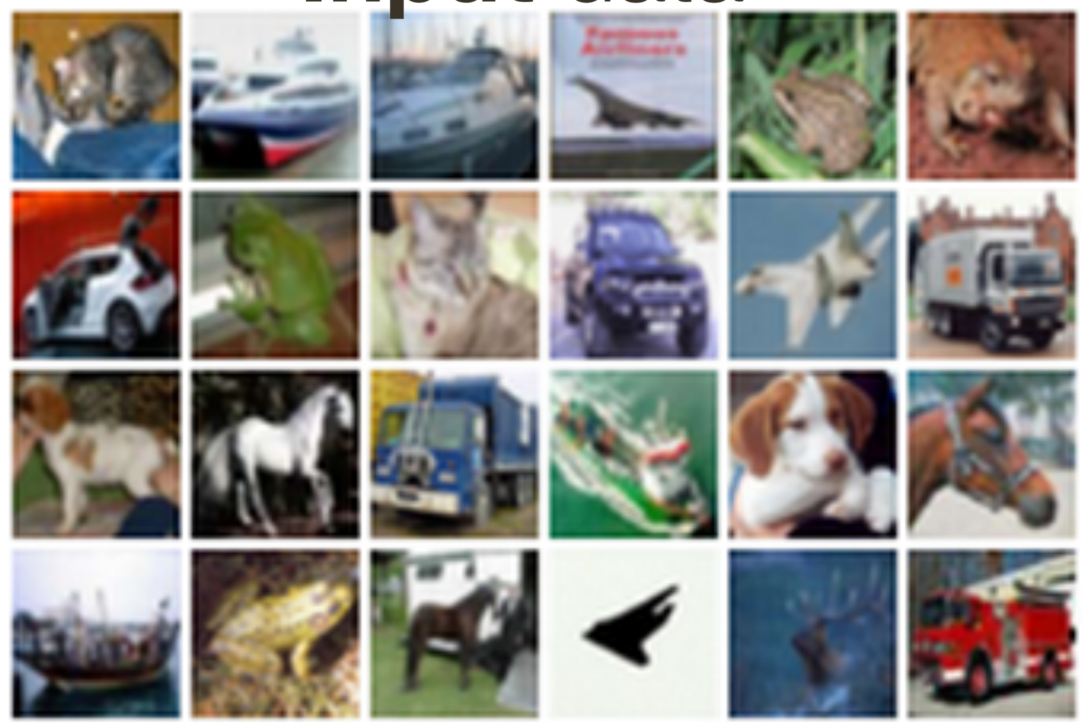
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

z usually smaller than x
(dimensionality reduction)

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



Input data



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

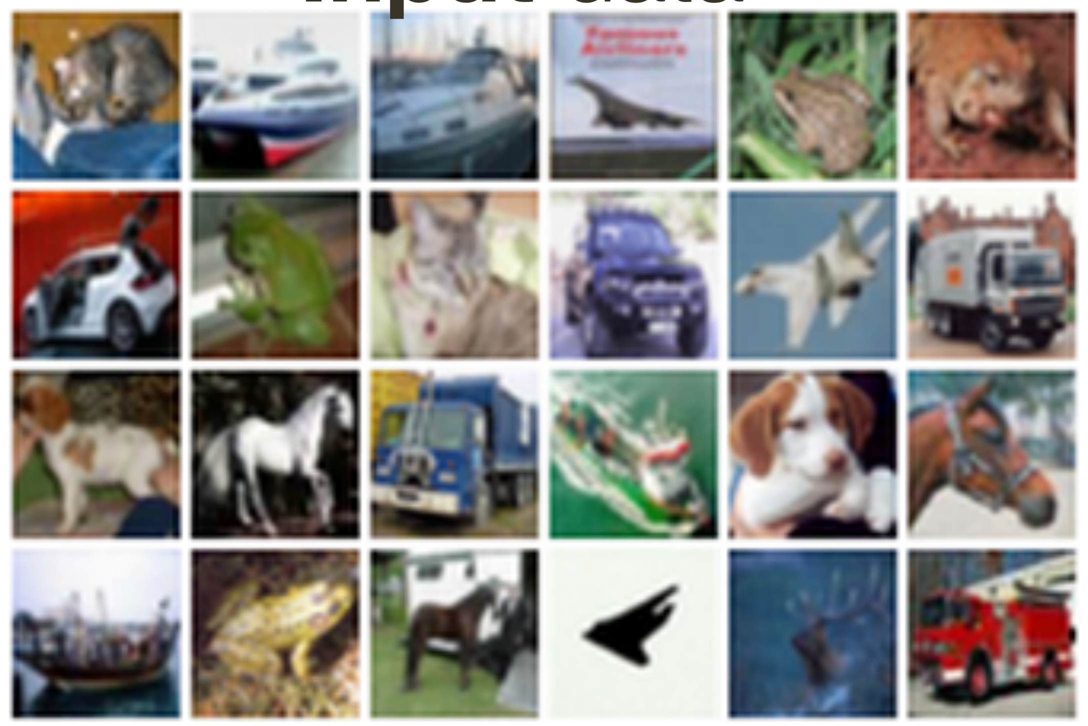
Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN

Features z

Encoder

Input data x

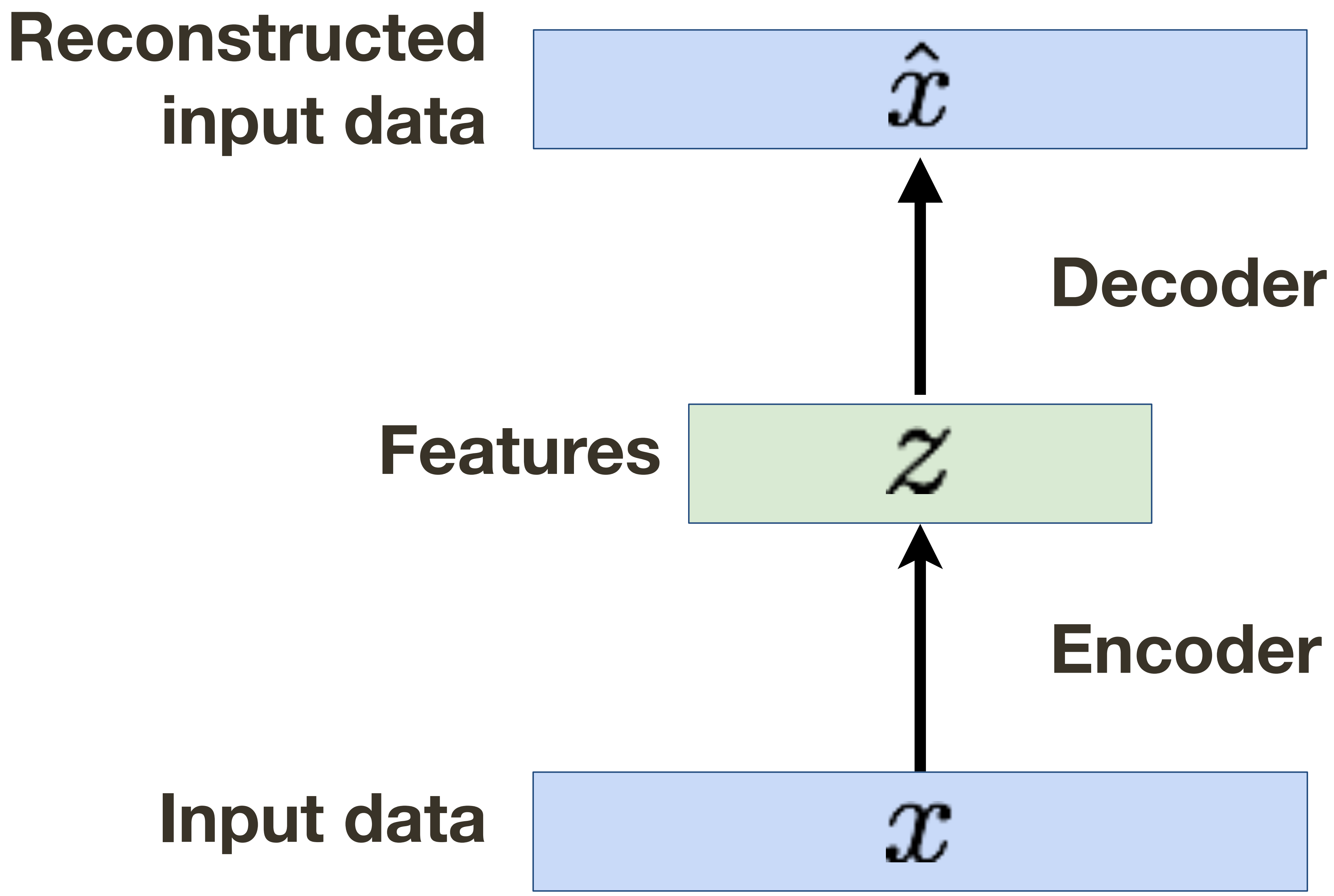
Input data

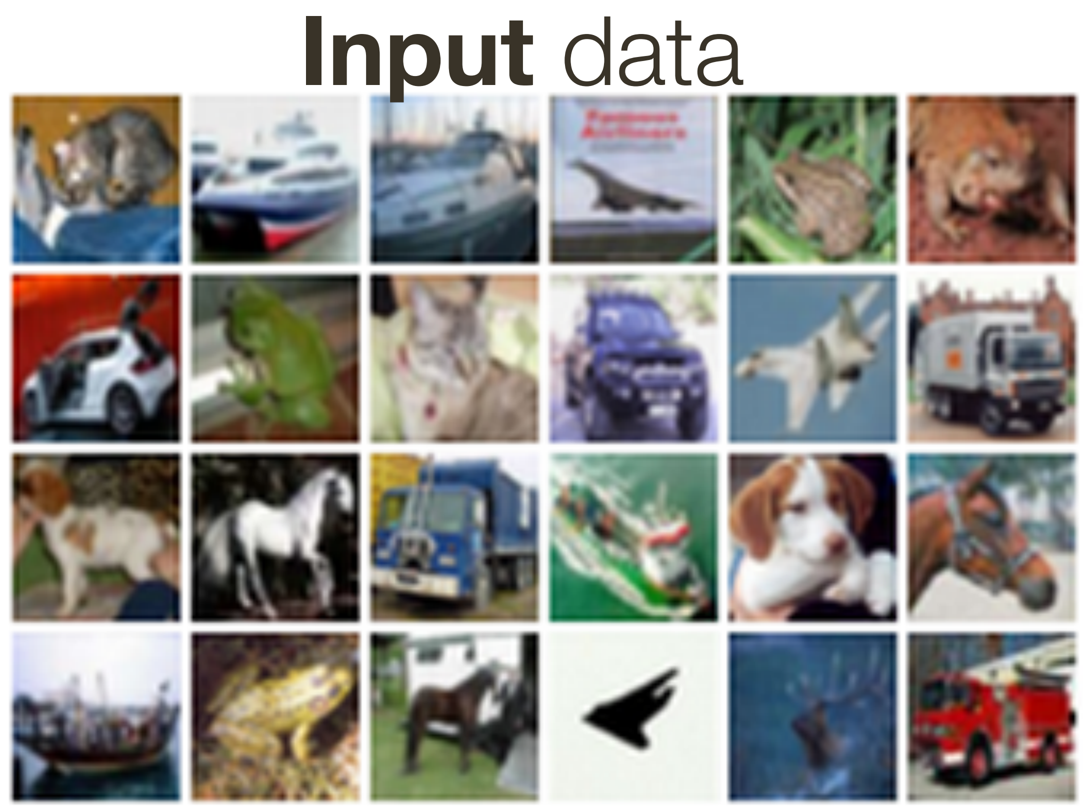


* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Autoencoders Reminder …

Train such that features can reconstruct original data best they can

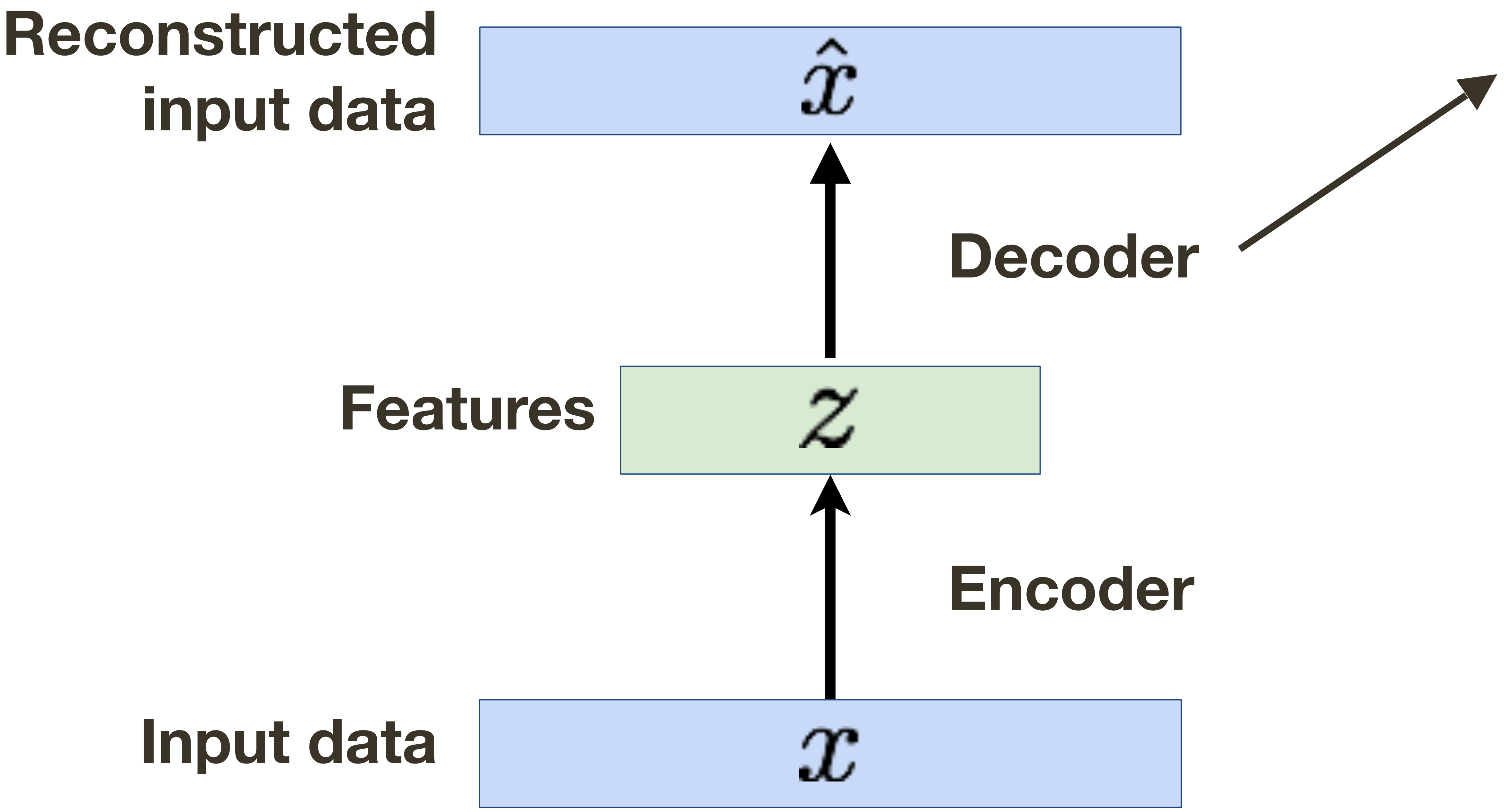




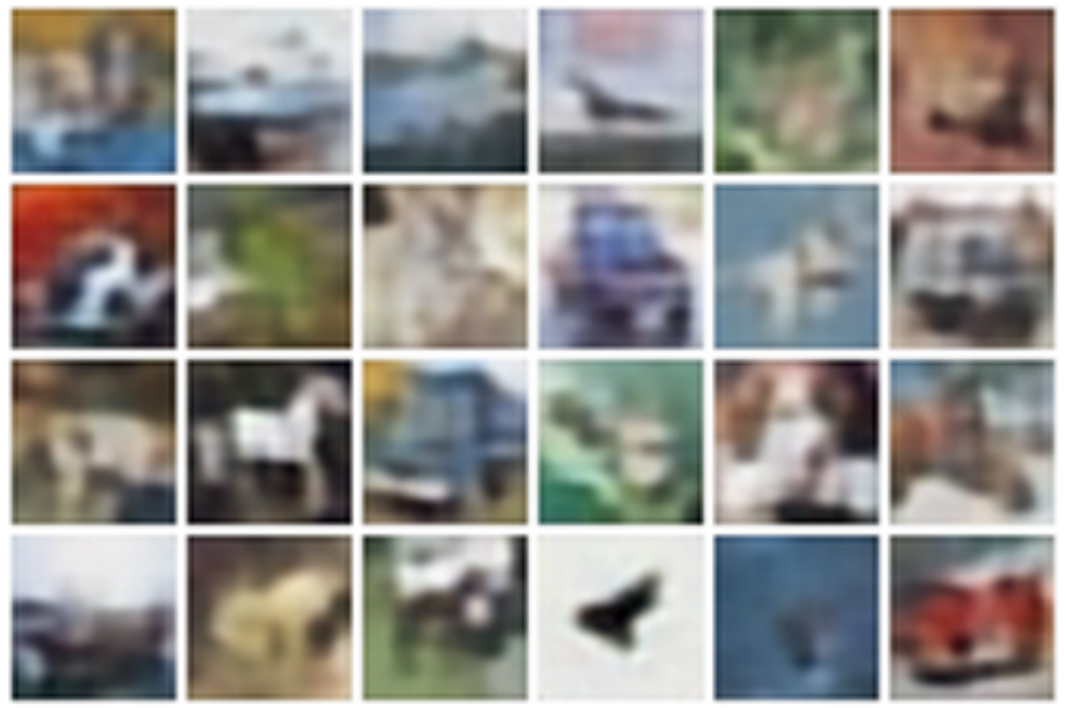
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Autoencoders Reminder …

Train such that features can reconstruct original data best they can

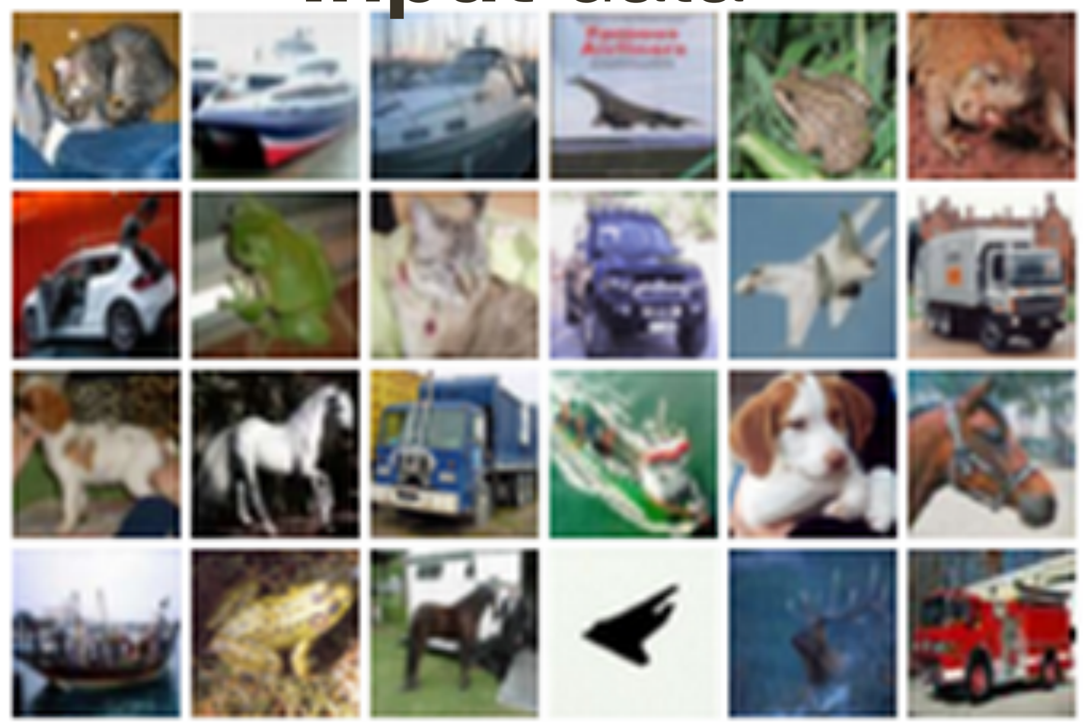


Reconstructed data

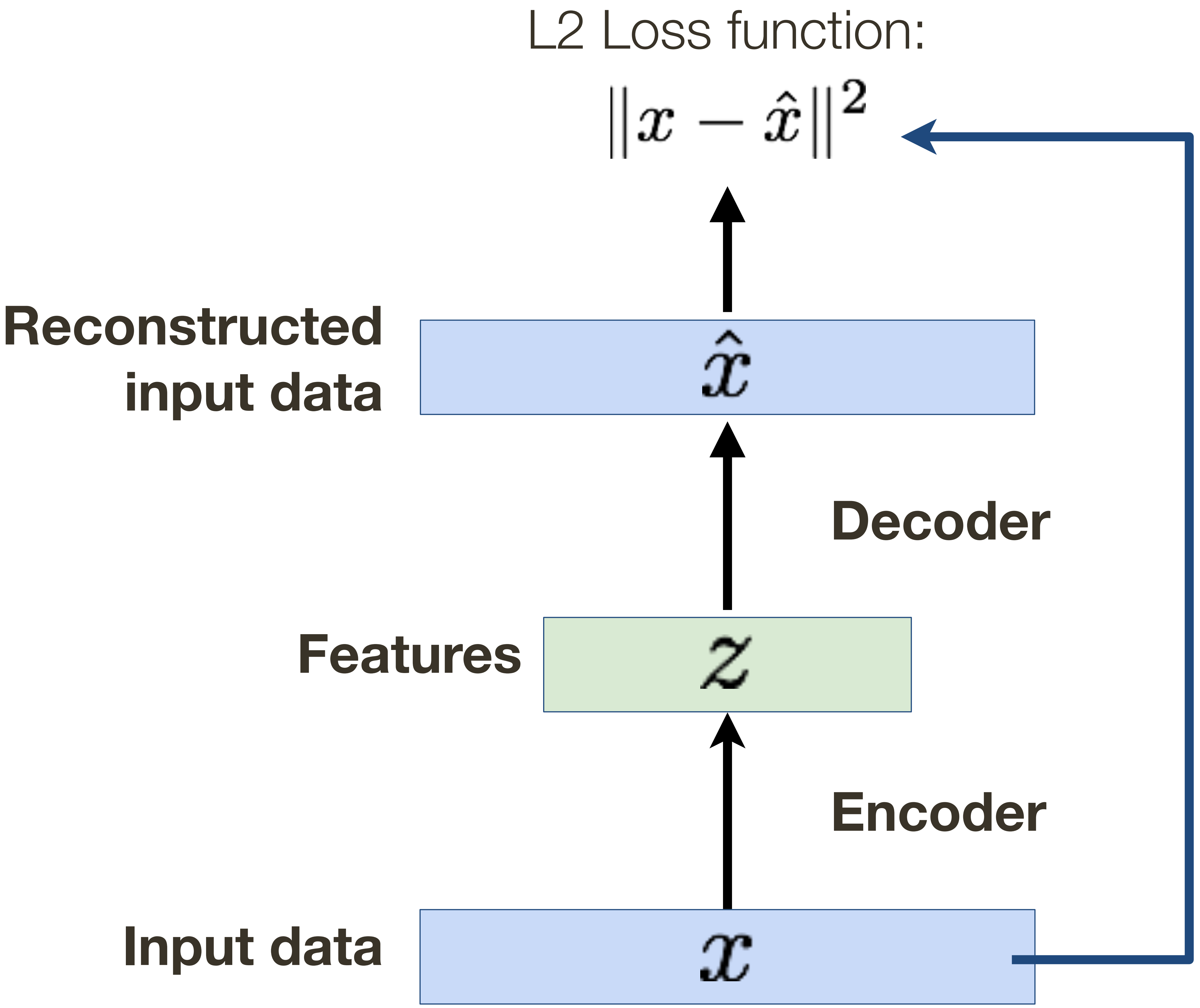


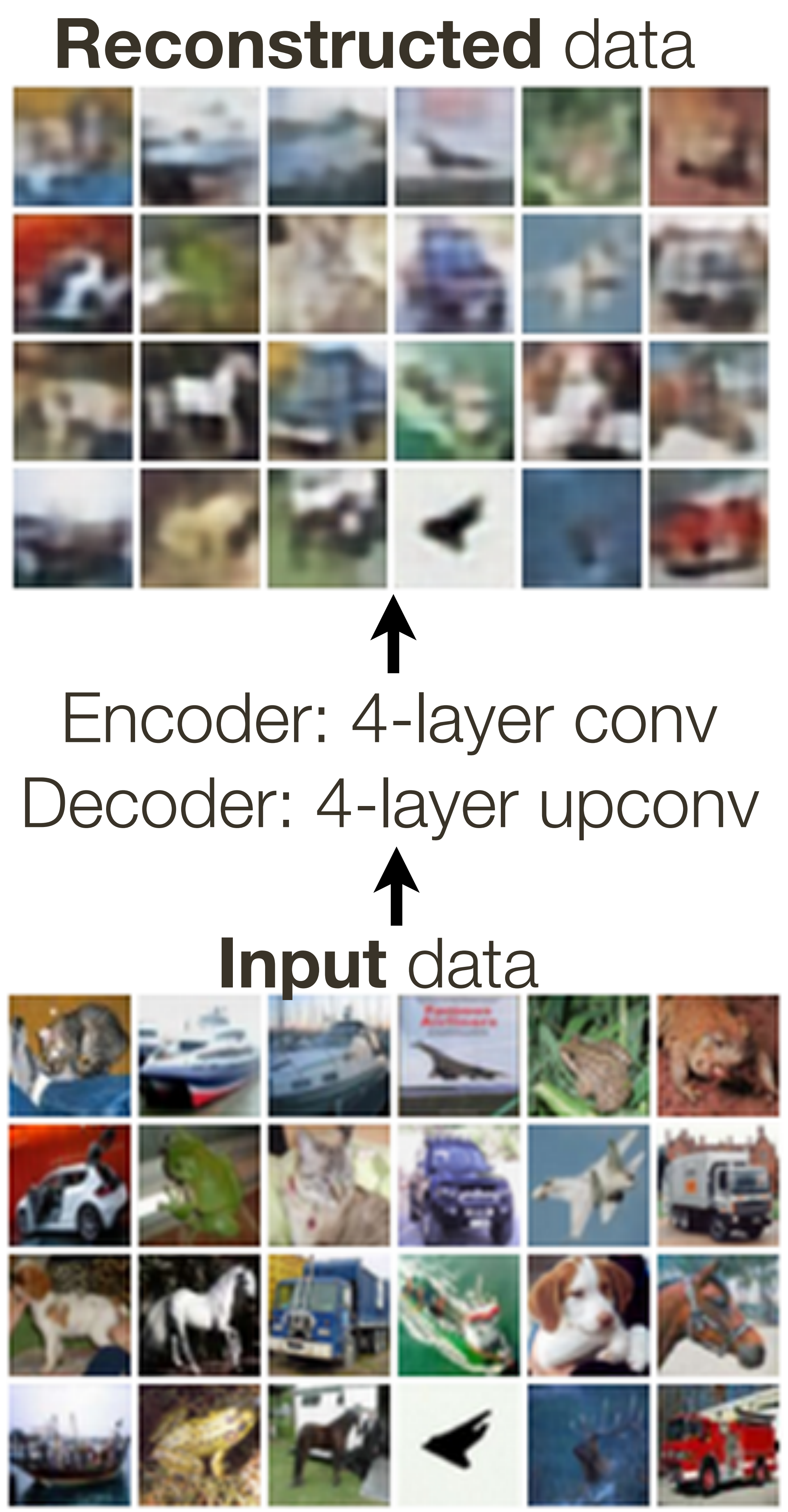
Encoder: 4-layer conv
Decoder: 4-layer upconv

Input data



Autoencoders Reminder …

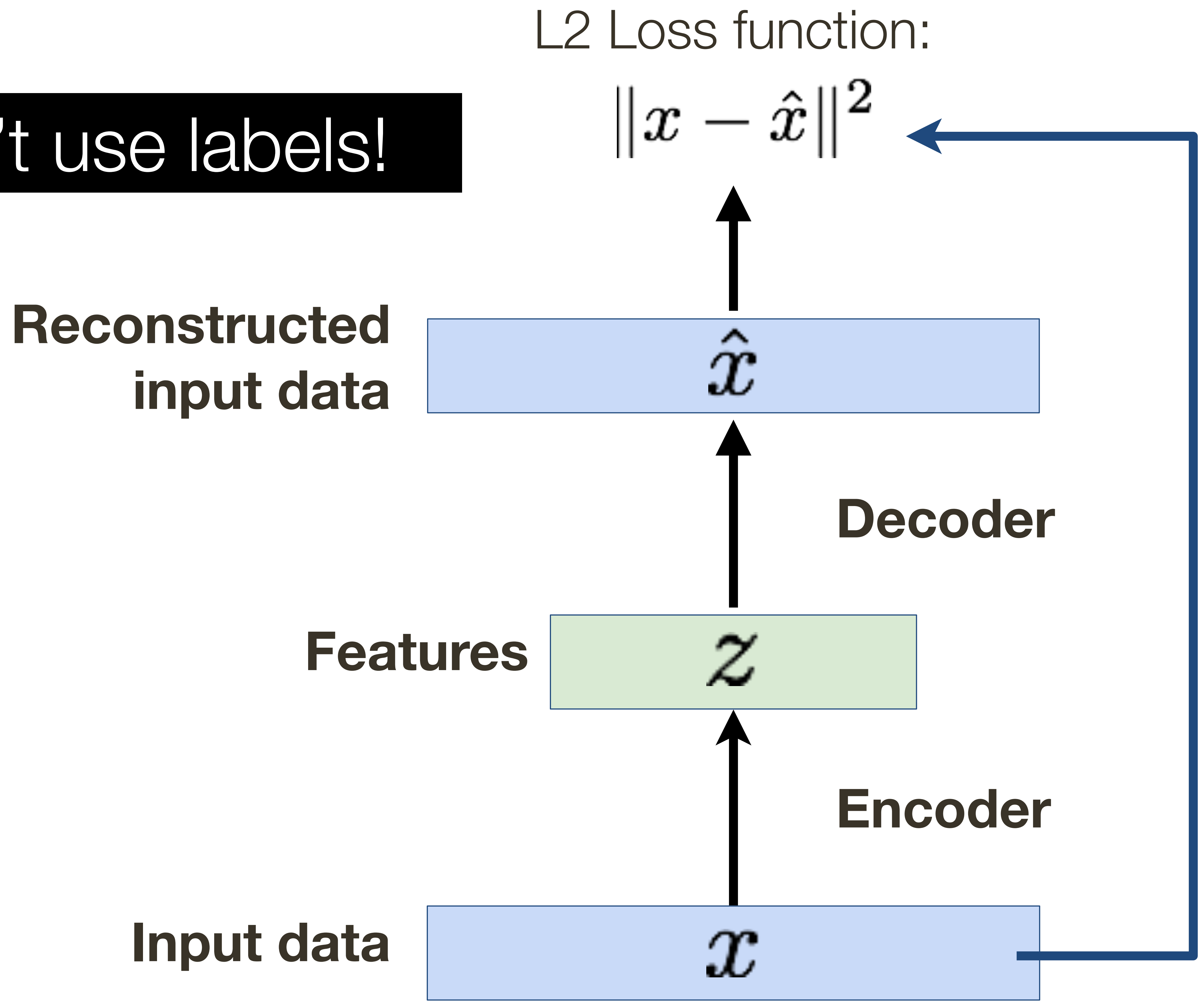


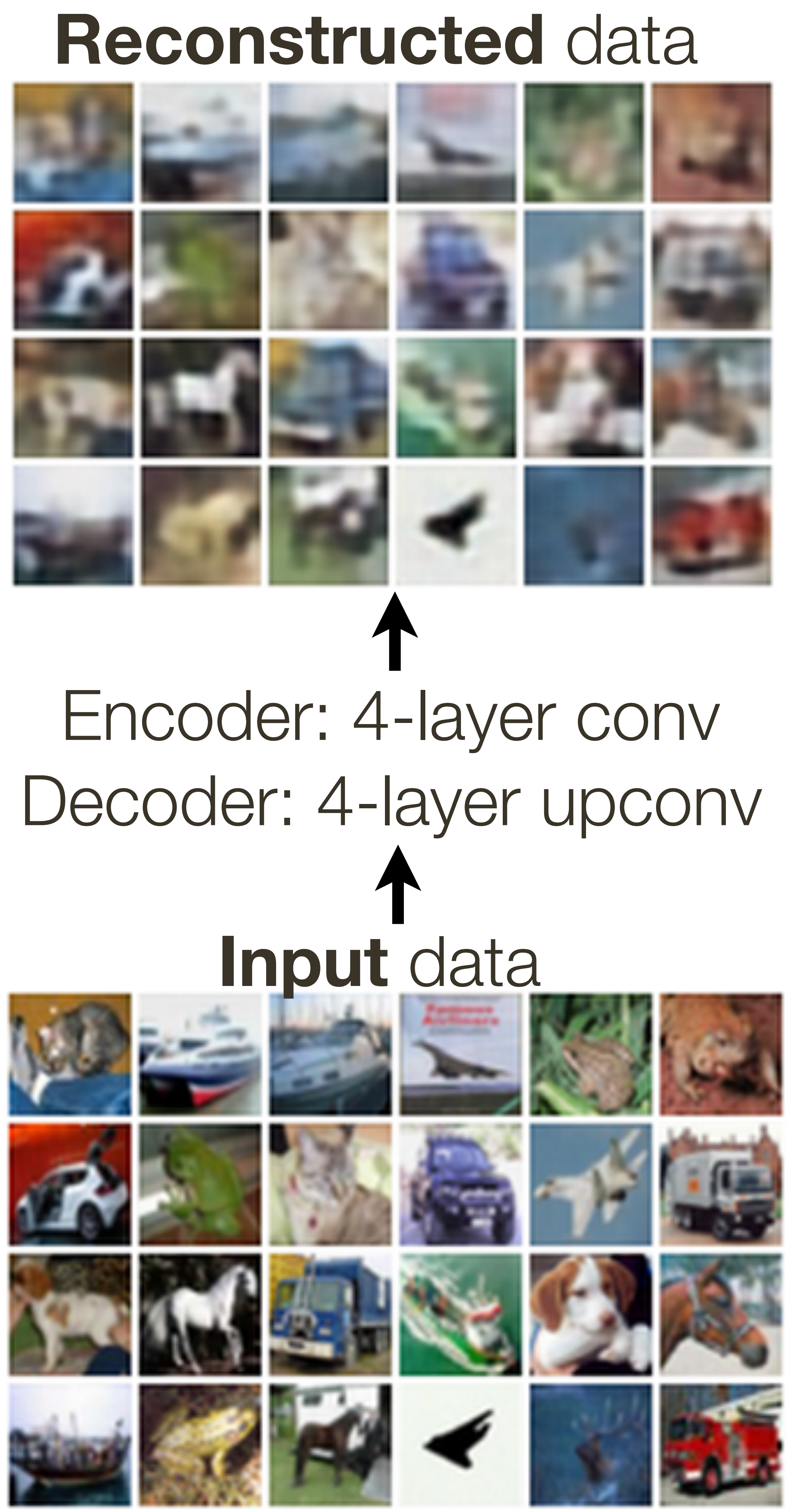


* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Autoencoders Reminder …

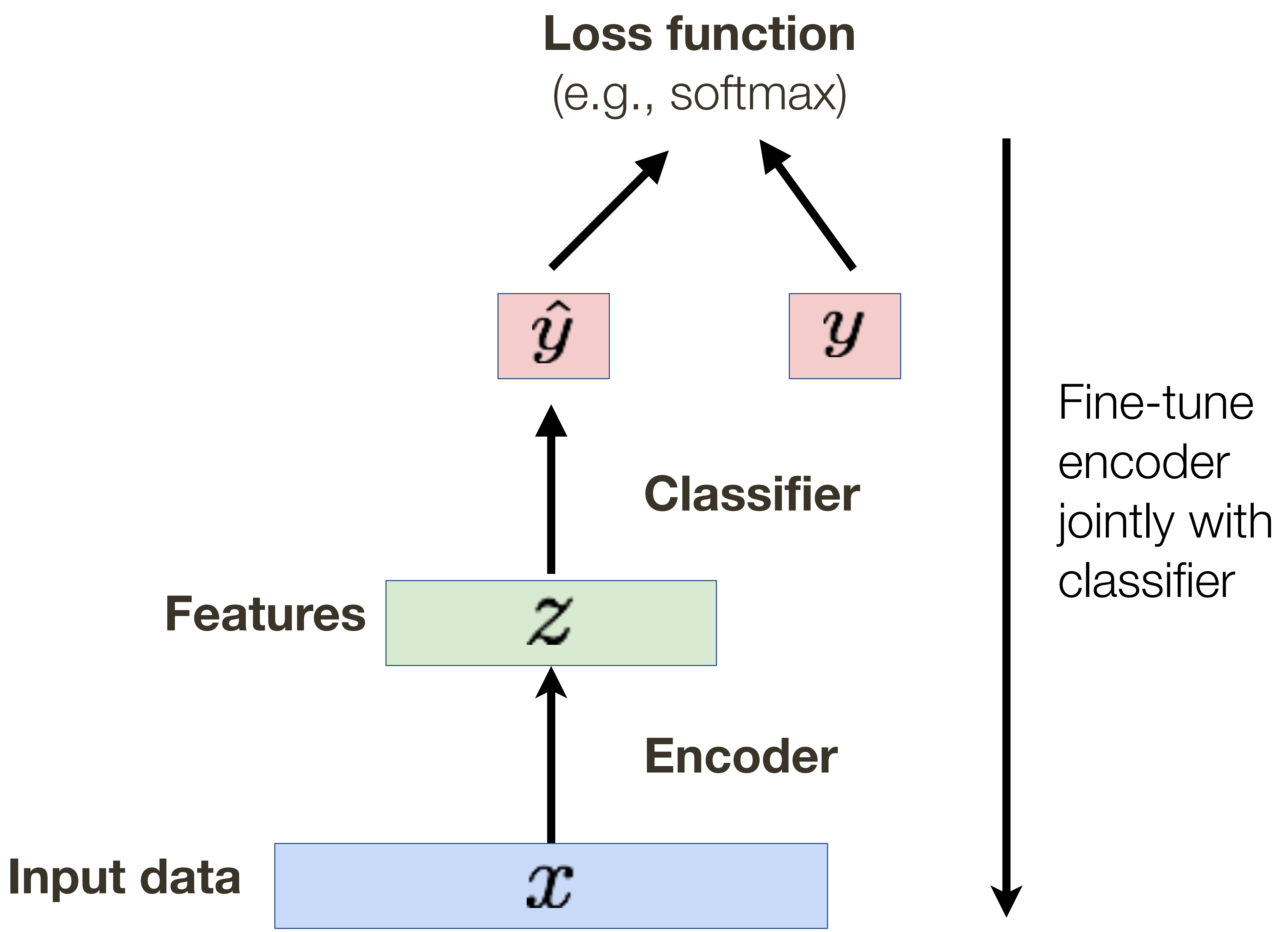
Doesn't use labels!



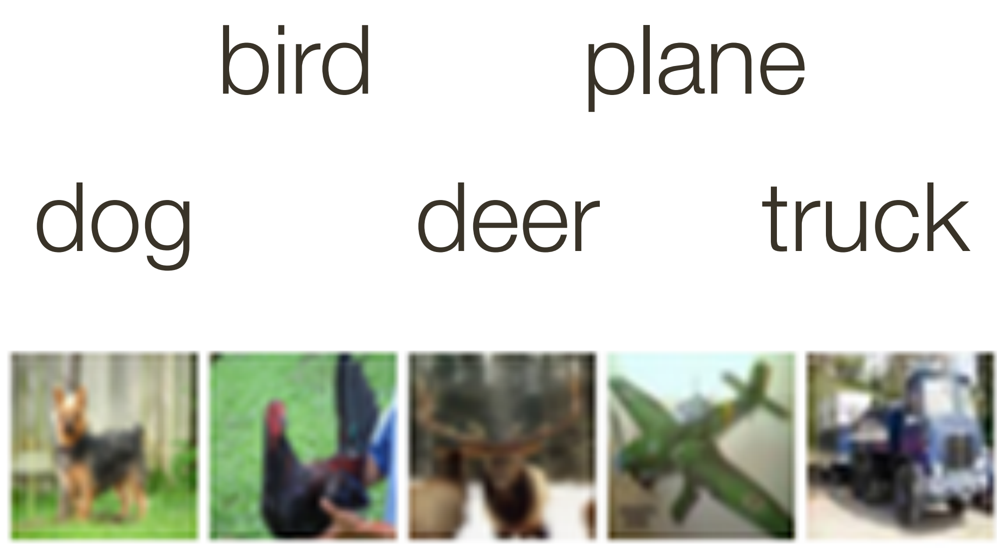


* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Autoencoders Reminder …



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



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

Variational Autoencoders

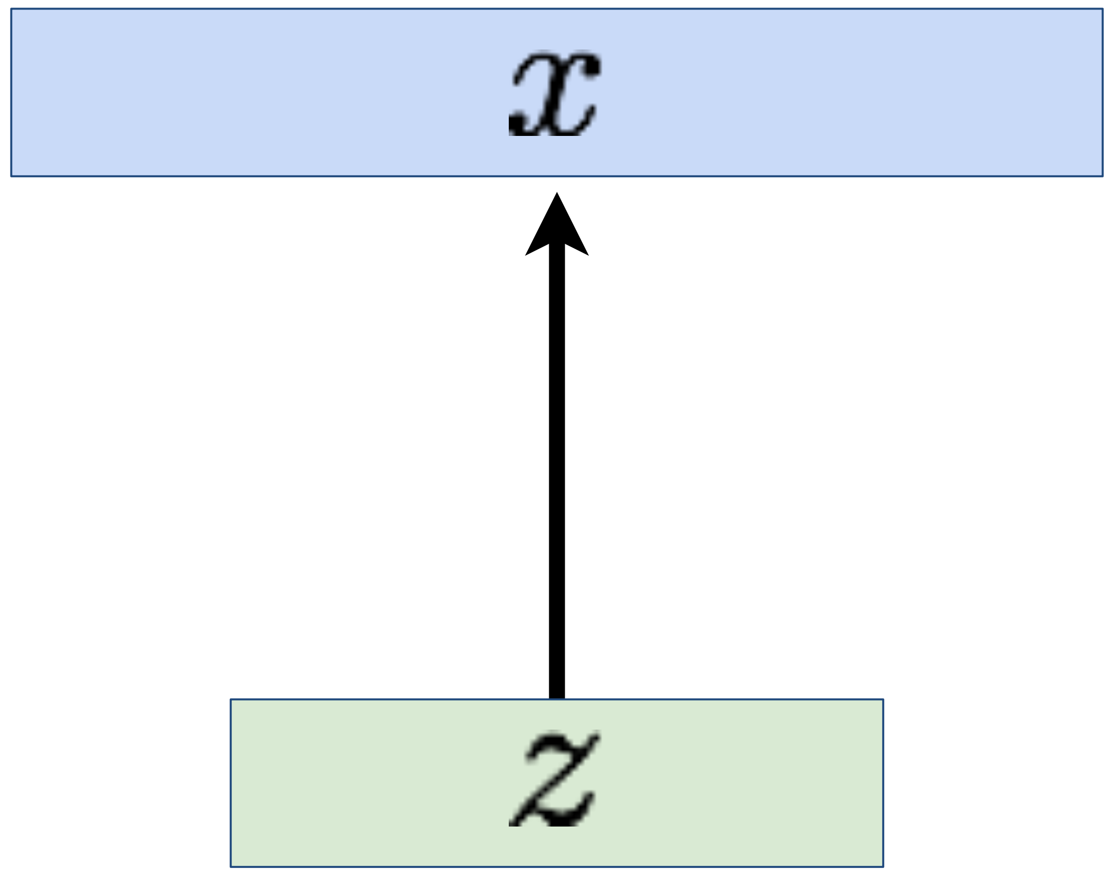
[Kingma and Welling, 2014]

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**

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

Variational Autoencoders

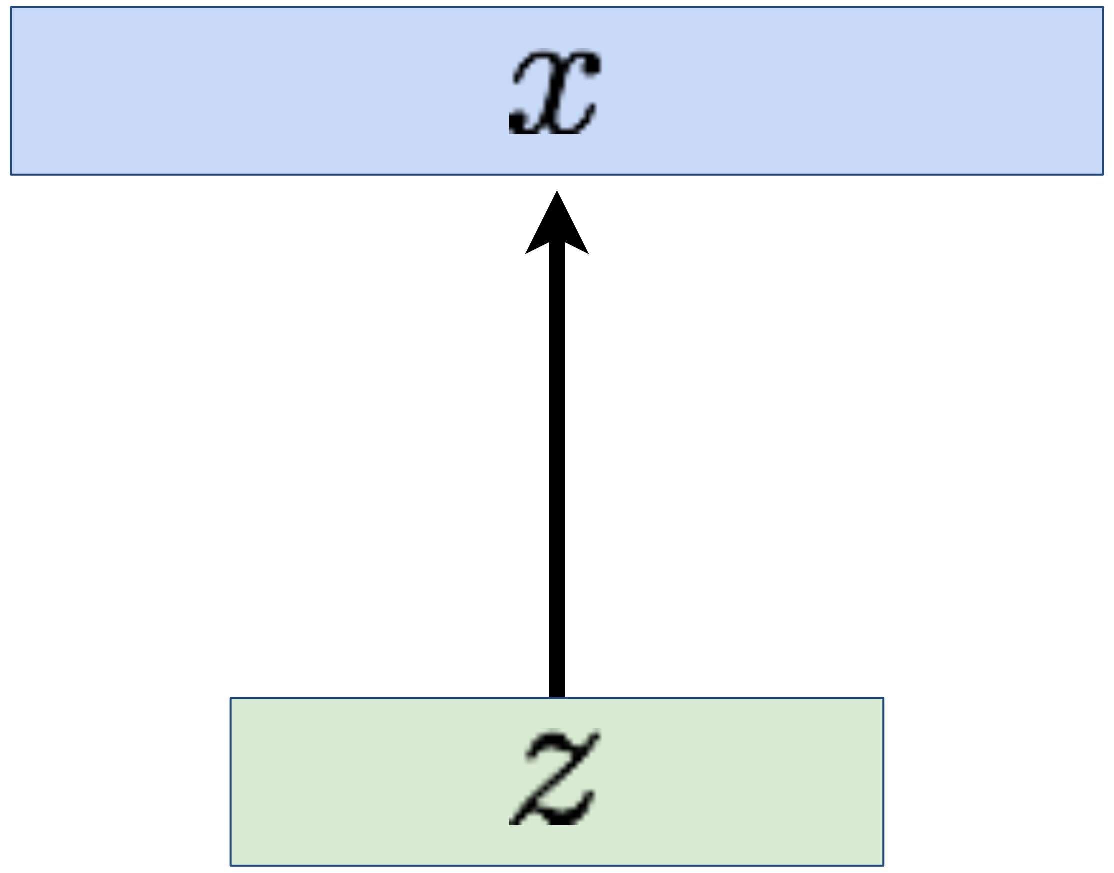
[Kingma and Welling, 2014]

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.*)

Variational Autoencoders

[Kingma and Welling, 2014]

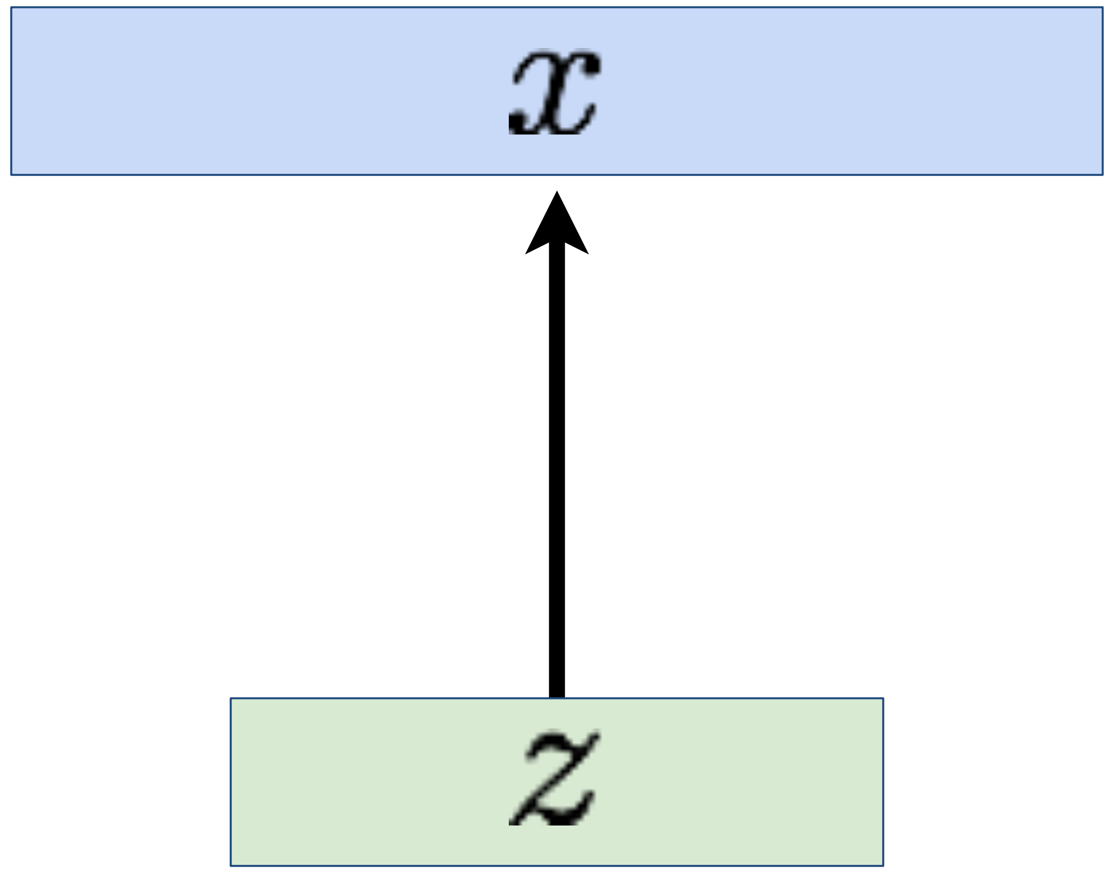
We want to **estimate the true parameters** θ^* of this generative model

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

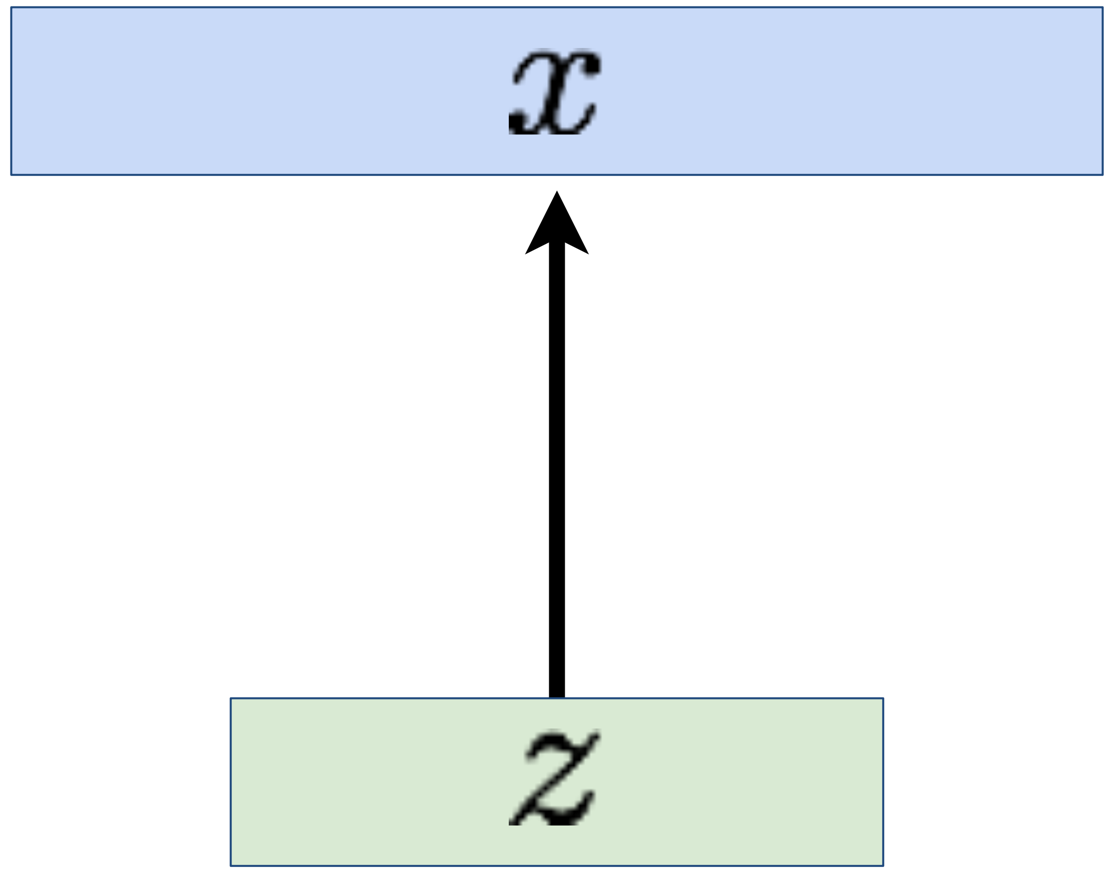
[Kingma and Welling, 2014]

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]

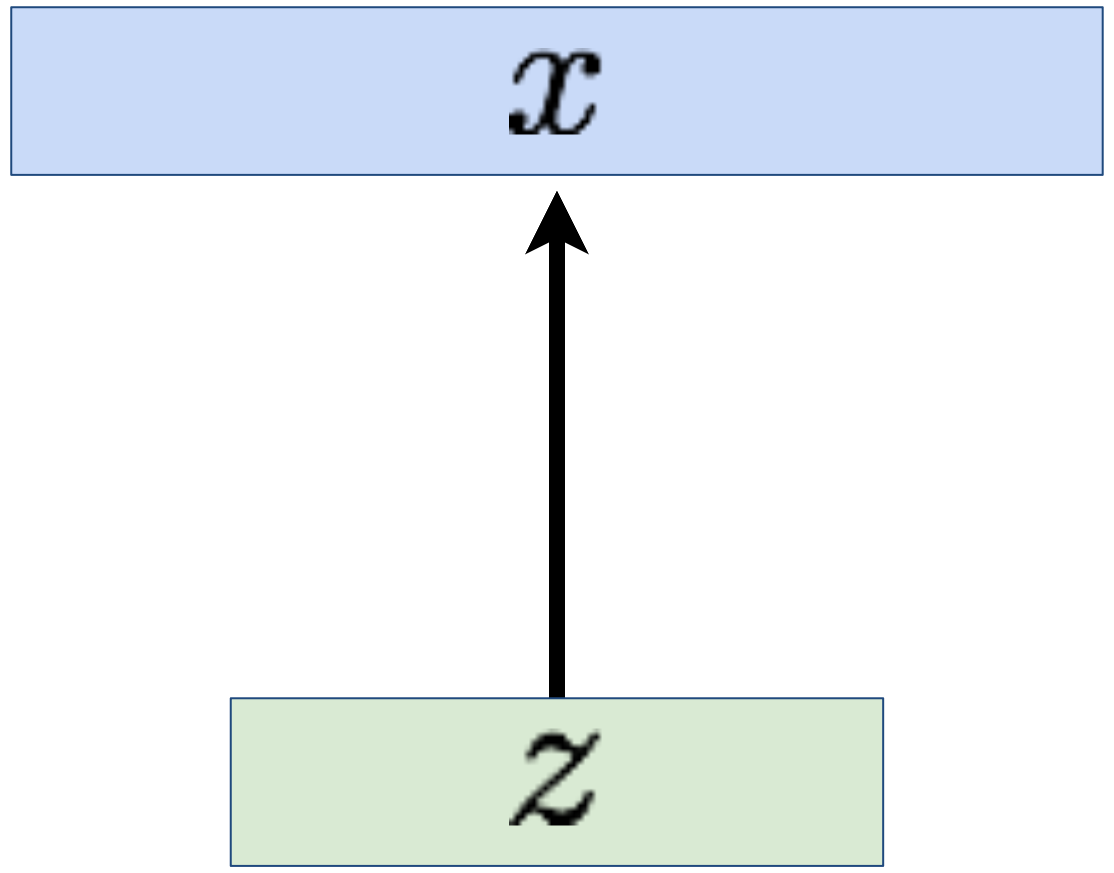
We want to **estimate the true parameters** θ^* of this generative model

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

Variational Autoencoders

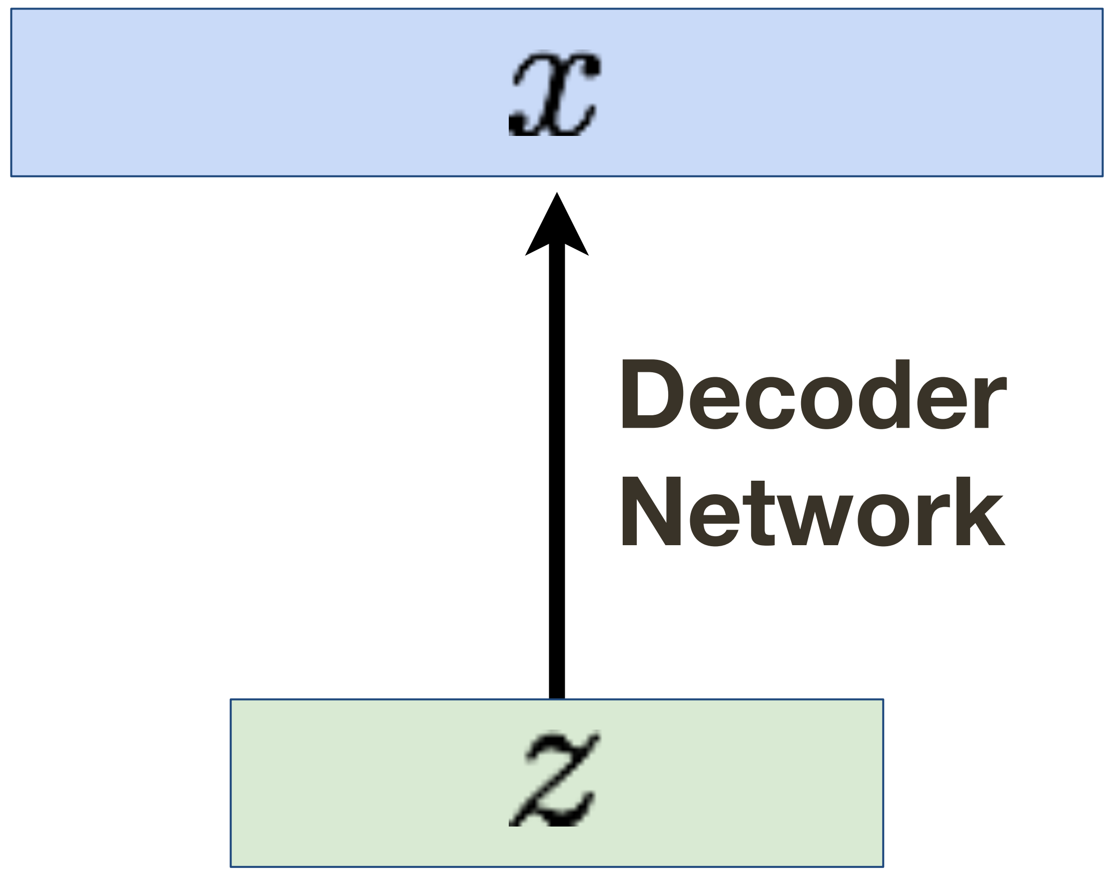
[Kingma and Welling, 2014]

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**

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

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

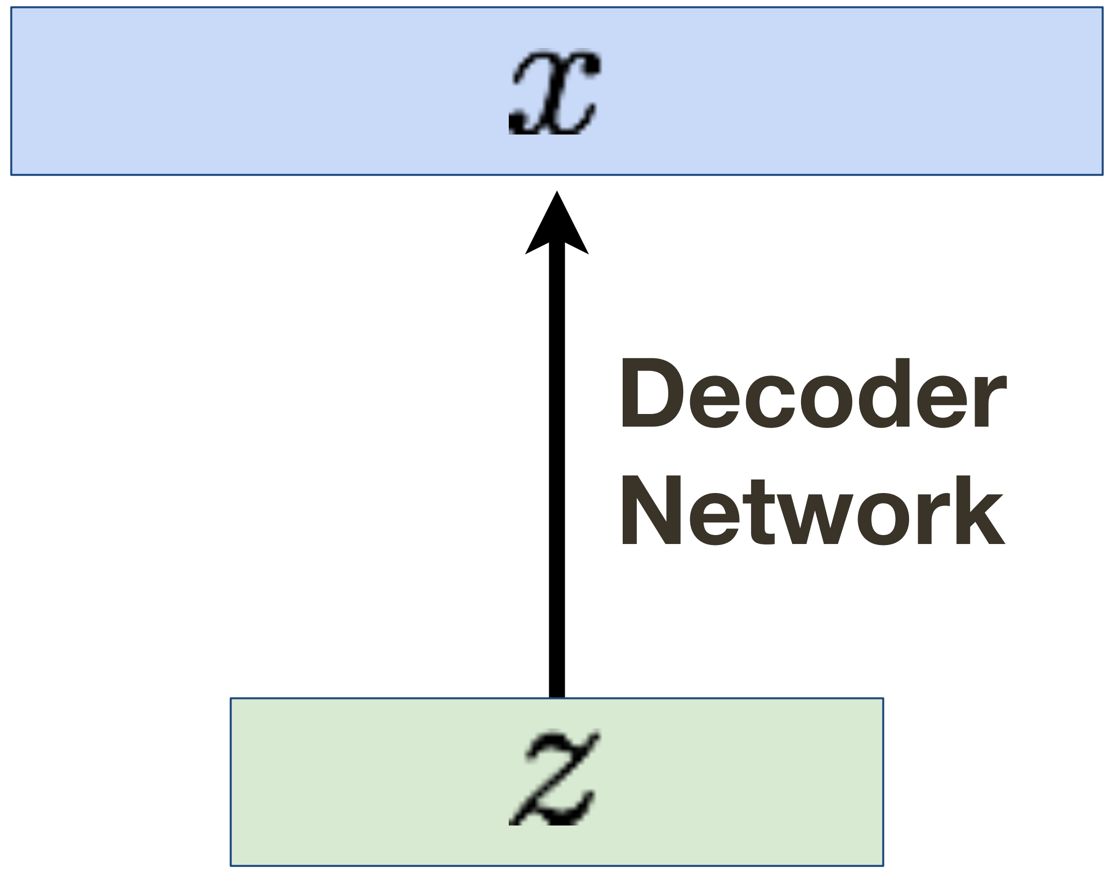
How do we **train** this model?

Sample from
true **conditional**

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

Sample from
true **prior**

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



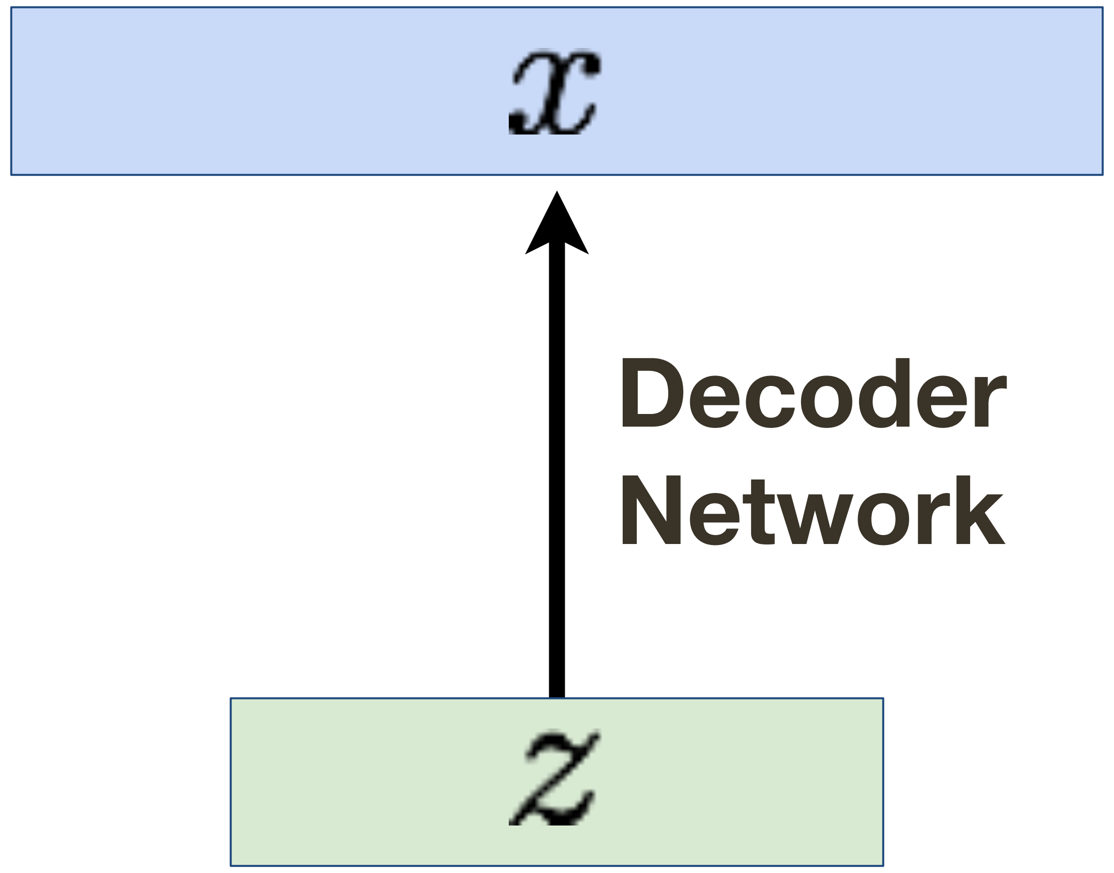
Variational Autoencoders

[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)

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

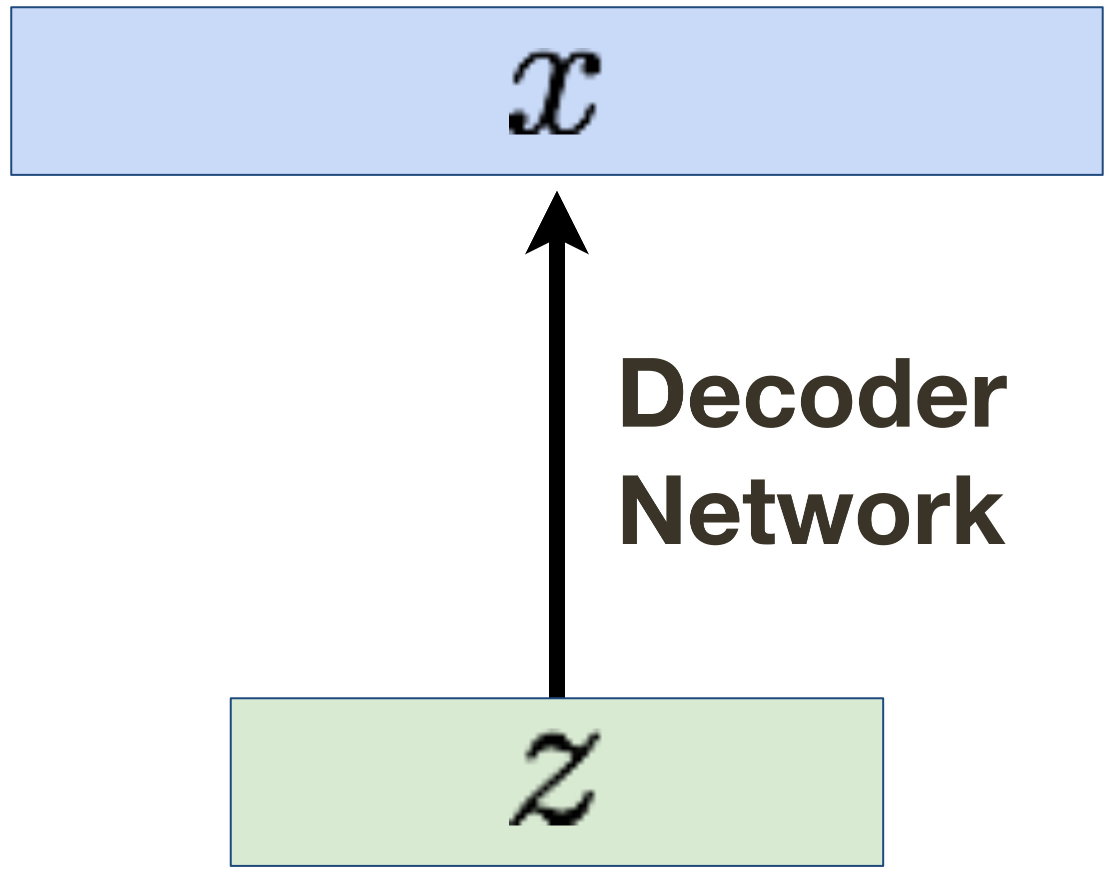
Variational Autoencoders

[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?

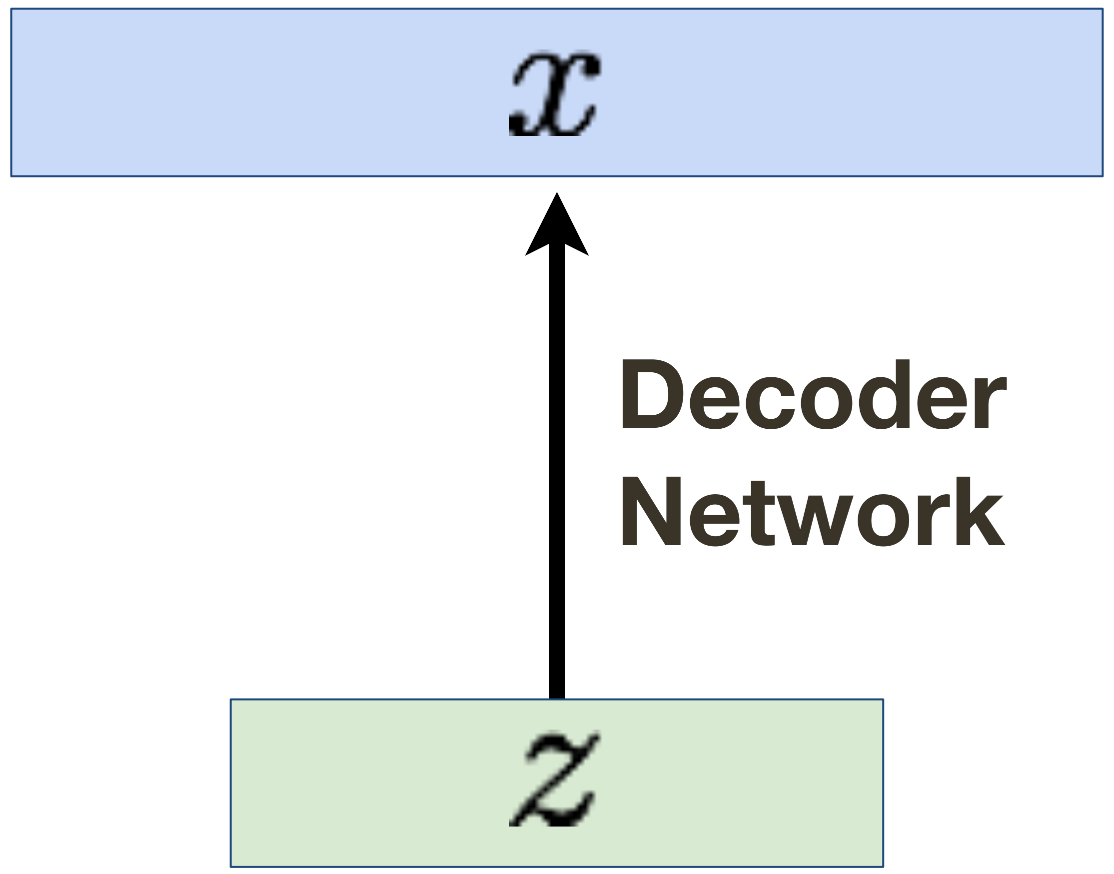
Variational Autoencoders

[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**: $$p_\theta(x) = \int \boxed{p_\theta(z)} p_\theta(x|z)dz$$

🙂

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

🙂

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: $p_\theta(z|x) = p_\theta(x|z)p_\theta(z)/\boxed{p_\theta(x)}$

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: $p_\theta(z|x) = p_\theta(x|z)p_\theta(z)/\boxed{p_\theta(x)}$

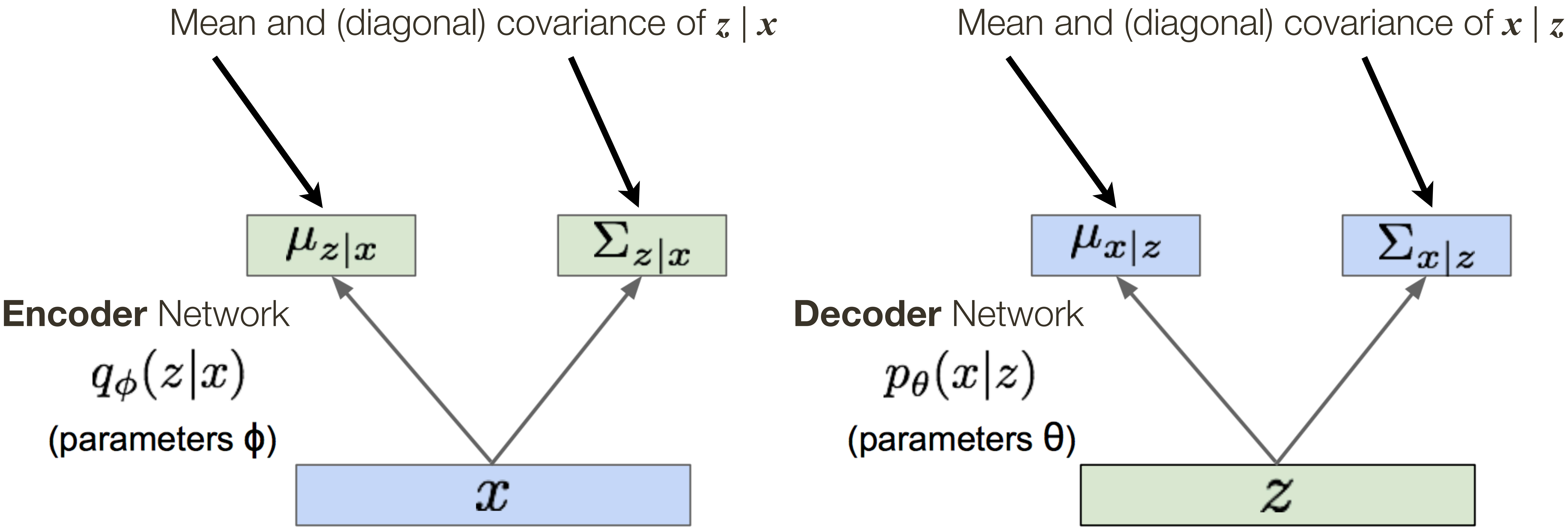
Solution: In addition to decoder network modeling $p_\theta(x|z)$, define additional encoder network $q_\phi(z|x)$ that approximates $p_\theta(z|x)$

— 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)

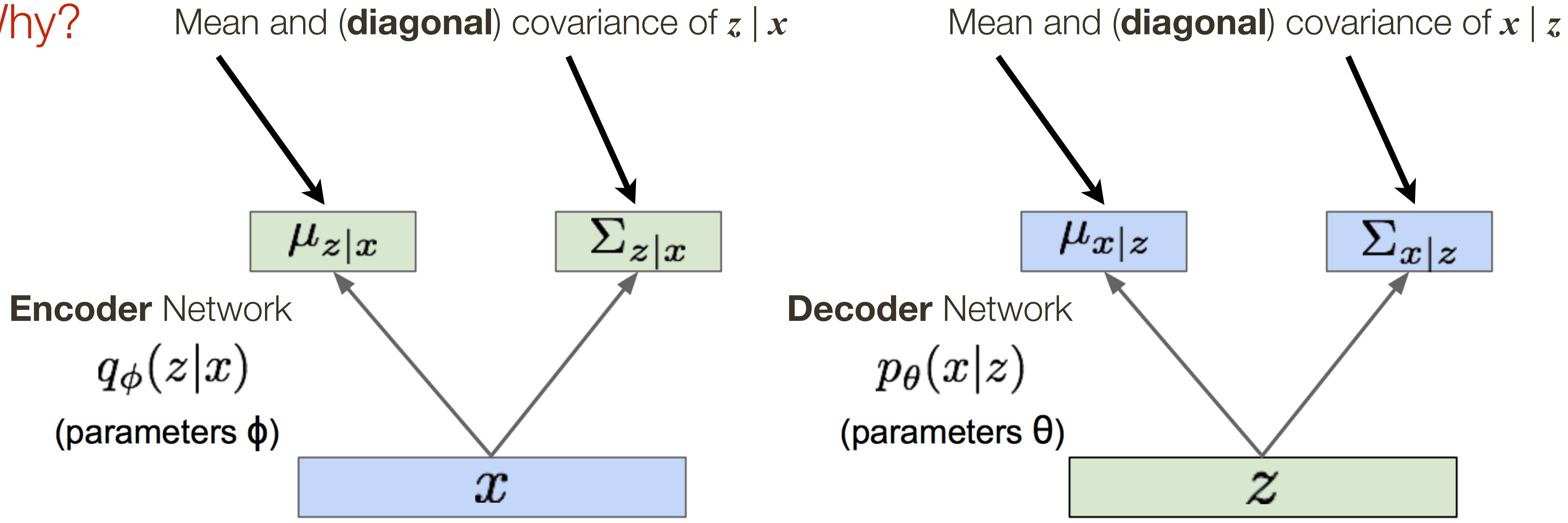


Variational Autoencoder

[Kingma and Welling, 2014]

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

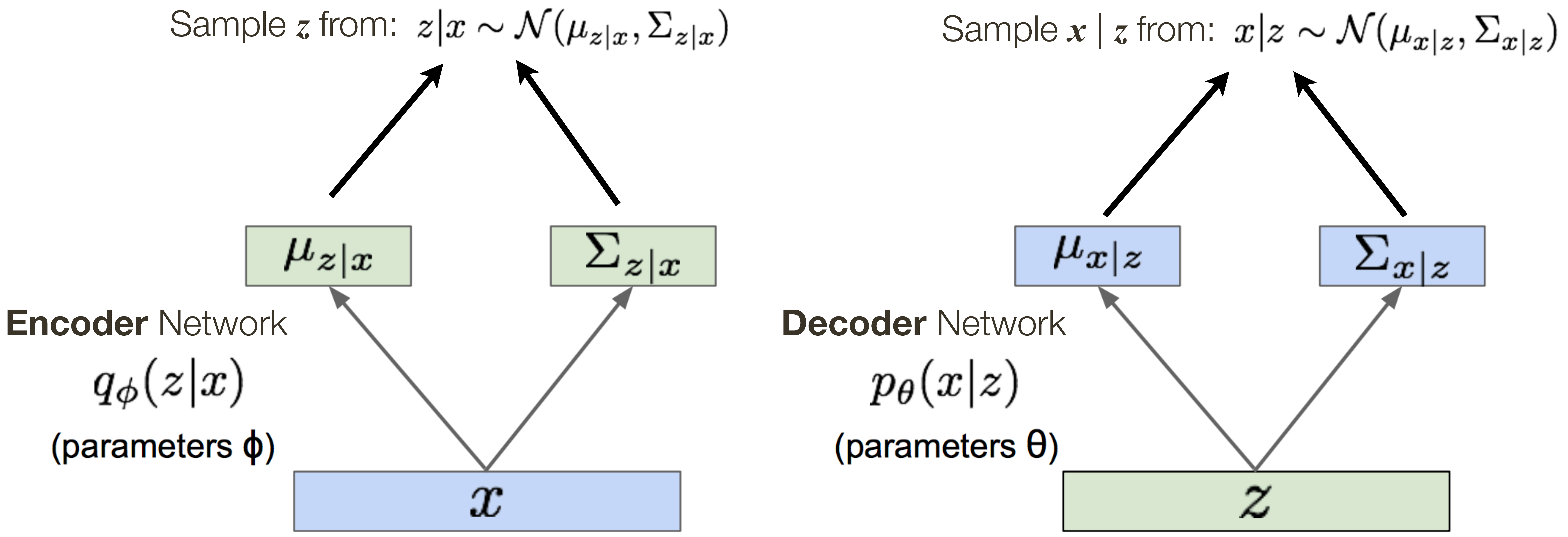
Why?



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:

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

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

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:

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

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

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

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:

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

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:

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

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.

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:

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

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

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

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

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

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

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

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

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

Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

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

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

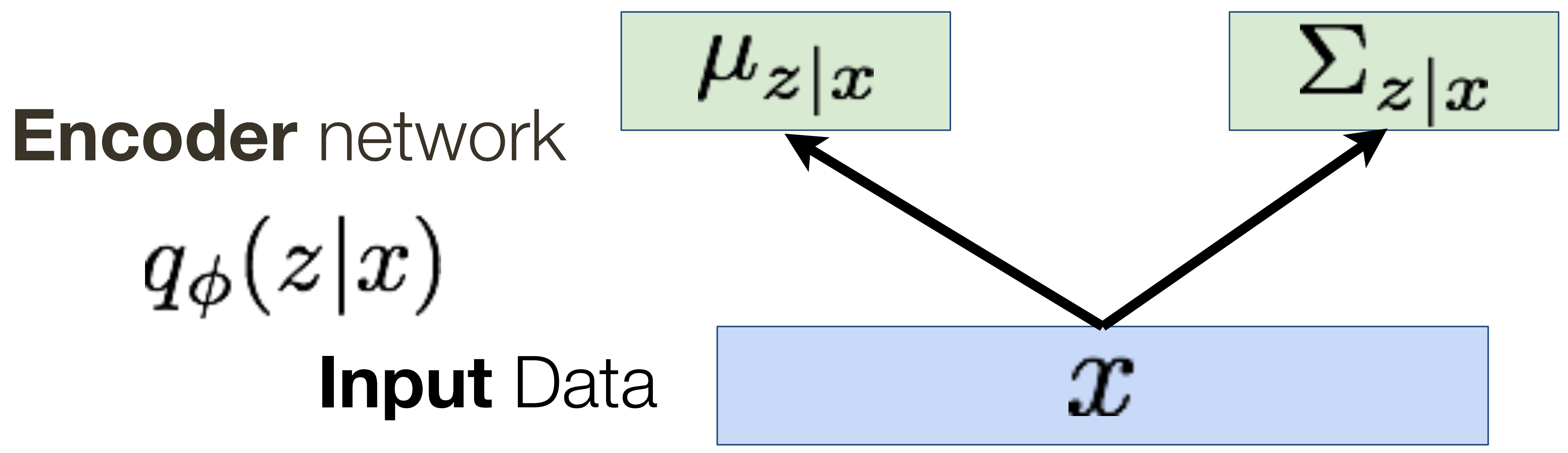
Input Data x

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

Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

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



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

Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

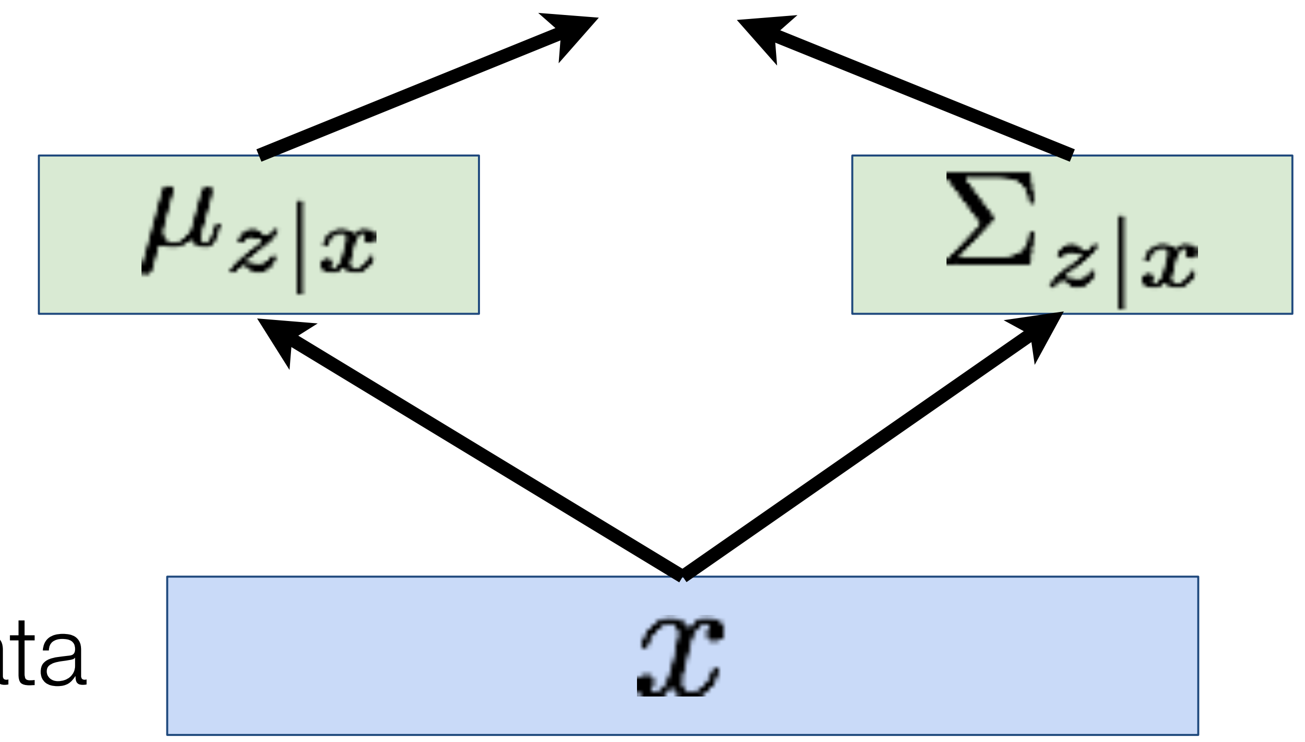
$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)}$$

Make approximate posterior distribution close to prior

Encoder network

$q_\phi(z|x)$

Input Data

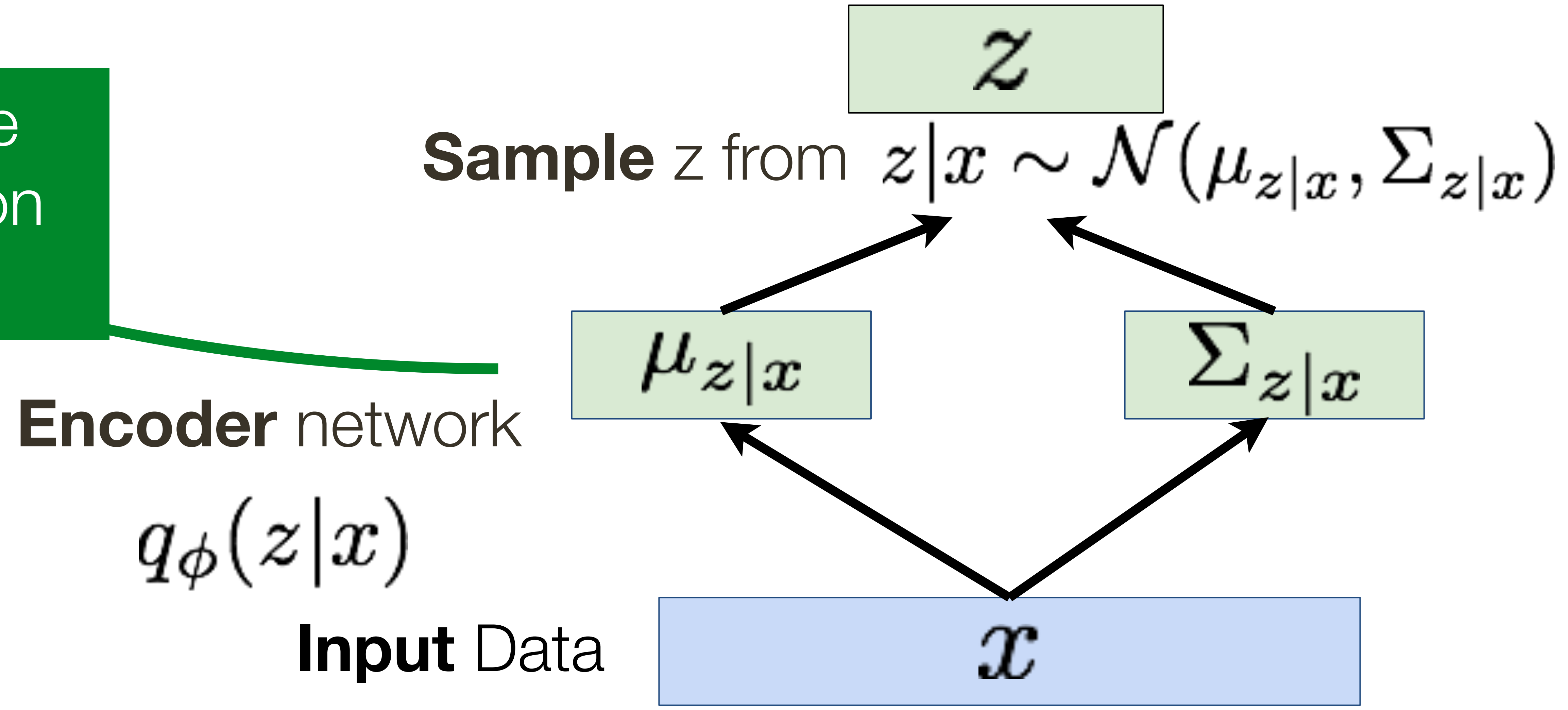


Variational Autoencoder: Learning

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



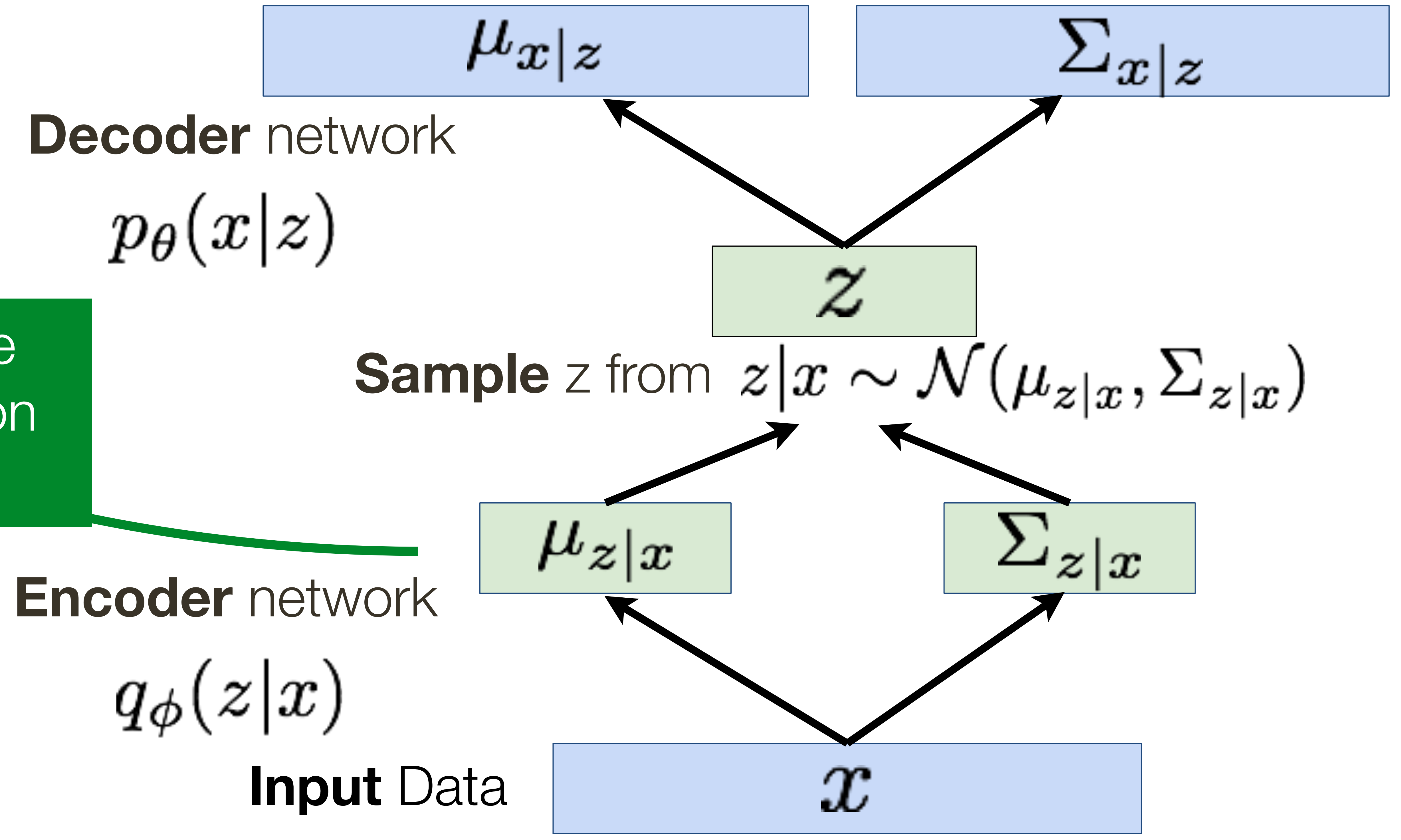
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

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



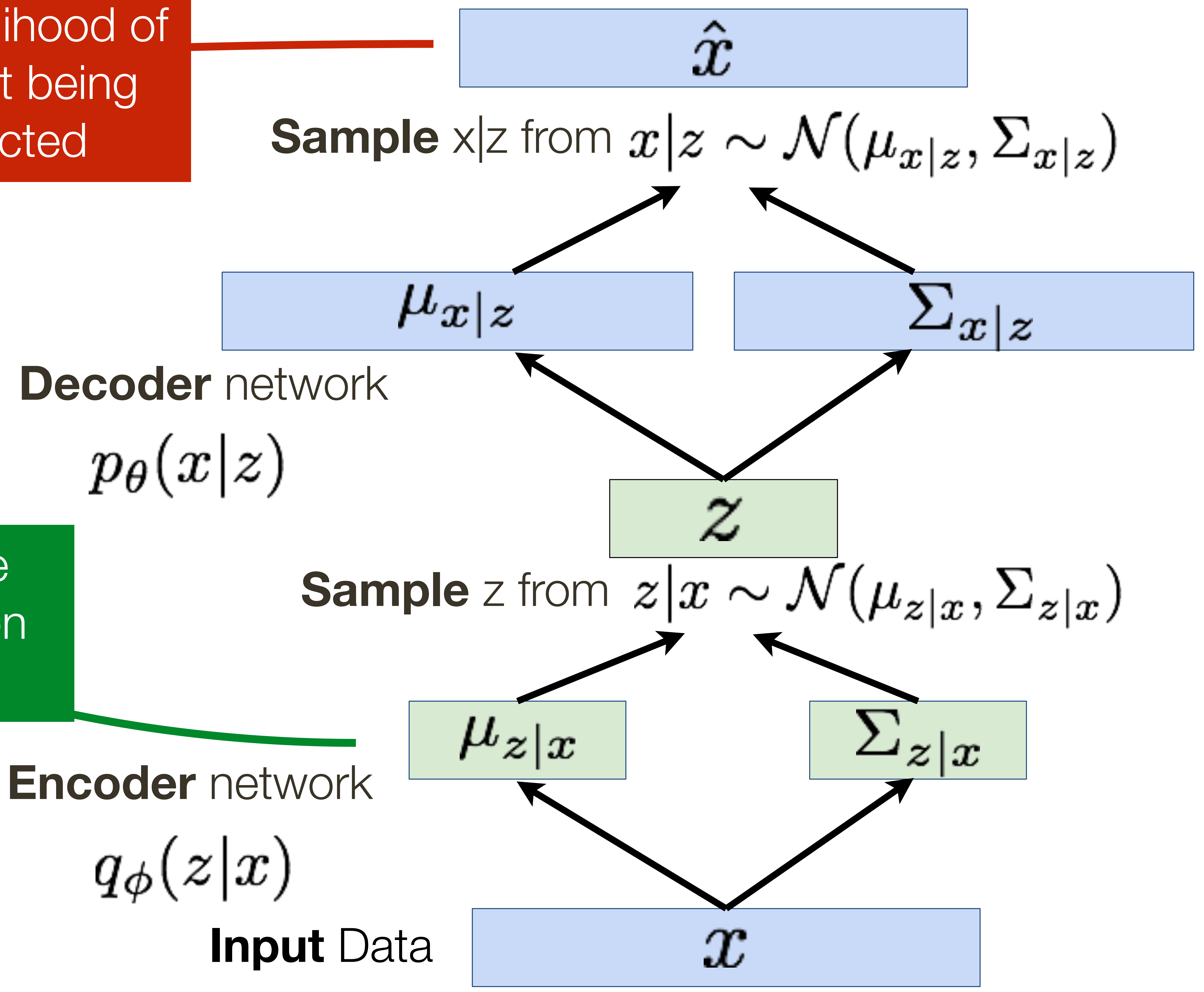
Variational Autoencoder: Learning

Putting it all together: maximizing the likelihood lower bound

Maximize likelihood of original input being reconstructed

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

Make approximate posterior distribution close to prior



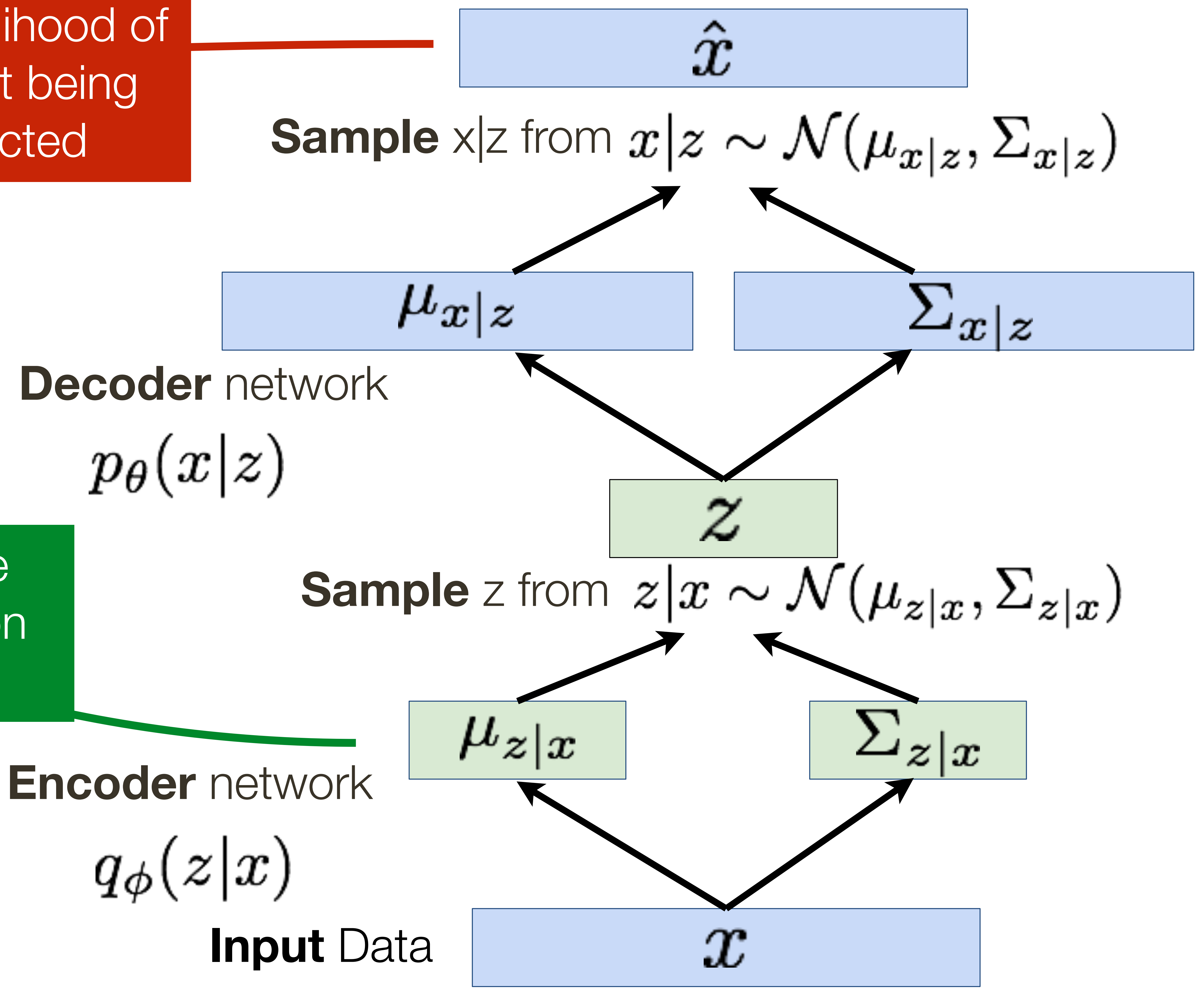
Variational Autoencoder: Learning

Putting it all together: maximizing the likelihood lower bound

Maximize likelihood of original input being reconstructed

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

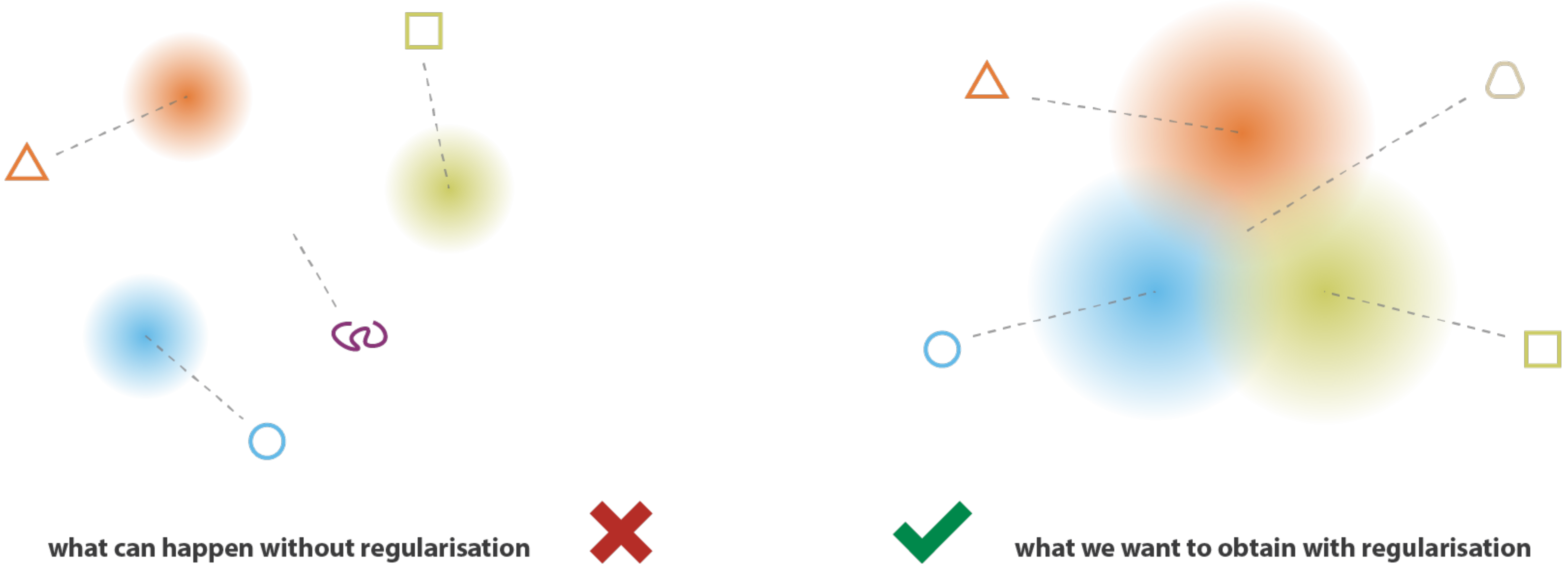
Make approximate posterior distribution close to prior



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

Variational Autoencoder: Learning

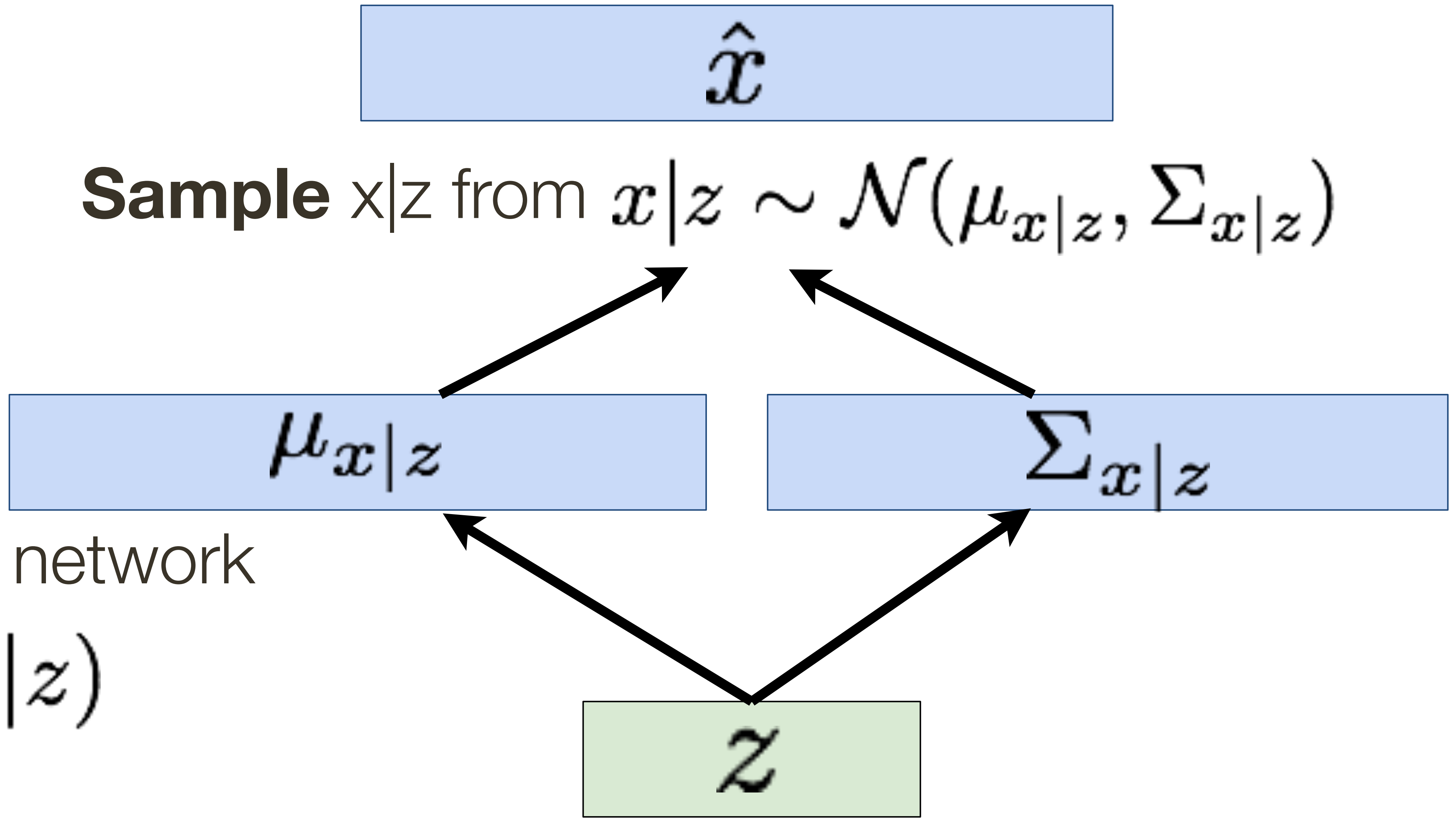
$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$



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

Variational Autoencoder: Generating Data

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

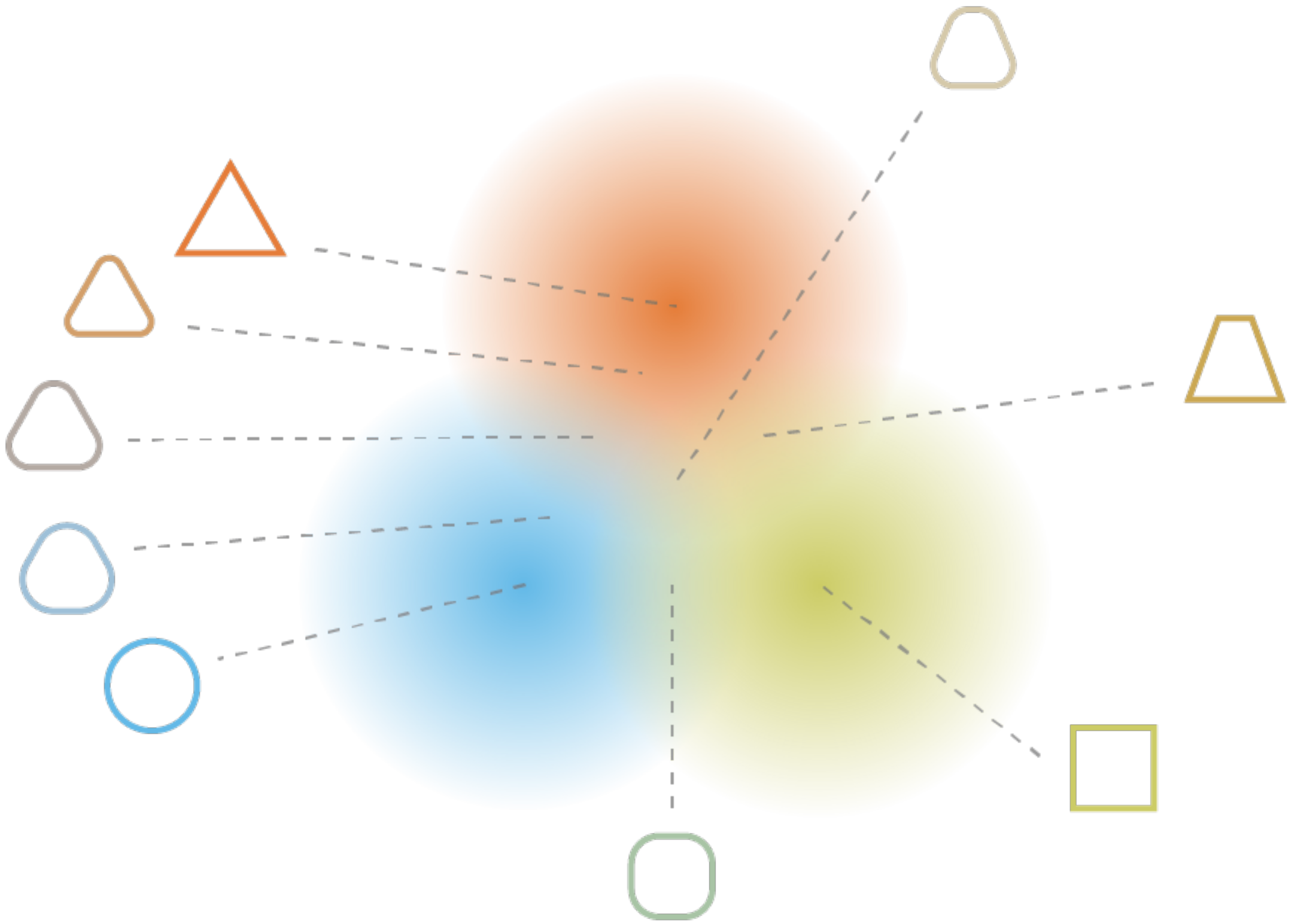


Decoder network

$p_\theta(x|z)$

Sample z from $z \sim \mathcal{N}(0, I)$

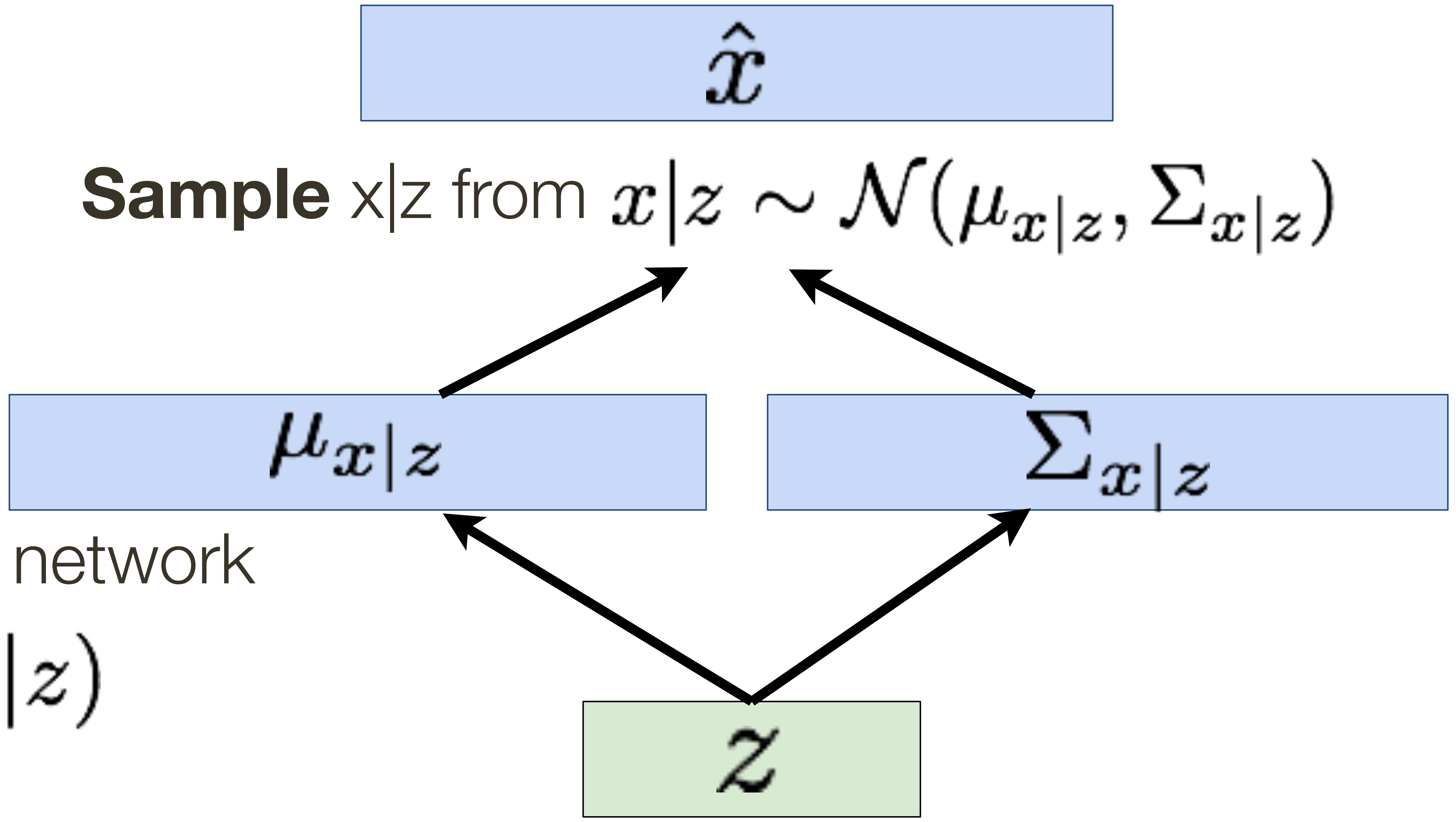
Variational Autoencoder: Generating Data



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

Variational Autoencoder: Generating Data

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



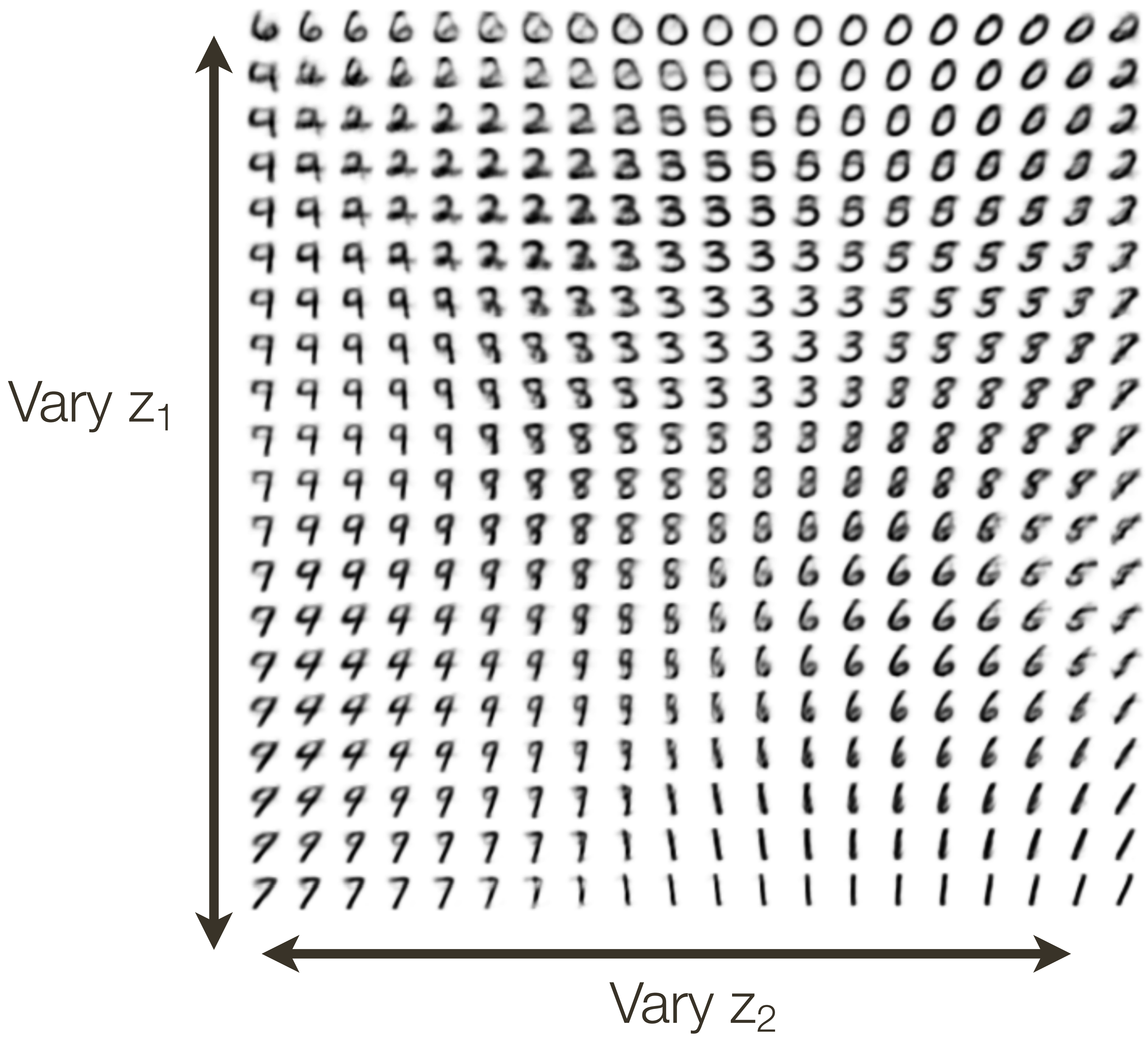
Sample x|z from $x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$

Decoder network

$p_\theta(x|z)$

Sample z from $z \sim \mathcal{N}(0, I)$

Data manifold for 2-d z



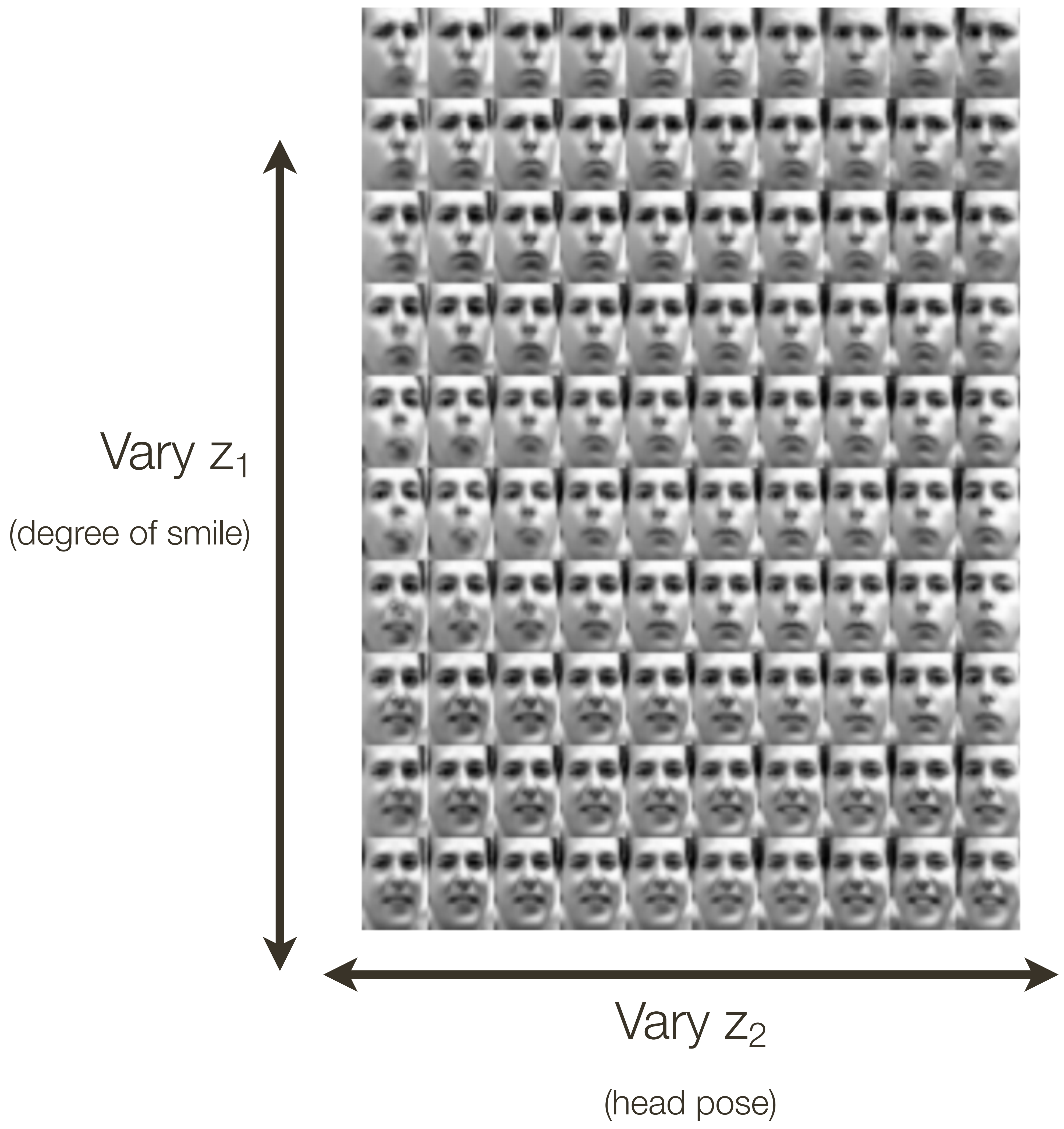
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

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**

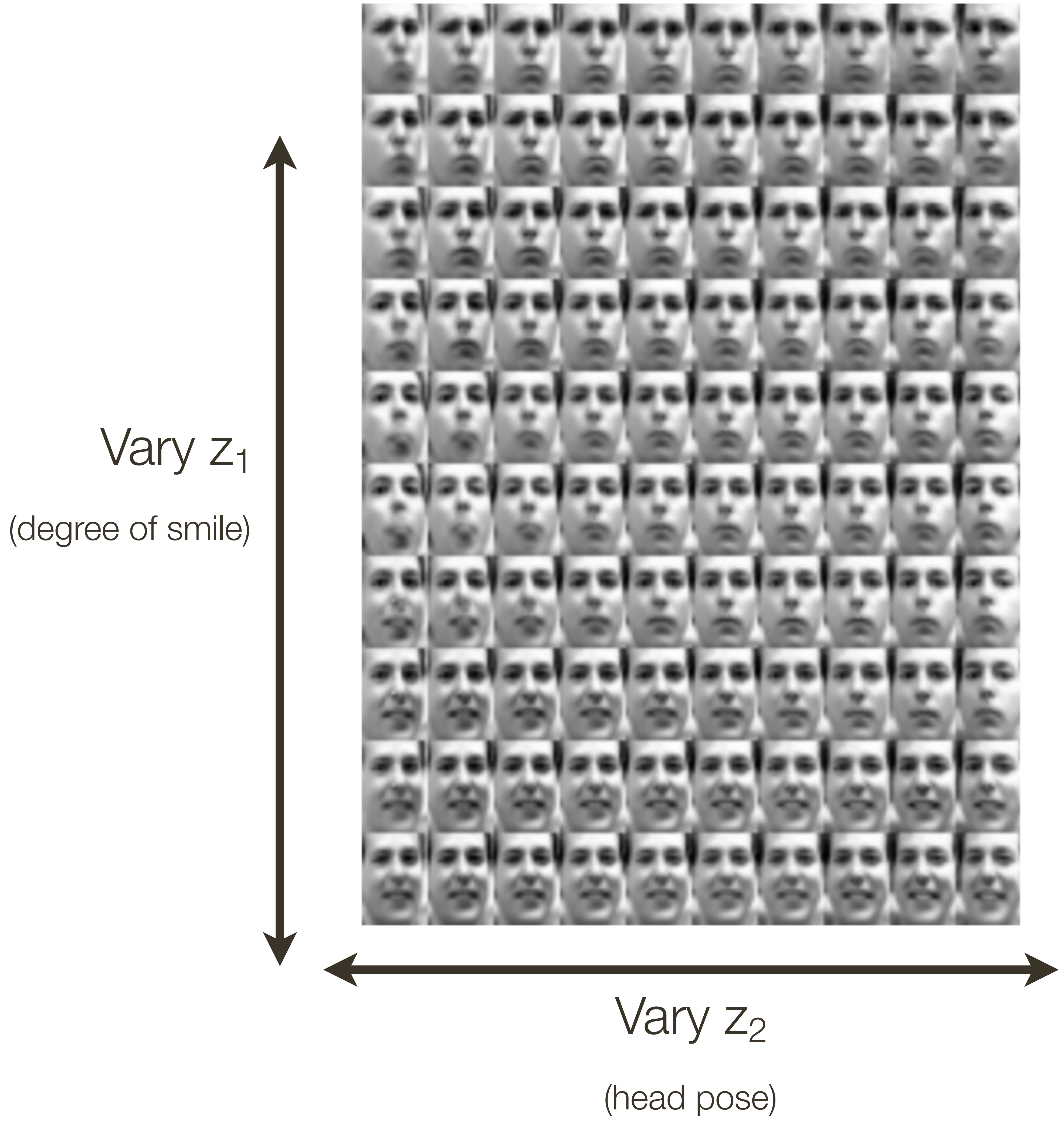
Variational Autoencoder: Generating Data

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



Variational Autoencoder: Generating Data

32x32 CIFAR-10

Labeled Faces in the Wild