THE UNIVERSITY OF BRITISH COLUMBIA

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

Lecture 17: Generative Models Cont. (VAE, GANs)

Course **Logistics**

- **Great set of projects!**
  - Modes of "sub-optimality" at this stage:

    (1) not enough thought into what alterations to base-model should be tested

    (2) motivation for architectures
  - What am I expecting for the project?
  - Feedback (still working on this)

- **Proposal** and **presentation** submission (Canvas on **Friday**)

- Paper **presentations** and list

So far …

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

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

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

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):

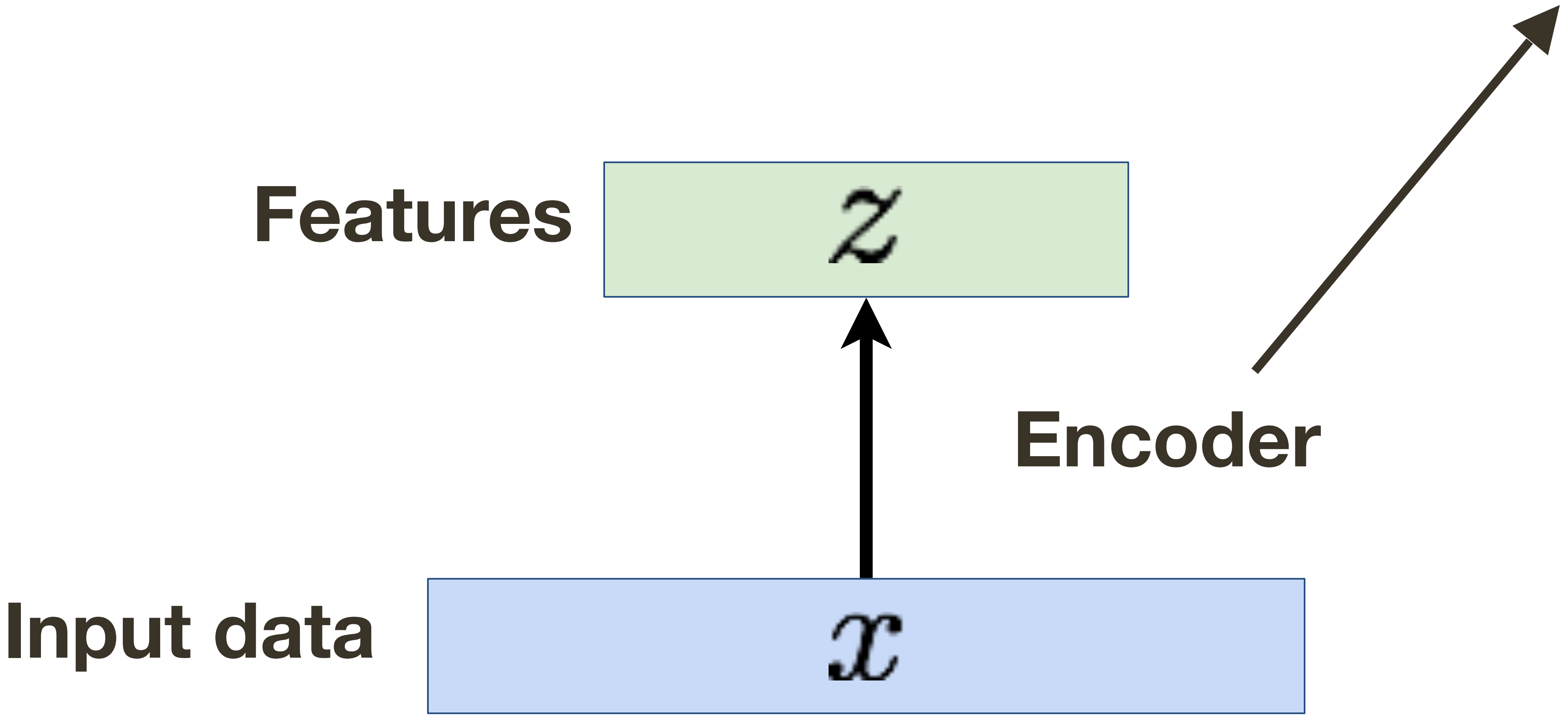
$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

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

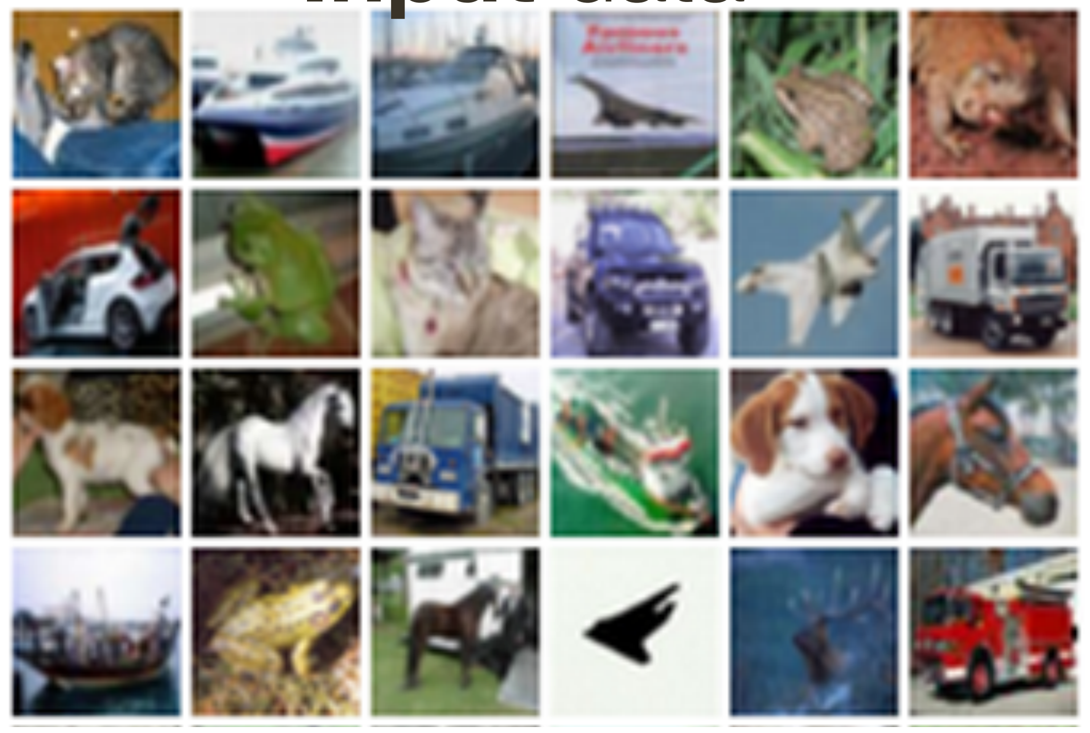
Autoencoders Reminder …

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



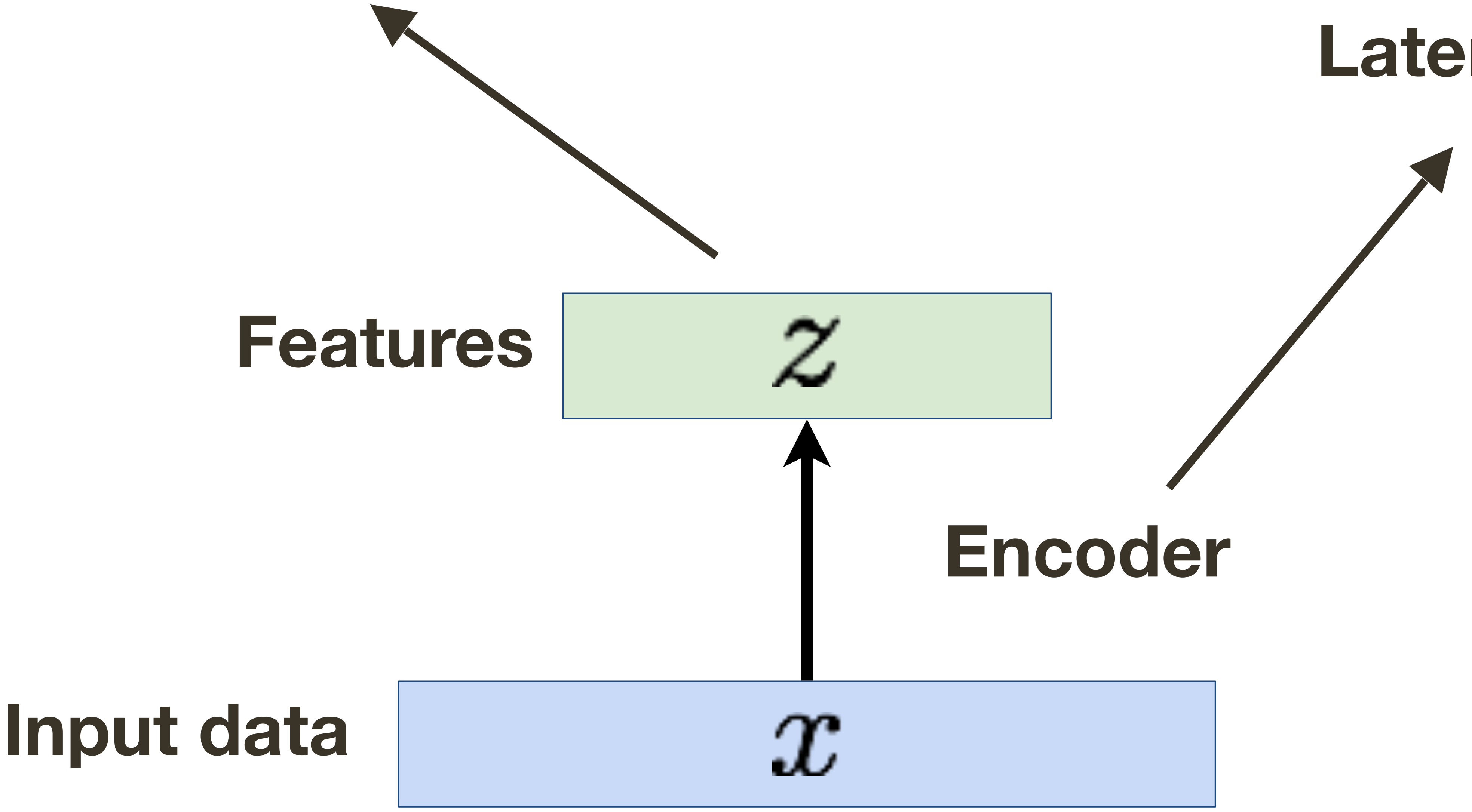
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Autoencoders Reminder …

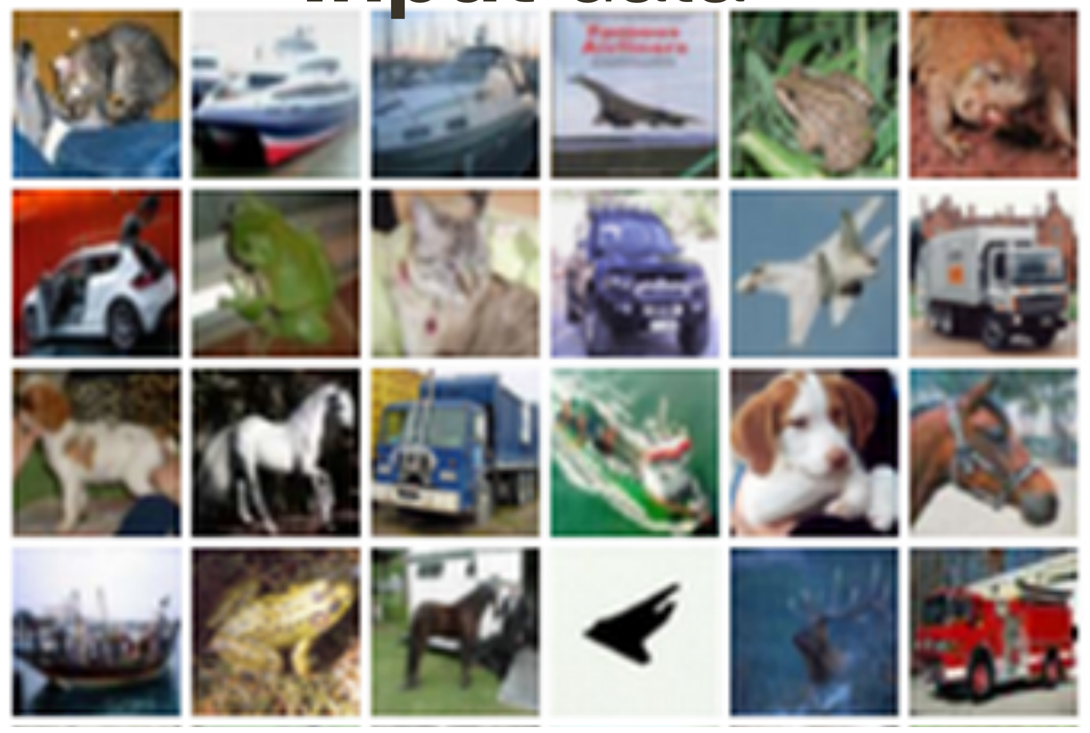
Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

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**

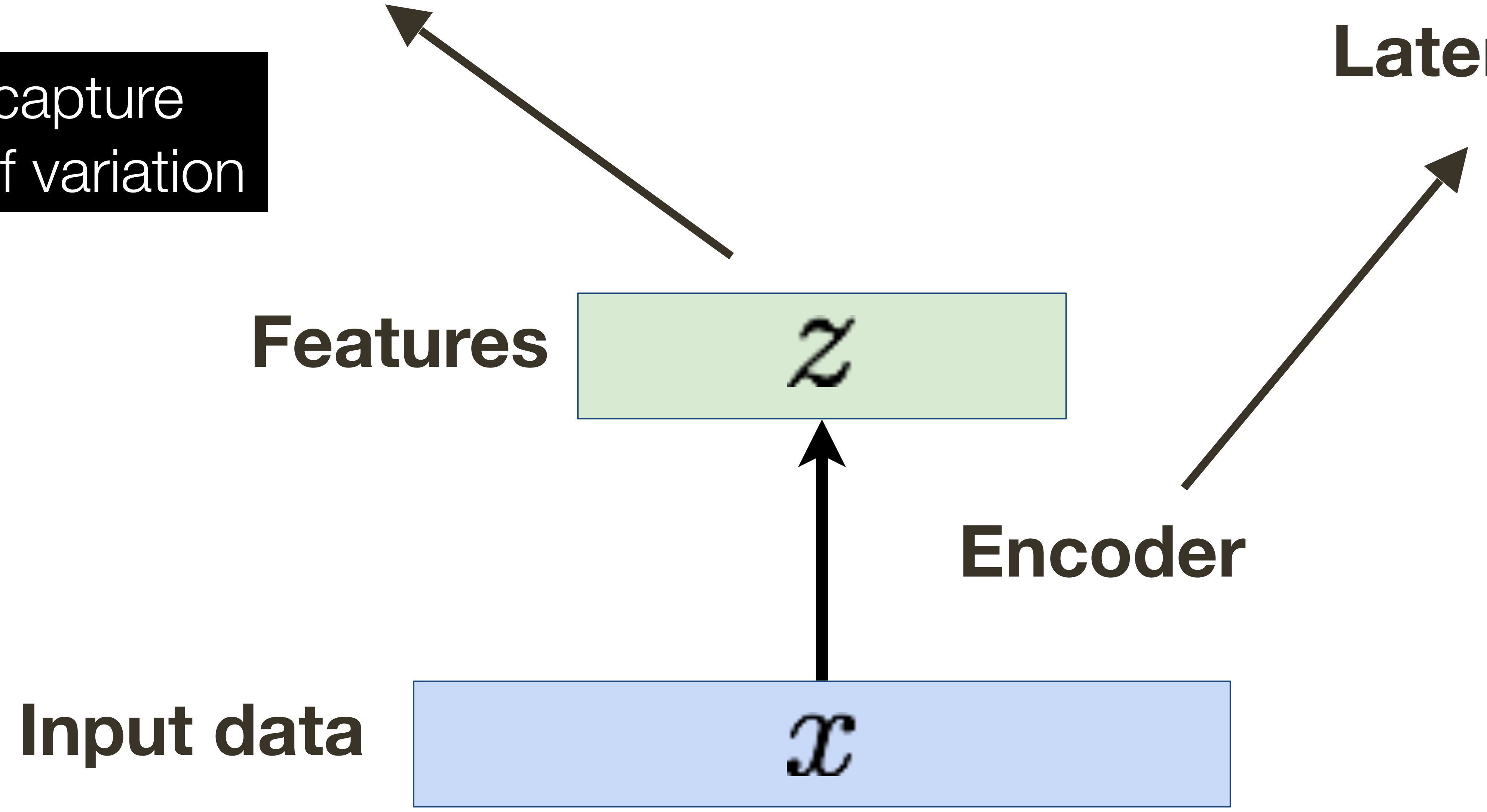
Autoencoders Reminder …

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

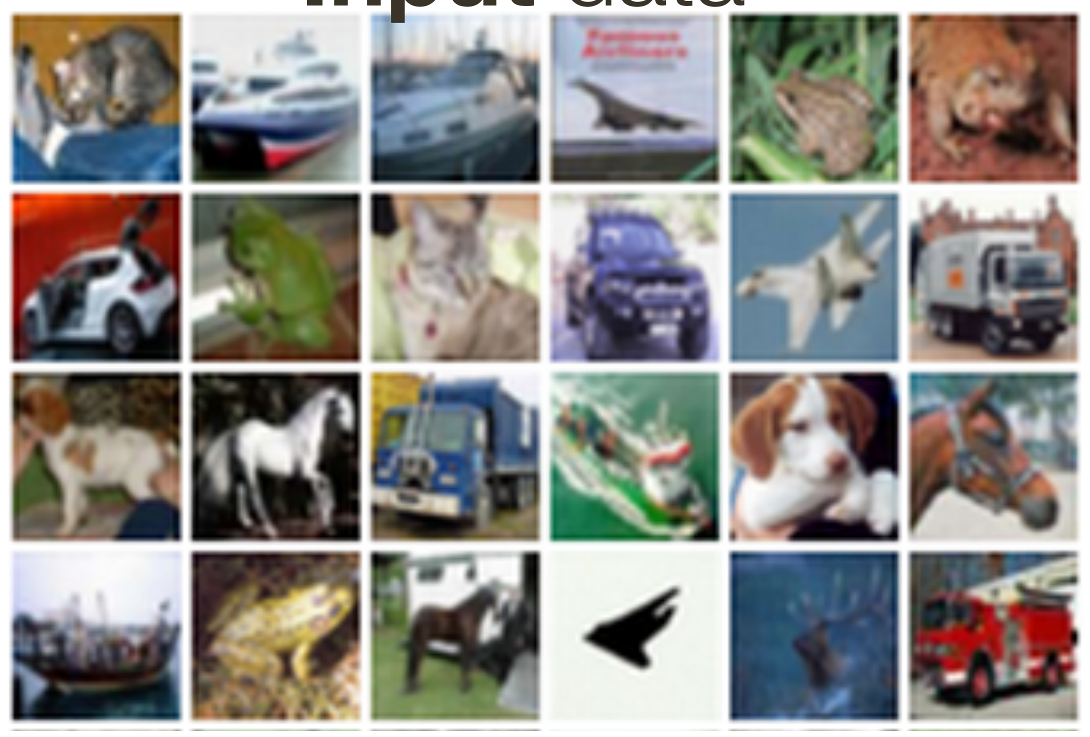
z usually smaller than x
(dimensionality reduction)

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

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



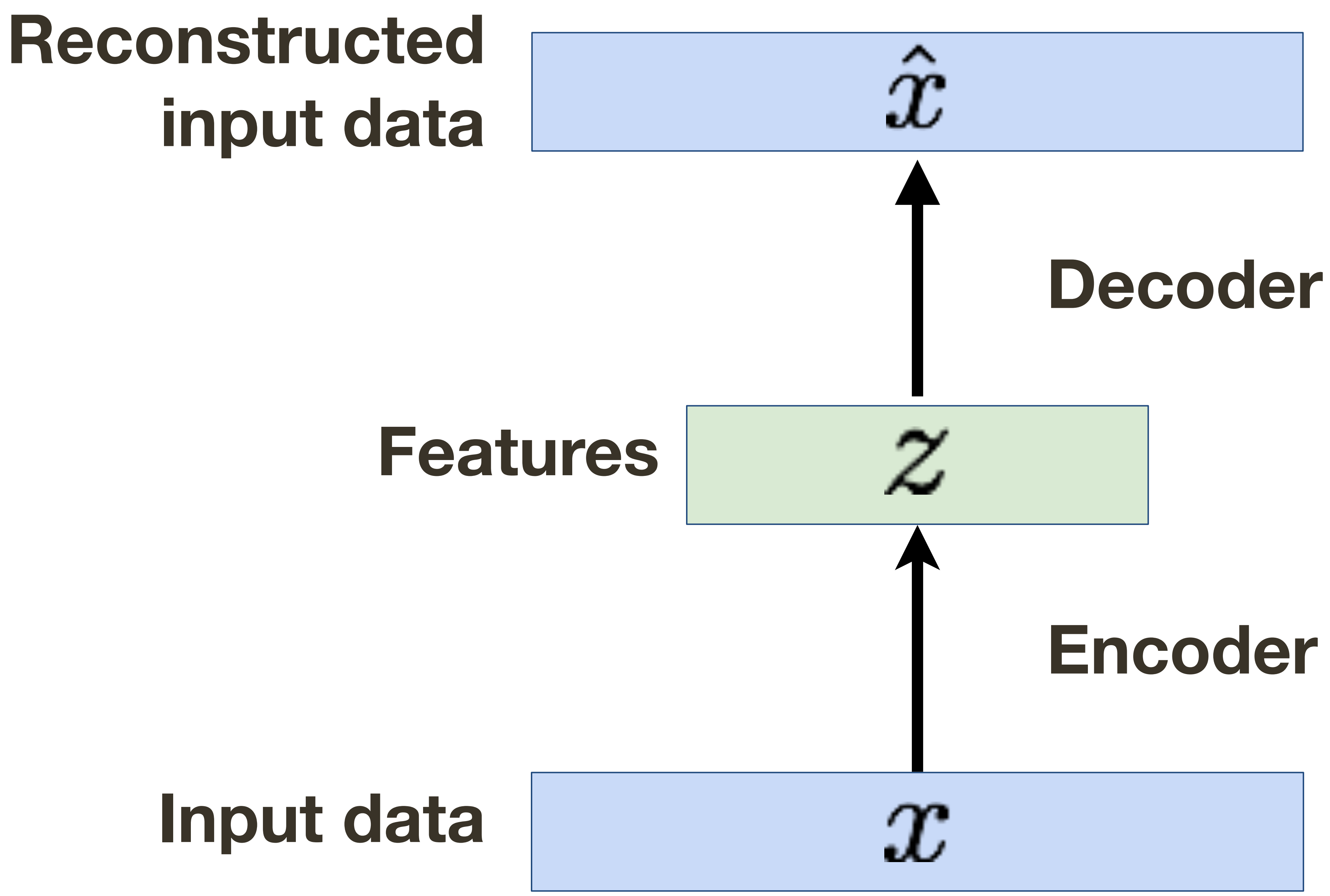
Input data



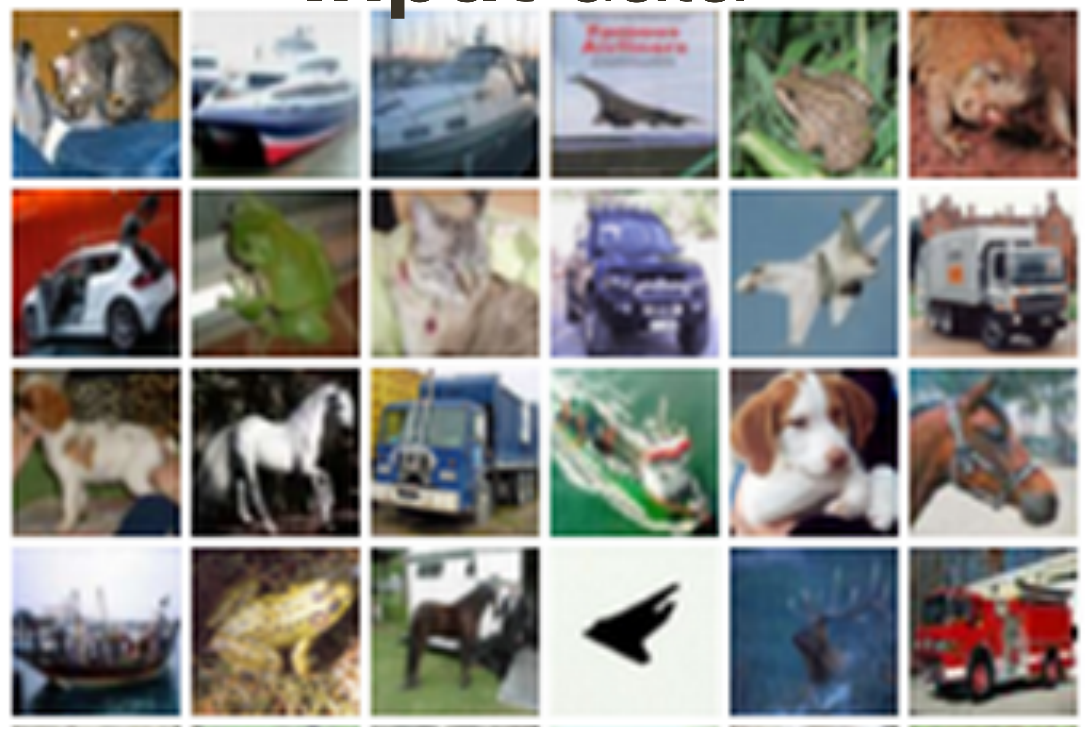
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Autoencoders Reminder …

Train such that features can reconstruct original data best they can



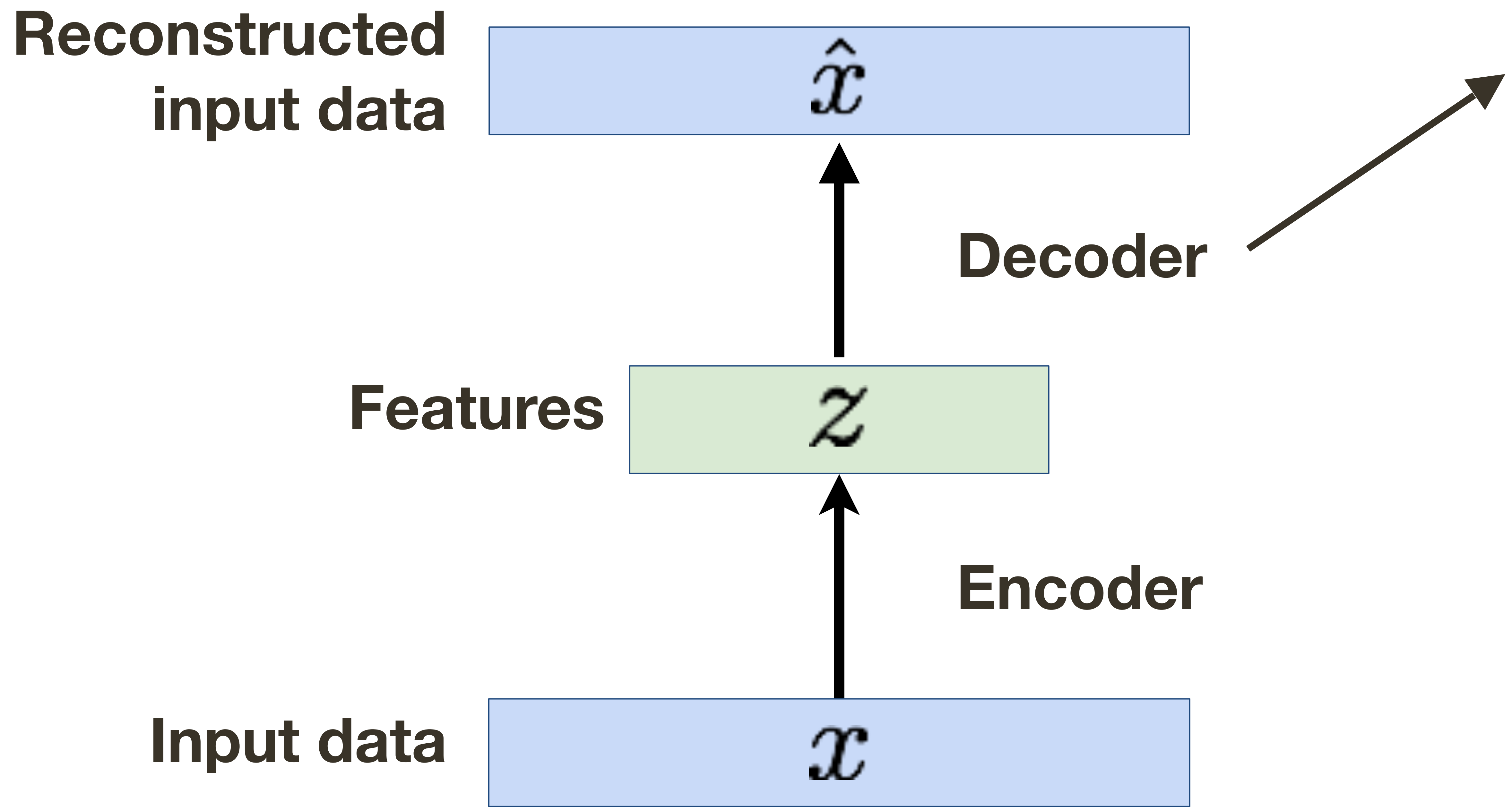
Input data



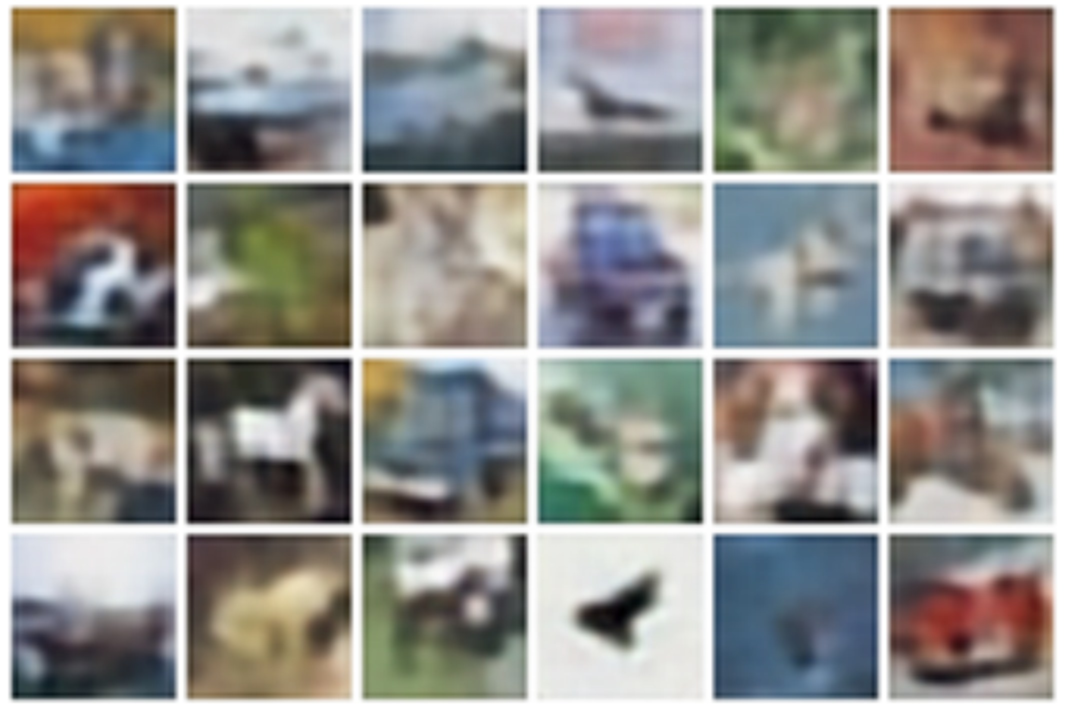
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Autoencoders Reminder …

Train such that features can reconstruct original data best they can

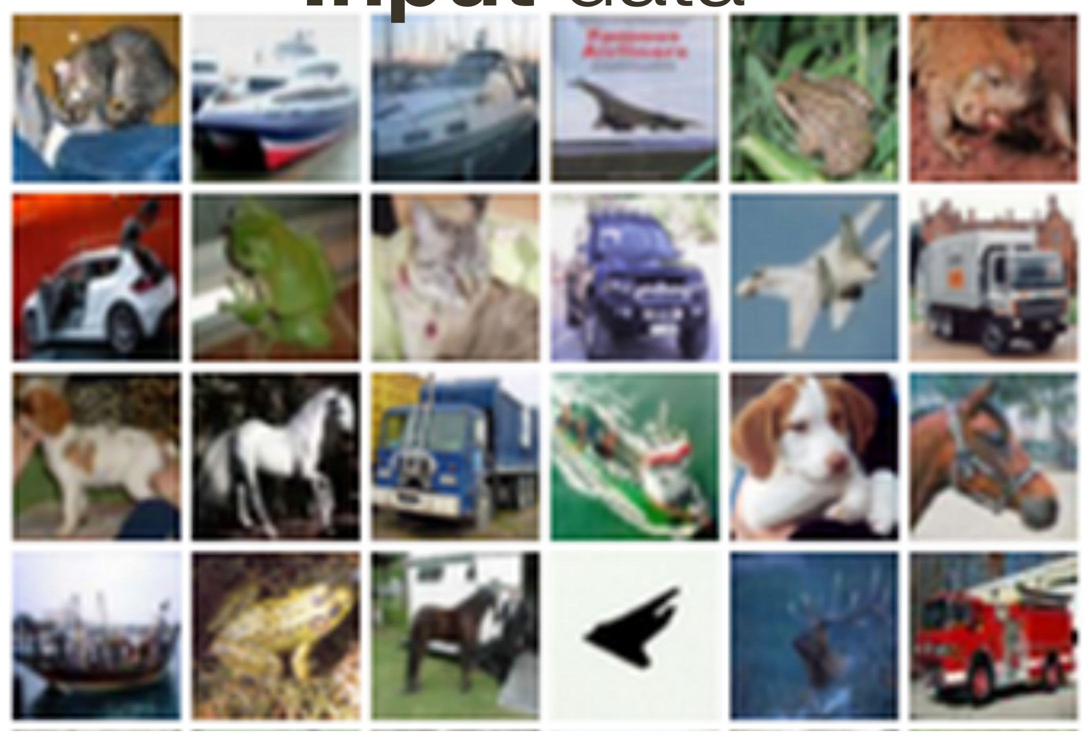


Reconstructed data

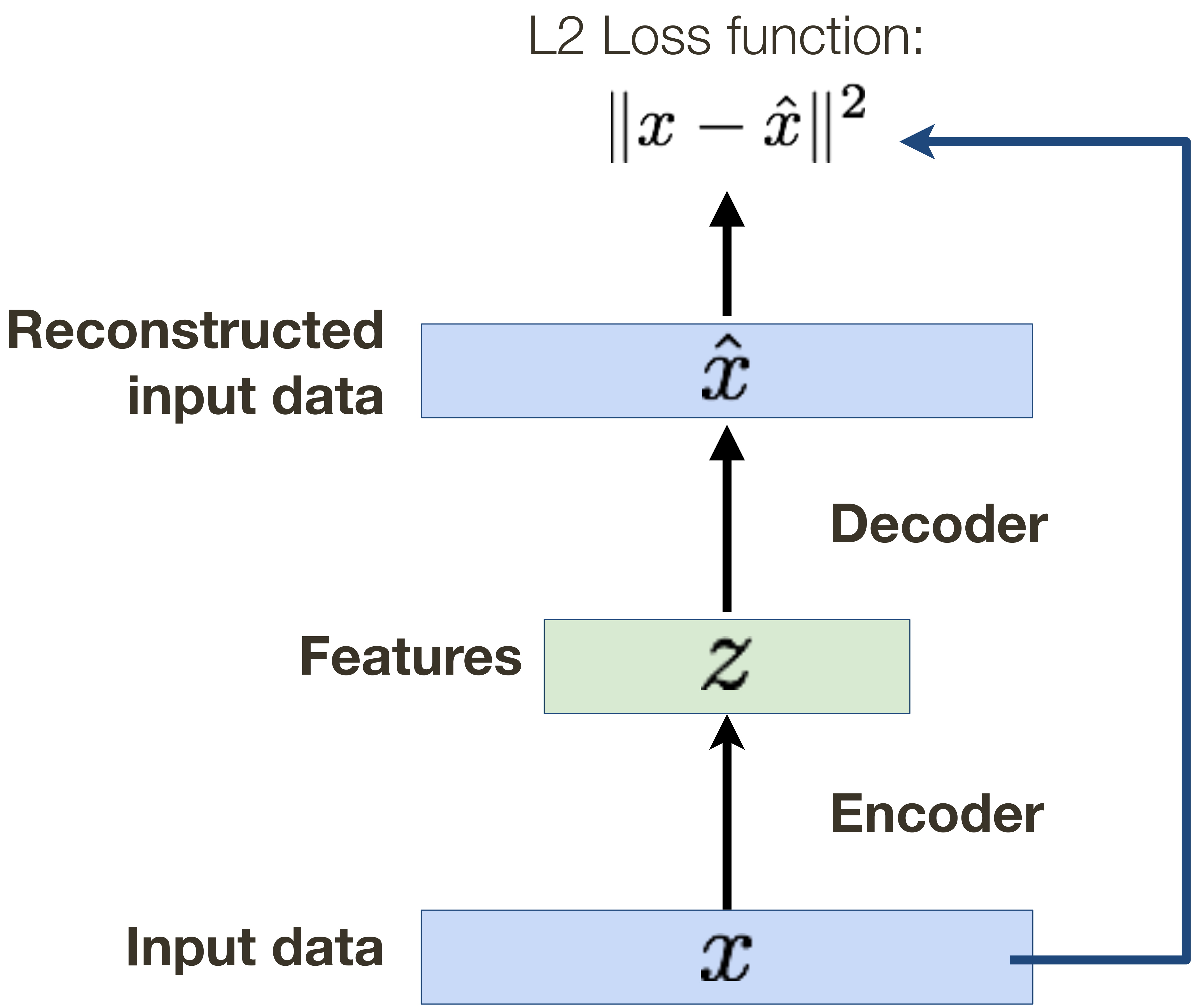


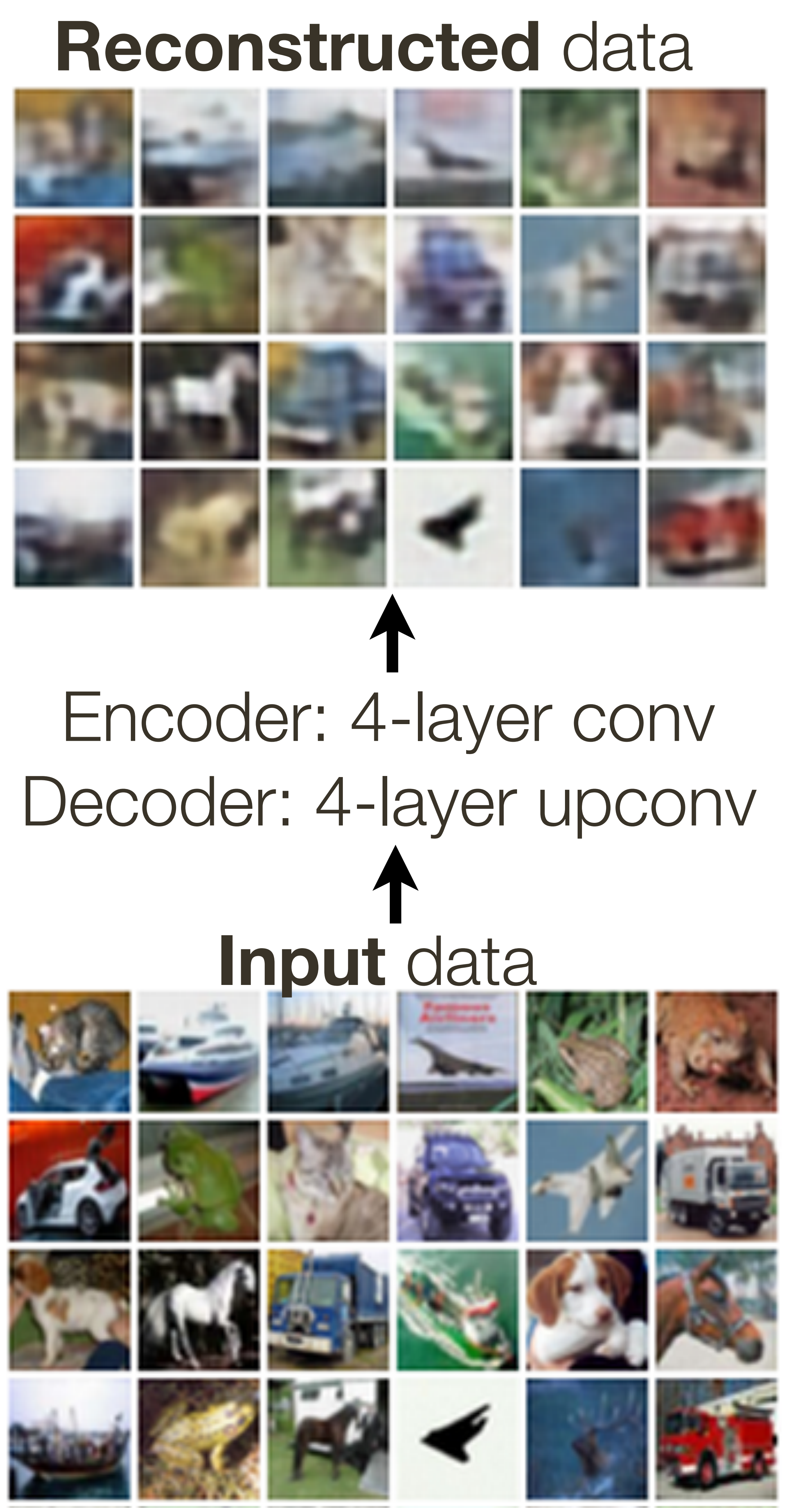
Encoder: 4-layer conv
Decoder: 4-layer upconv

Input data



Autoencoders Reminder …





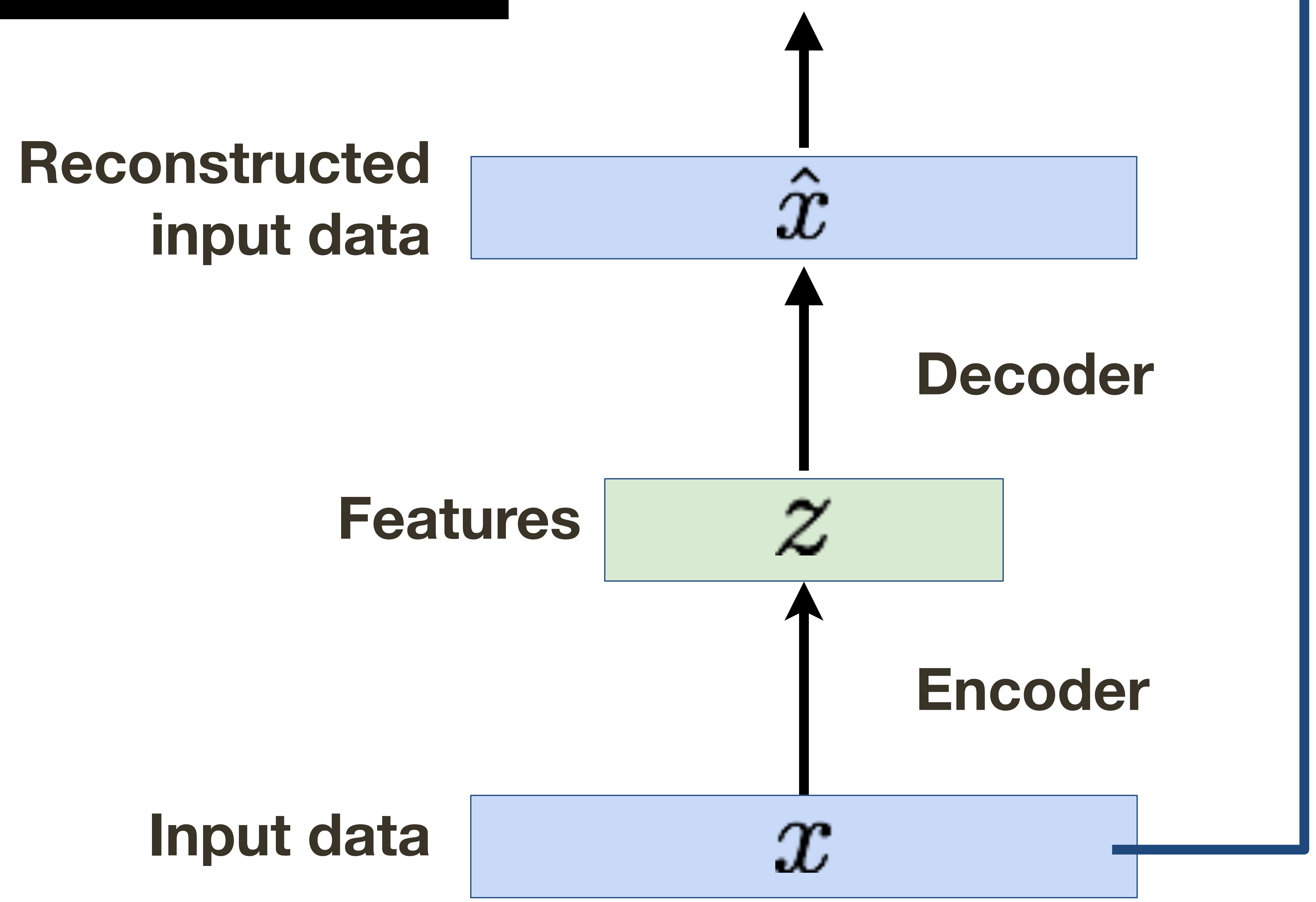
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

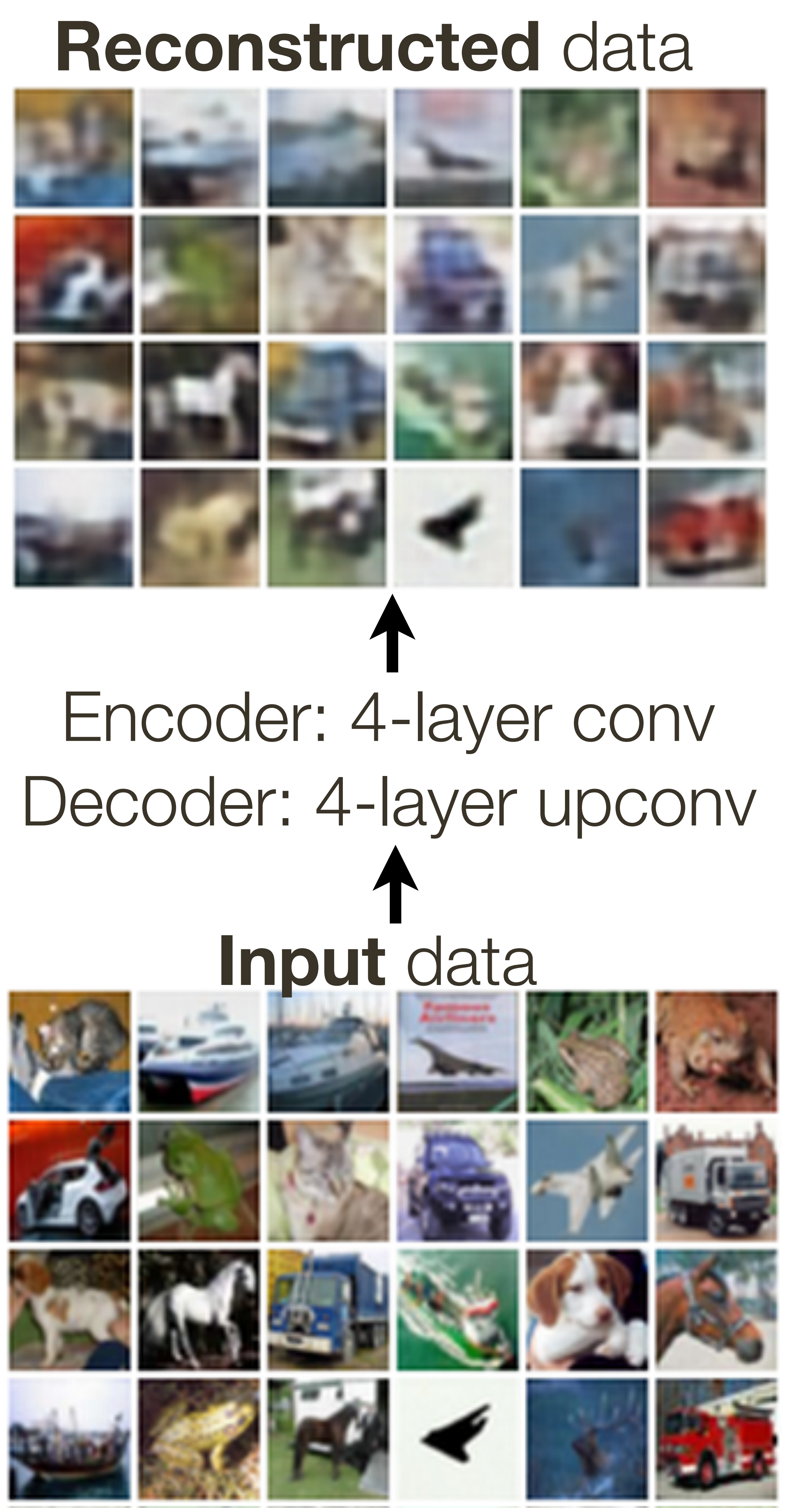
Autoencoders Reminder …

Doesn't use labels!

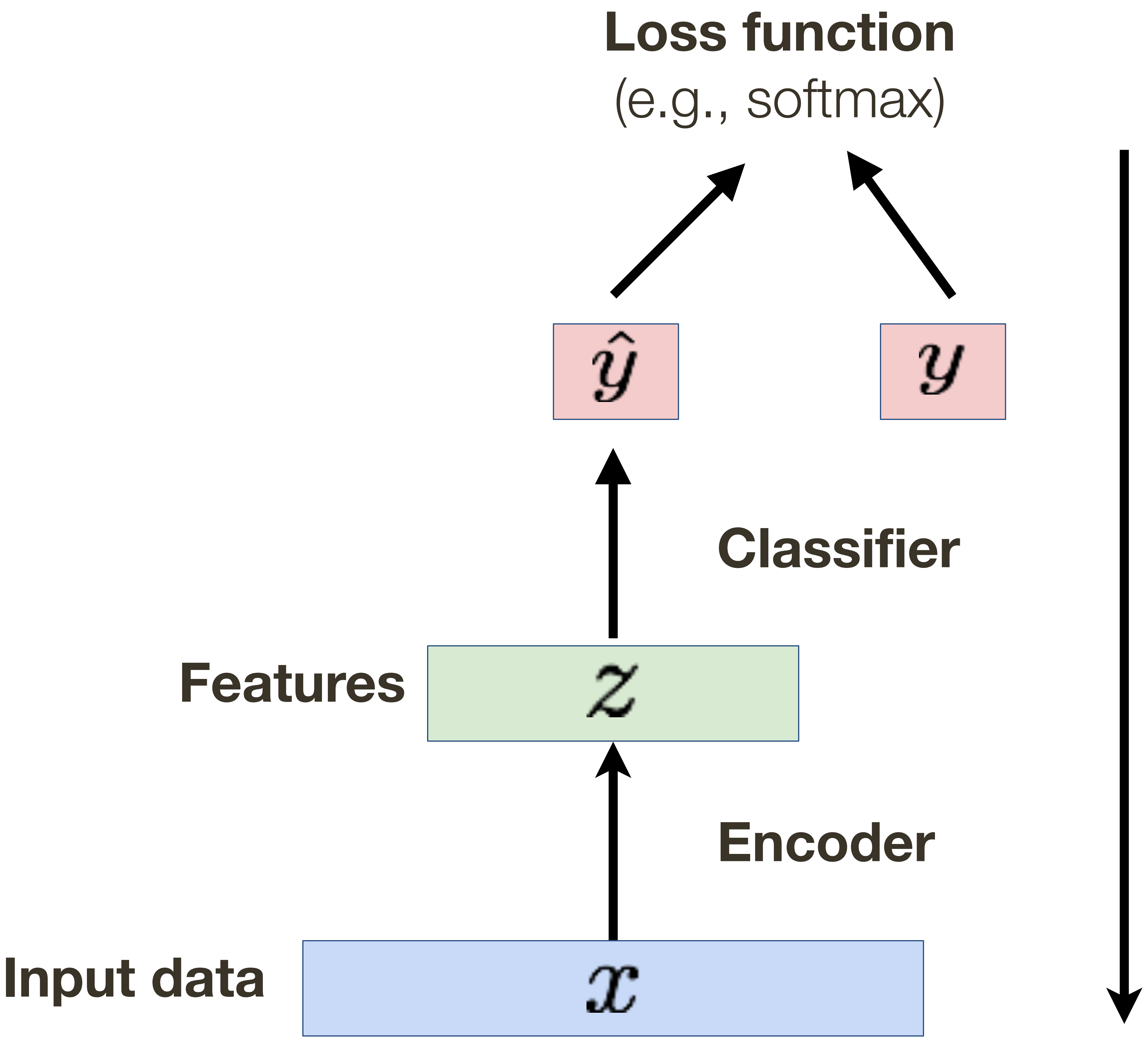
L2 Loss function:

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





Autoencoders Reminder …



Fine-tune encoder jointly with classifier

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

bird plane

dog deer truck

Variational Autoencoders

[Kingma and Welling, 2014]

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

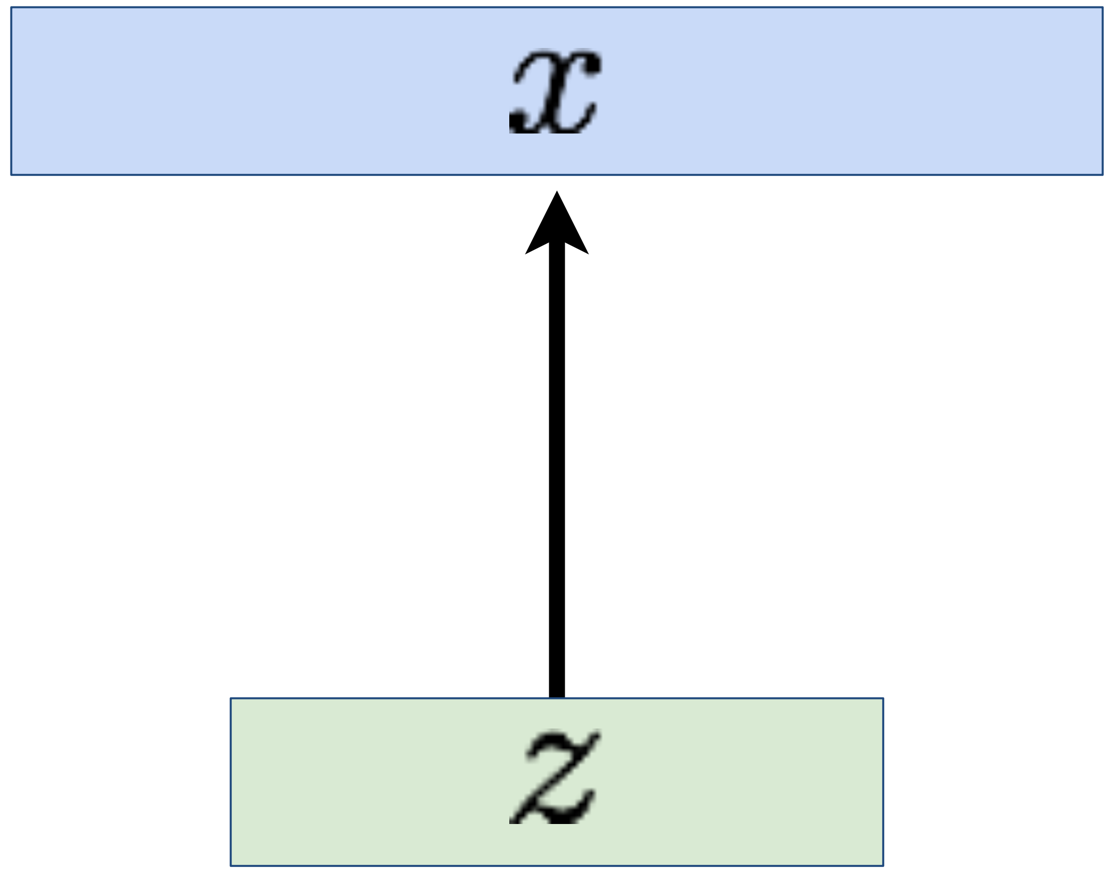
Assume training data is generated from underlying unobserved (latent) representation z

Sample from
true **conditional**

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

Sample from
true **prior**

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



Variational Autoencoders

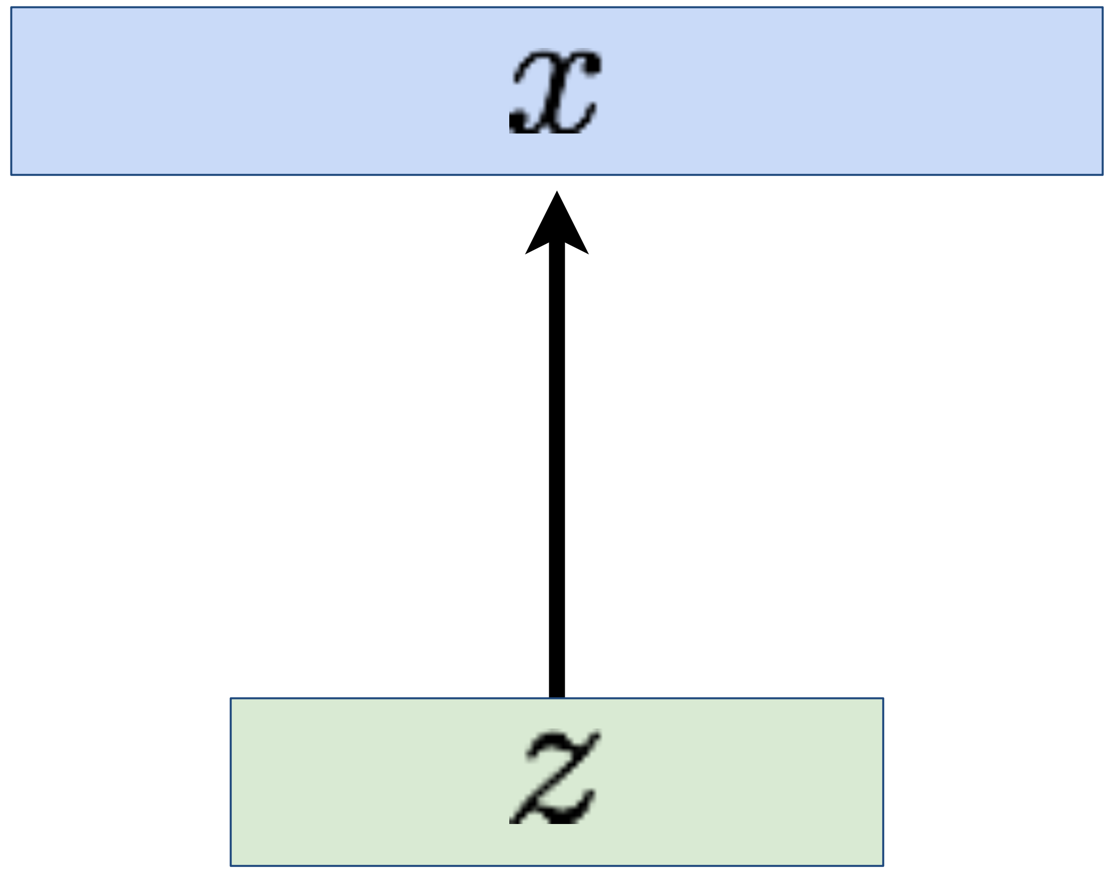
[Kingma and Welling, 2014]

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.*)

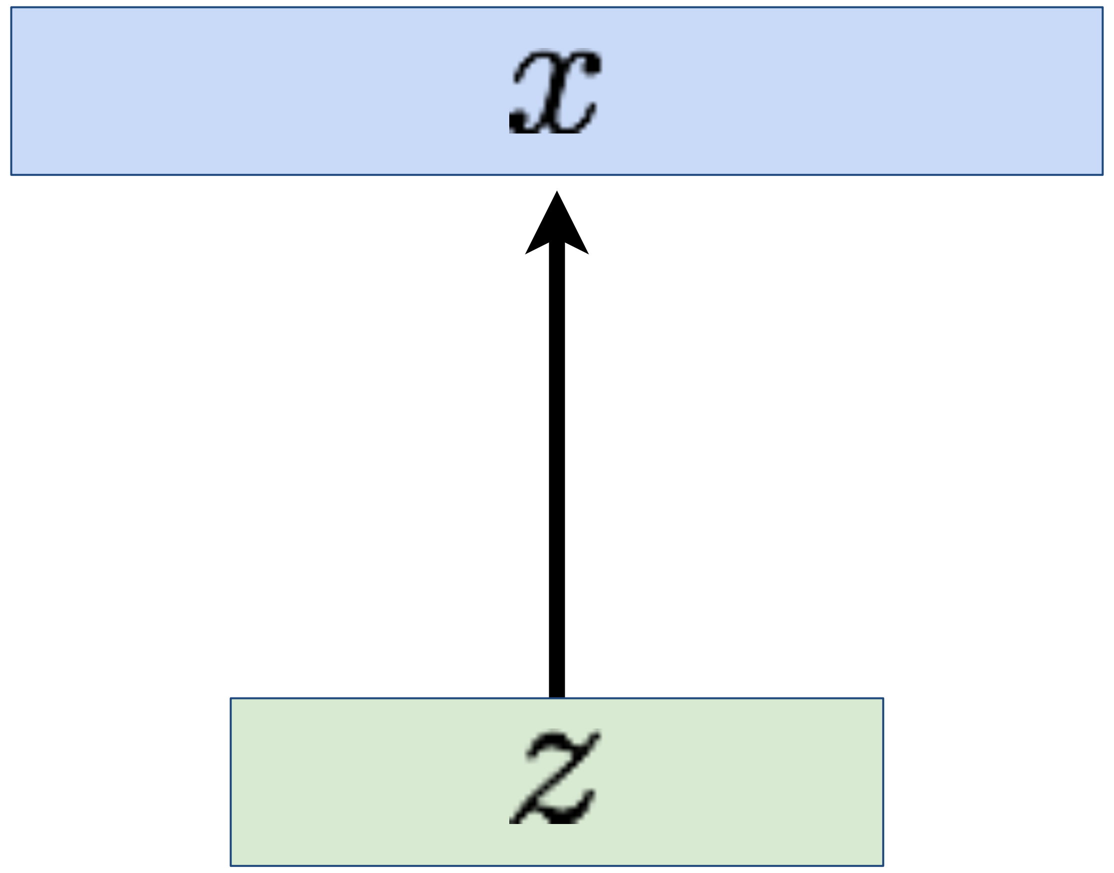
Variational Autoencoders

[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

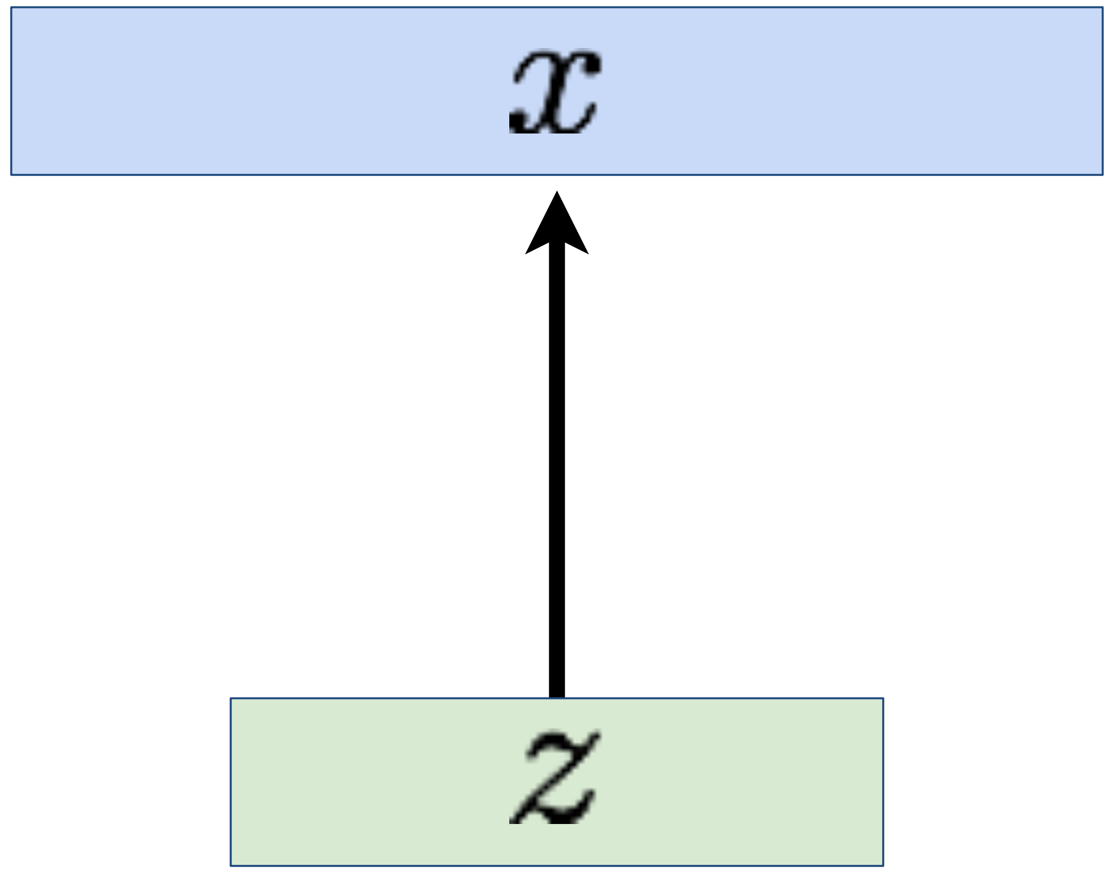
How do we **represent** this model?

Sample from
true **conditional**

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

Sample from
true **prior**

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



* 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

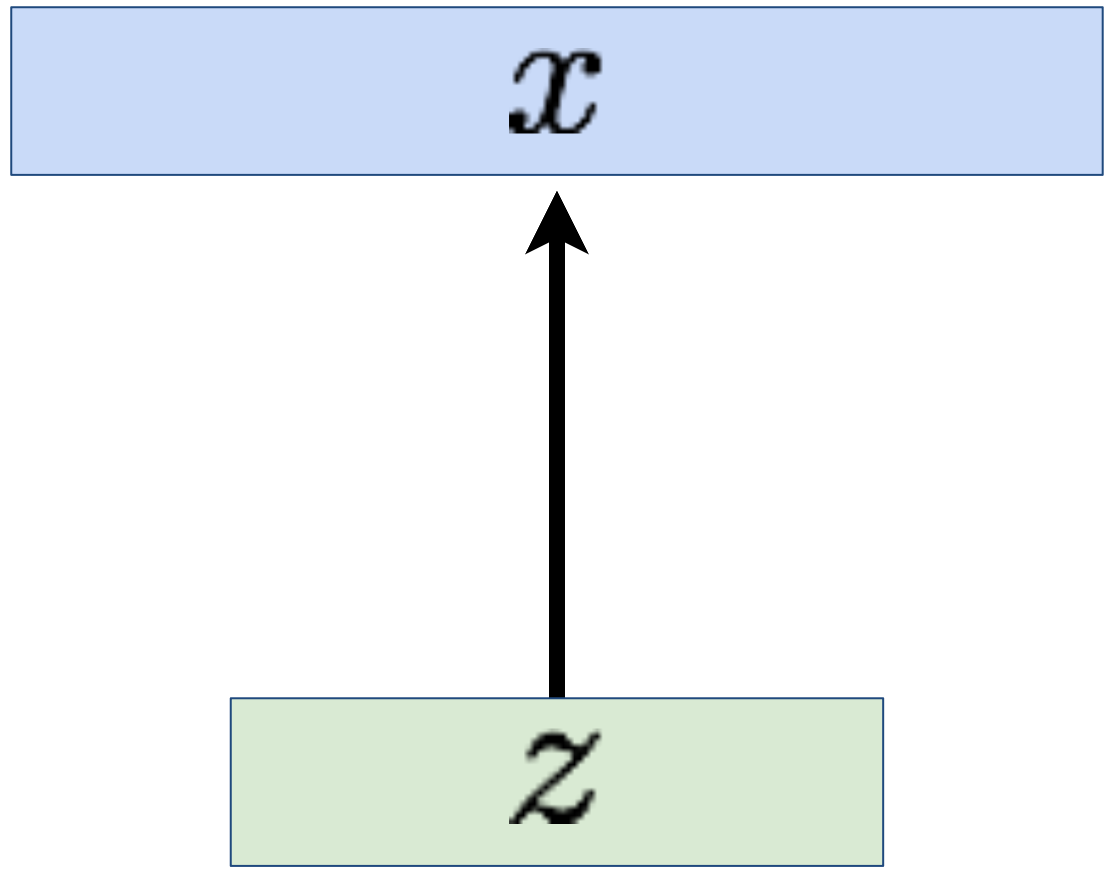
How do we **represent** this model?

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**

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

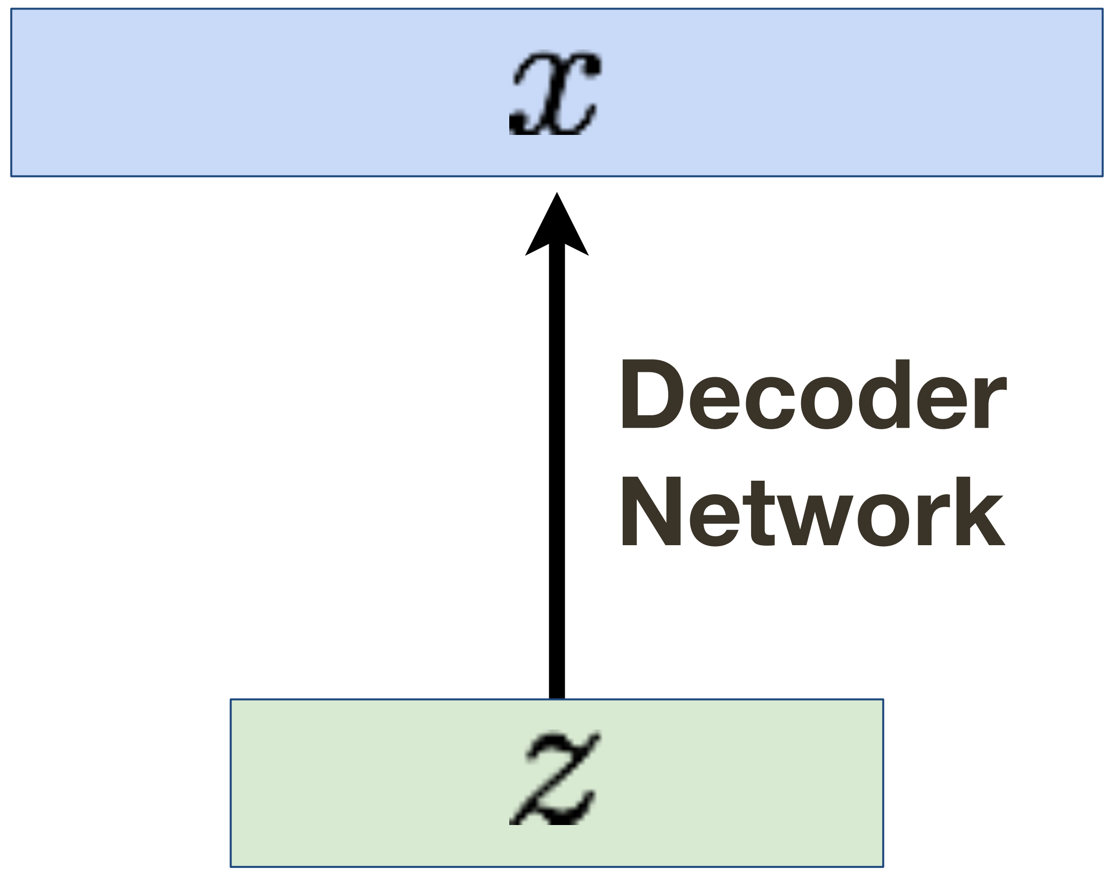
Variational Autoencoders

[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

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

Variational Autoencoders

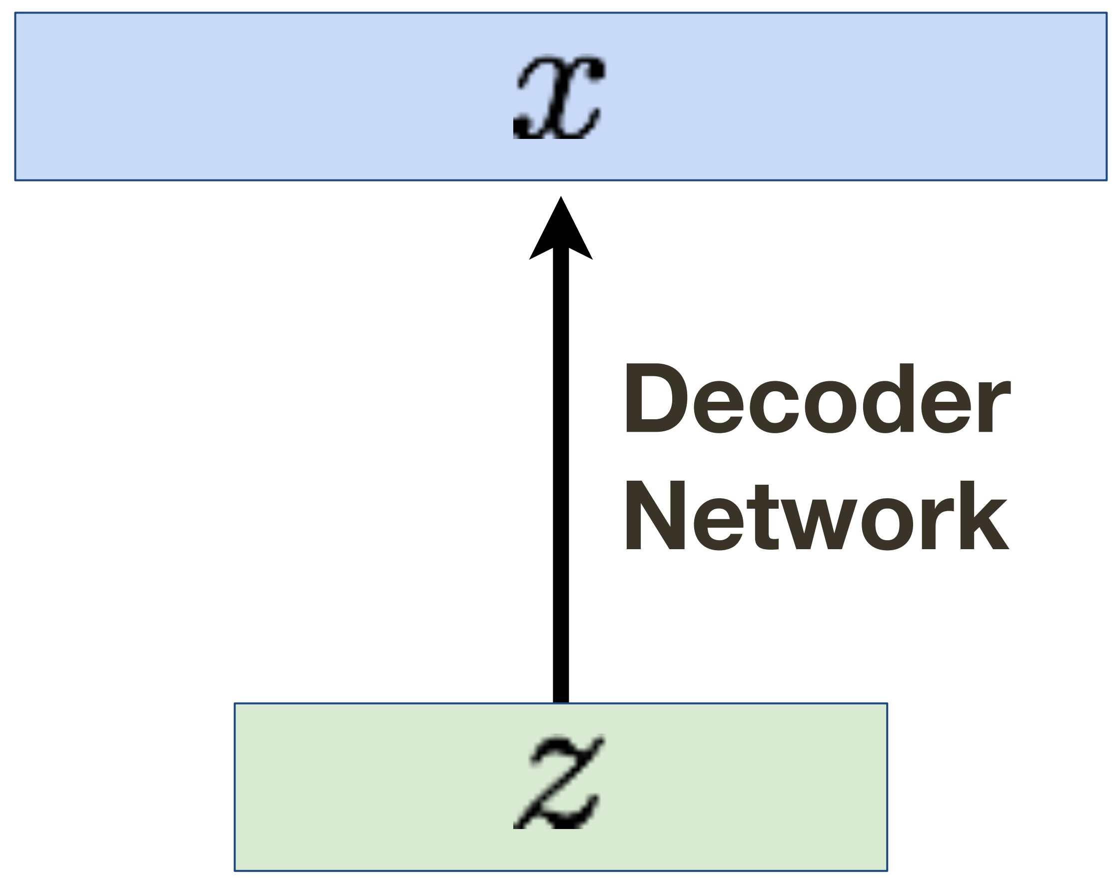
[Kingma and Welling, 2014]

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

How do we **train** this model?

Sample from
true **conditional**

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



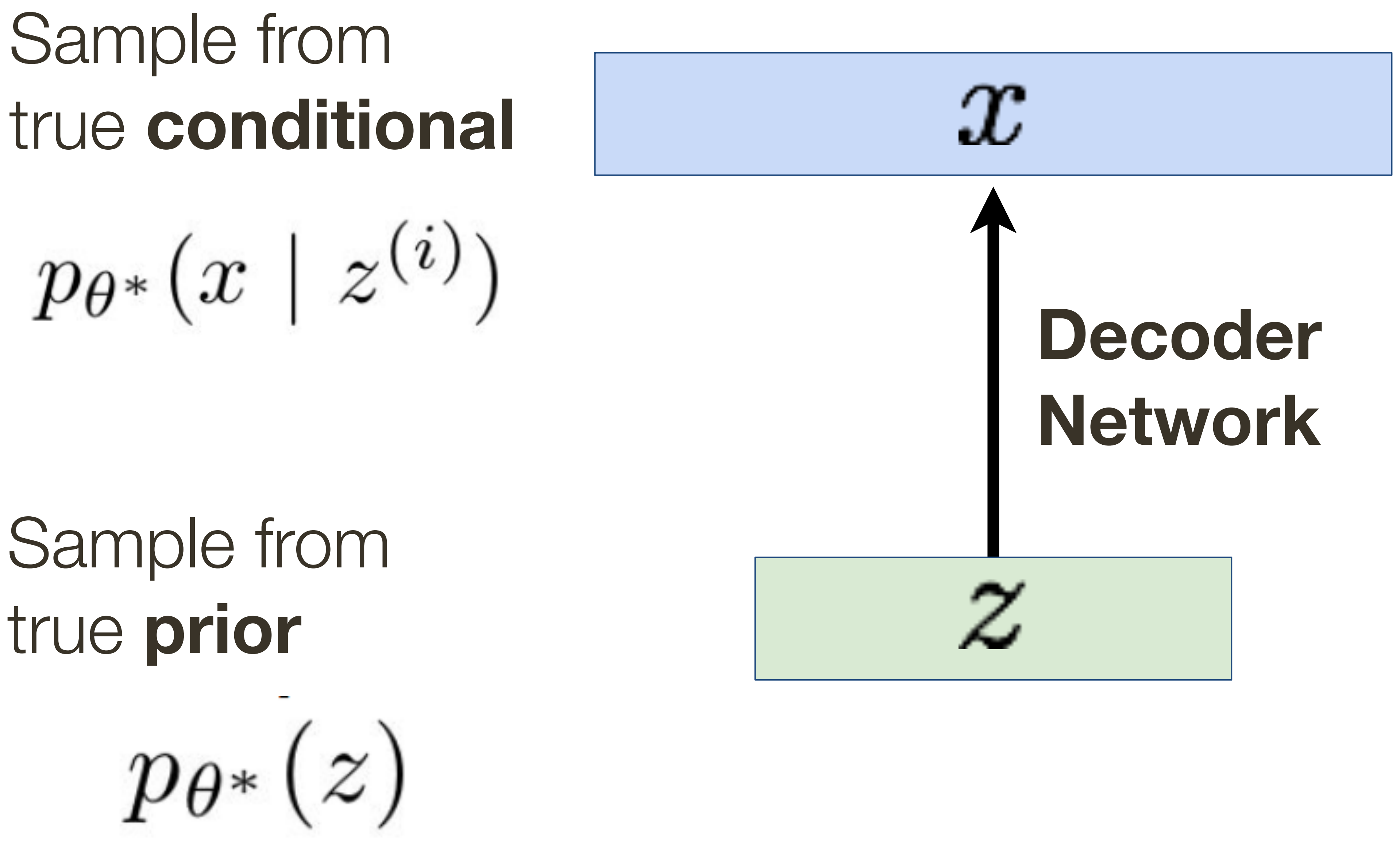
Sample from
true **prior**

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

Variational Autoencoders

[Kingma and Welling, 2014]

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



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)

Variational Autoencoders

[Kingma and Welling, 2014]

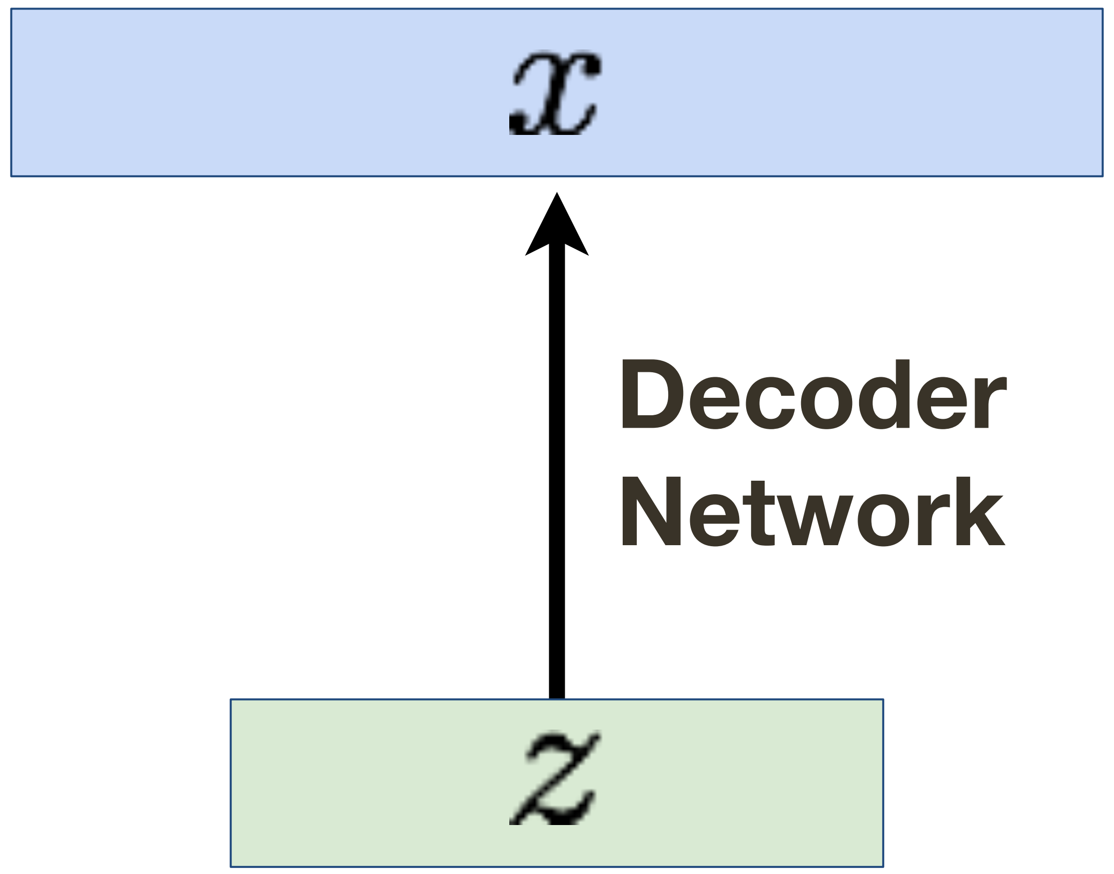
We want to **estimate the true parameters** θ^* of this generative model

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?

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

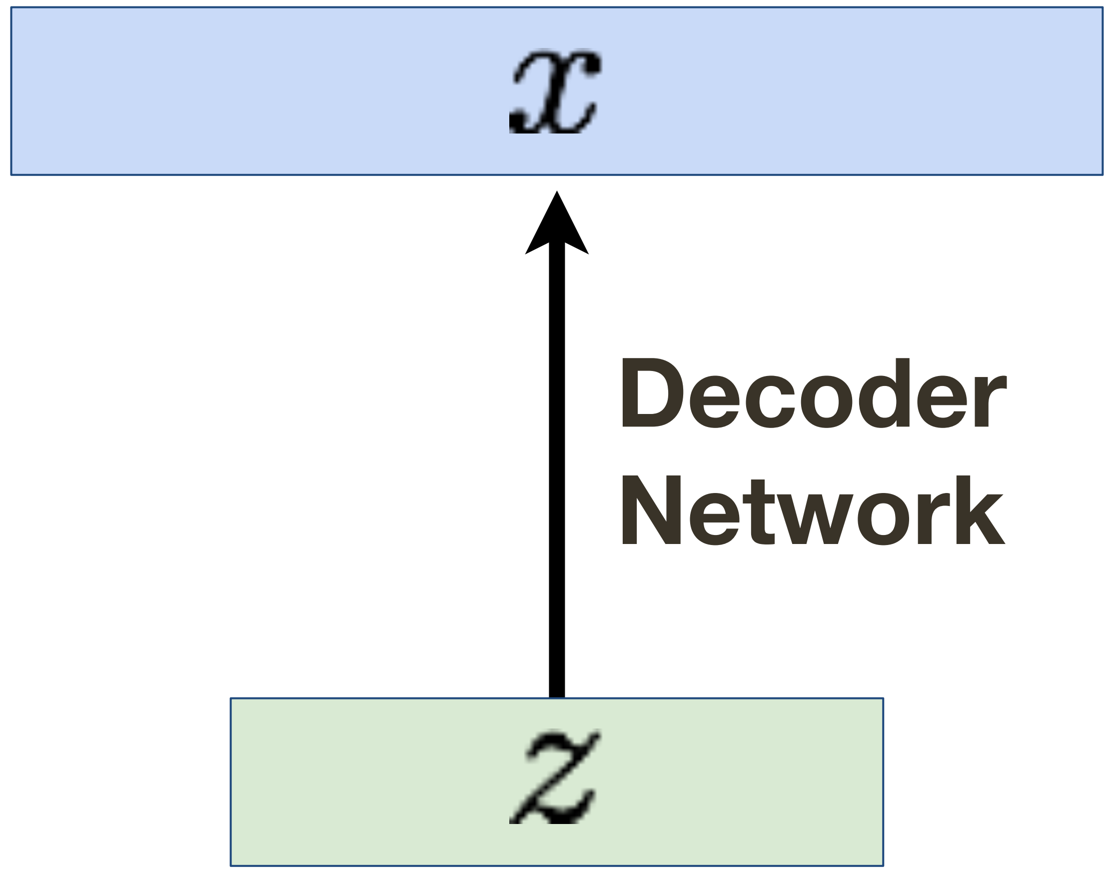
Variational Autoencoders

[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)

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

Intractability in Variational Autoencoder

[Kingma and Welling, 2014]

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

Decoder Neural Network 🙂

🙂 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: $p_\theta(z|x) = p_\theta(x|z)p_\theta(z)/p_\theta(x)$

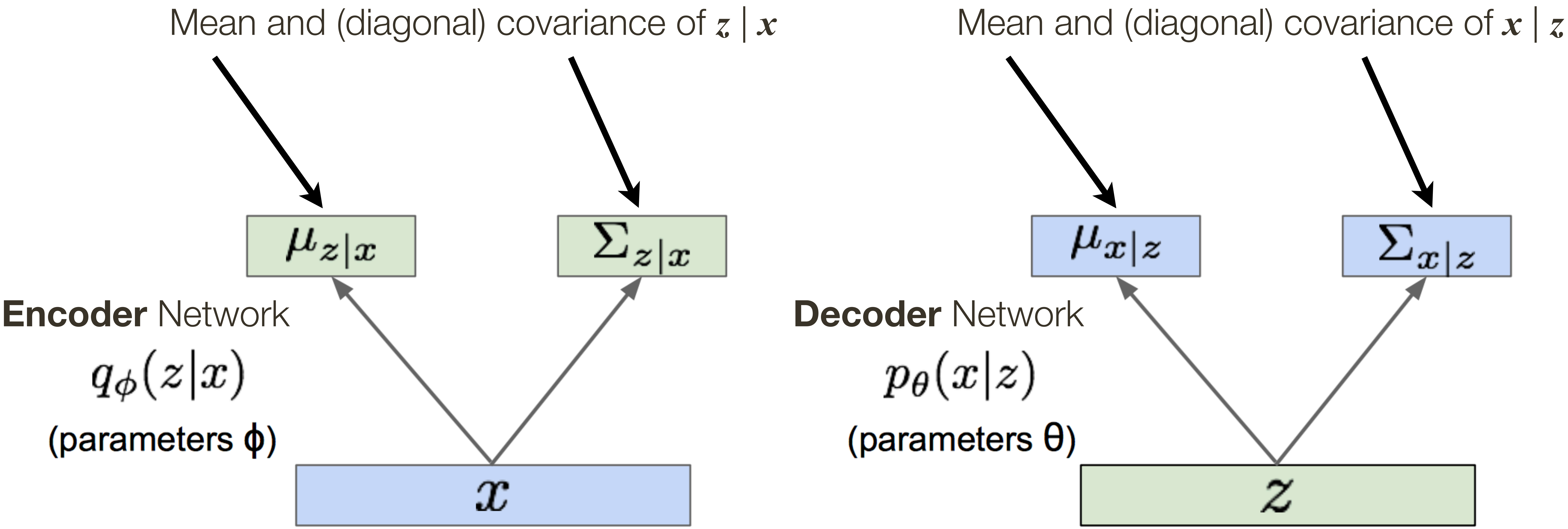
Solution: In addition to decoder network modeling $p_\theta(x|z)$, define additional encoder network $q_\phi(z|x)$ that approximates $p_\theta(z|x)$

— 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)

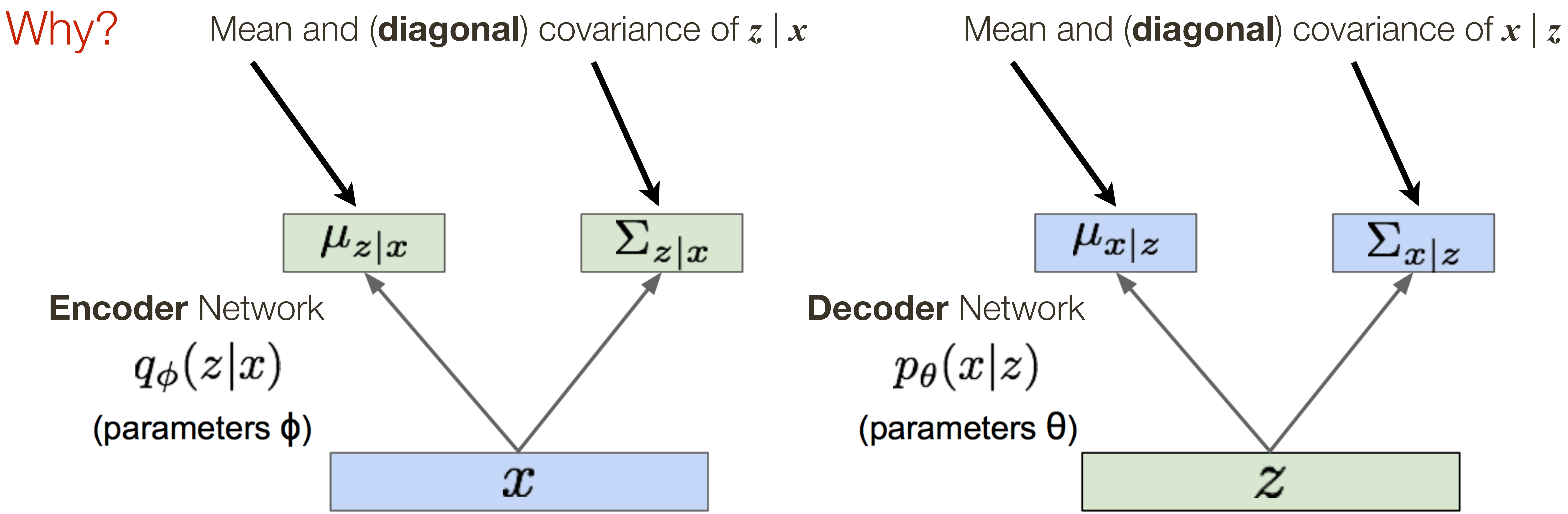


* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder

[Kingma and Welling, 2014]

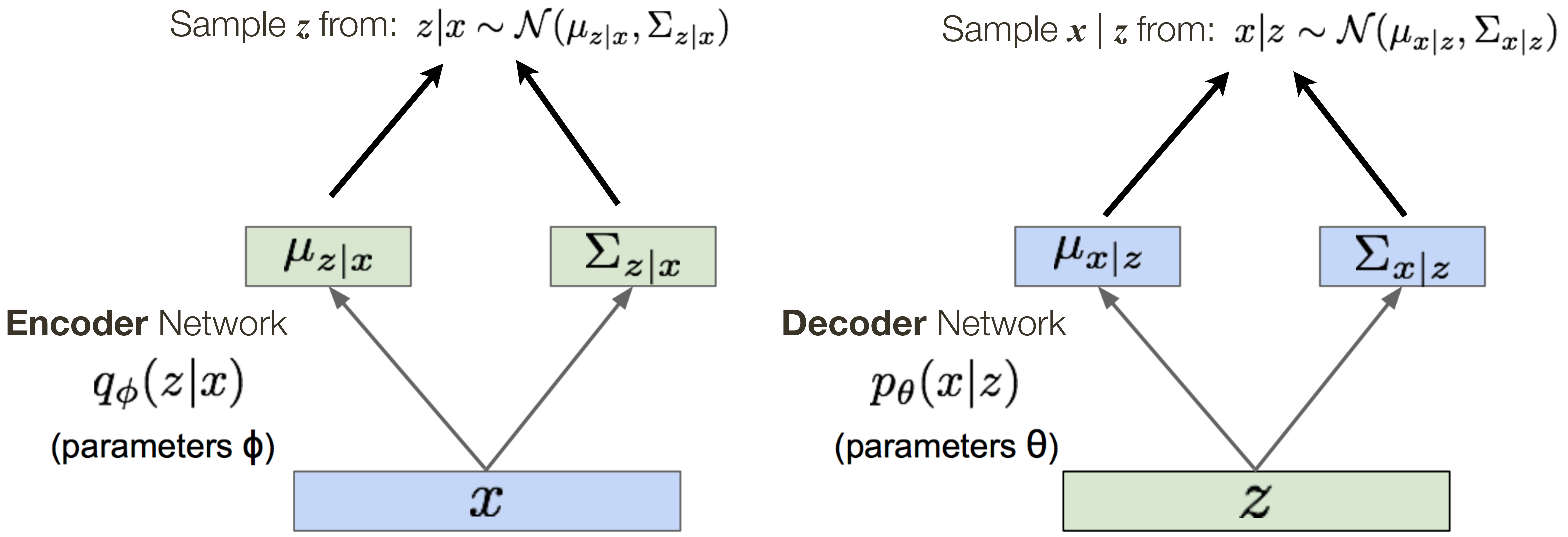
Since we are modeling probabilistic generation of data, encoder and decoder networks are probabilistic (they model distributions)



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]

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:

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

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

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

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

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

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:

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

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

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

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

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

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:

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

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

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

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

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

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:

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

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

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

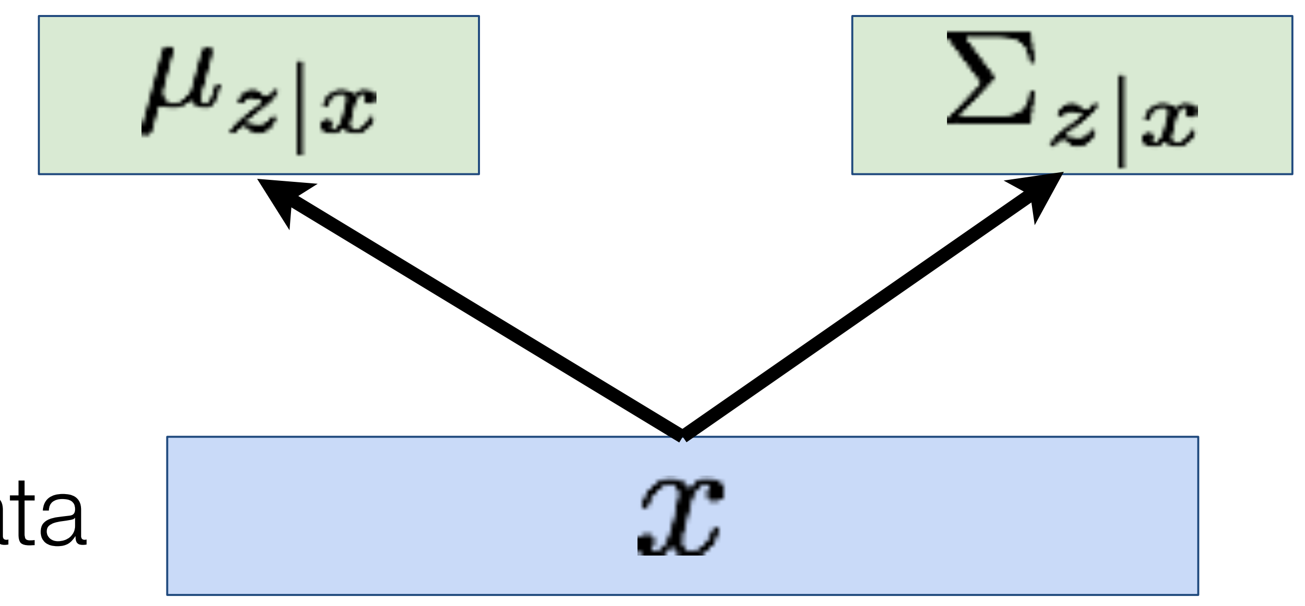
Putting it all together:
maximizing the likelihood lower bound

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

Encoder network

$q_\phi(z|x)$

Input Data



Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

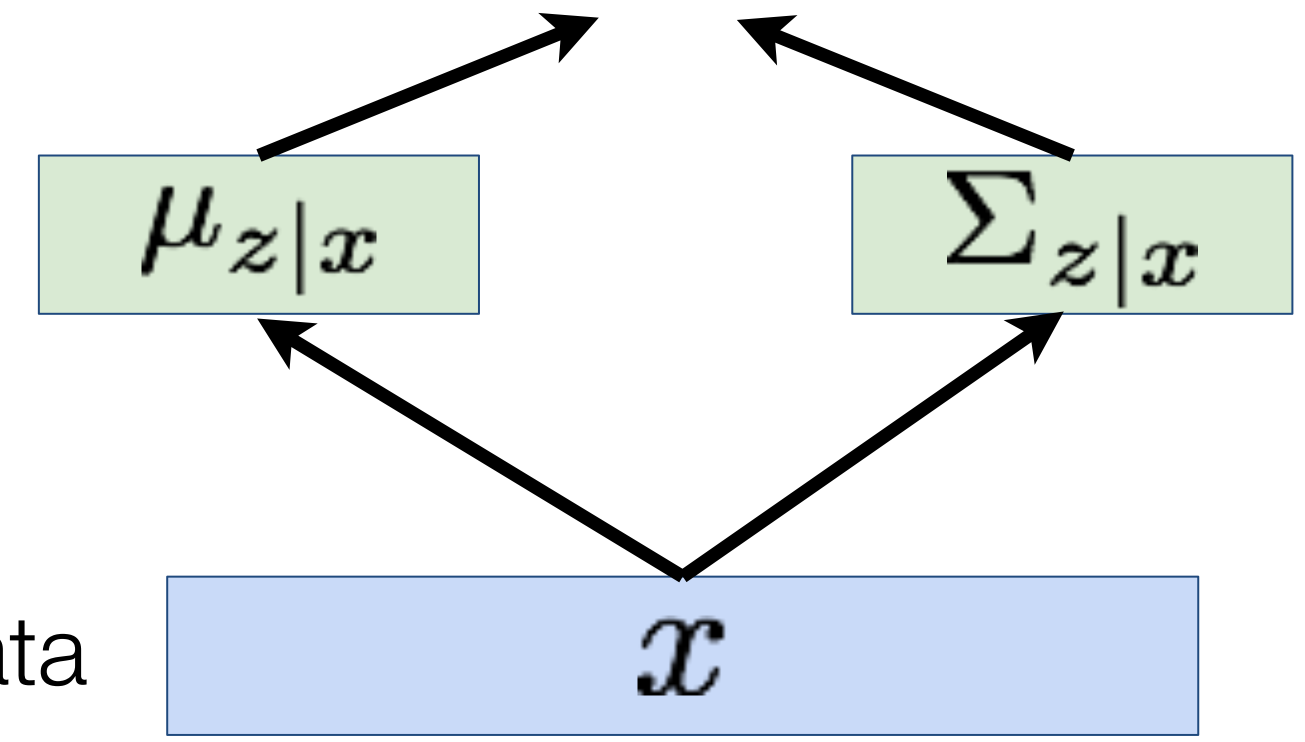
$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)}, \theta, \phi)}$$

Make approximate posterior distribution close to prior

Encoder network

$q_\phi(z|x)$

Input Data

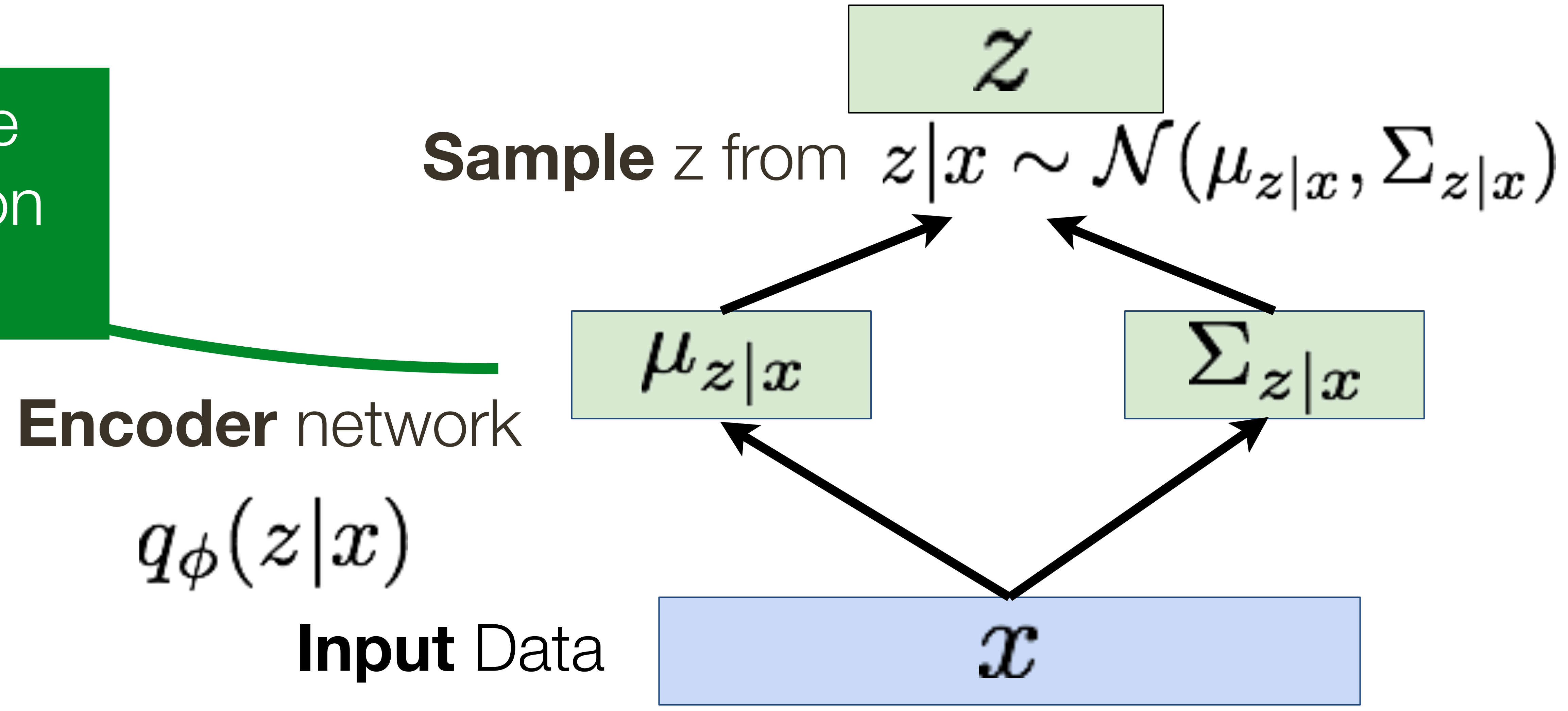


Variational Autoencoder: Learning

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

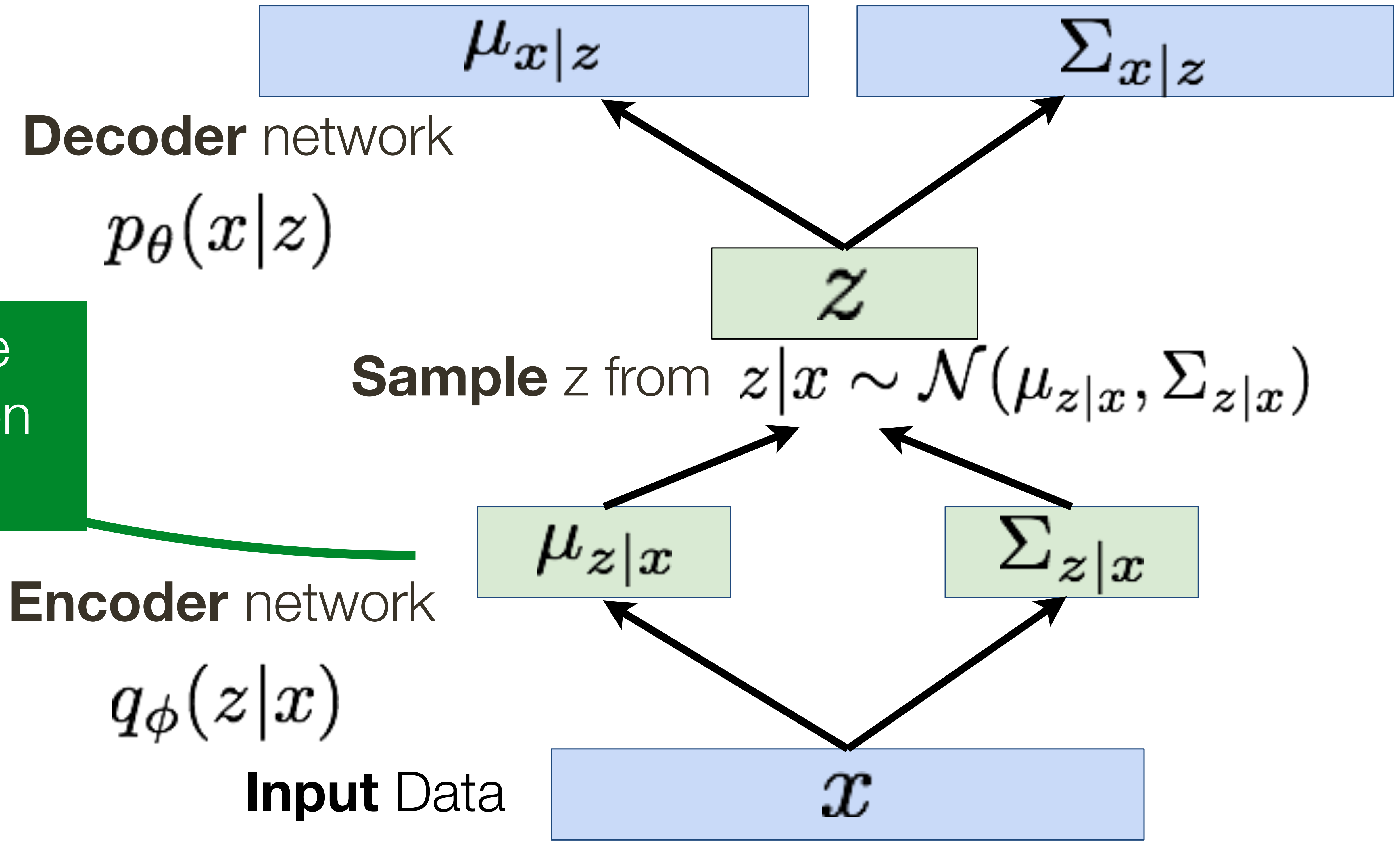


Variational Autoencoder: Learning

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**

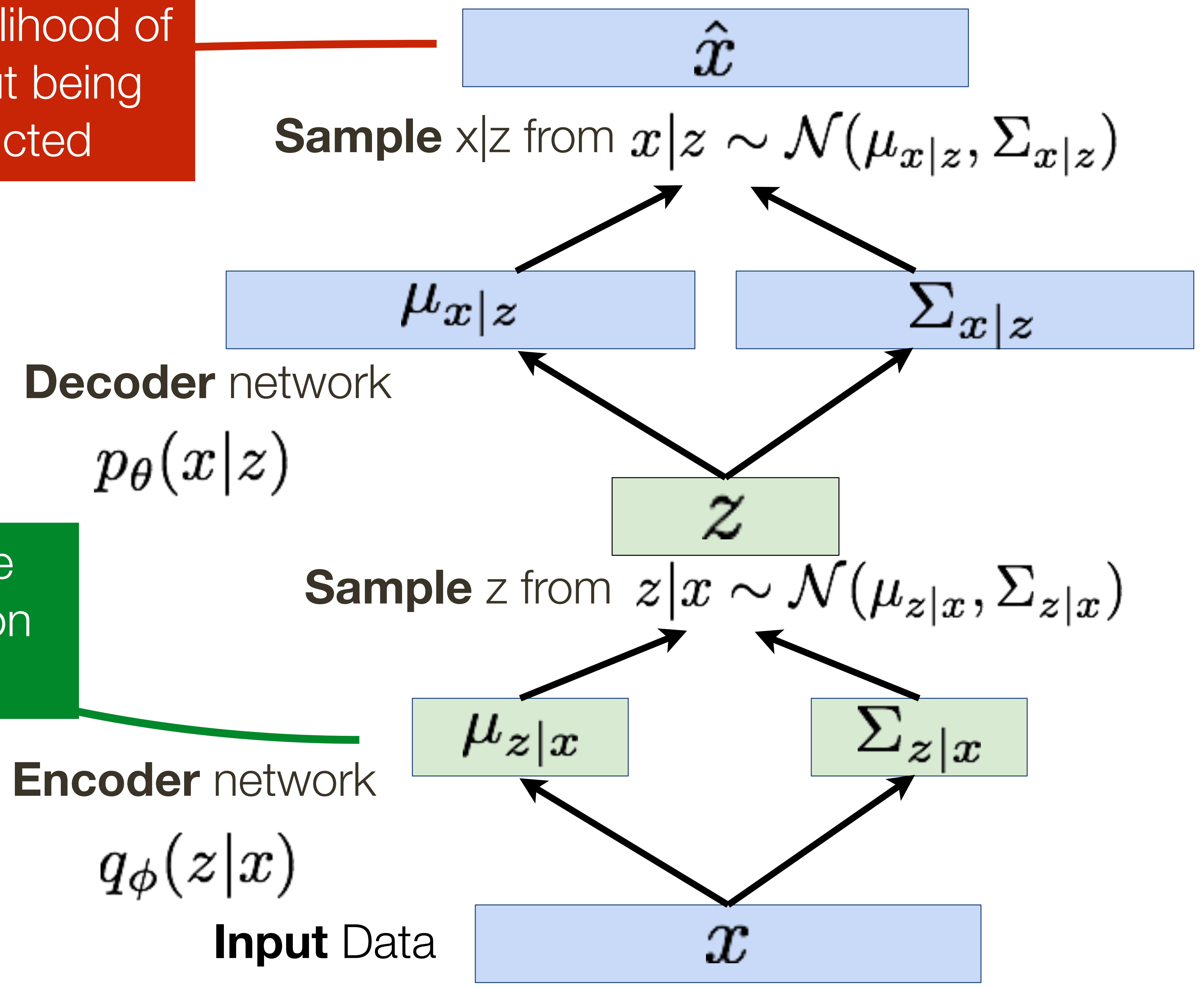
Variational Autoencoder: Learning

Putting it all together: maximizing the likelihood lower bound

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



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

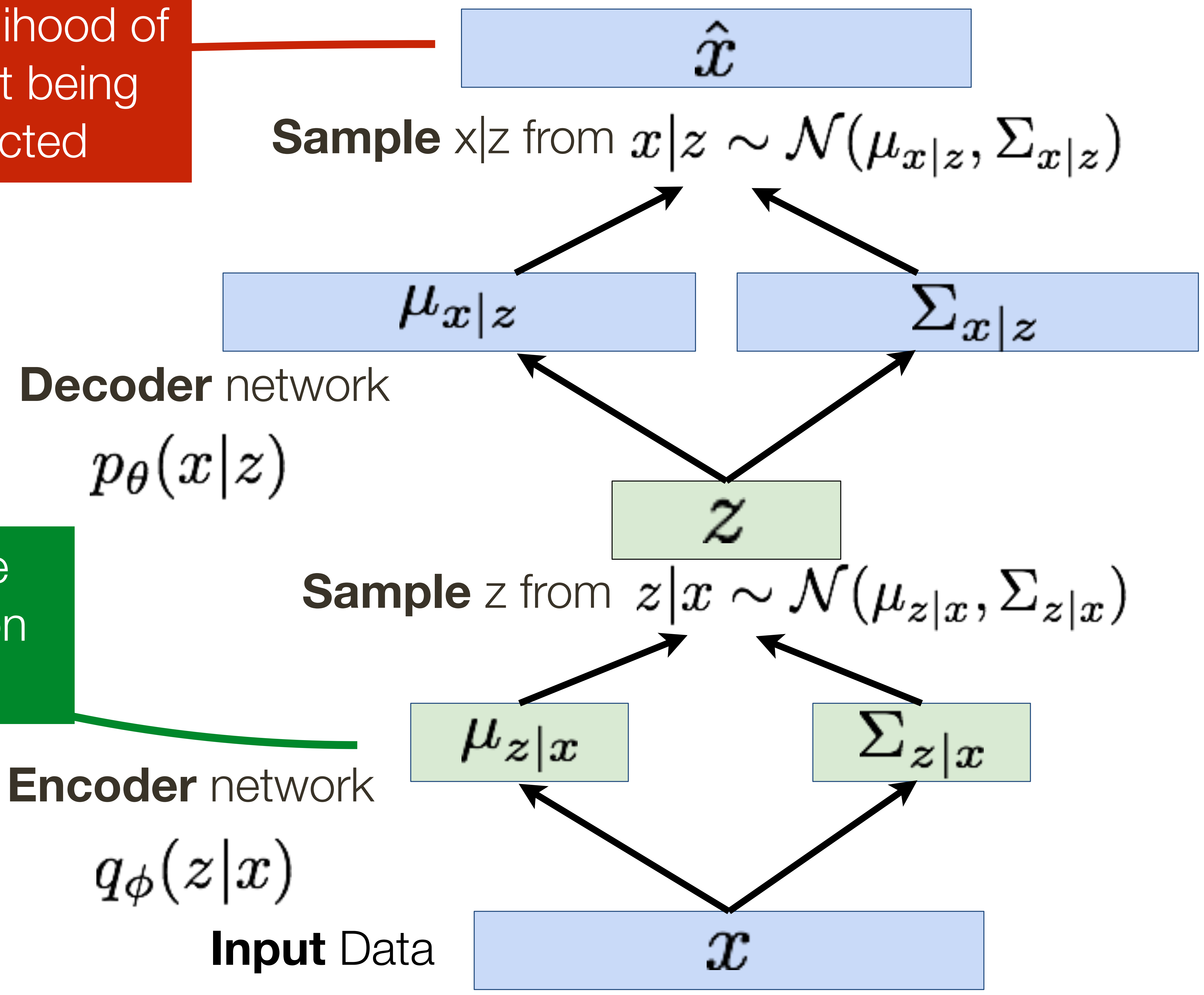
Variational Autoencoder: Learning

Putting it all together: maximizing the likelihood lower bound

Maximize likelihood of original input being reconstructed

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

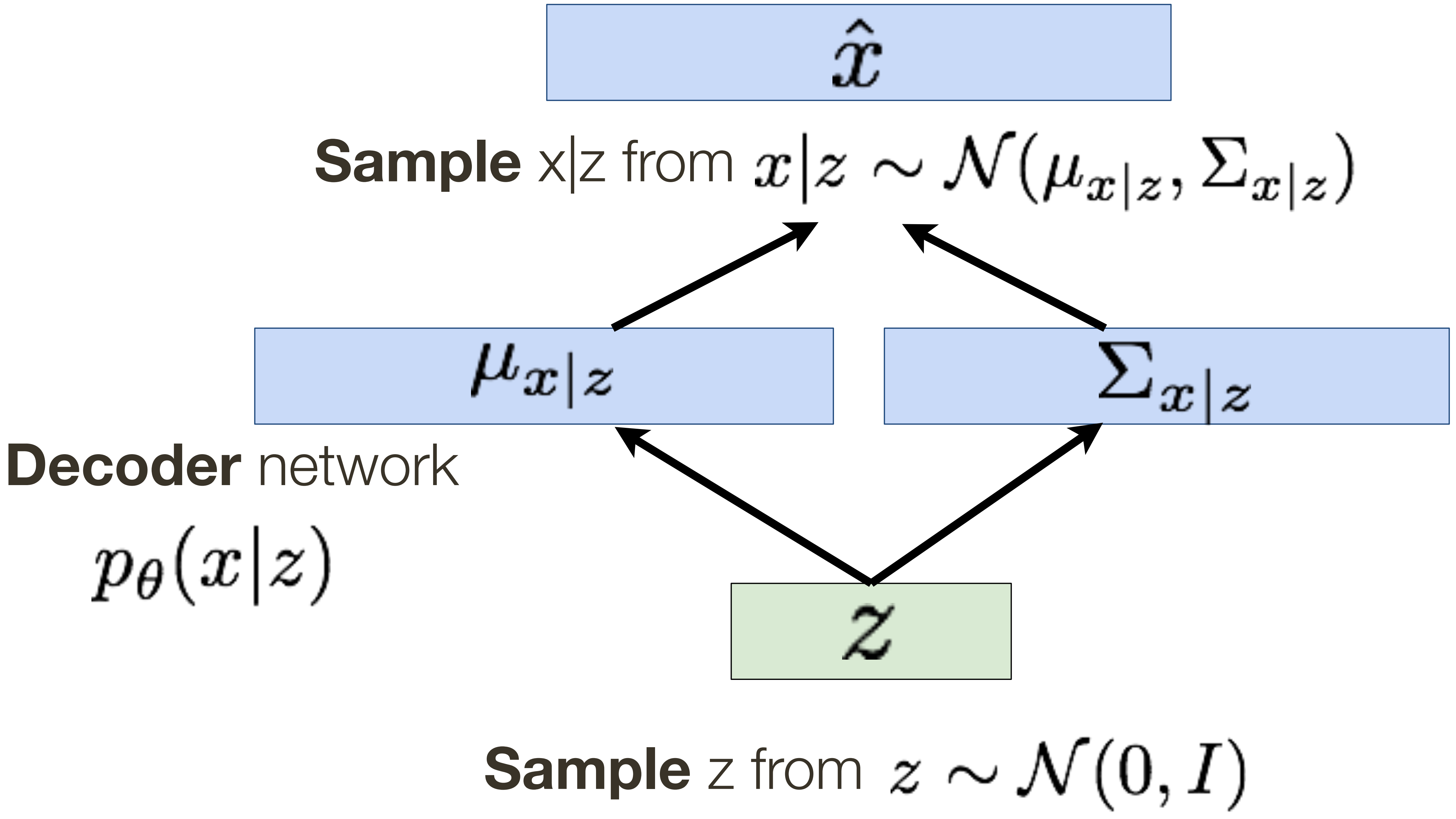
Make approximate posterior distribution close to prior



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

Variational Autoencoder: Generating Data

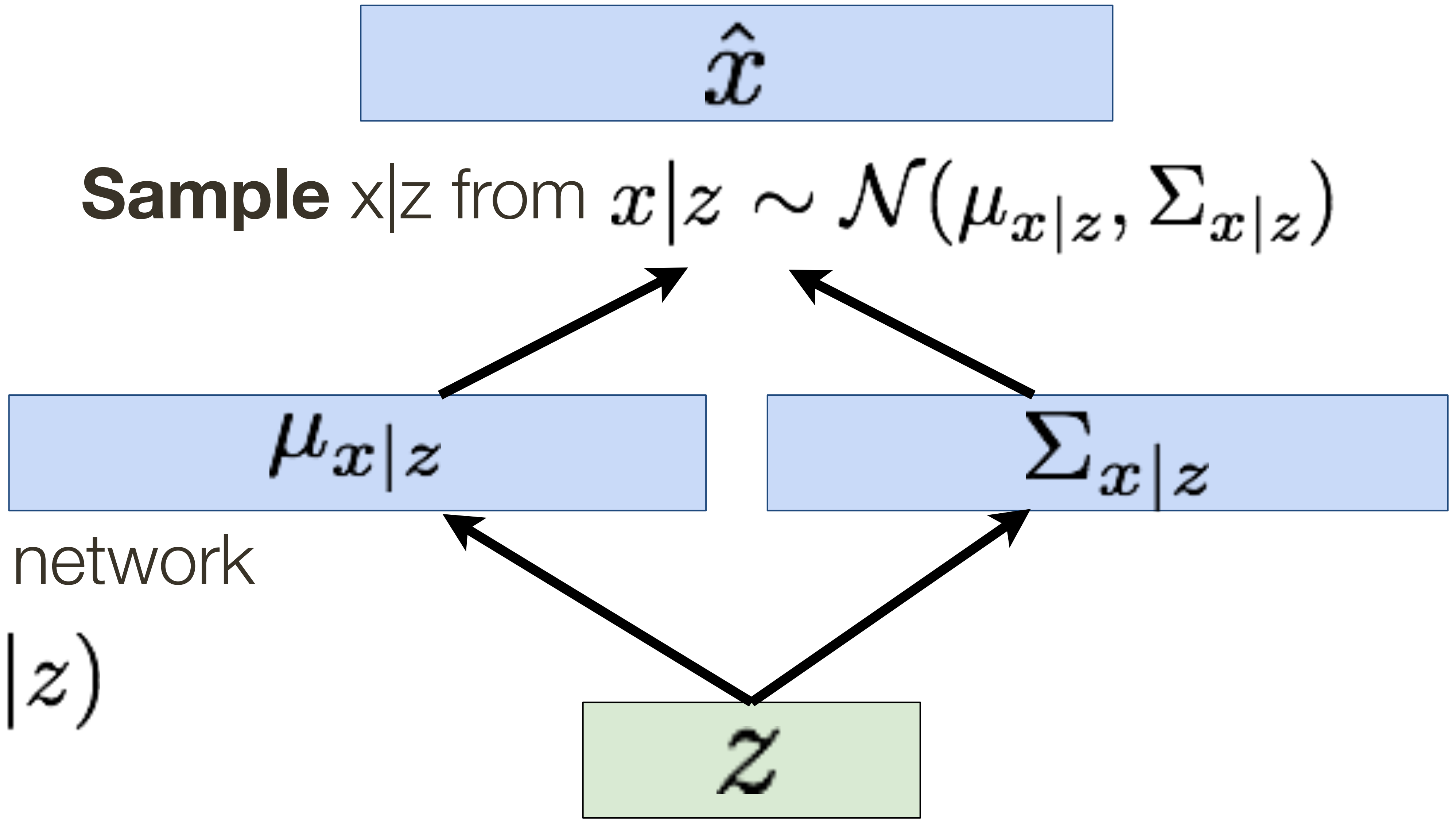
Use decoder network and sample z from **prior**



Variational Autoencoder: Generating Data

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

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

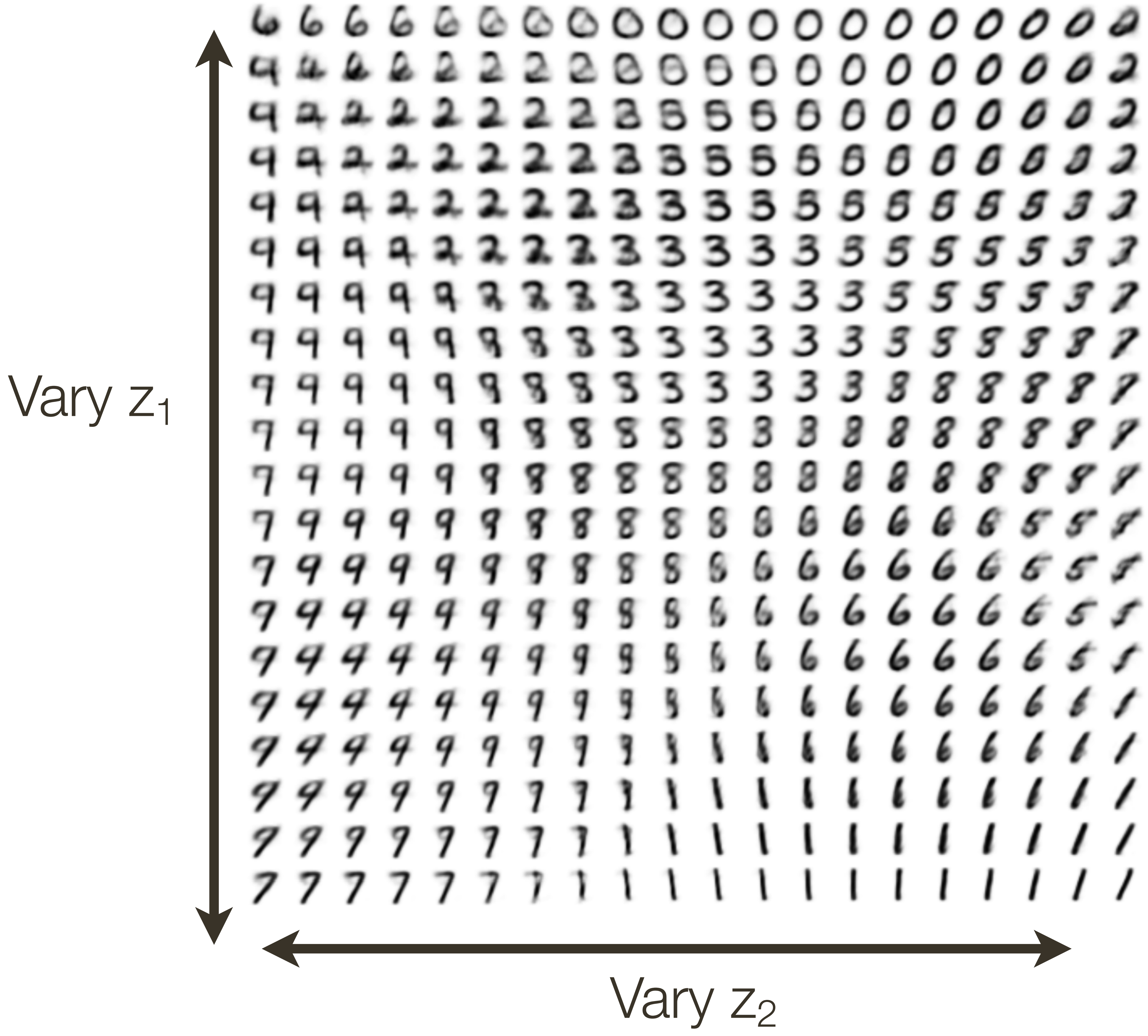


Decoder network

$p_\theta(x|z)$

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

Data manifold for 2-d z



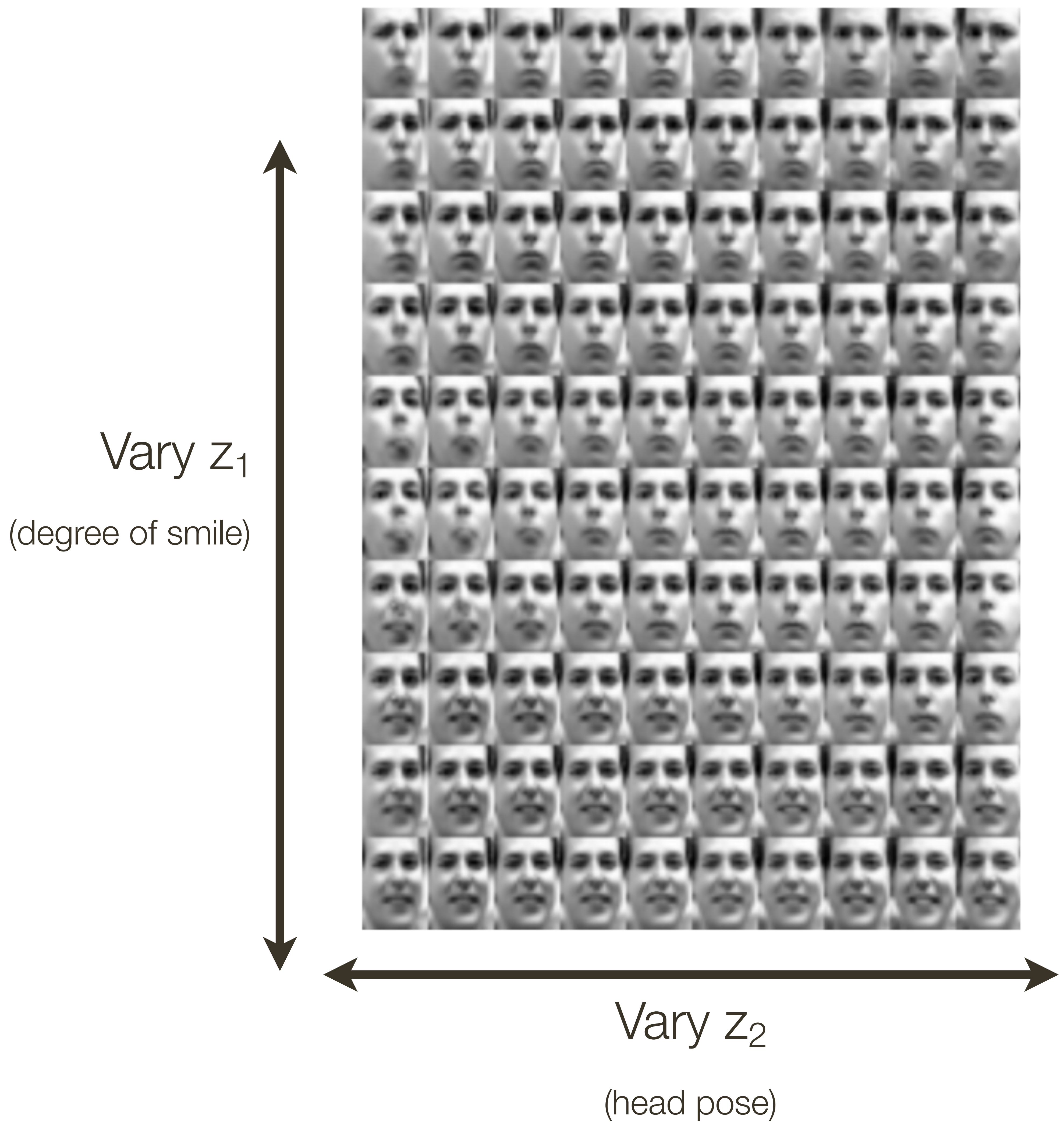
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

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**

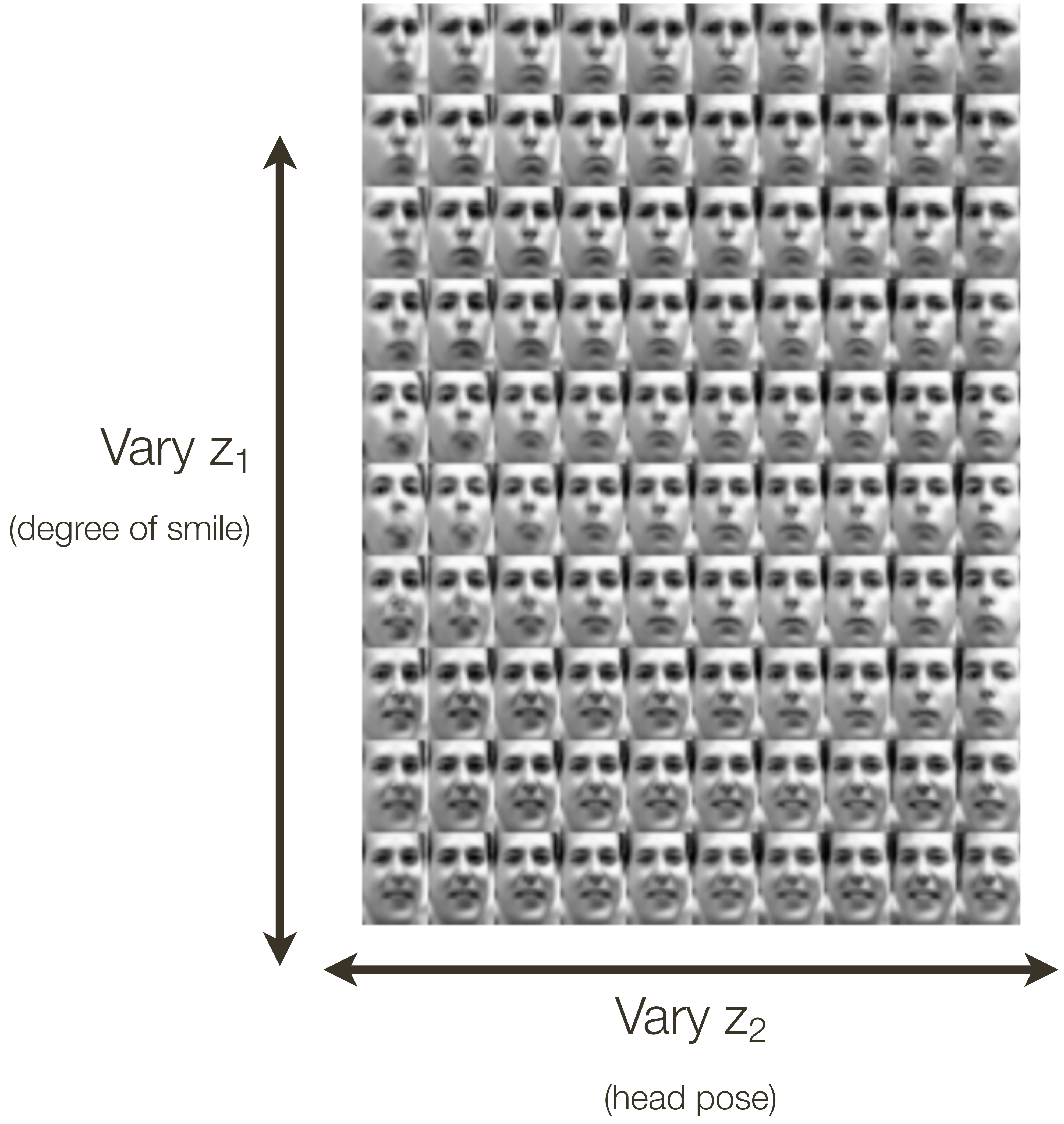
Variational Autoencoder: Generating Data

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

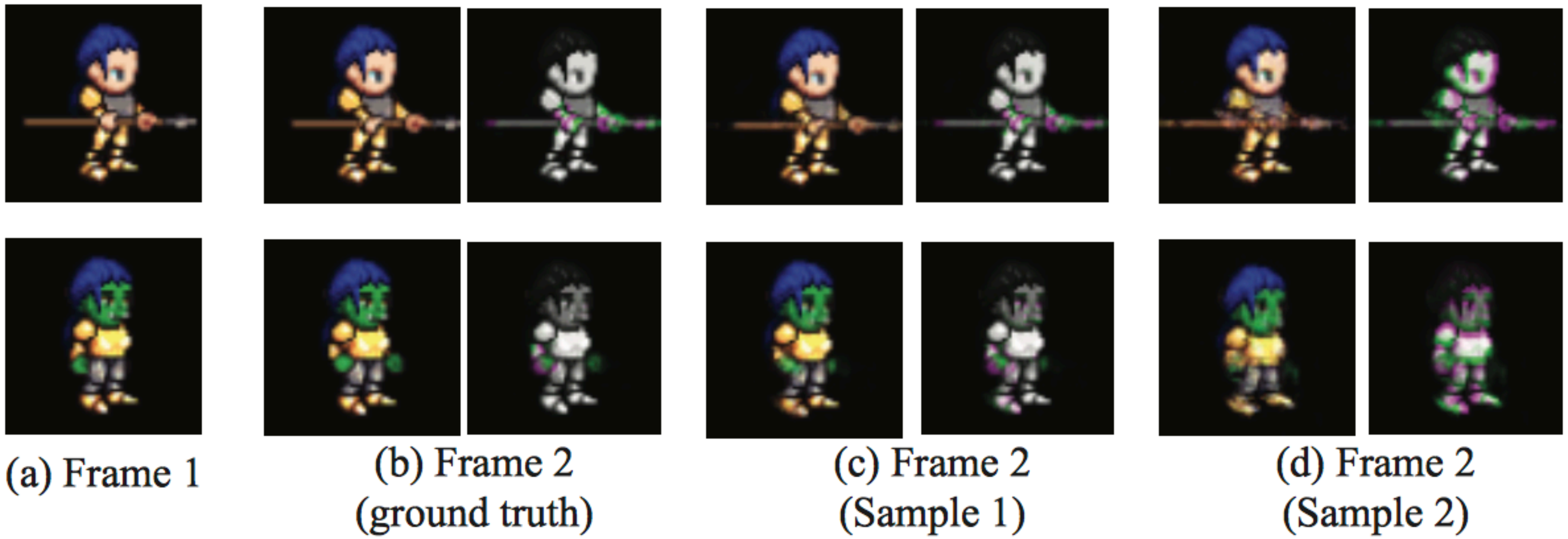
32x32 CIFAR-10

Labeled Faces in the Wild

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

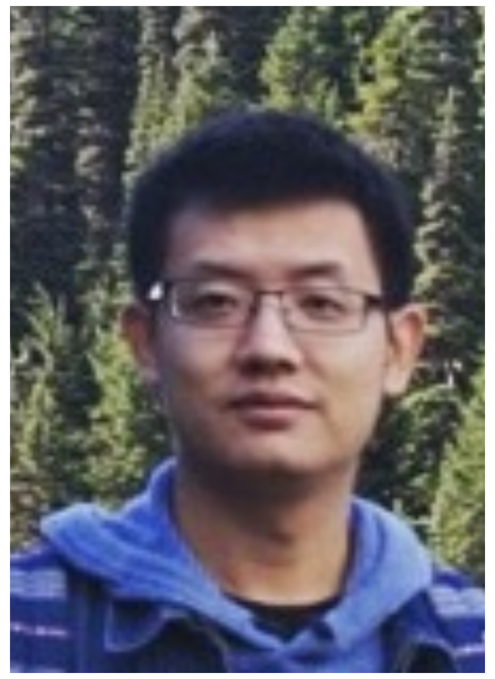
Conditional VAEs

[Xue et al., 2016]



ECCV 2018

Probabilistic Video Generation using Holistic Attribute Control

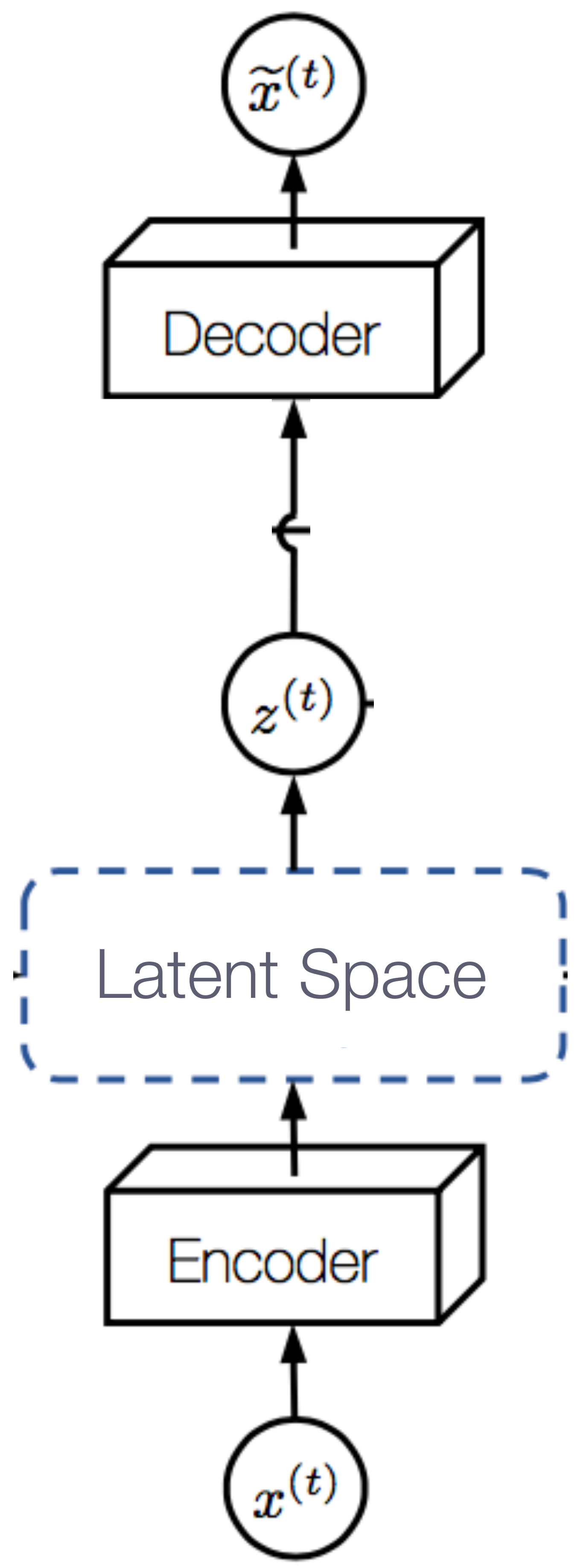




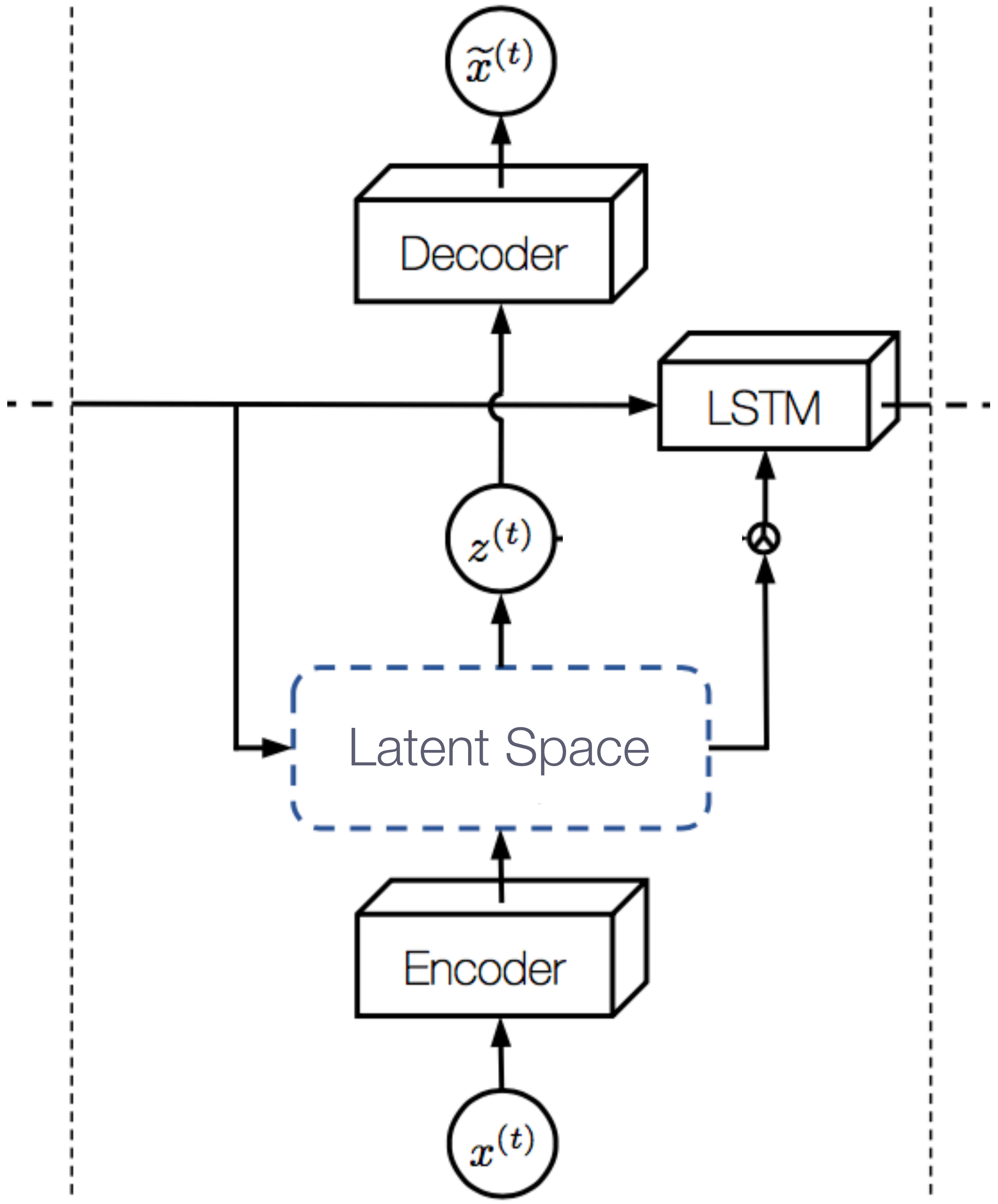
Jiawei (Eric) He | Andreas Lehrmann | Joe Marino | Greg Mori | Leonid Sigal

SFU | Disney Research, oculus | California Institute of Technology 1891 | SFU | UBC

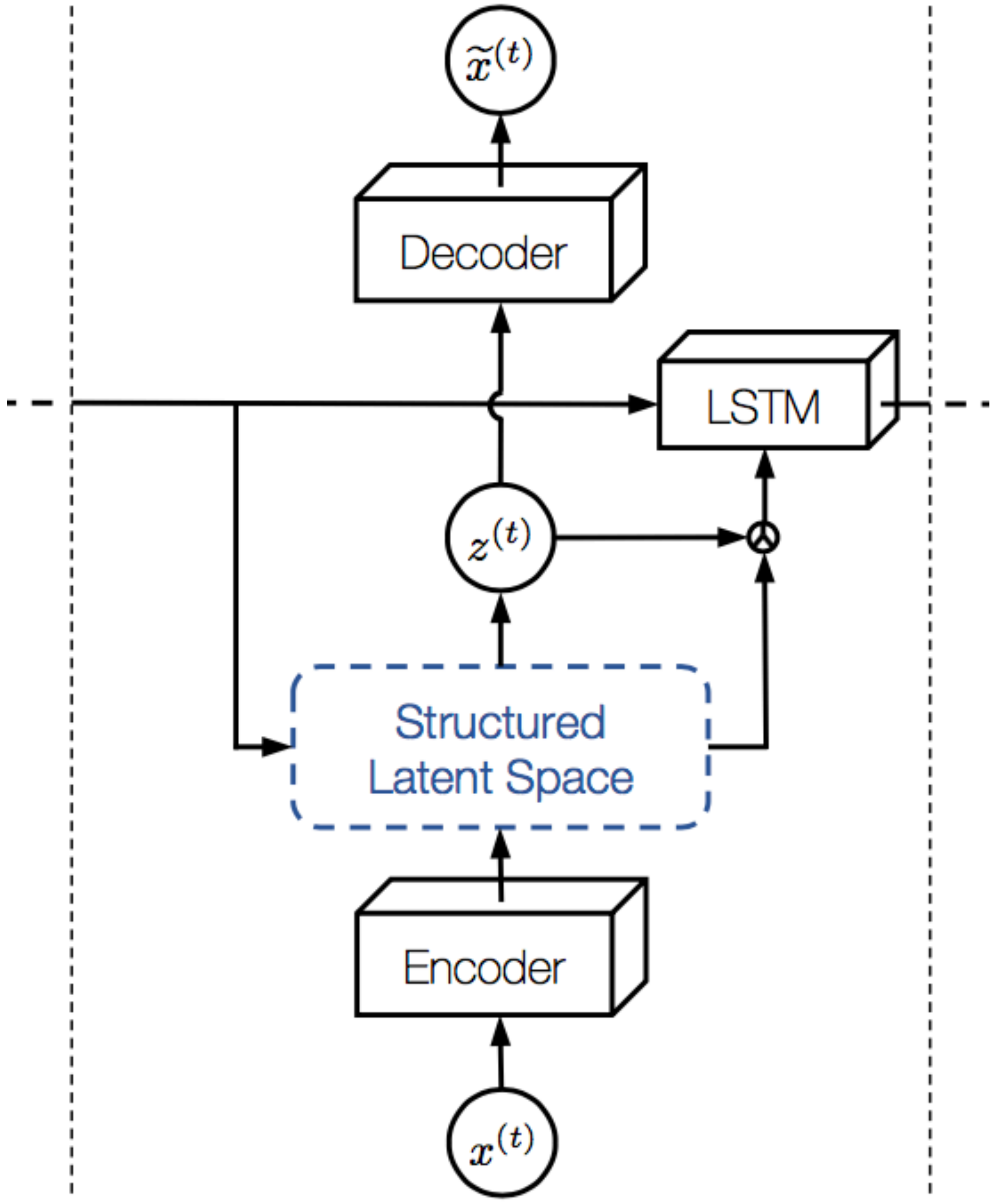
Variational Autoencoder (VAE)

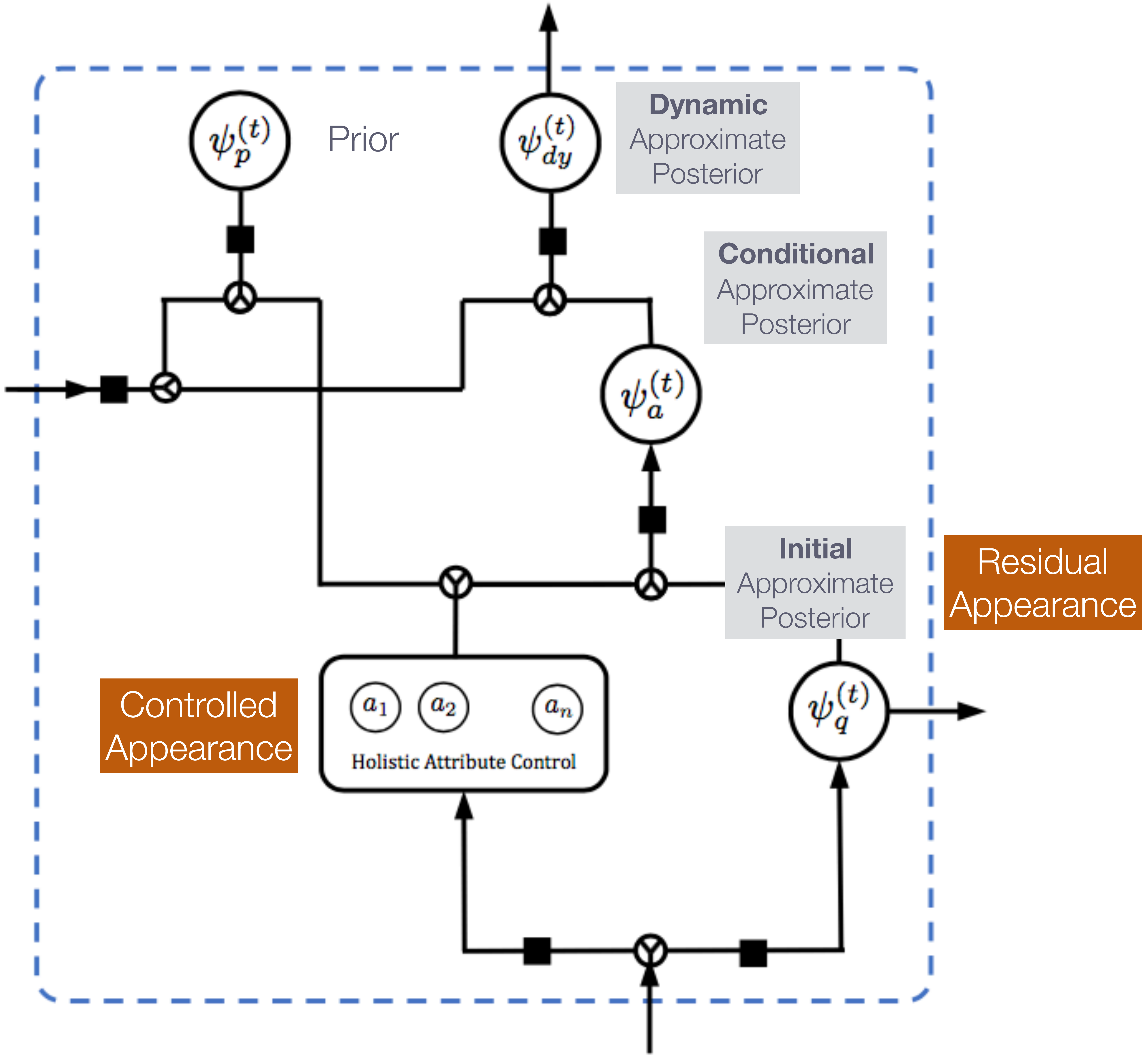


Variational Autoencoder (VAE) **+ LSTM**

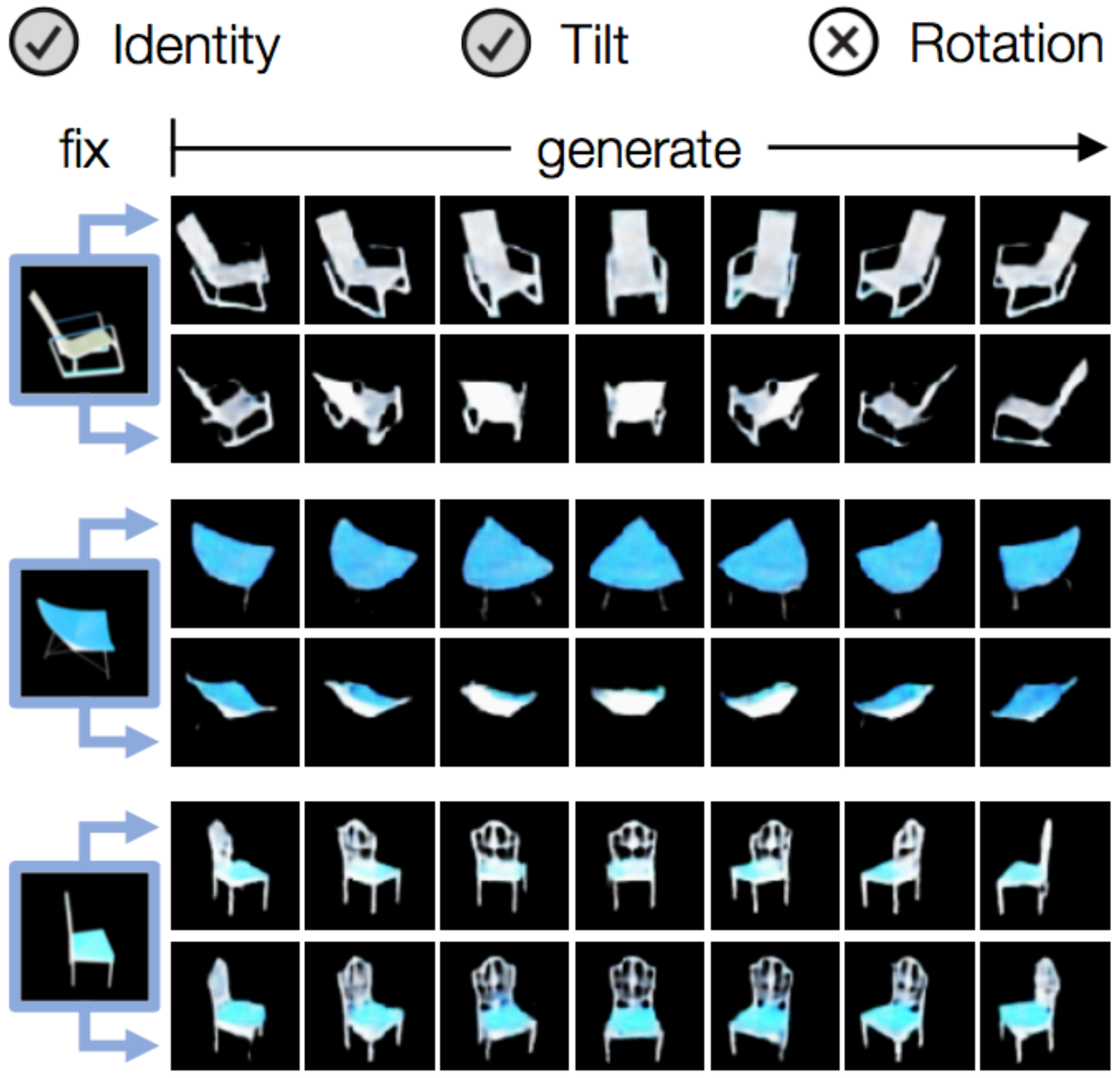


VAE + LSTM with Structured Latent Space

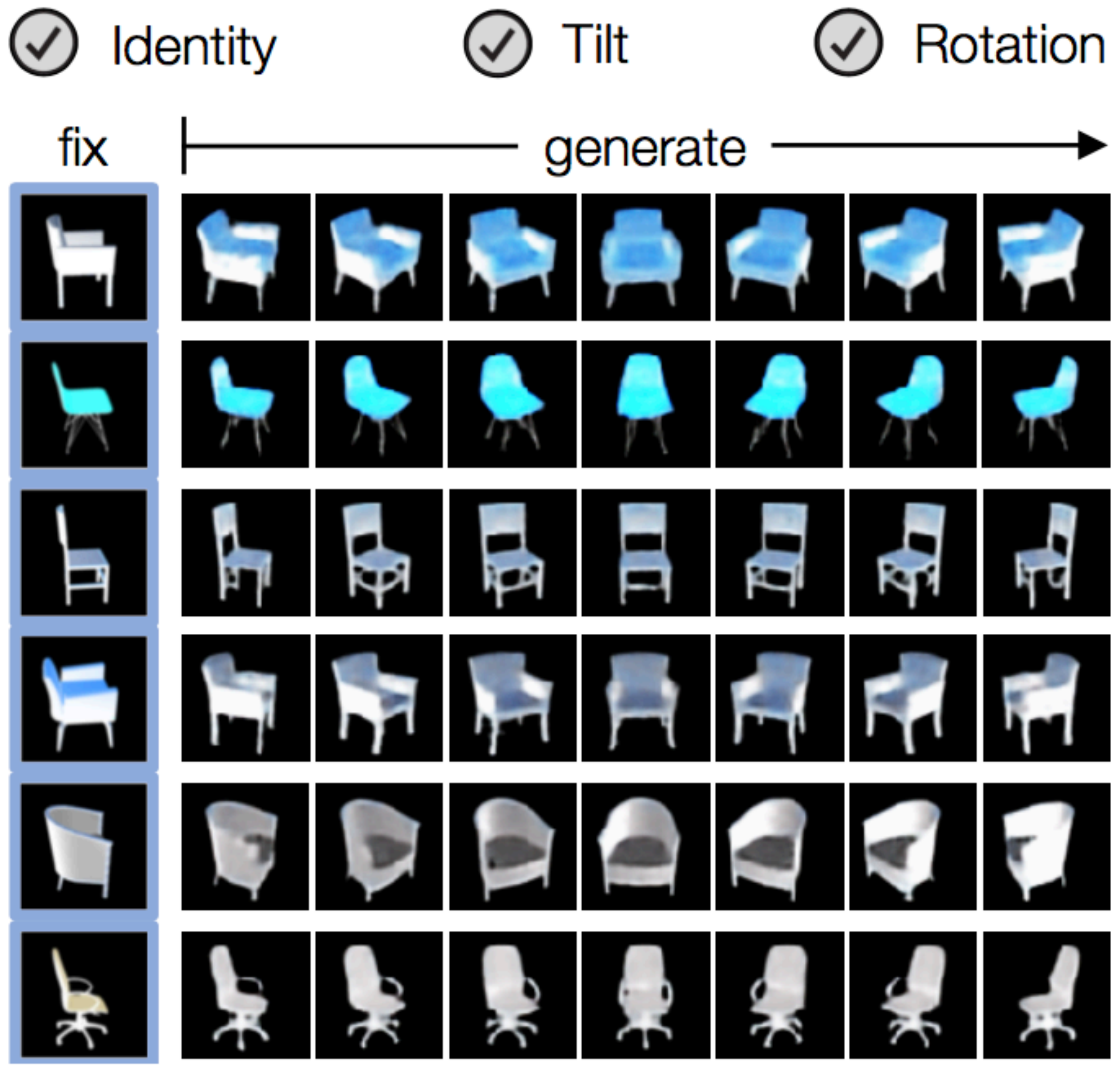




Results: Chair CAD dataset



(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**

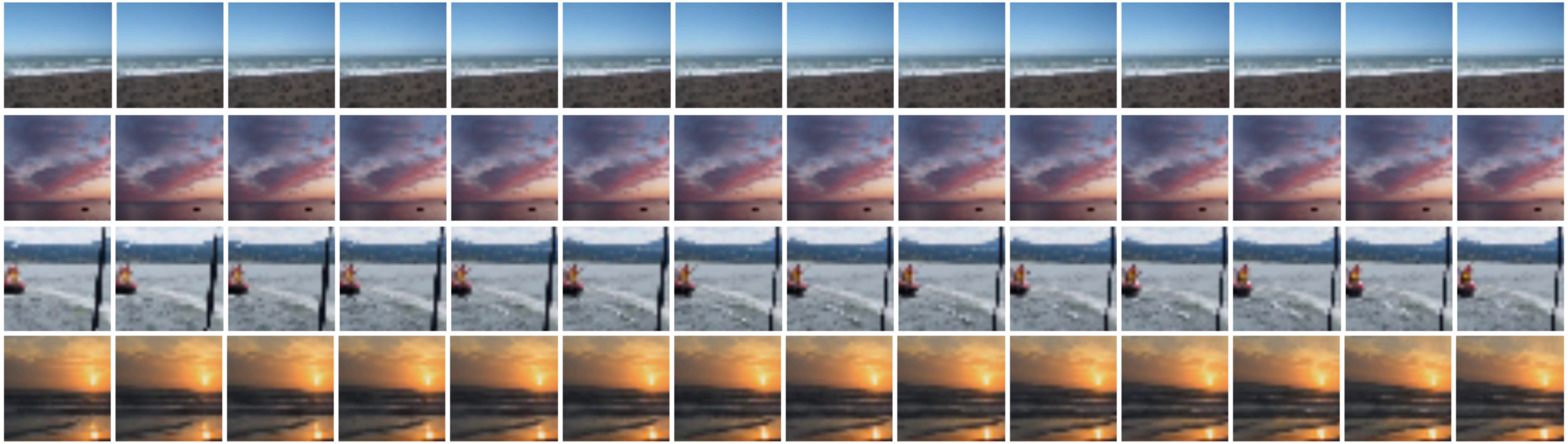
Quantitative

		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

	Weizmann Human Action [2]			
	Bound	MoCoGAN [32] ○	VideoVAE (ours) ○	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



		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)

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

Generative Adversarial Networks (GANs)

Generative Adversarial Networks

[Goodfellow et al., 2014]

Problem: Want to sample from complex, high-dimensional training distribution. There is no direct way to do this!

Generative Adversarial Networks

[Goodfellow et al., 2014]

Problem: Want to sample from complex, high-dimensional training distribution. There is no direct way to do this!

Solution: Sample from a simple distributions, e.g., random noise. Learn transformation to the training distribution

Generative Adversarial Networks

[Goodfellow et al., 2014]

Problem: Want to sample from complex, high-dimensional training distribution. There is no direct way to do this!

Solution: Sample from a simple distributions, e.g., random noise. Learn transformation to the training distribution

Question: What can we use to represent complex transformation function?

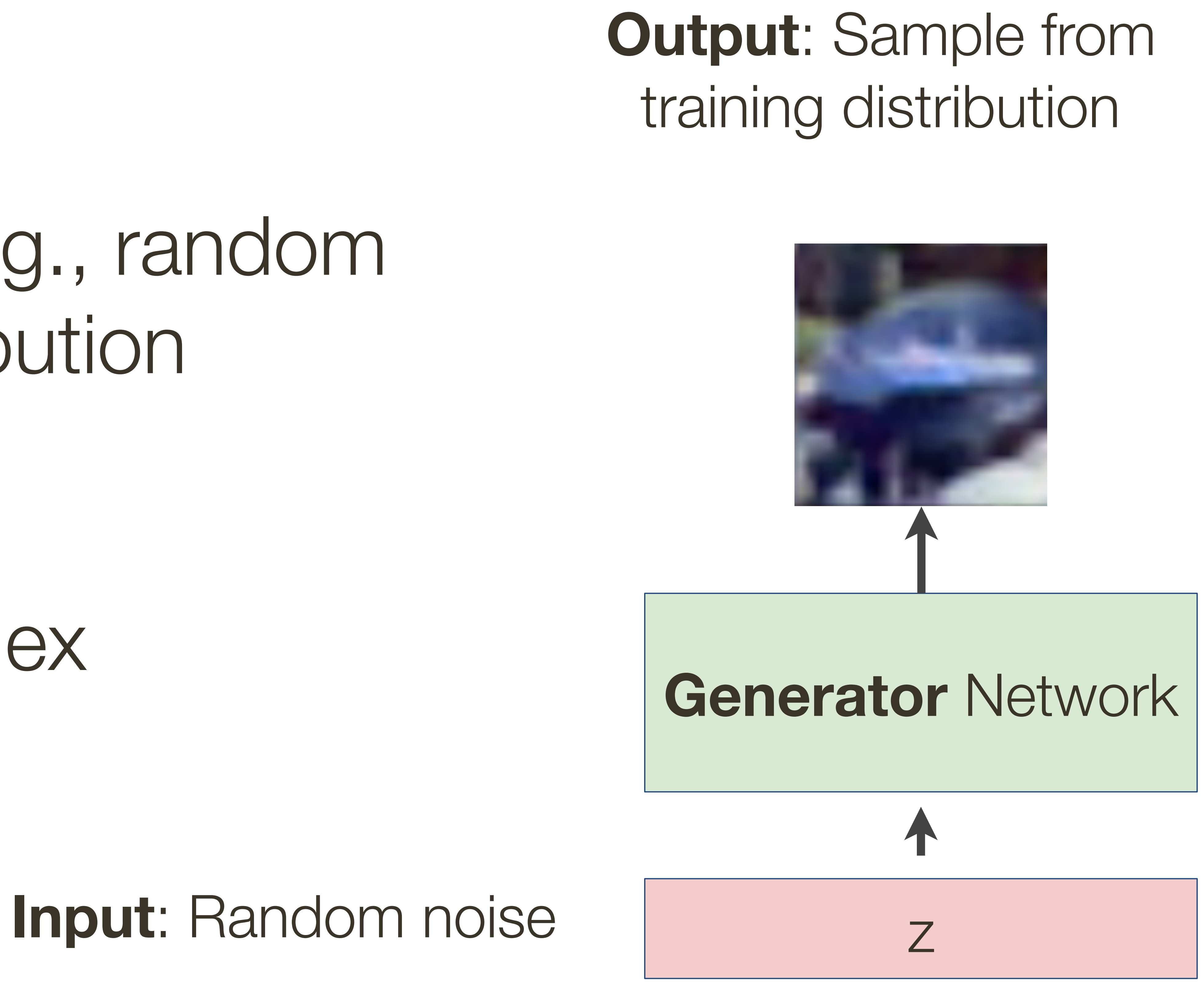
Generative Adversarial Networks

[Goodfellow et al., 2014]

Problem: Want to sample from complex, high-dimensional training distribution. There is no direct way to do this!

Solution: Sample from a simple distributions, e.g., random noise. Learn transformation to the training distribution

Question: What can we use to represent complex transformation function?



Training GANs: Two-player Game

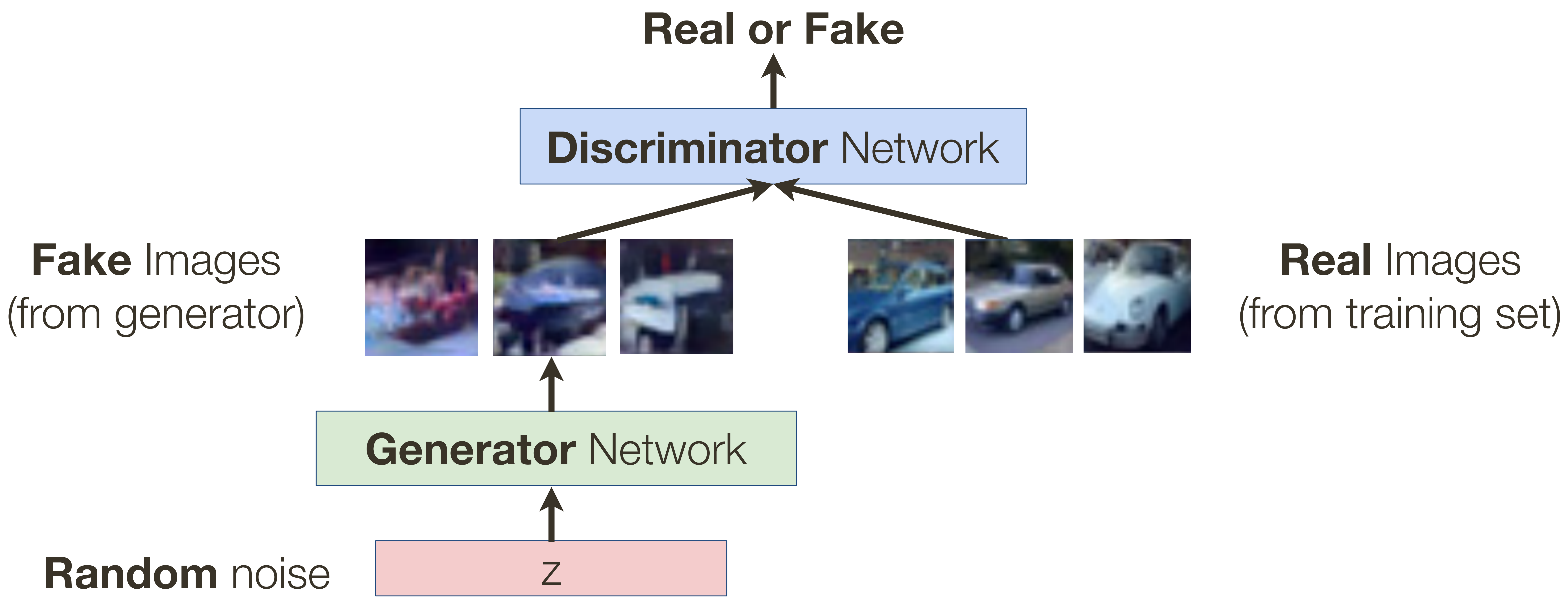
[Goodfellow et al., 2014]

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

Training GANs: Two-player Game

[Goodfellow et al., 2014]

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



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

Training GANs: Two-player Game

[Goodfellow et al., 2014]

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Minimax objective function:

Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Discriminator output for real data x

Discriminator output for generated fake data G(z)

- **Discriminator** (θ_d) wants to maximize objective such that D(x) is close to 1 (real) and D(G(z)) is close to 0 (fake)
- **Generator** (θ_g) wants to minimize objective such that D(G(z)) is close to 1 (discriminator is fooled into thinking generated G(z) is real)

Training GANs: Two-player Game

[Goodfellow et al., 2014]

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. Gradient **ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. Gradient **descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Training GANs: Two-player Game

[Goodfellow et al., 2014]

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. Gradient **ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. Gradient **descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

Training GANs: Two-player Game

[Goodfellow et al., 2014]

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. Gradient **ascent** on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

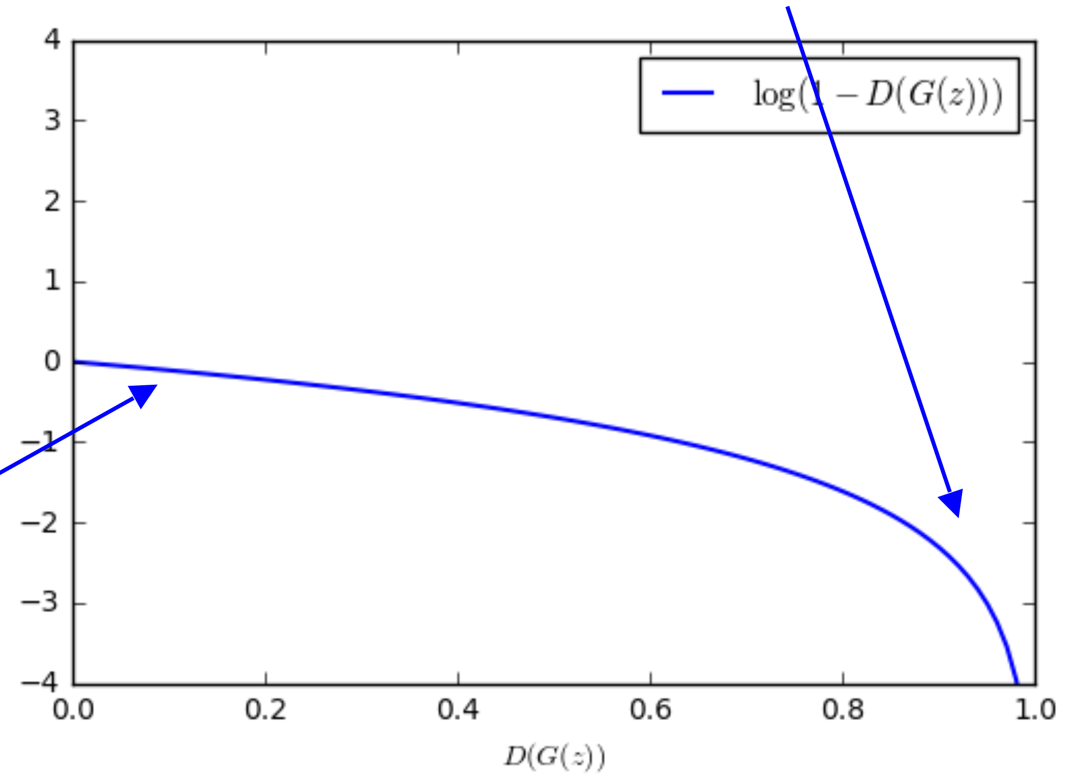
2. Gradient **descent** on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

Gradient signal dominated by region where sample is already good

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!



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

Training GANs: Two-player Game

[Goodfellow et al., 2014]

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. Gradient **ascent** on discriminator

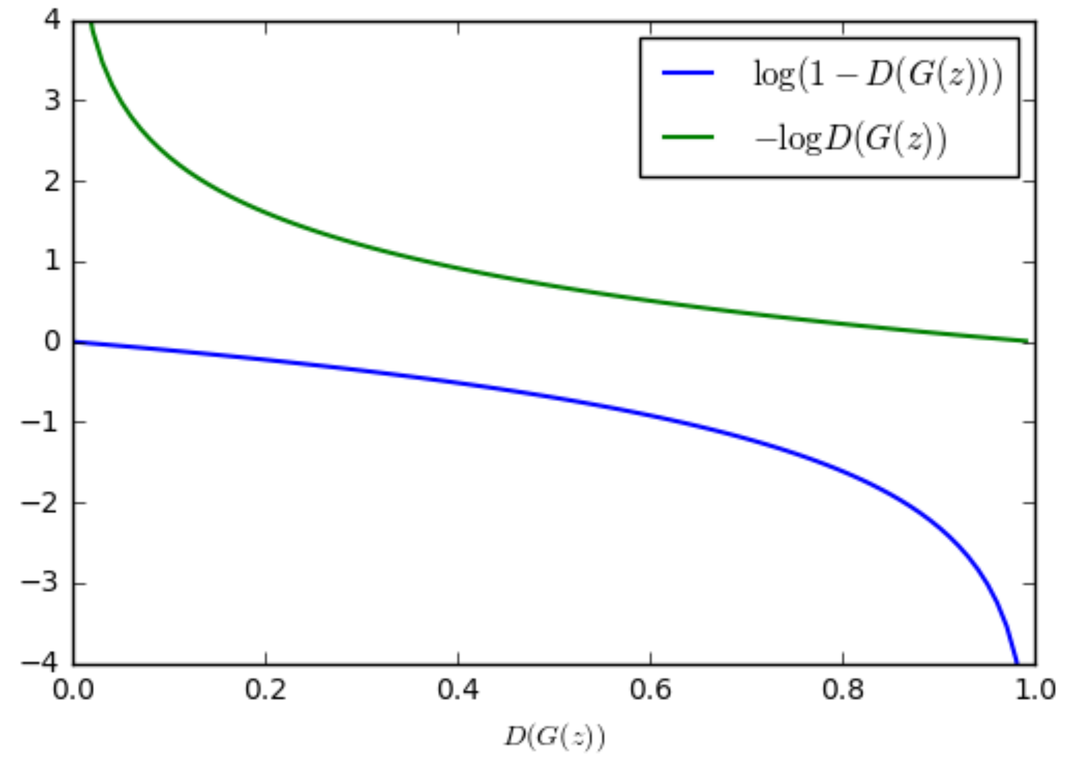
$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. Instead, gradient **ascent** on generator, different objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

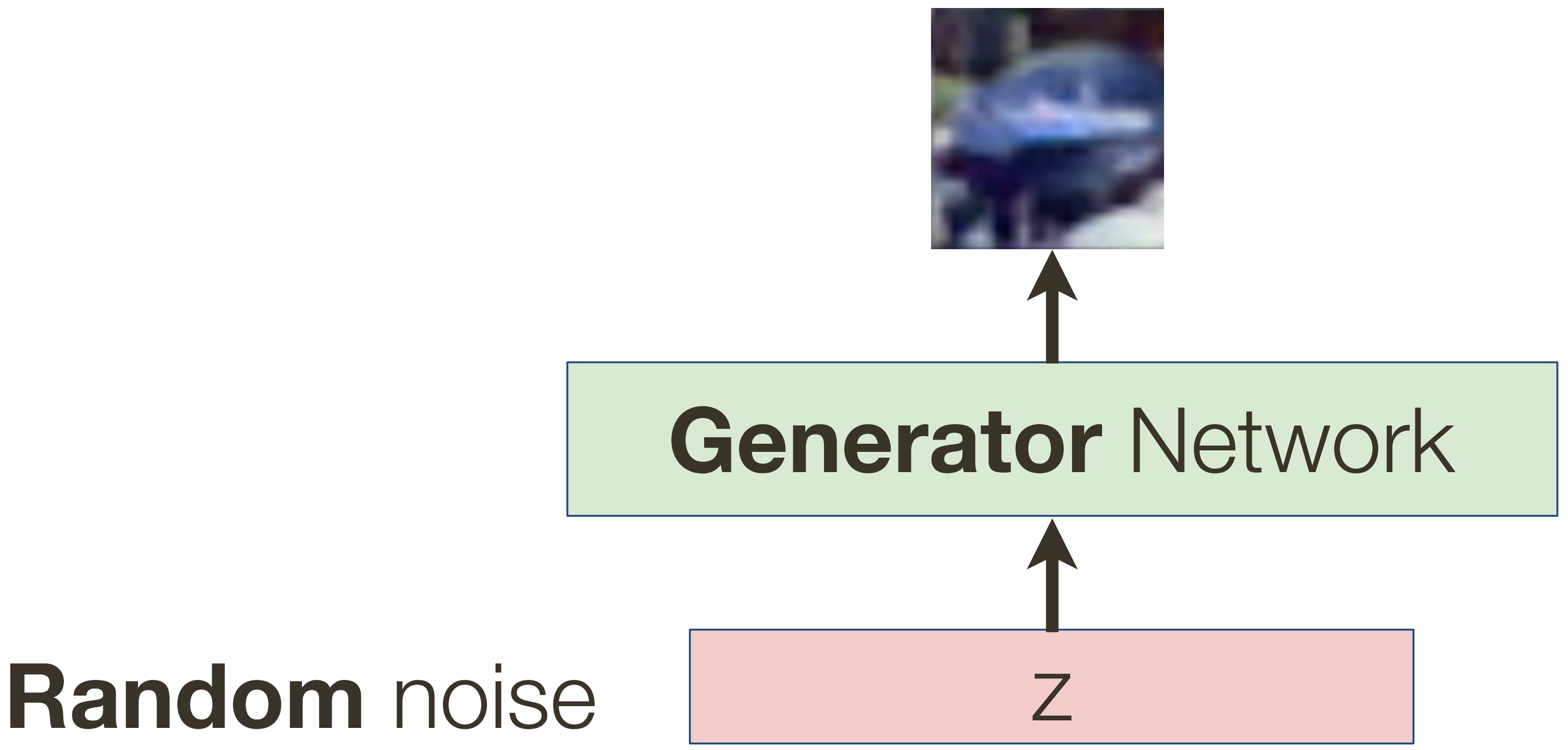
Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.

Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.



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

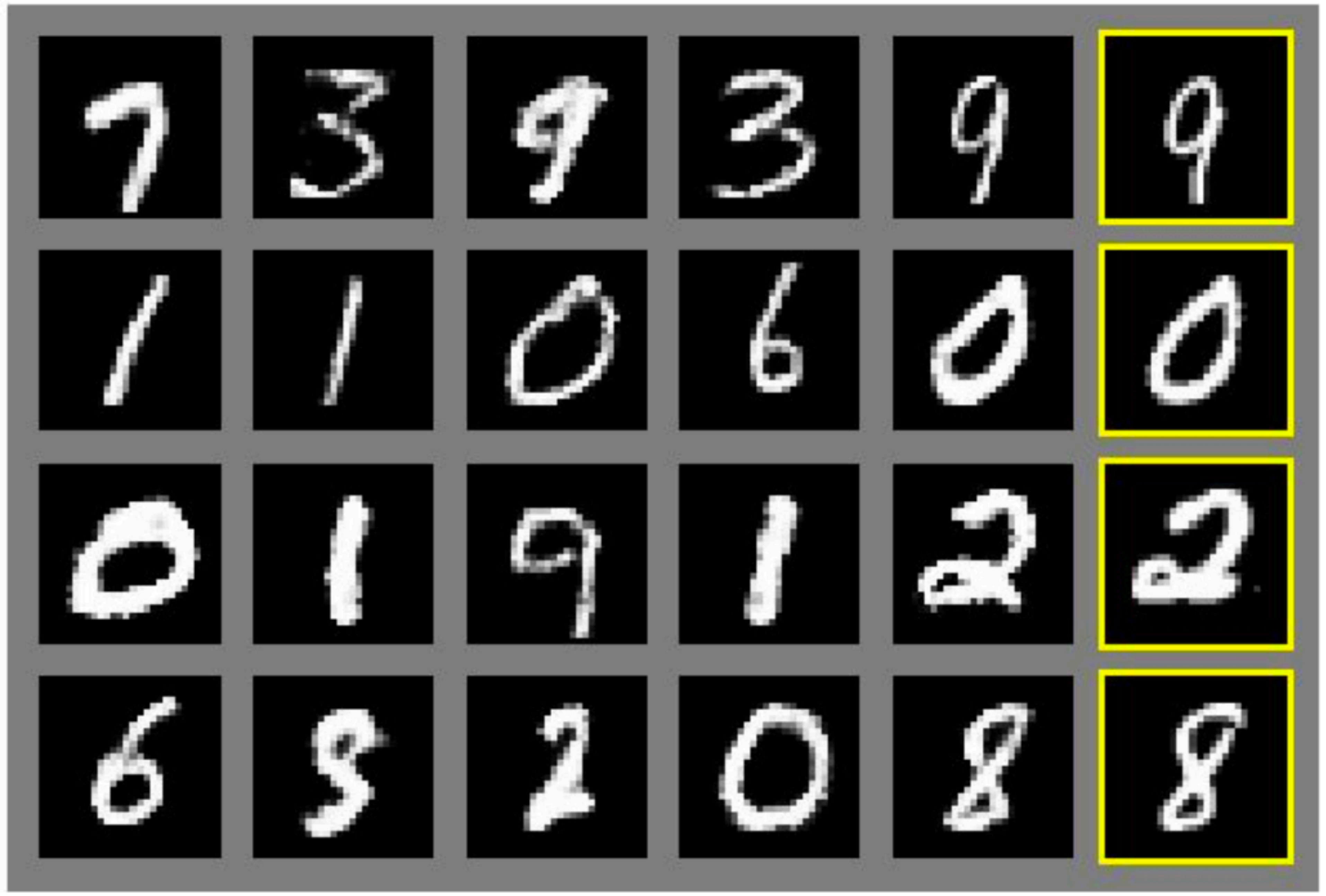
Sampling **GAN**s



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

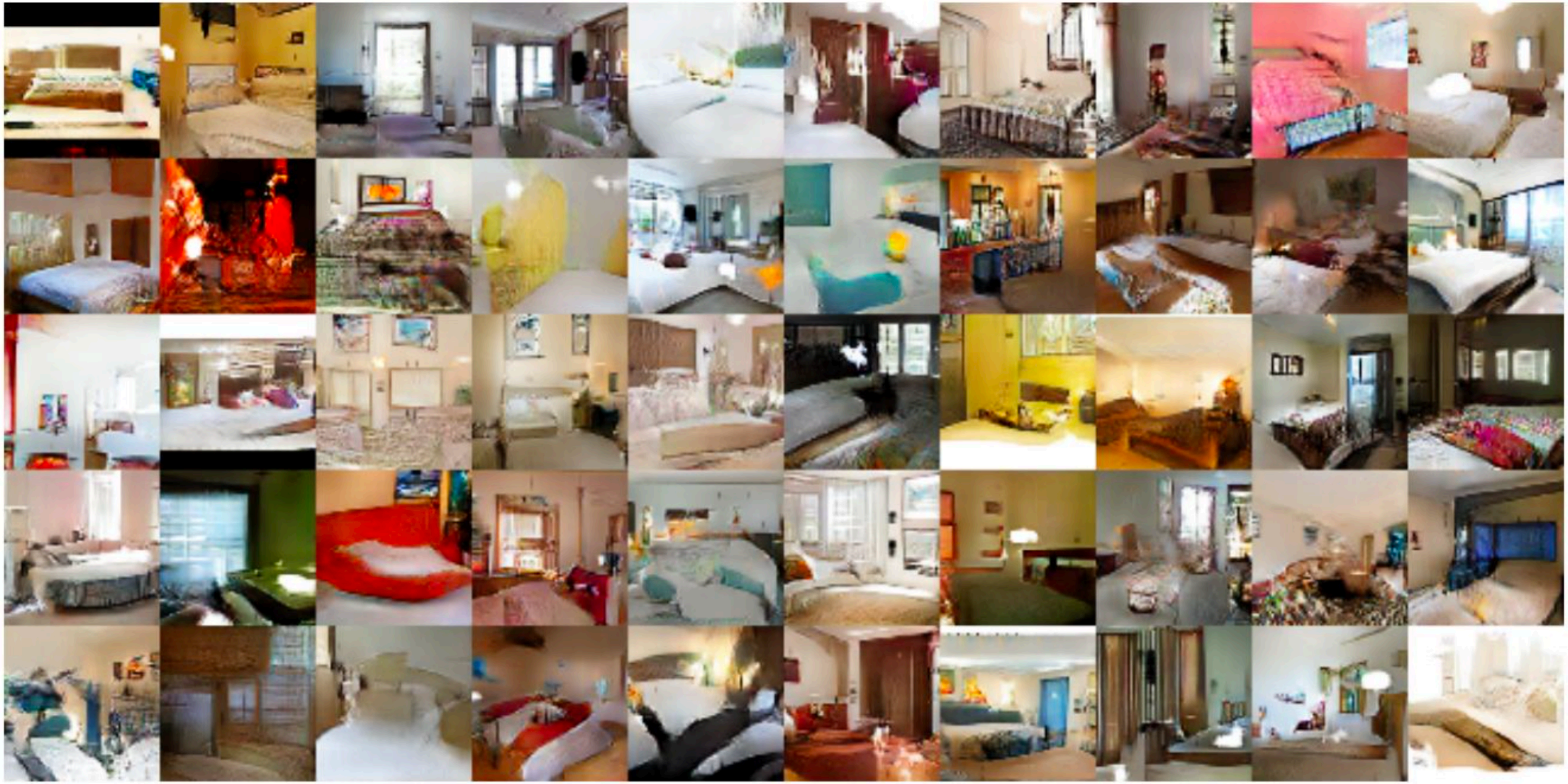
Generative Adversarial Nets

Generated Samples



GANs with Convolutional Architectures

[Radford et al., 2016]

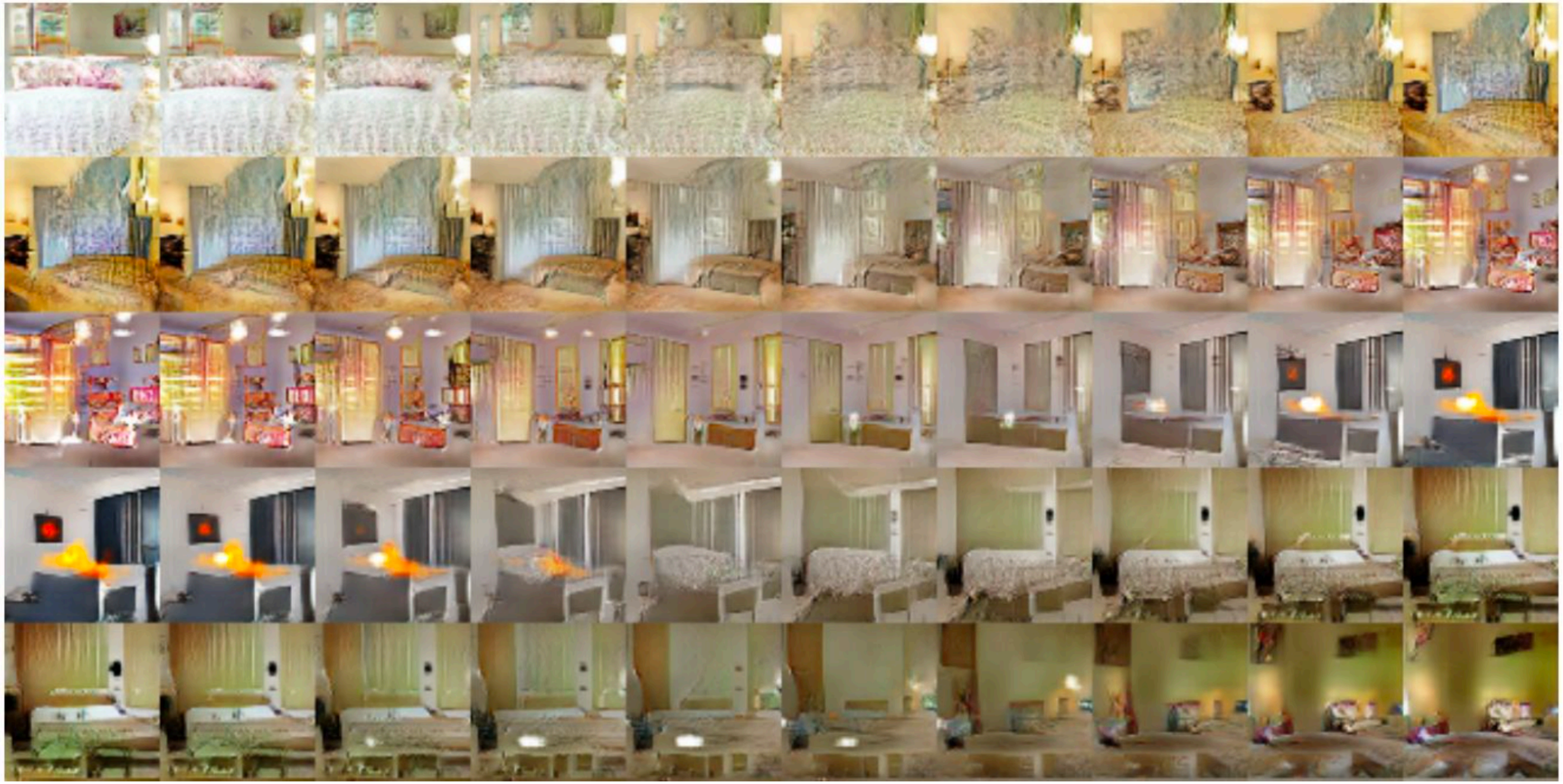


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GANs with Convolutional Architectures

[Radford et al., 2016]

Interpolating between points in latent space



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

GANs: Interpretable Vector Math

[Radford et al., 2016]

Smiling woman Neutral womai Neutral man

Samples from the model

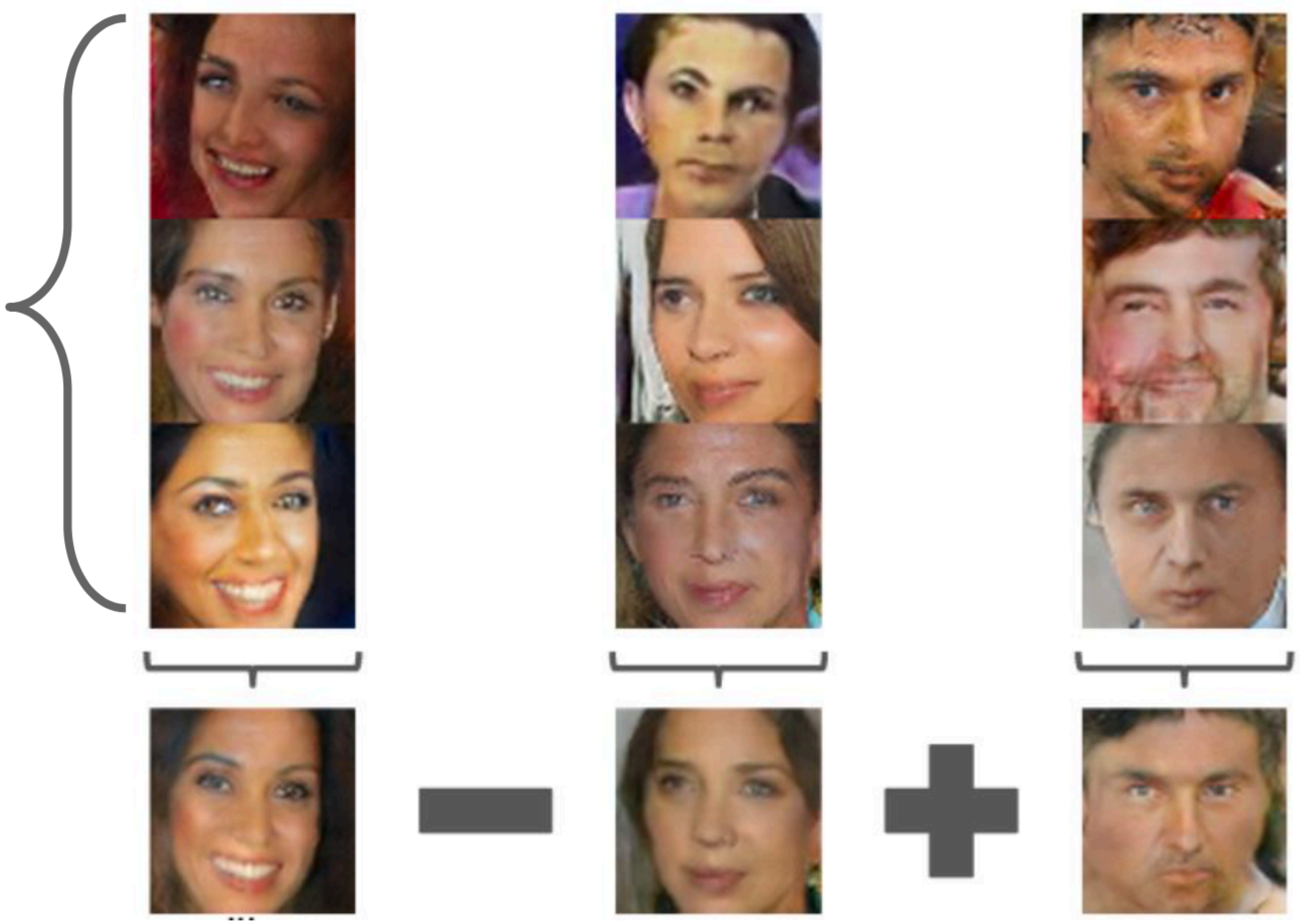
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

GANs: Interpretable Vector Math

[Radford et al., 2016]

Smiling woman   Neutral womai   Neutral man

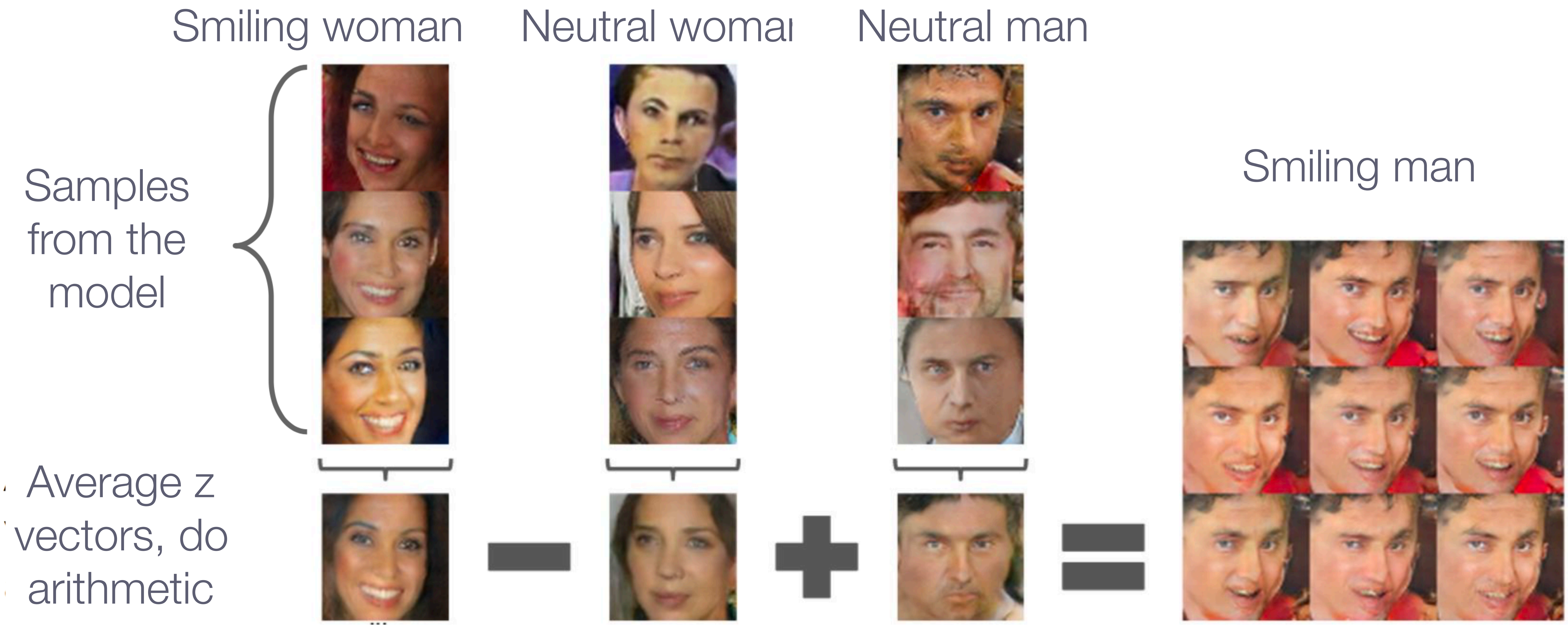
Samples from the model



Average z vectors, do arithmetic

GANs: Interpretable Vector Math

[Radford et al., 2016]



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

GANs: Interpretable Vector Math

[Radford et al., 2016]

Glasses Man | No Glasses Man | No Glasses Woman

Samples from the model

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

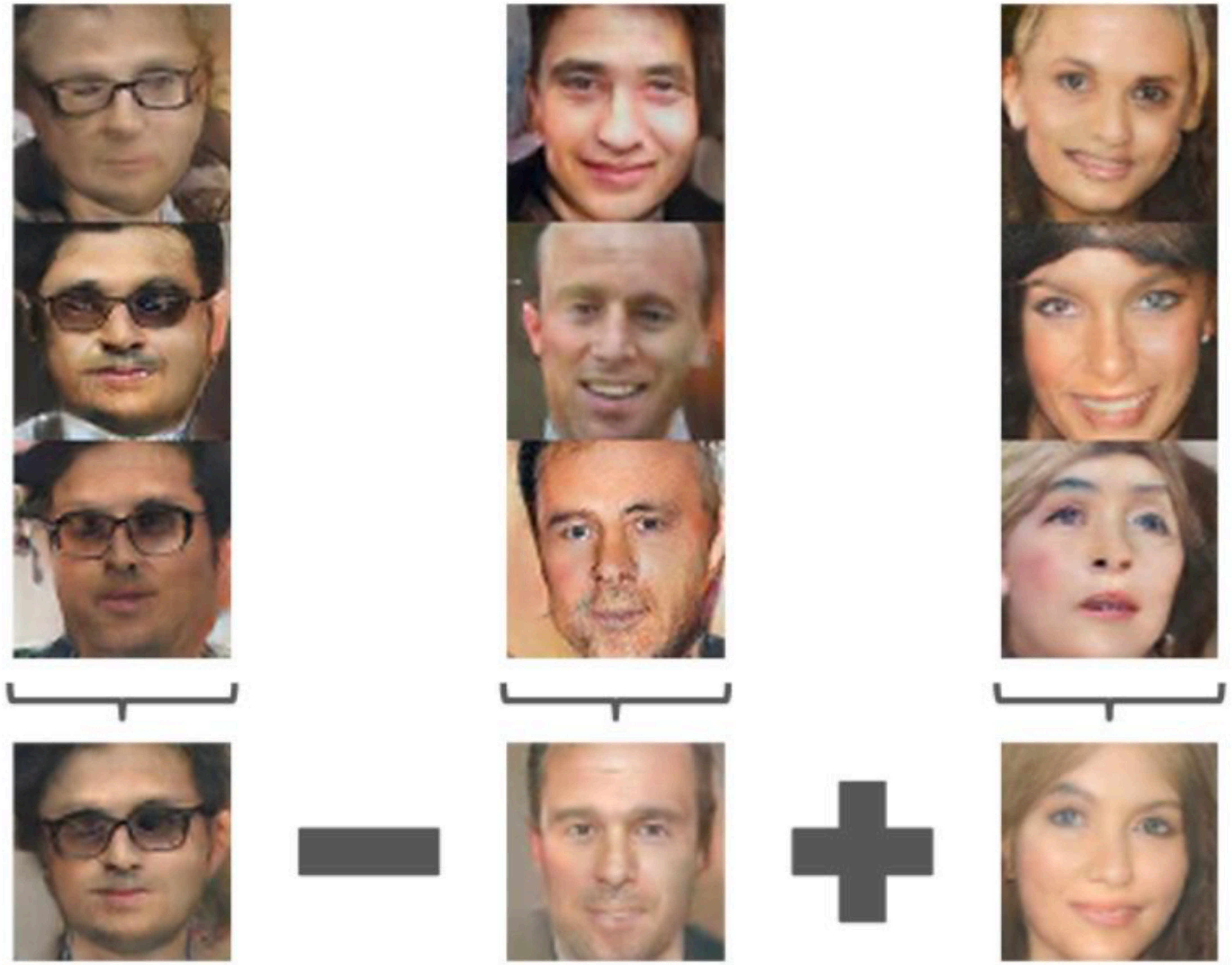
GANs: Interpretable Vector Math

[Radford et al., 2016]

Glasses Man | No Glasses Man | No Glasses Woman

Samples from the model

Average z vectors, do arithmetic



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

GANs: Interpretable Vector Math

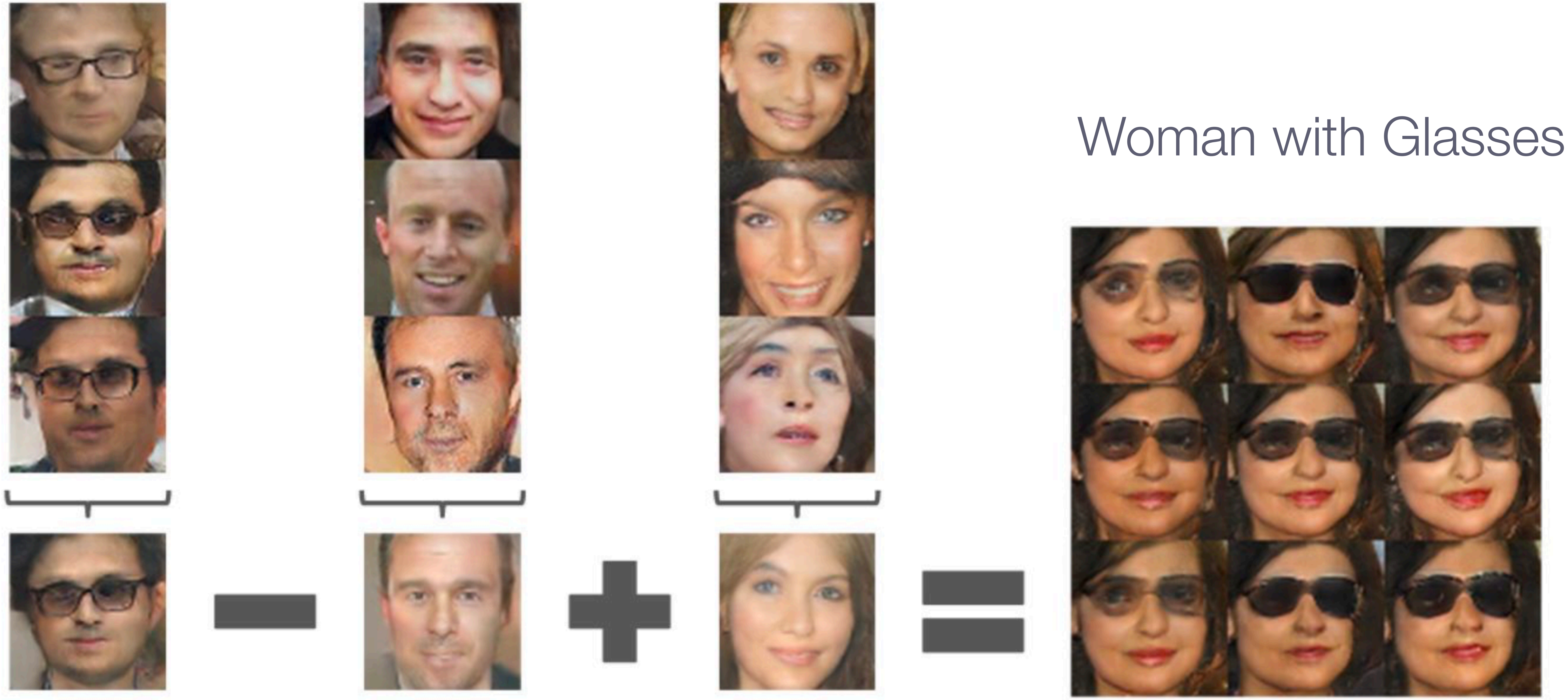
[Radford et al., 2016]

Radford et al, ICLR 2016

Glasses Man | No Glasses Man | No Glasses Woman

Samples from the model

Average z vectors, do arithmetic

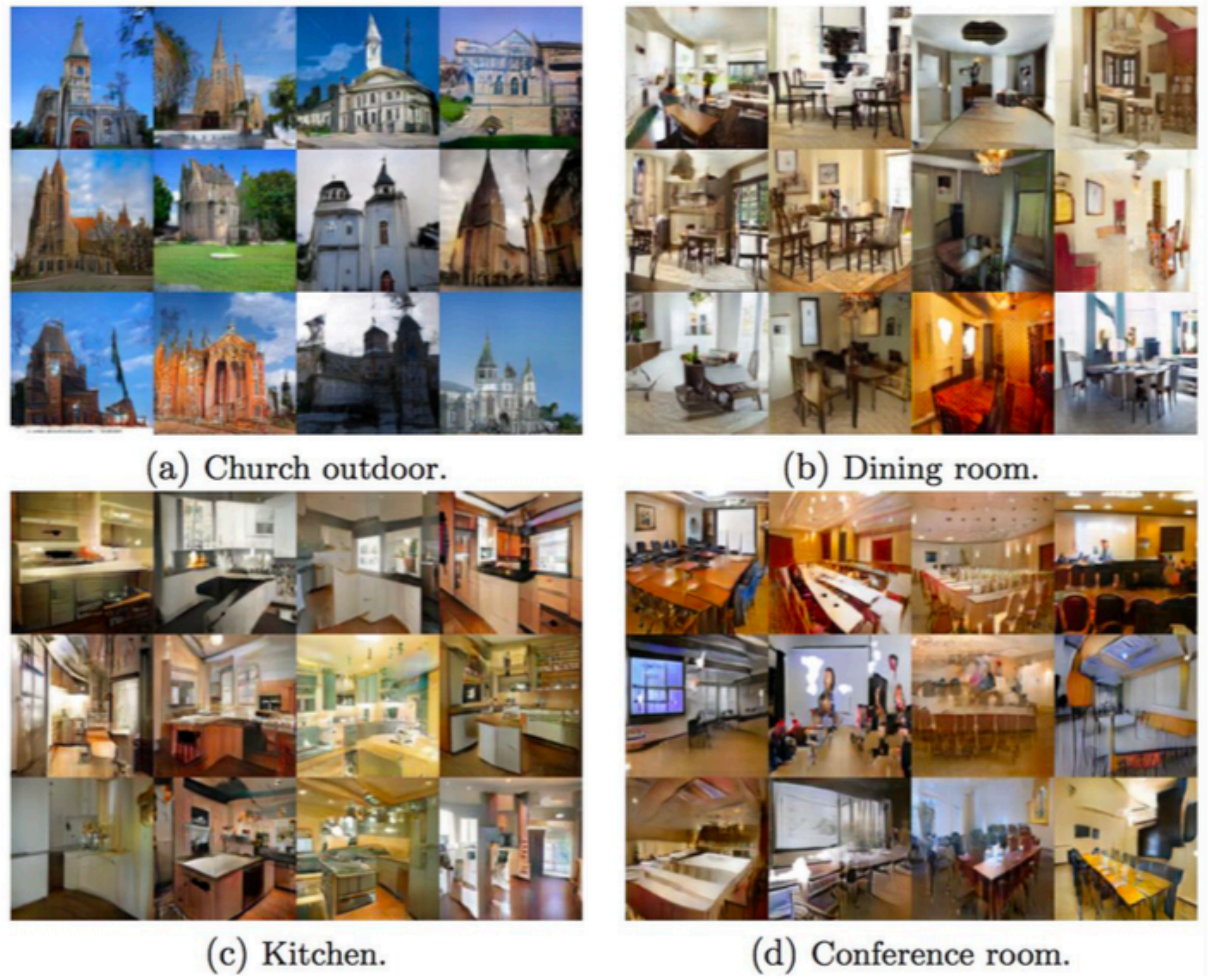


Woman with Glasses

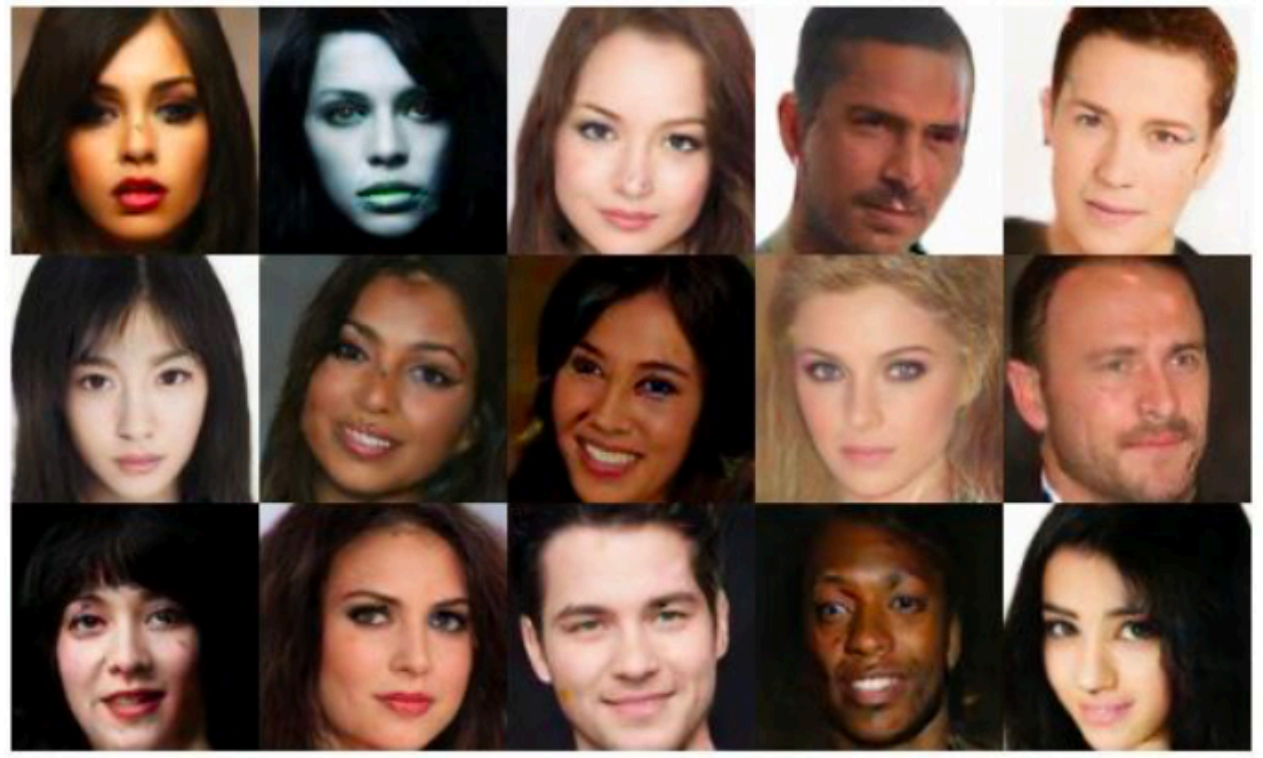
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Year of the **GAN**

Better training and generation



LSGAN. Mao et al. 2017.



BEGAN. Bertholet et al. 2017.

Source->Target domain transfer

CycleGAN. Zhu et al. 2017.

Text -> Image Synthesis

Reed et al. 2017.

Many GAN applications

Pix2pix. Isola 2017. Many examples at https://phillipi.github.io/pix2pix/

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

Year of the **GAN**

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

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

GANs

Don't work with an explicit density function

Take game-theoretic approach: learn to generate from training distribution through 2-player game

Pros:

— Beautiful, state-of-the-art samples!

Cons:

— Trickier / more unstable to train

— Can't solve inference queries such as p(x), p(z|x)

Active area of research:

— Better loss functions, more stable training (Wasserstein GAN, LSGAN, many others)

— Conditional GANs, GANs for all kinds of applications

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