THE UNIVERSITY OF BRITISH COLUMBIA

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

Lecture 12: Generative Models

Supervised vs. **Unsupervised** Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x→y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, *etc.*

→ Cat

Classification

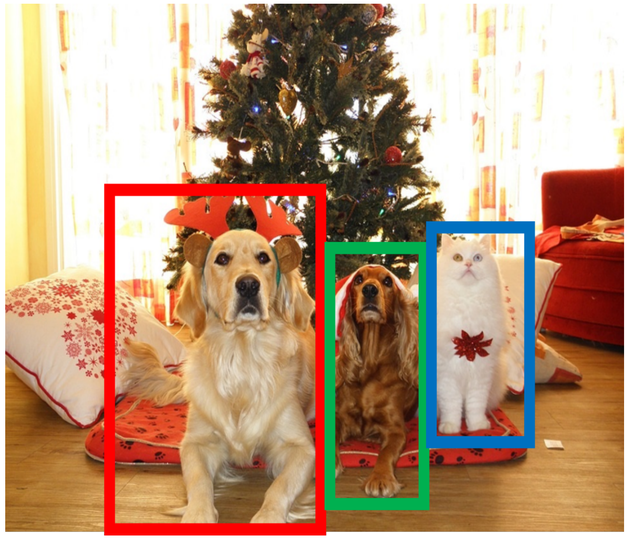
Supervised vs. **Unsupervised** Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x→y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, *etc.*



DOG, **DOG**, **CAT**

Object Detection

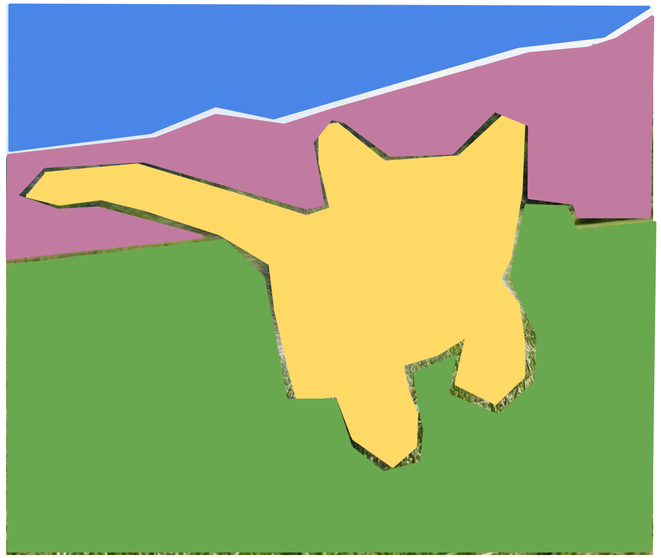
Supervised vs. **Unsupervised** Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x→y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, *etc.*



GRASS, **CAT**, **TREE**, **SKY**

Semantic Segmentation

Supervised vs. **Unsupervised** Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x→y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, *etc.*

A cat sitting on a suitcase on the floor

Image Captioning

Supervised vs. **Unsupervised** Learning

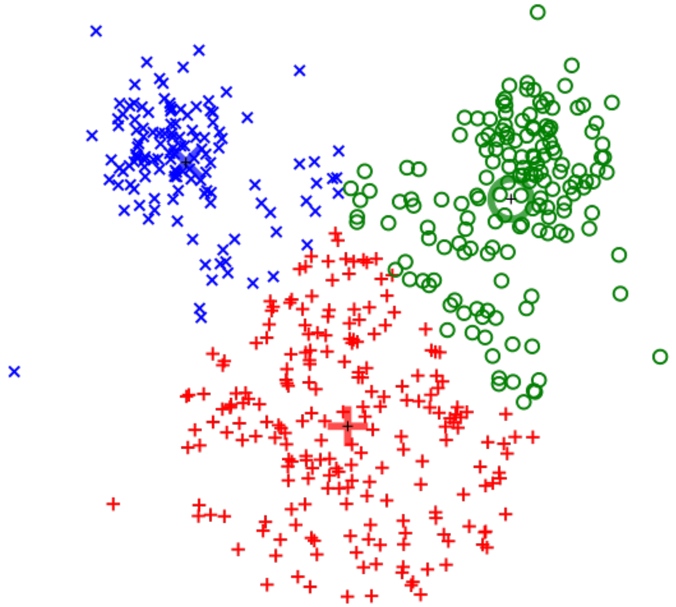
Unsupervised Learning

Data: x

Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, *etc.*



k-means clustering

Supervised vs. **Unsupervised** Learning

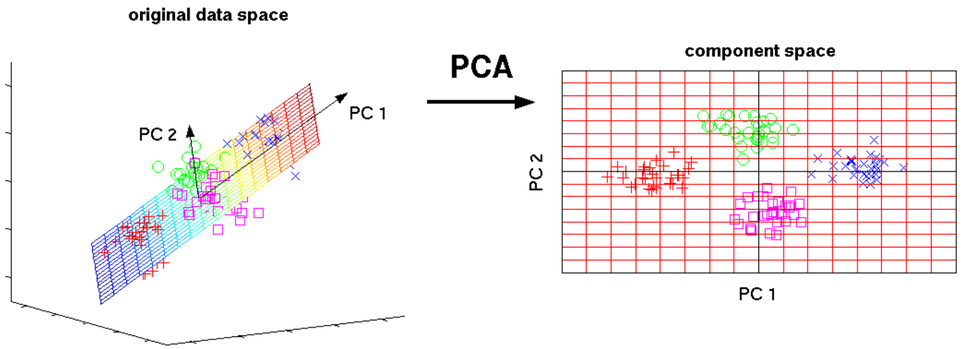
Unsupervised Learning

Data: x

Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, *etc.*



dimensionality reduction

Supervised vs. **Unsupervised** Learning

Unsupervised Learning

Data: x

Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, *etc.*

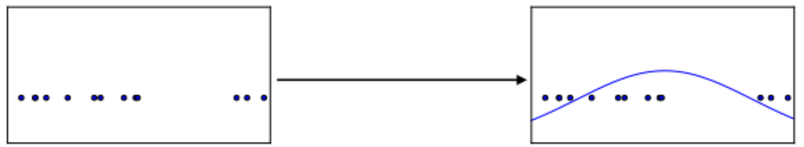
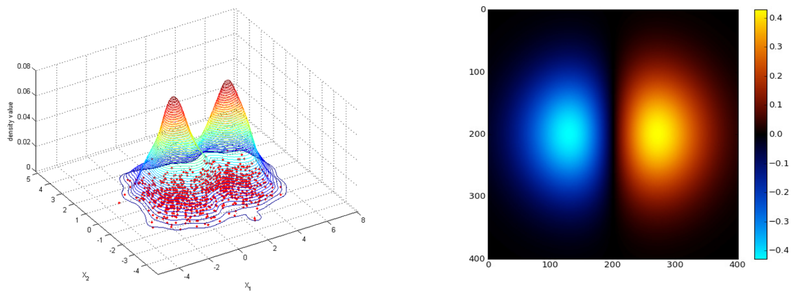


Figure copyright Ian Goodfellow, 2016. Reproduced with permission.

1-dim density estimation



2-dim density estimation

2-d density images left and right are CC0 public domain

Supervised vs. **Unsupervised** Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x→y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, *etc.*

Unsupervised Learning

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

Generative Models

Given training data, generate new samples from the same distribution

Training data ~ $p_{\text{data}}(\boldsymbol{x})$

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

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

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

Generative Models

Given training data, generate new samples from the same distribution

Training data ~ $p_{\text{data}}(\boldsymbol{x})$

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

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

Addresses **density estimation**, a core problem in unsupervised learning

— **Explicit** density estimation: explicitly define and solve for $p_{\text{model}}(\boldsymbol{x})$

— **Implicit** density estimation: learn model that can sample from $p_{\text{model}}(\boldsymbol{x})$ w/o explicitly defining it

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

Taxonomy of Generative Models

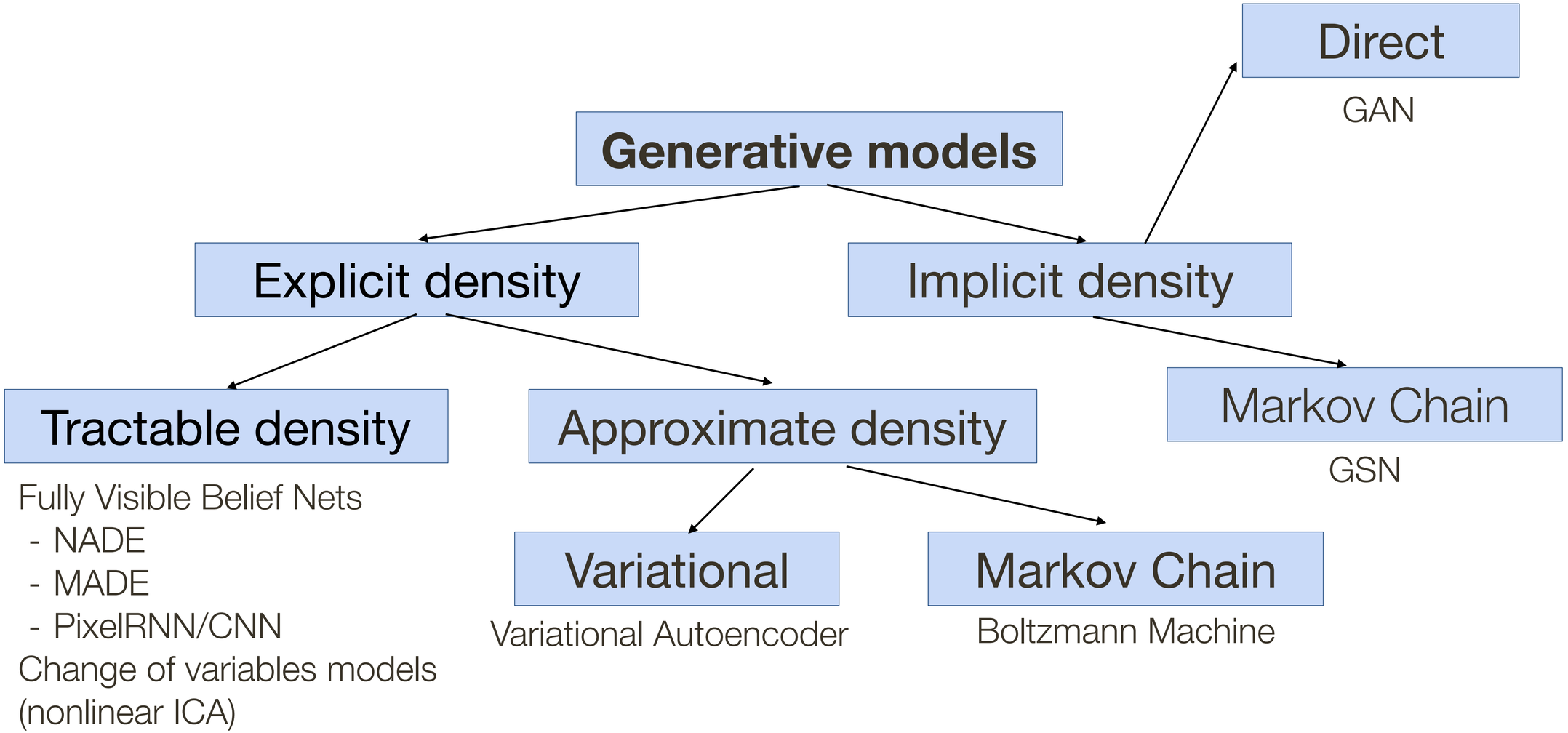


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

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

Taxonomy of Generative Models

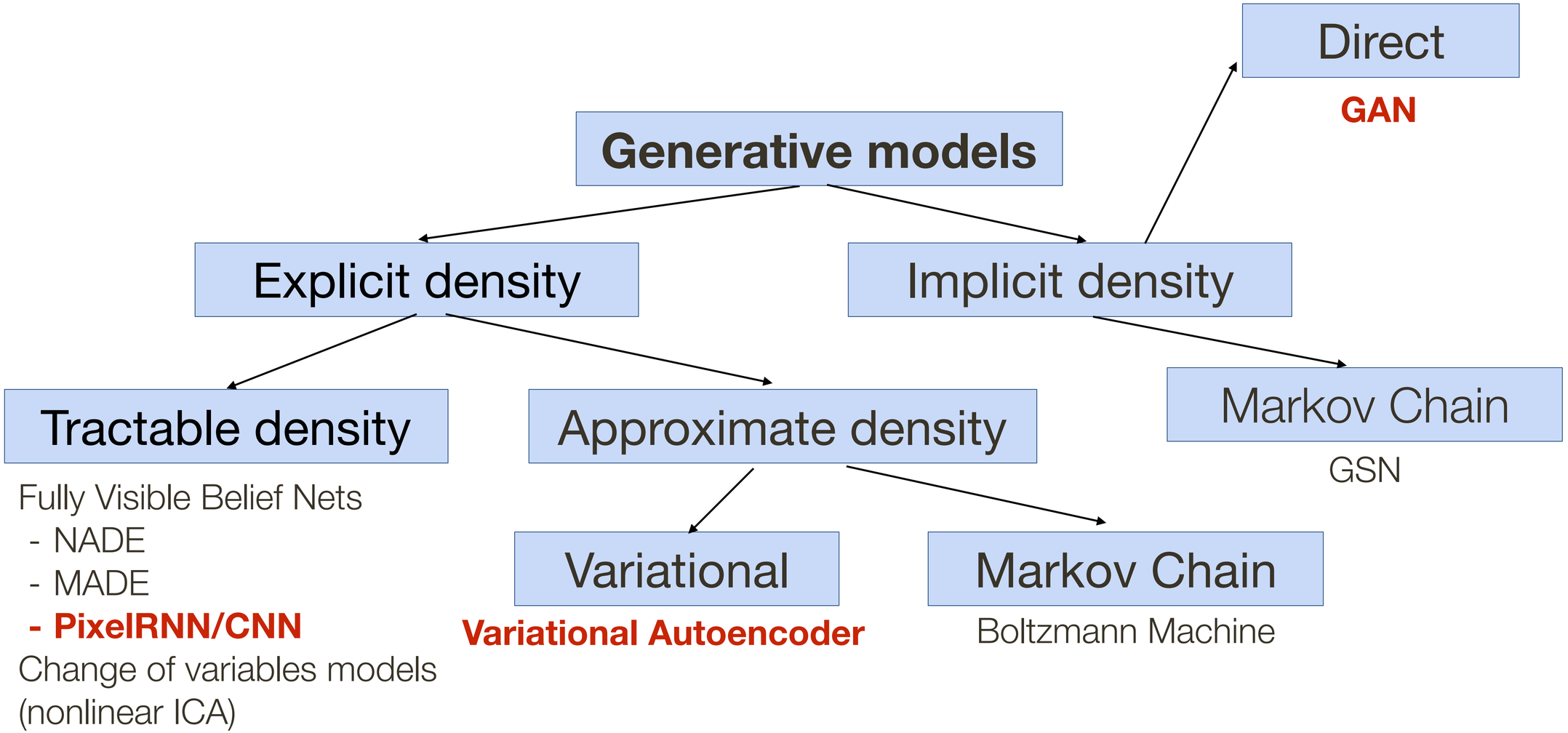
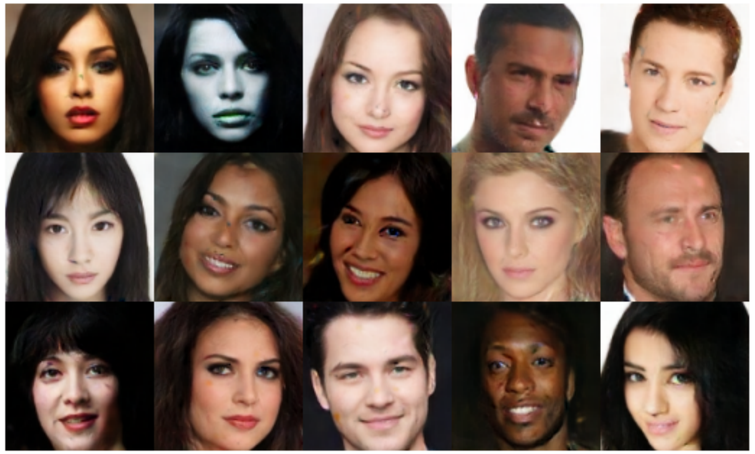


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

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

Why **Generative** Models?

— Realistic **samples** for artwork, super-resolution, colorization, *etc.*



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

Why **Generative** Models?

— Realistic **samples** for artwork, super-resolution, colorization, *etc.*

— Generative models of time-series data can be used for **simulation**, **predictions** and planning (reinforcement learning applications)

— Training generative models can also enable inference of latent representation that can be useful as **general features**

— **Dreaming** / hypothesis visualization

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

PixelRNN and PixelCNN

PixelRNN

[van der Oord et al., 2016]

Explicit Density model

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

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

Likelihood of image $\boldsymbol{x}$

Probability of i'th pixel value given all previous pixels

then maximize likelihood of training data

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

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

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

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**

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

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

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

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"

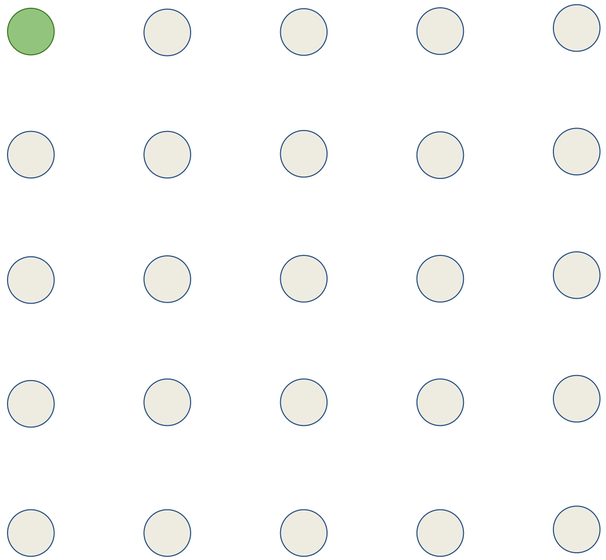
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

PixelRNN

[van der Oord et al., 2016]

Generate image pixels starting from the corner

Dependency on previous pixels model using an RNN (LSTM)



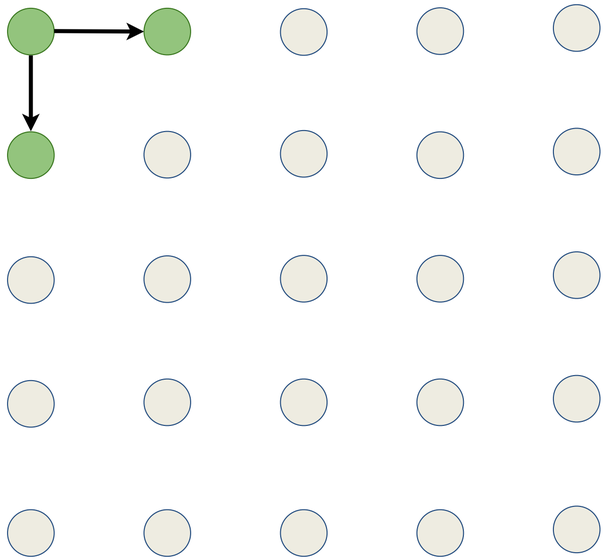
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

PixelRNN

[van der Oord et al., 2016]

Generate image pixels starting from the corner

Dependency on previous pixels model using an RNN (LSTM)



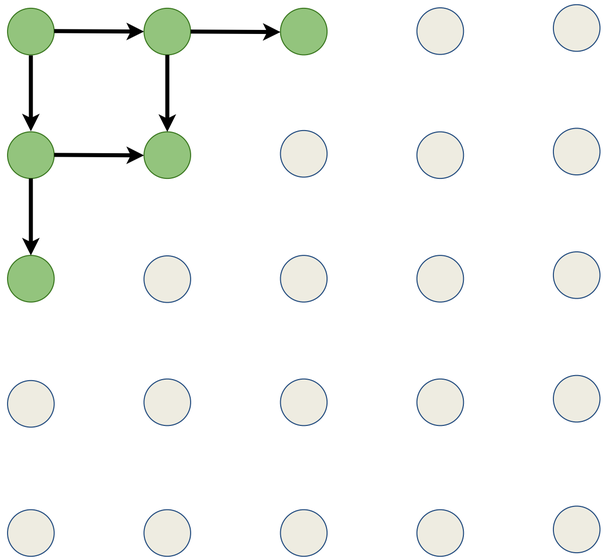
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

PixelRNN

[van der Oord et al., 2016]

Generate image pixels starting from the corner

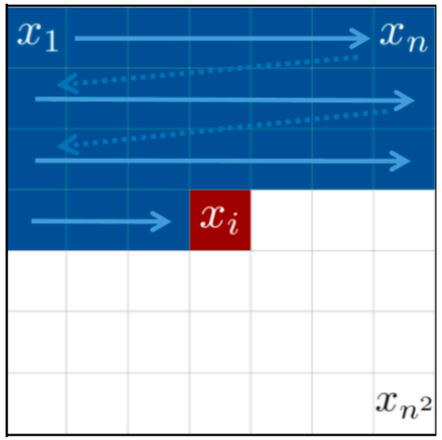
Dependency on previous pixels model using an RNN (LSTM)

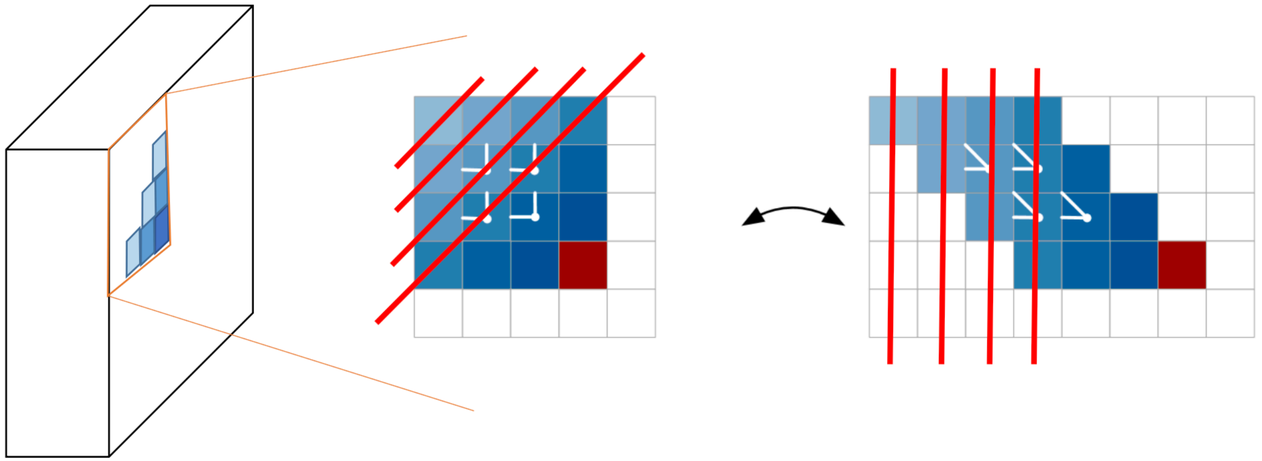


* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

PixelRNN

[van der Oord et al., 2016]





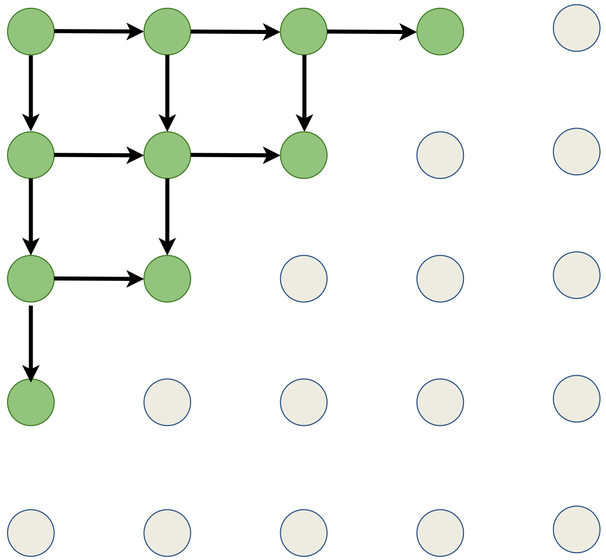
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

PixelRNN

[van der Oord et al., 2016]

Generate image pixels starting from the corner

Dependency on previous pixels model using an RNN (LSTM)



Problem: sequential generation is slow

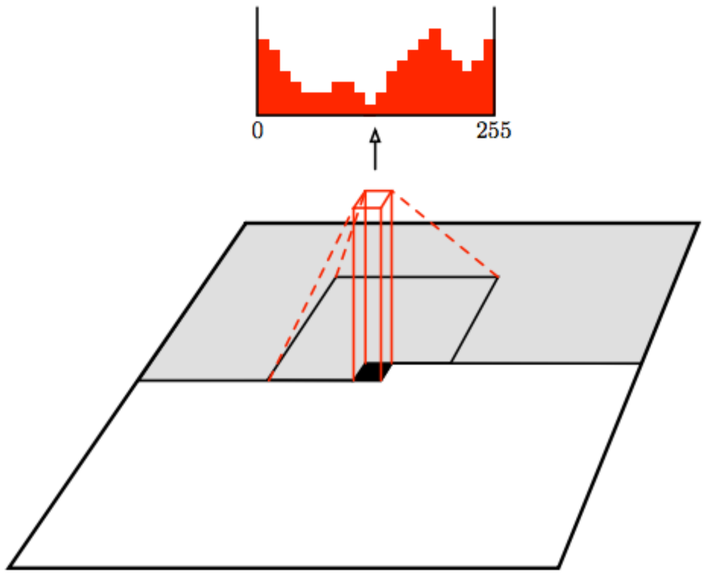
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

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



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

PixelCNN

[van der Oord et al., 2016]

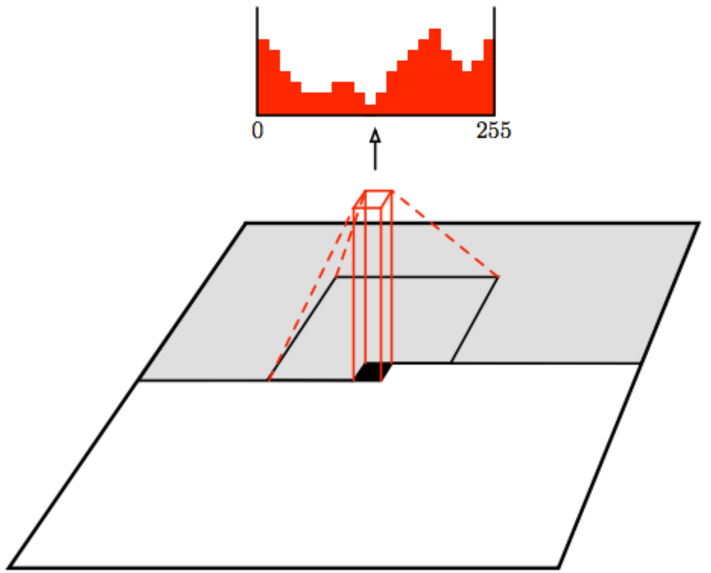
Still generate image pixels starting from the corner

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

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



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

PixelCNN

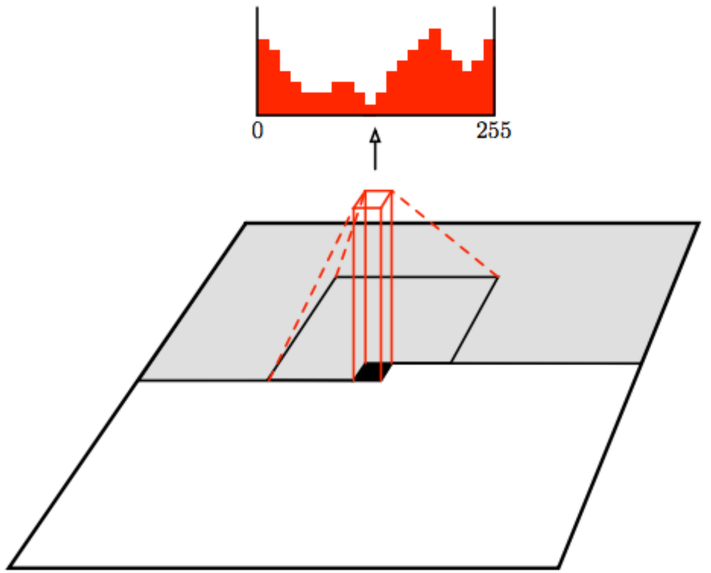
[van der Oord et al., 2016]

Still generate image pixels starting from the corner

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

Training: maximize likelihood of training images

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

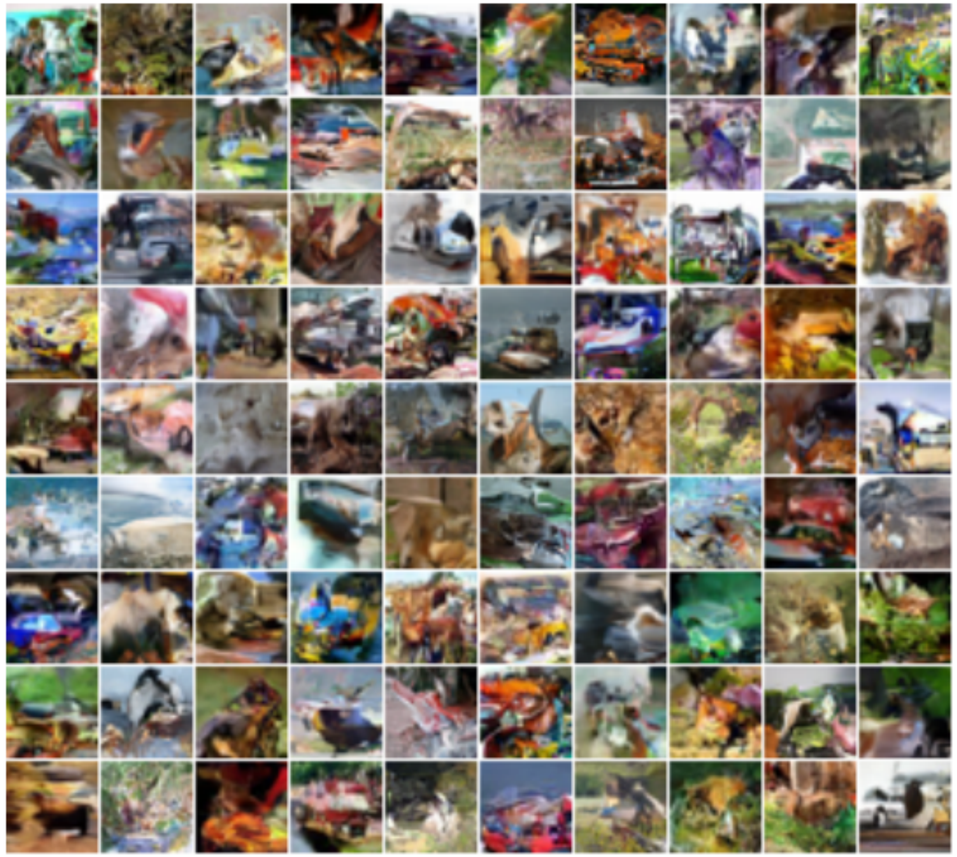


Generation is still slow (sequential), but learning is faster

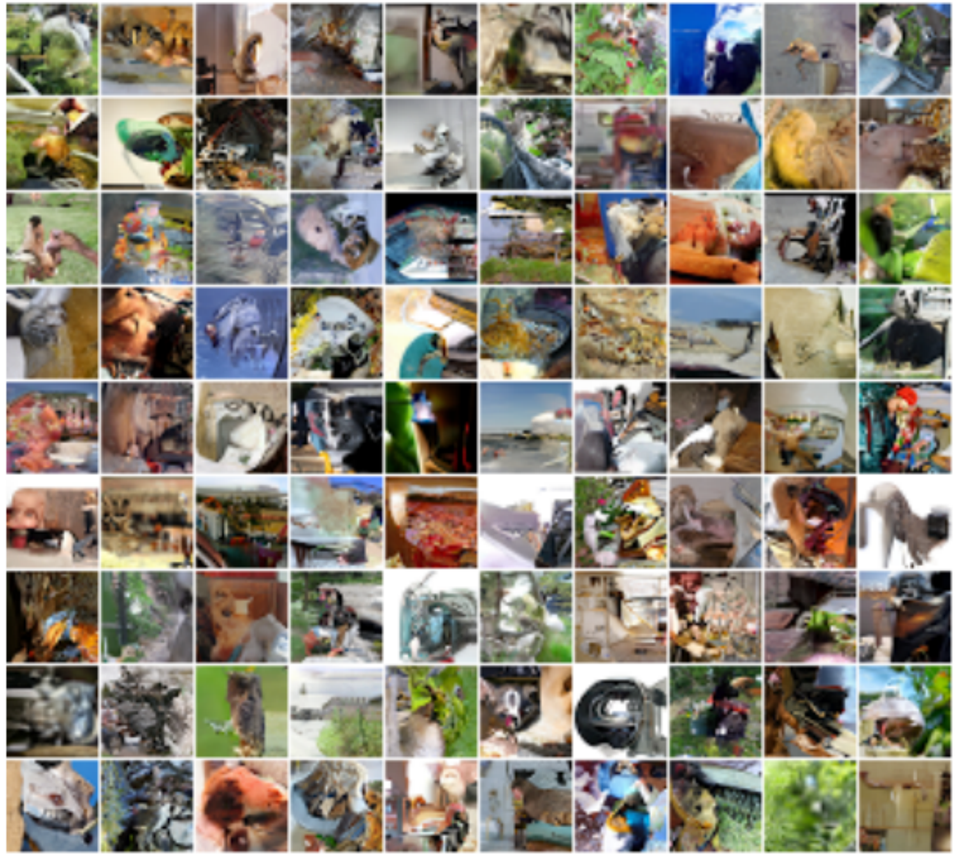
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Generated Samples

[van der Oord et al., 2016]



32x32 **CIFAR-10**



32x32 **ImageNet**

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

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…

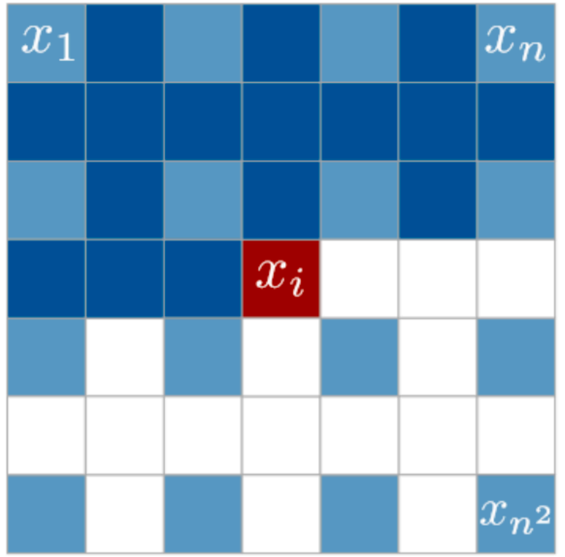
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Multi-scale PixelRNN

[van der Oord et al., 2016]

Take sub-sampled pixels as additional input pixels

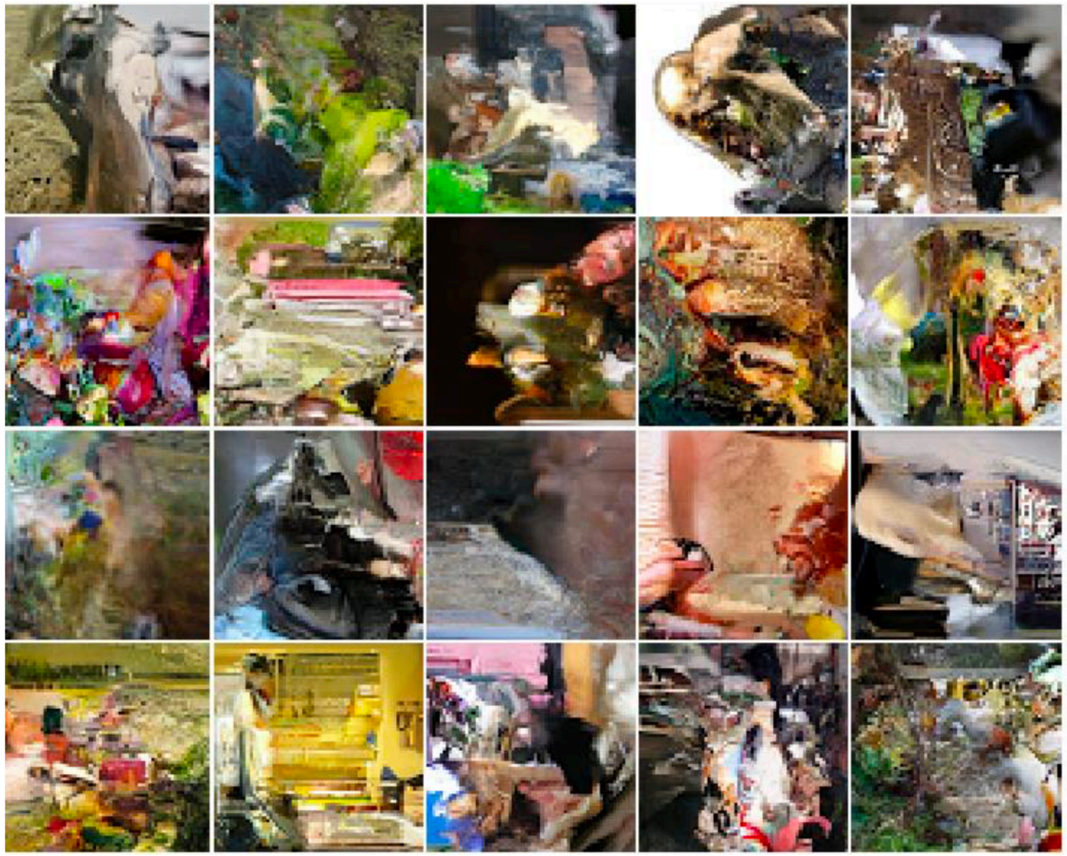
Can capture better global information (more visually coherent)

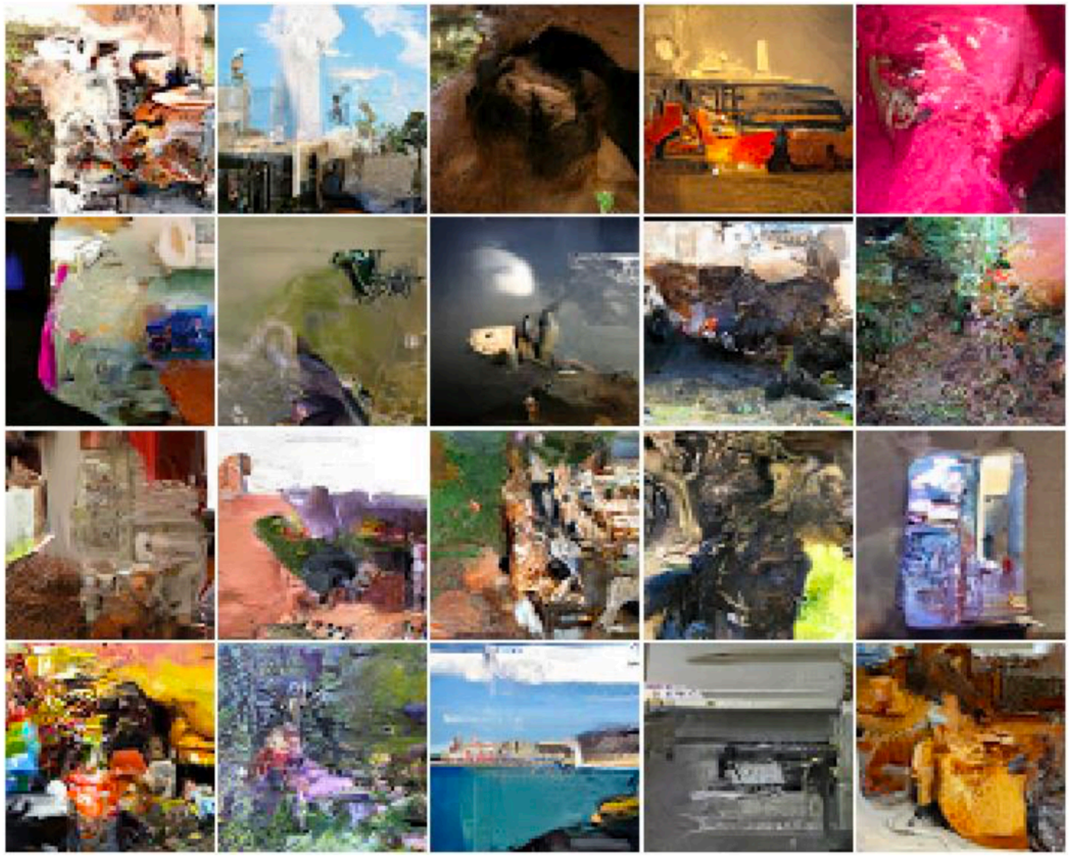


* slide from Hsiao-Ching Chang, Ameya Patil, Anand Bhattad

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 $\mathbf{h}$

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

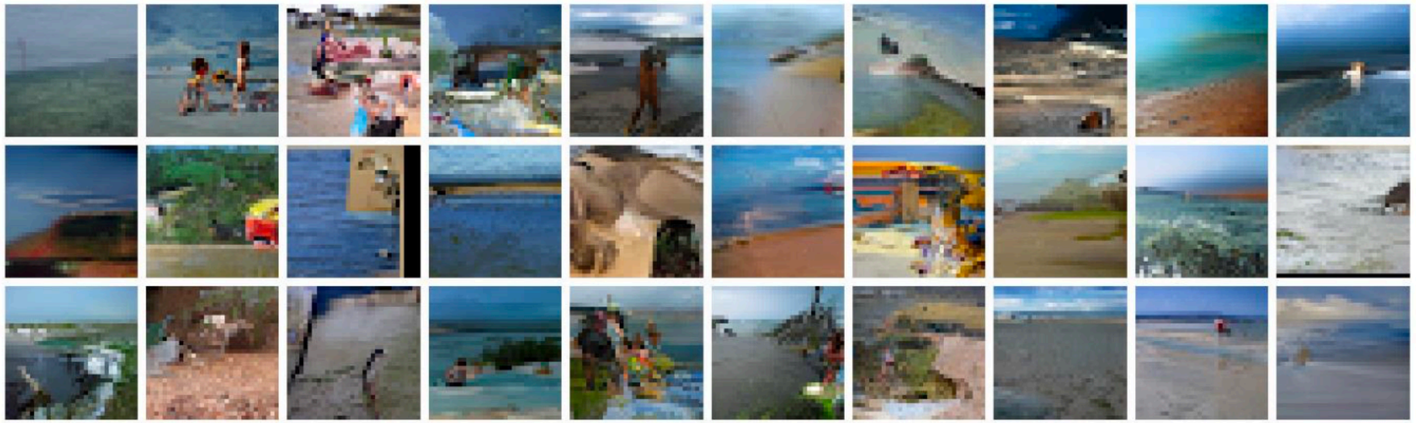
$$p(\mathbf{x}|\mathbf{h}) = p(x_1, x_2, ..., x_{n^2}|\mathbf{h})$$

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

Conditional Image Generation

[van der Oord et al., 2016]

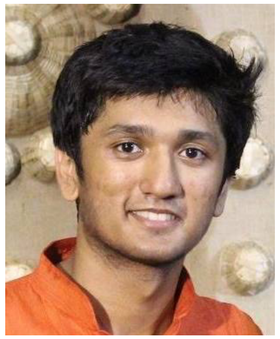
African elephant



Sandbar

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

Attention RNN: Structured Spatial Attention Mechanism

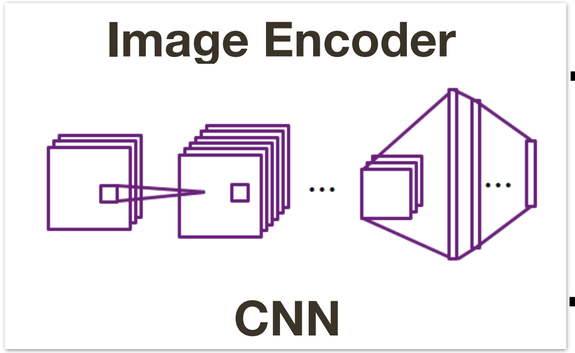


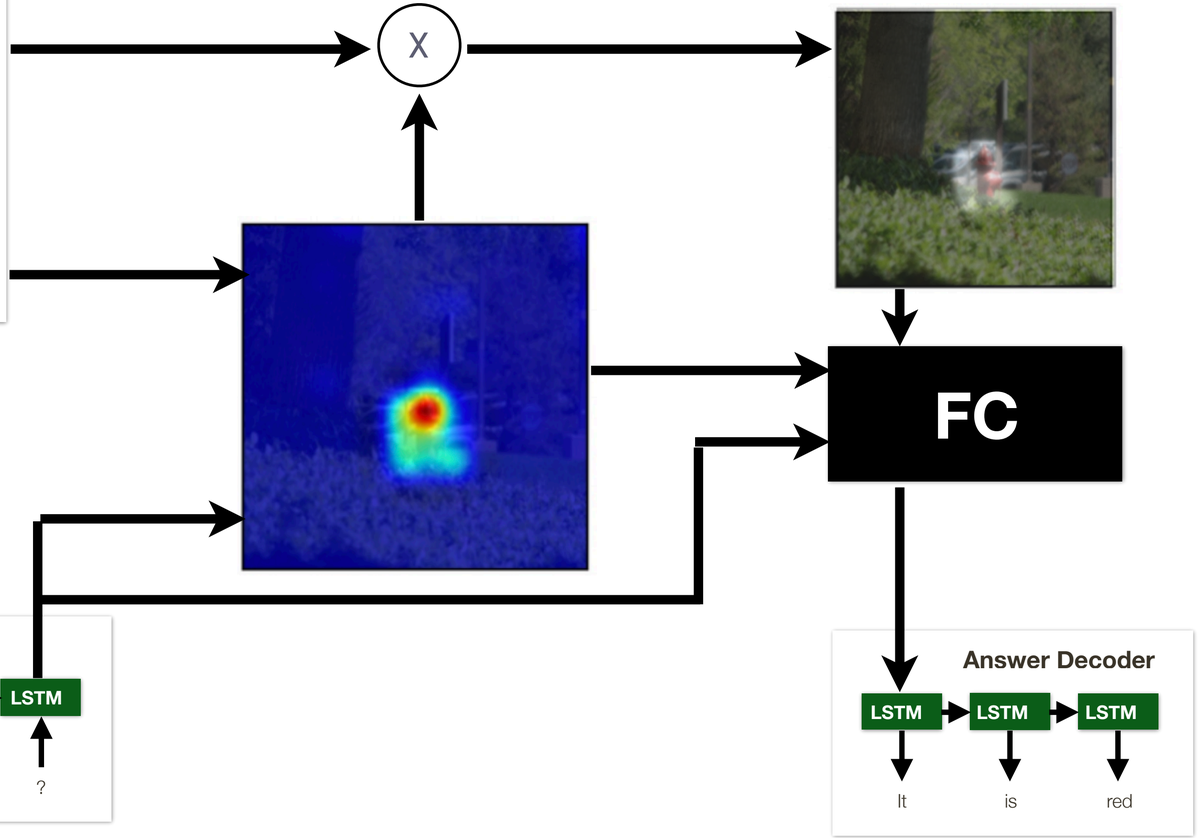
Siddhesh Khandelwal

Leonid Sigal

Motivation

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





Q: What color is a hydrant?

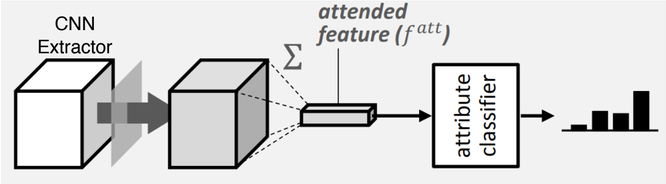
A: It is red

Motivation

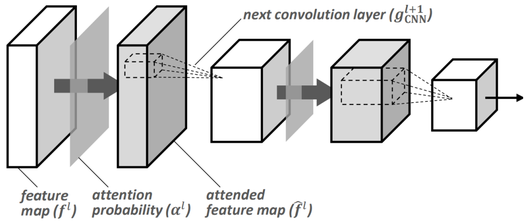
Attention is widely used in vision: helps identify relevant regions of the image

Attention is **applied**:

On the **output of CNN** architecture



After **each CNN** layer



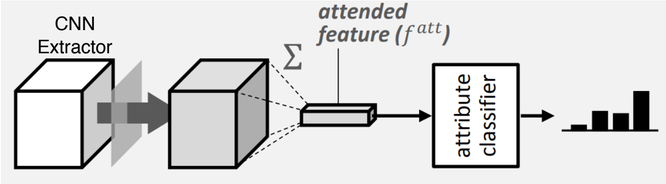
(Image taken from Seo et al. BMVC 2018)

Motivation

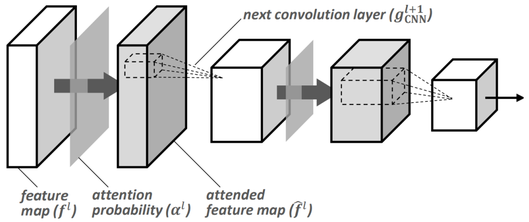
Attention is widely used in vision: helps identify relevant regions of the image

Attention is **applied**:

On the **output of CNN** architecture



After **each CNN** layer



(Image taken from Seo et al. BMVC 2018)

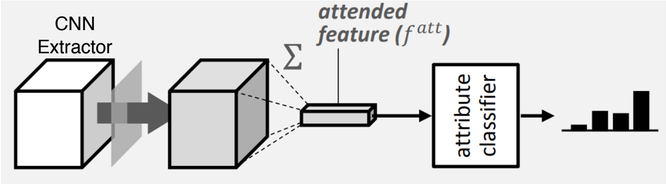
Existing attention mechanisms are either **global** or **local**

Motivation

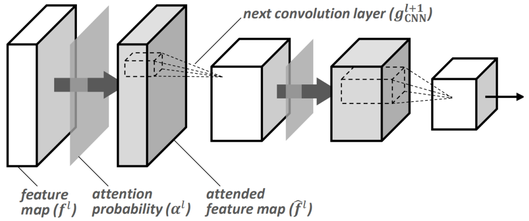
Attention is widely used in vision: helps identify relevant regions of the image

Attention is **applied**:

On the **output of CNN** architecture

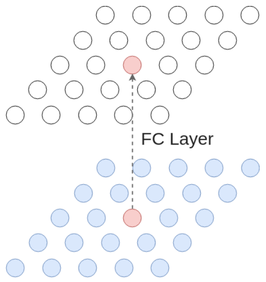


After **each CNN** layer



(Image taken from Seo et al. BMVC 2018)

Existing attention mechanisms are either **global** or **local**



Global:

Pros:

— Can model arbitrary context

Cons:

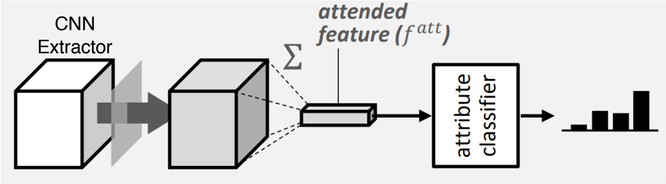
— Cannot be applied at high res.

Motivation

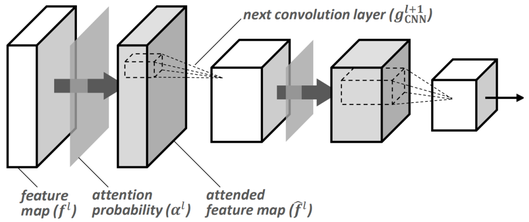
Attention is widely used in vision: helps identify relevant regions of the image

Attention is **applied**:

On the **output of CNN** architecture

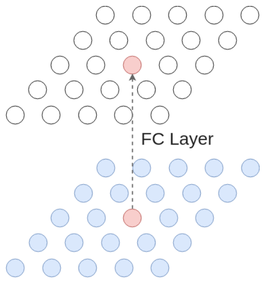


After **each CNN** layer



(Image taken from Seo et al. BMVC 2018)

Existing attention mechanisms are either **global** or **local**



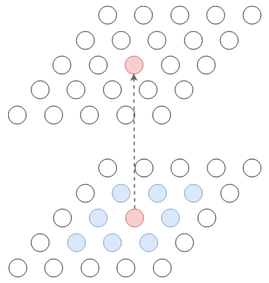
Global:

Pros:

— Can model arbitrary context

Cons:

— Cannot be applied at high res.



Local:

Pros:

— Can be applied at any resolution

Cons:

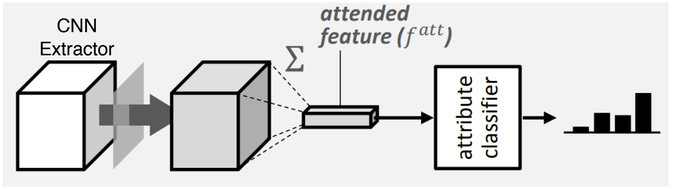
— Can only model local context

Motivation

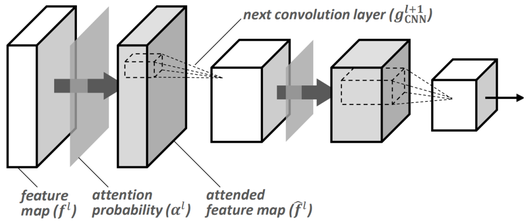
Attention is widely used in vision: helps identify relevant regions of the image

Attention is **applied**:

On the **output of CNN** architecture

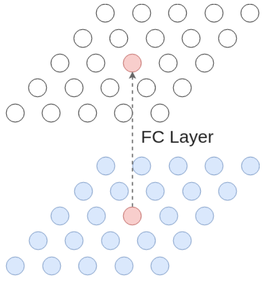


After **each CNN** layer



(Image taken from Seo et al. BMVC 2018)

Existing attention mechanisms are either **global** or **local**



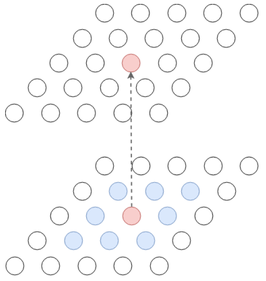
Global:

Pros:

— Can model arbitrary context

Cons:

— Cannot be applied at high res.



Local:

Pros:

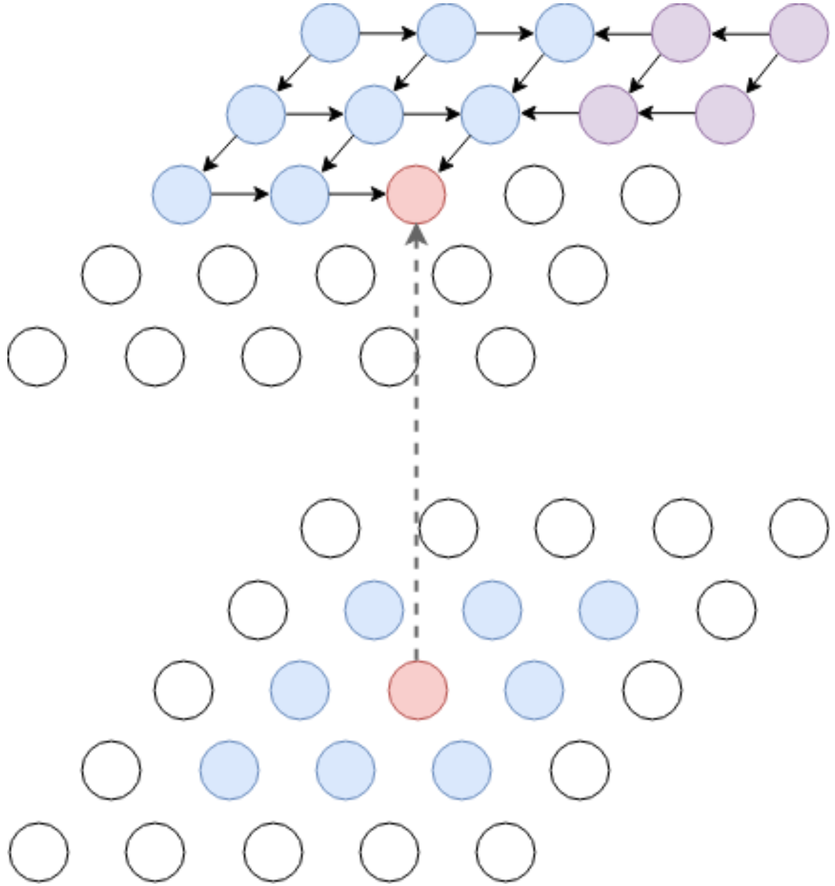
— Can be applied at any resolution

Cons:

— Can only model local context

Neither can account (explicitly) for **structure** in the attention variables

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

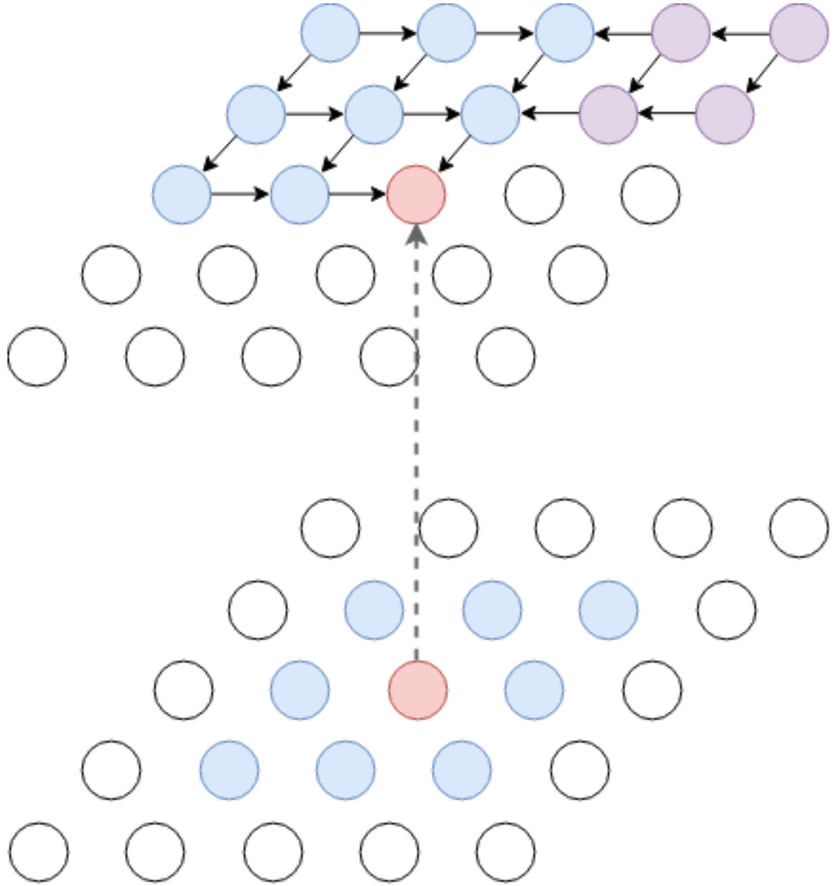
— Previously generated attention values

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

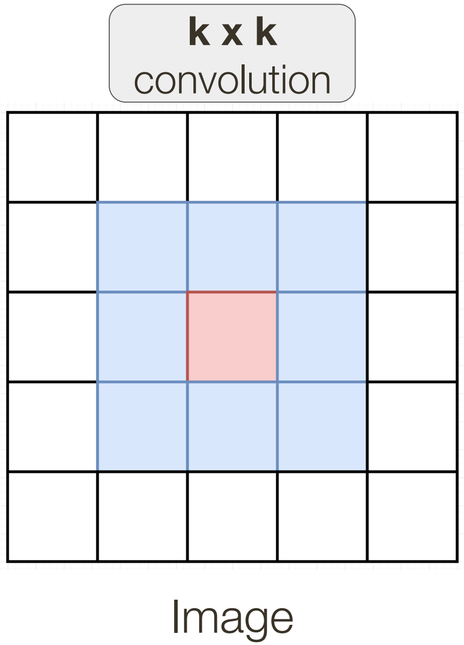


AttentionRNN: Structured Spatial Attention

Each **attention value** depends on

— **Local image context**

— Previously generated attention values

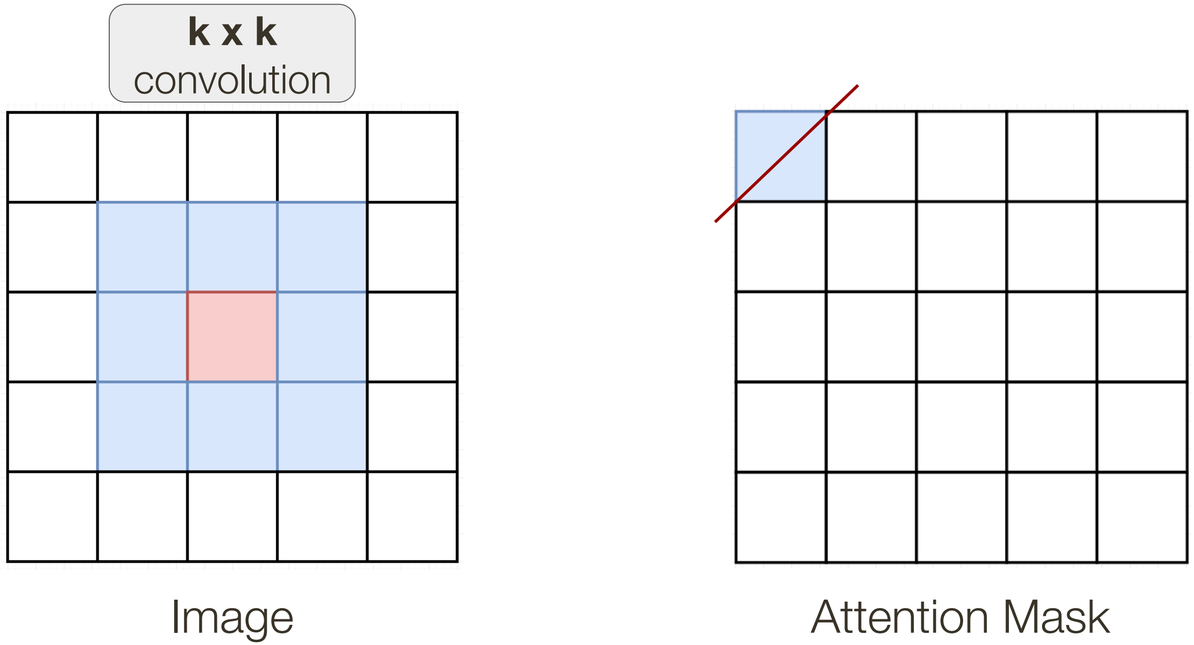


AttentionRNN: Structured Spatial Attention

Each **attention value** depends on

— Local image context

— **Previously generated attention values**

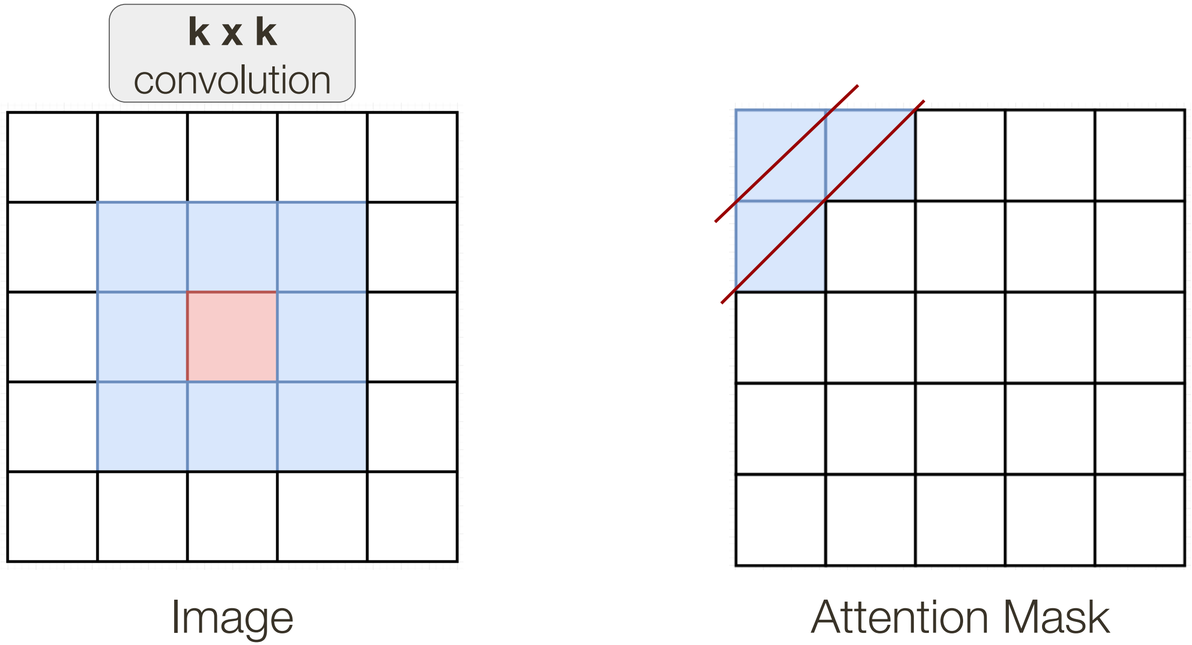


AttentionRNN: Structured Spatial Attention

Each **attention value** depends on

— Local image context

— **Previously generated attention values**

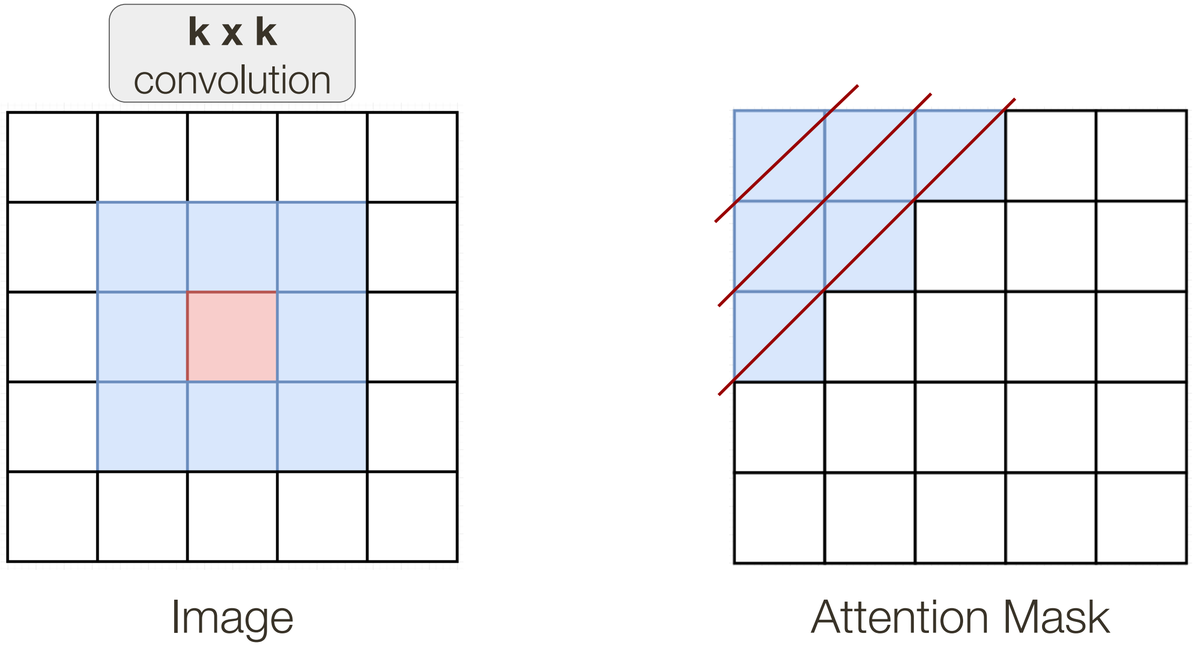


AttentionRNN: Structured Spatial Attention

Each **attention value** depends on

— Local image context

— **Previously generated attention values**

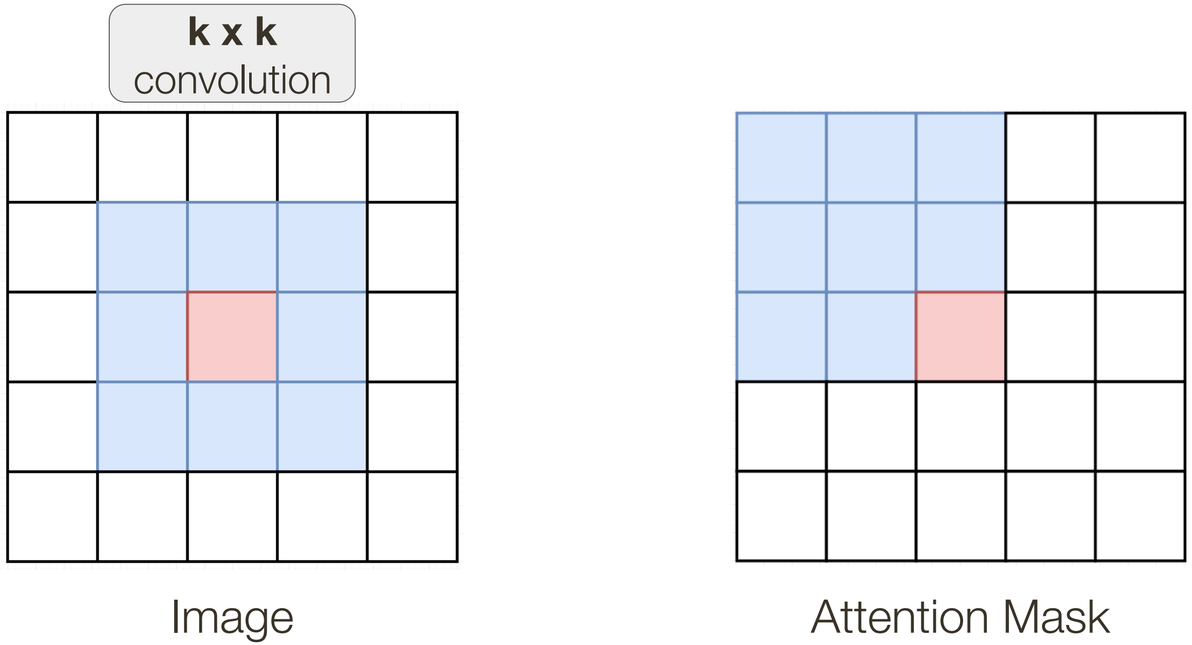


AttentionRNN: Structured Spatial Attention

Each **attention value** depends on

— Local image context

— **Previously generated attention values**

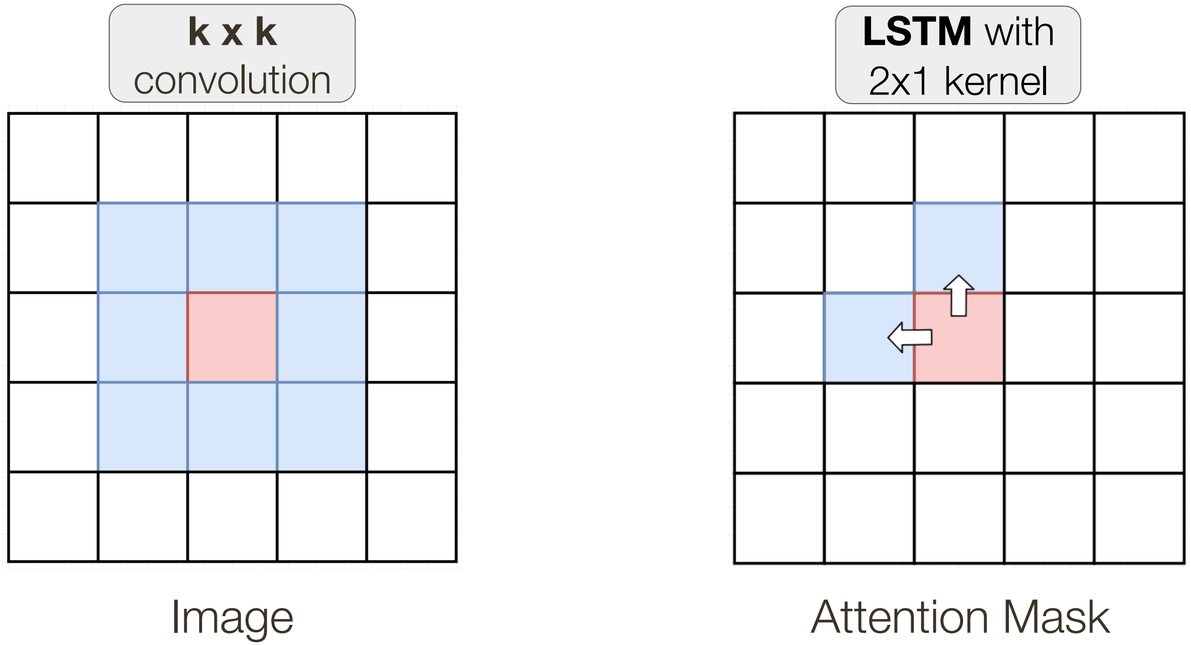


AttentionRNN: Structured Spatial Attention

Each **attention value** depends on

— Local image context

— Previously generated attention values

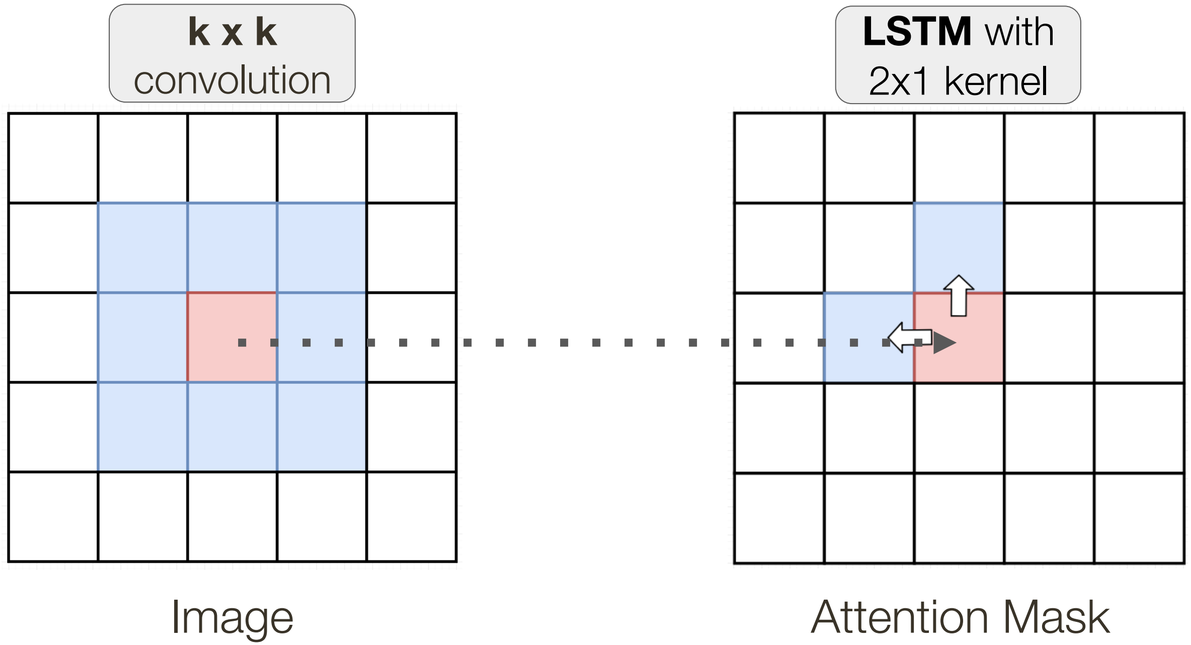


AttentionRNN: Structured Spatial Attention

Each **attention value** depends on

— Local image context

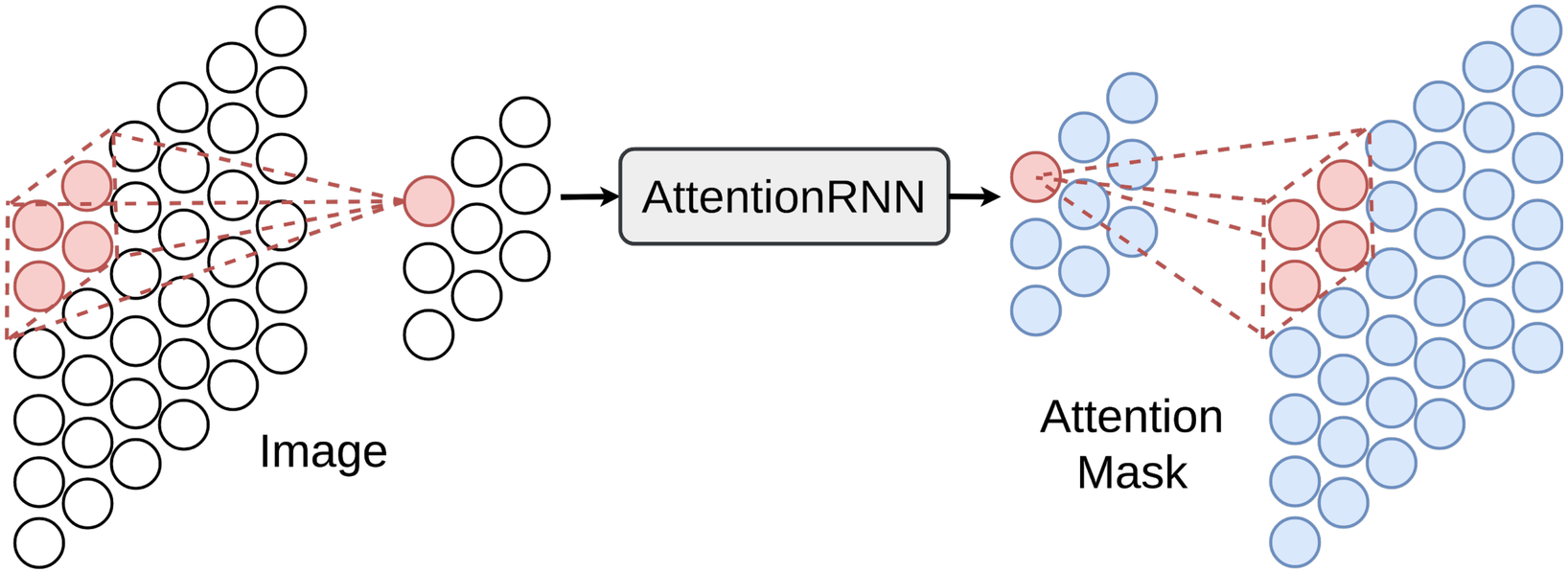
— Previously generated attention values



Block Attention RNN: Scalability

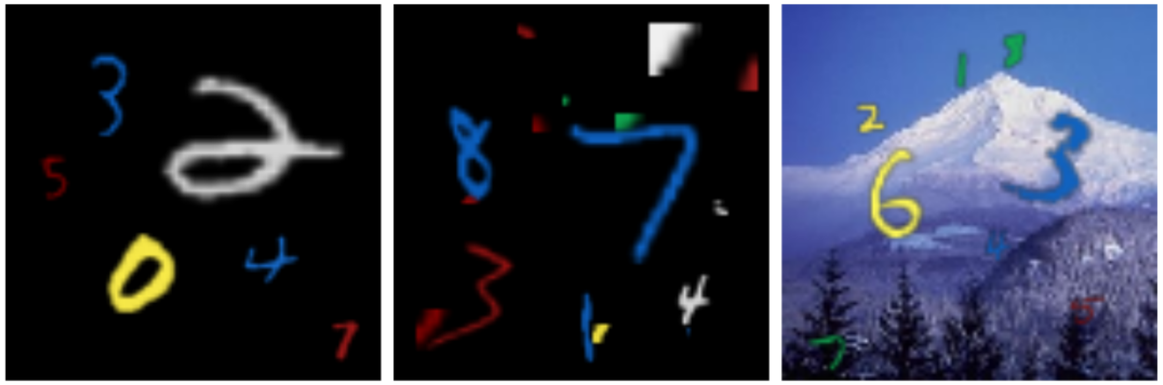
LSTMs don’t work well over large sequences (say beyond 50 x 50)

We utilize symmetric down-sampling and up-sampling scheme to deal with larger resolution attention maps



Experiments: Visual Attribute Prediction

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



(a) MREF (b) MDIST (c) MBG

	MREF	MDIST	MBG
CNN+SAN [23]	81.28	78.06	52.36
CNN+¬CTX [23]	94.95	90.06	66.83
CNN+CTX [23]	97.92	95.80	79.26
CNN+ARNN	**99.11**	**97.04**	**86.07**

Global (only top layer)

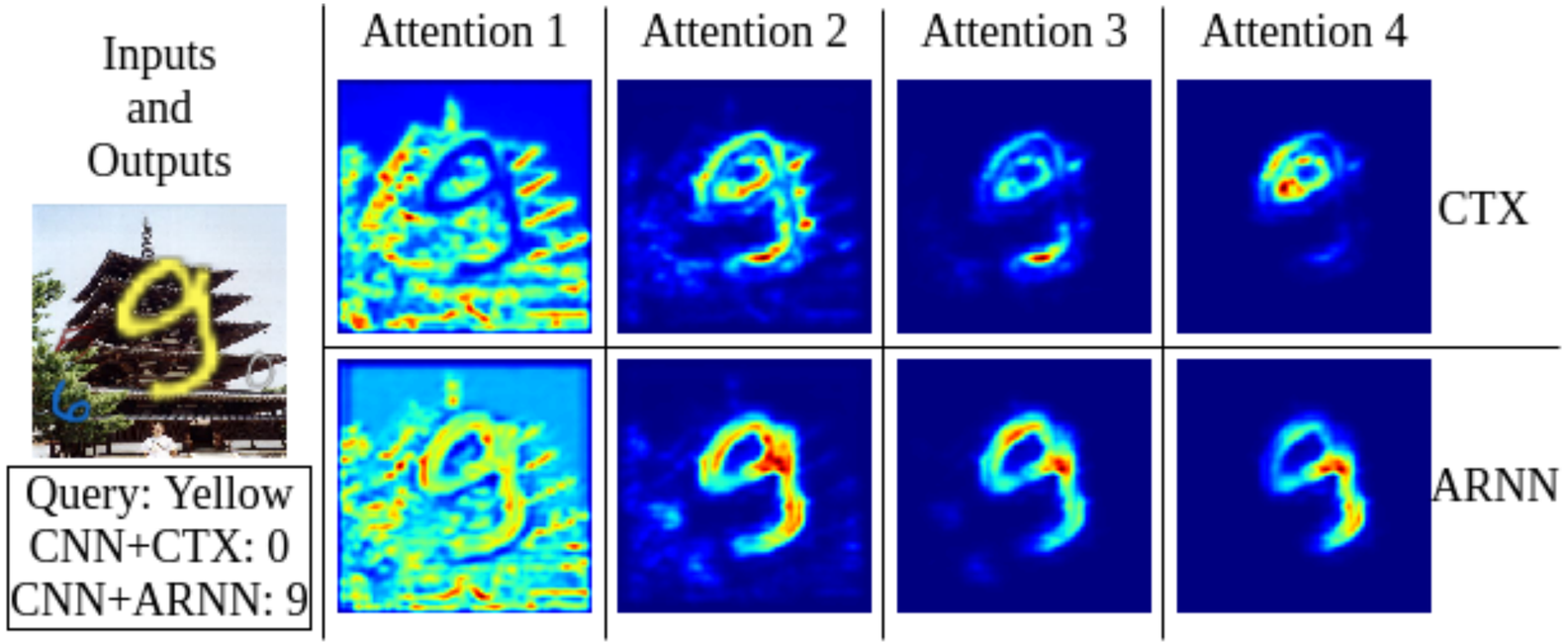
Local (all layers)

Local with context (all layers)

Local with context + structure (all layers)

Experiments: Visual Digit Prediction

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



	SAN	¬ CTX	CTX	ARNN
Correctness	0.1258	0.2017	0.2835	**0.3729**

Experiments: Visual Question Answering

Task: Visual Question Answering

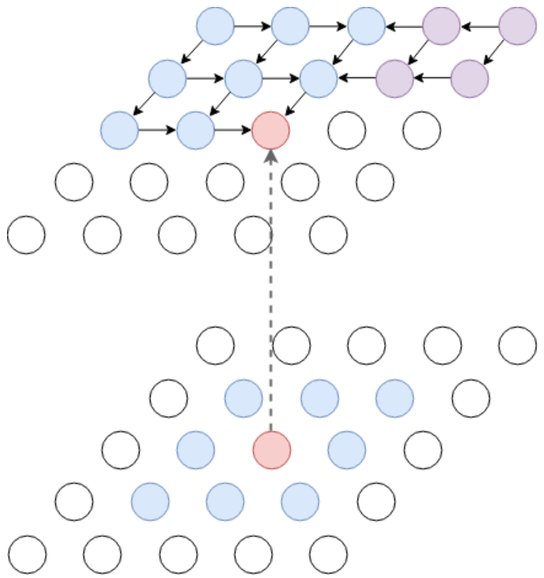
	Accuracy
Prior	24.04
HCA [18]	51.27
HCA+ARNN	**53.23**

HCA — Hierarchical Co-attention

[Lu et al., NIPS 17]

Conclusions

— Modeling **structure** among the latent attention variables is useful



Variational Autoencoders (VAE)

So far …

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

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

* 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):

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

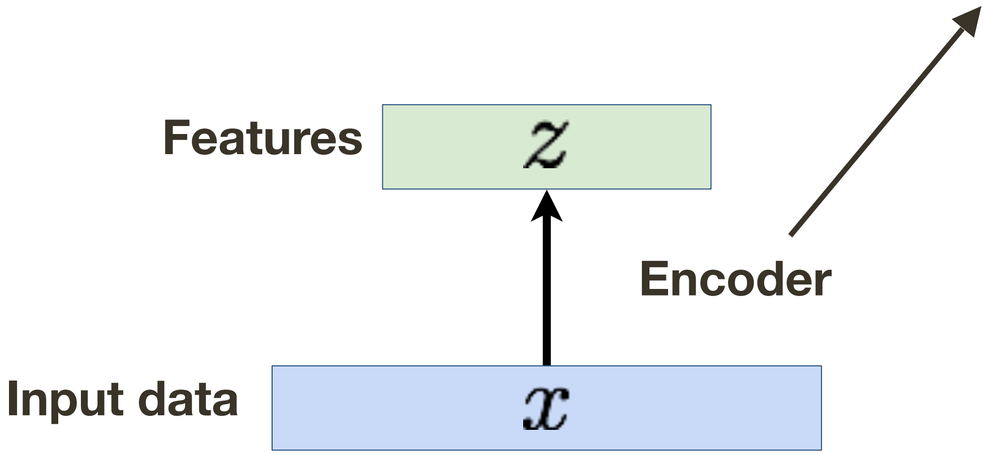
cannot optimize directly, derive and optimize lower bound of likelihood instead

* 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

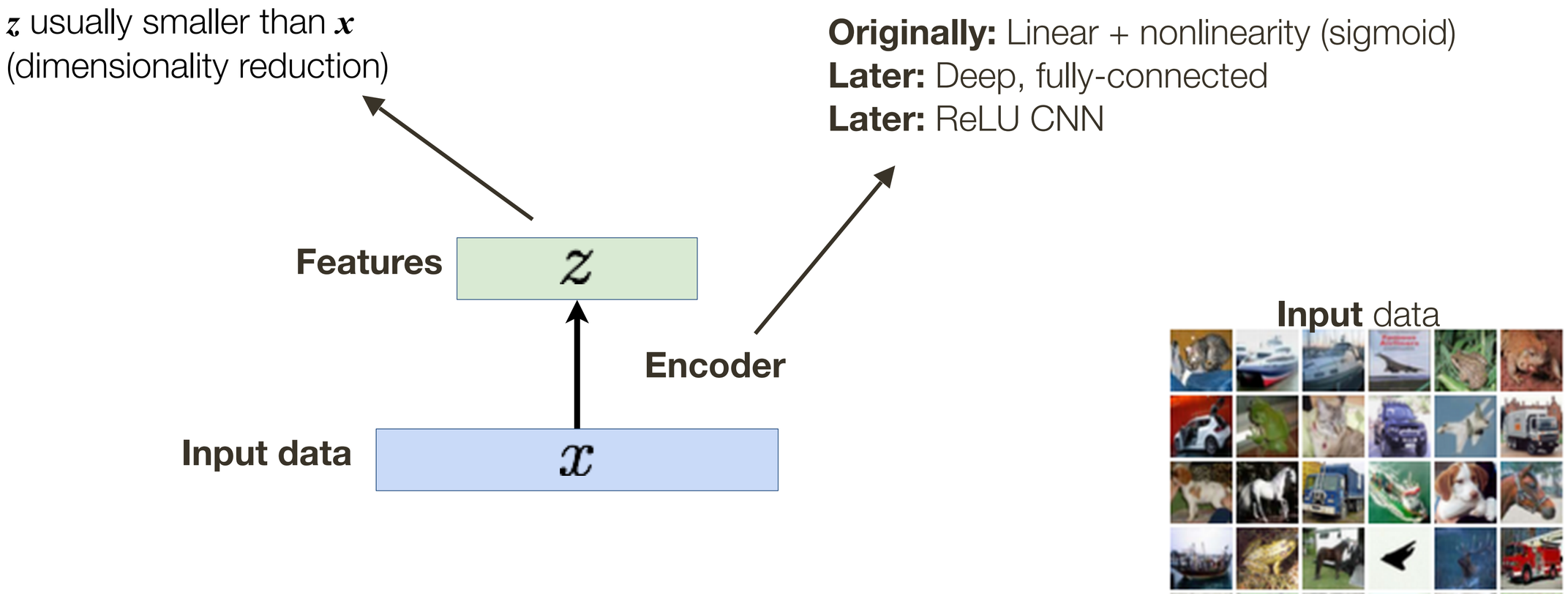


Input data

* 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



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

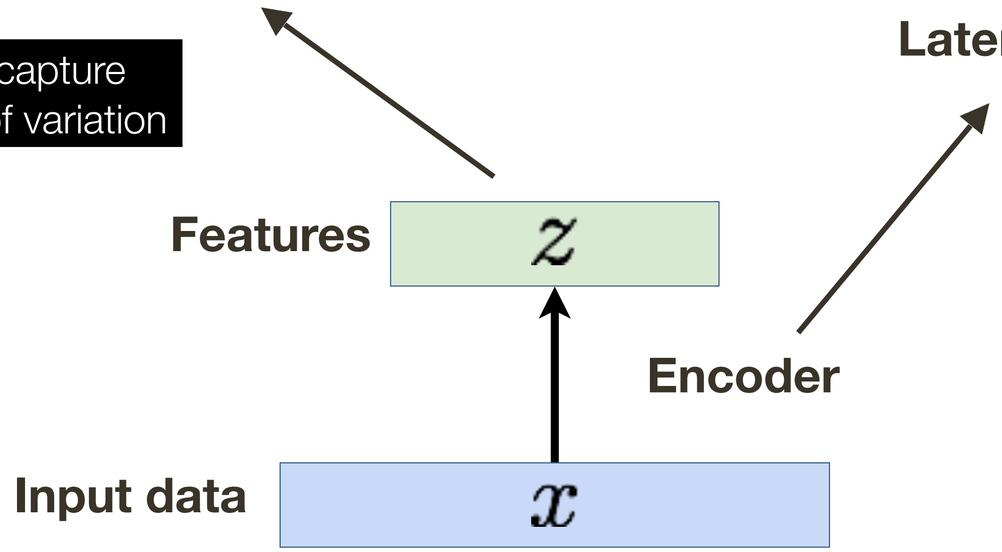
Autoencoders Reminder …

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

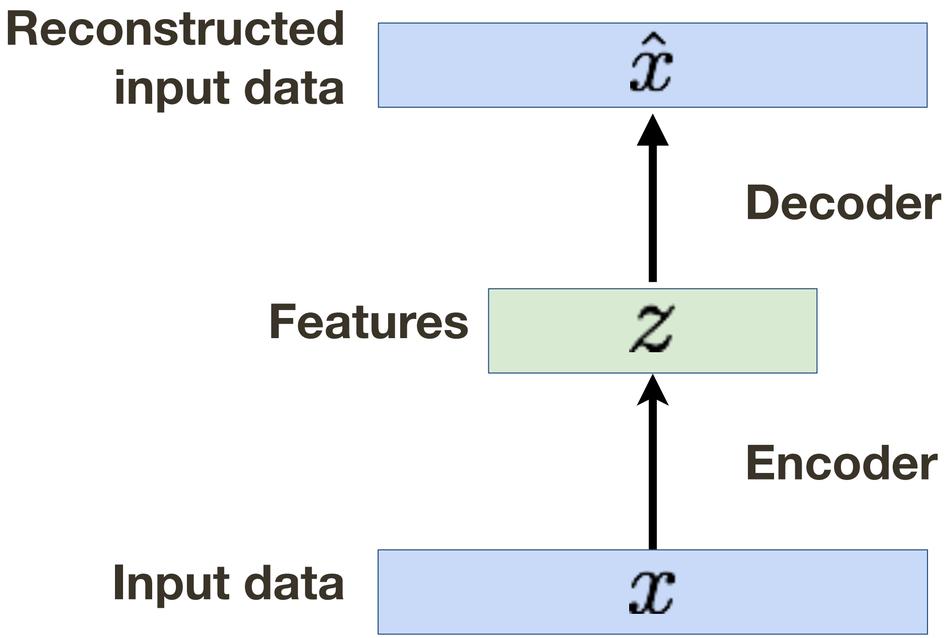


Input data

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

Autoencoders Reminder …

Train such that features can reconstruct original data best they can

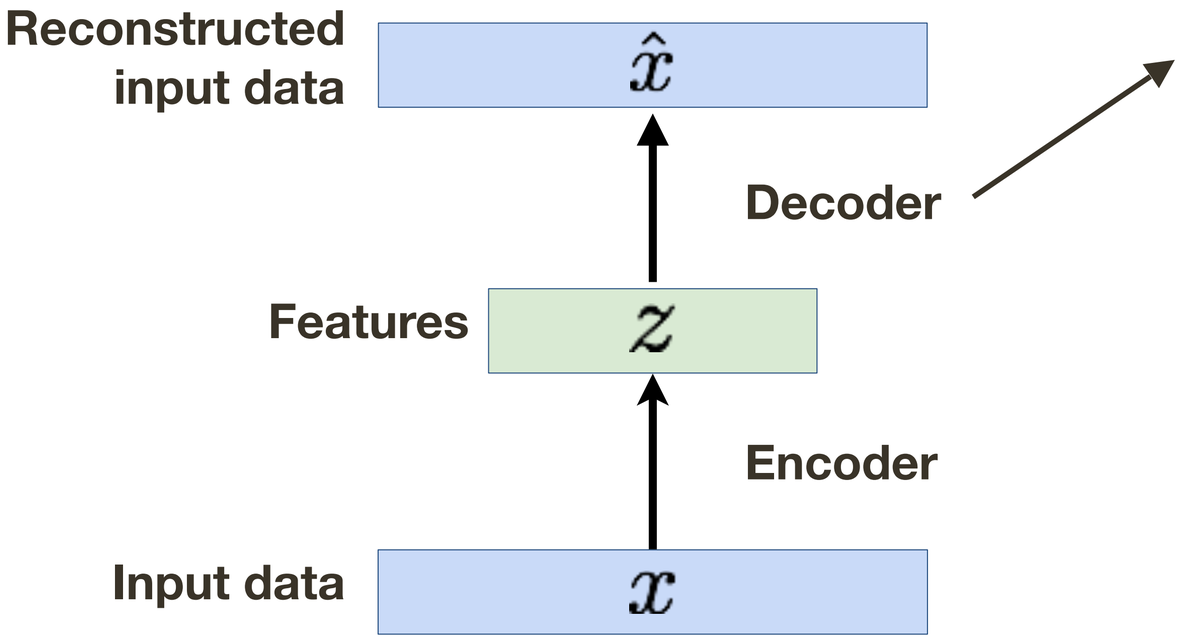


Input data

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

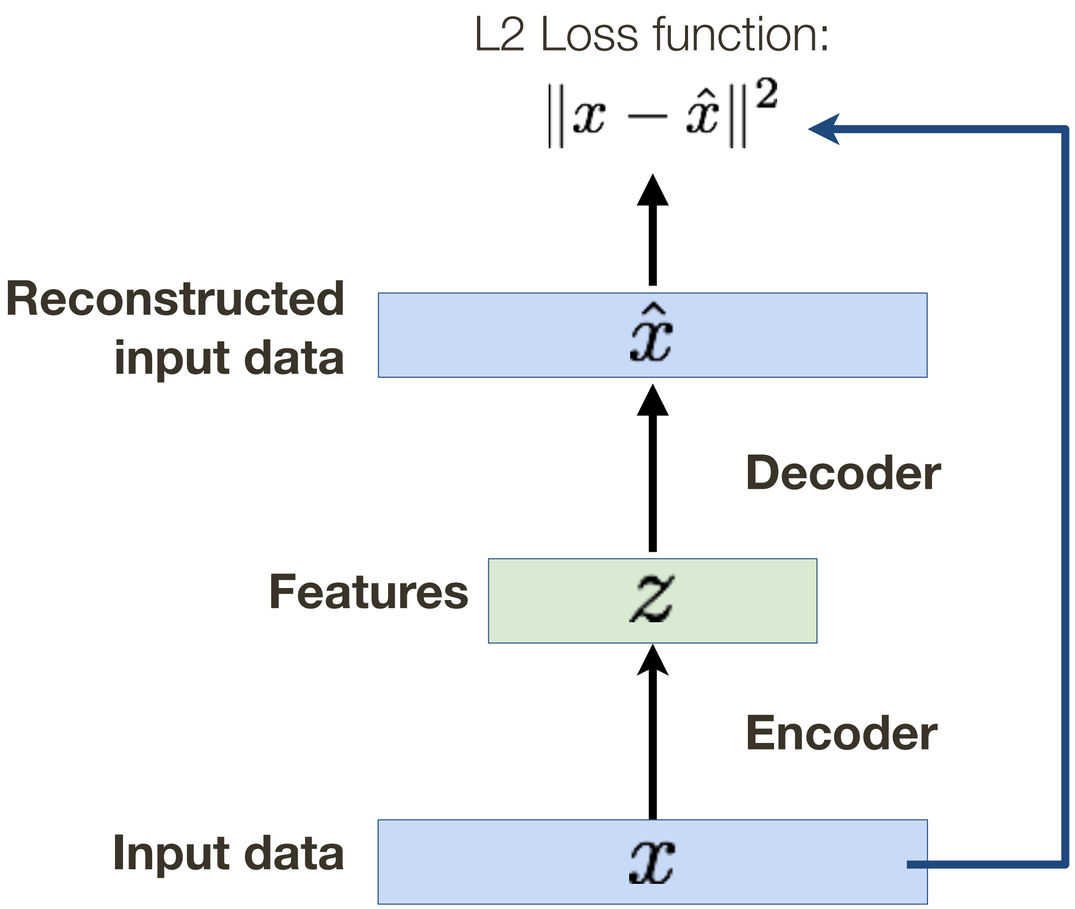
Autoencoders Reminder …

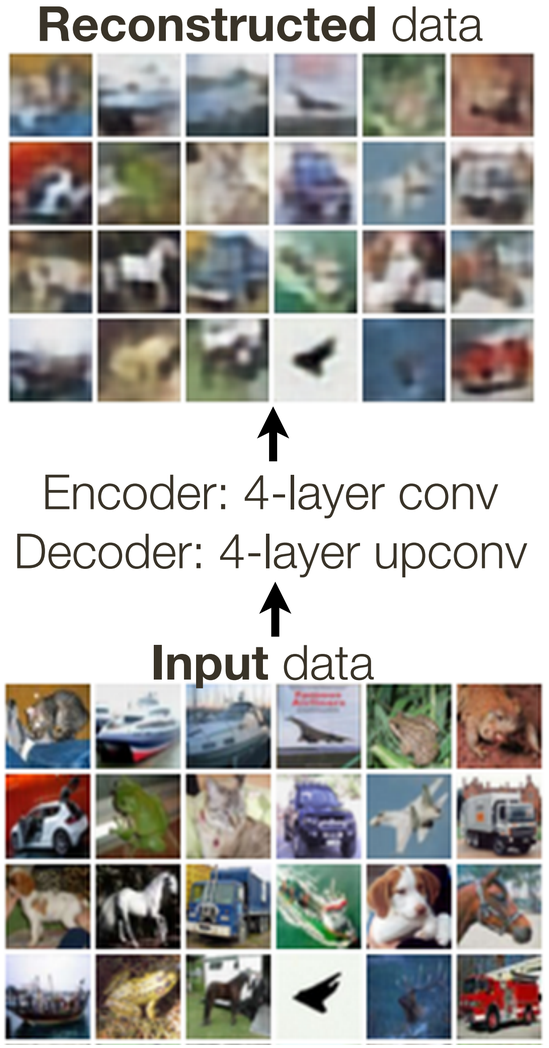
Train such that features can reconstruct original data best they can



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

Autoencoders Reminder …

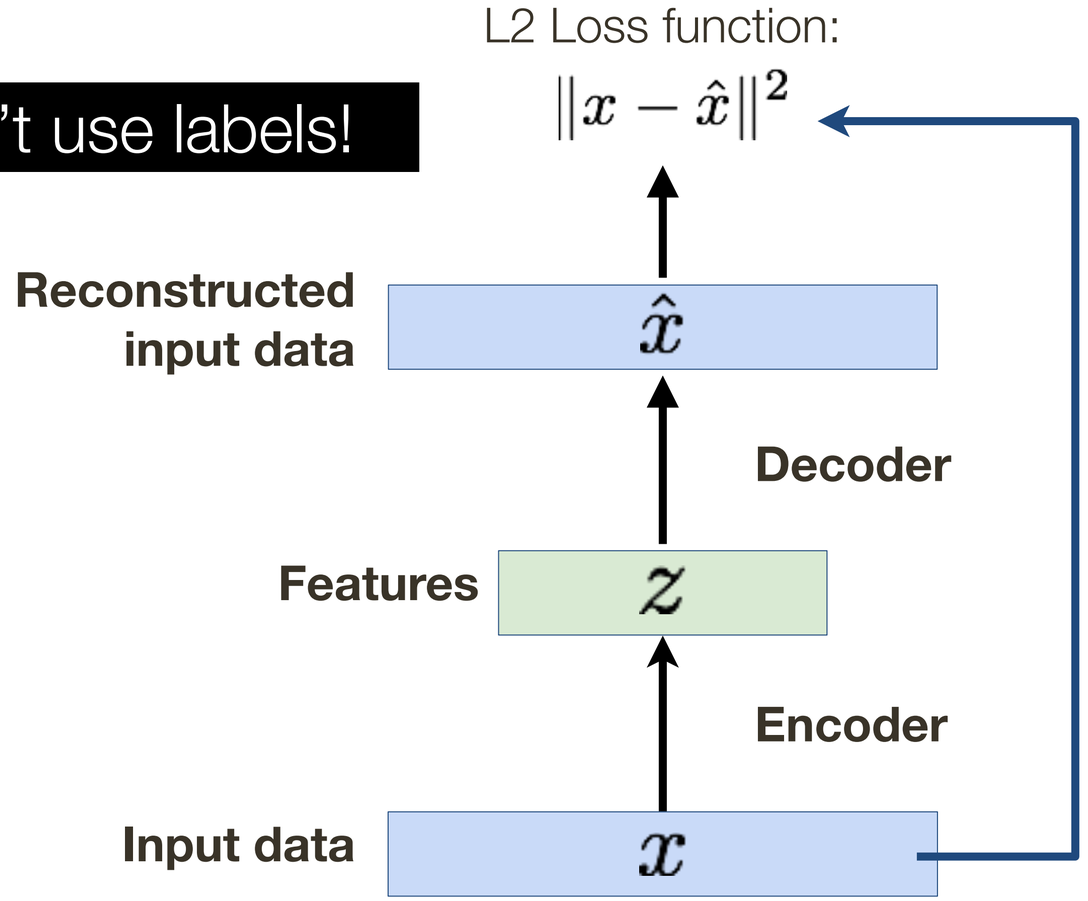




* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

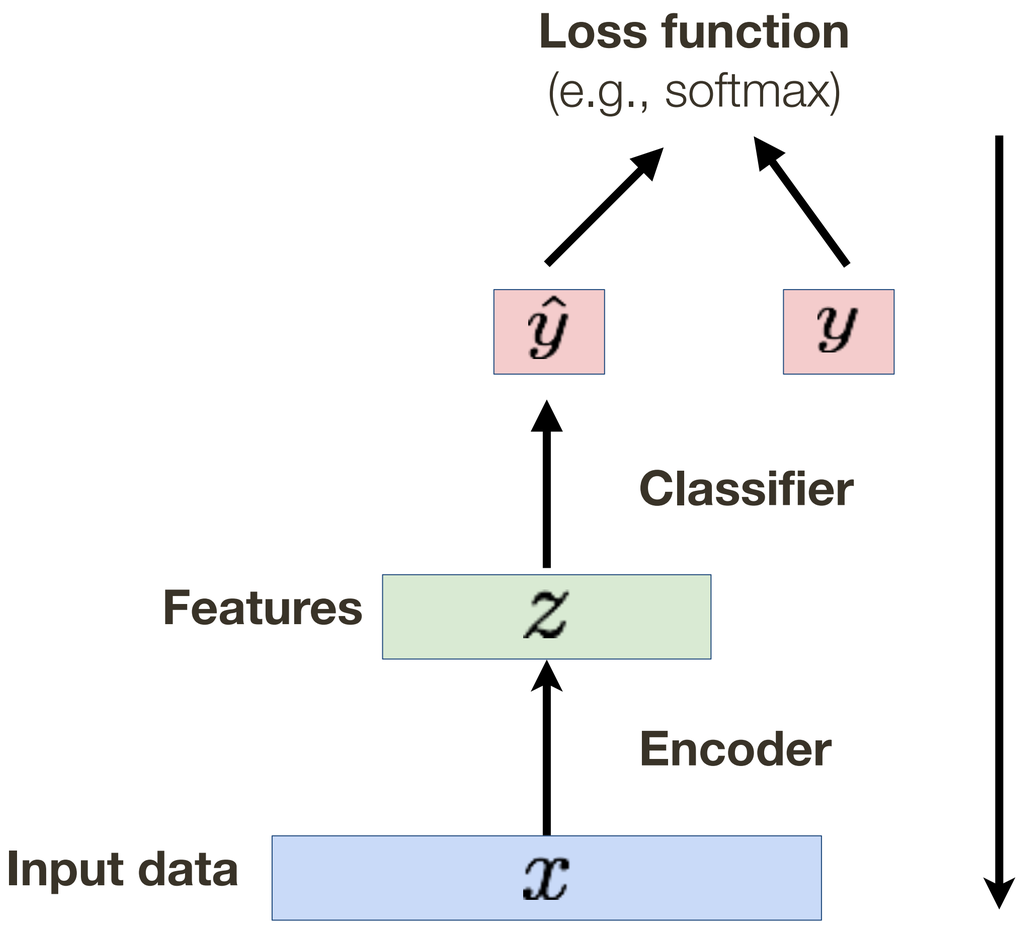
Autoencoders Reminder …

Doesn't use labels!



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

Autoencoders Reminder …



Fine-tune encoder jointly with classifier

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

bird plane
dog deer truck

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

Variational Autoencoders

[Kingma and Welling, 2014]

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

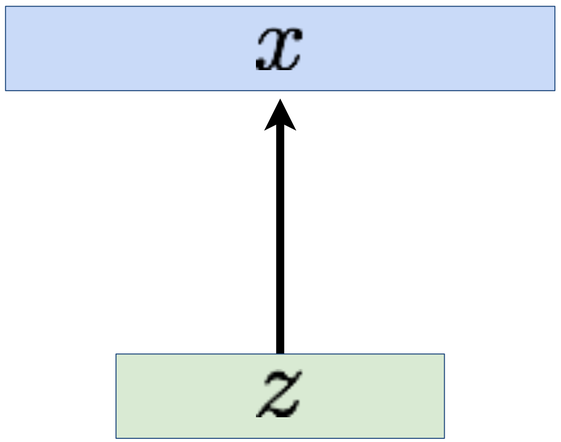
Assume training data is generated from underlying unobserved (latent) representation z

Sample from true **conditional**

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

Sample from true **prior**

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



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

Variational Autoencoders

[Kingma and Welling, 2014]

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

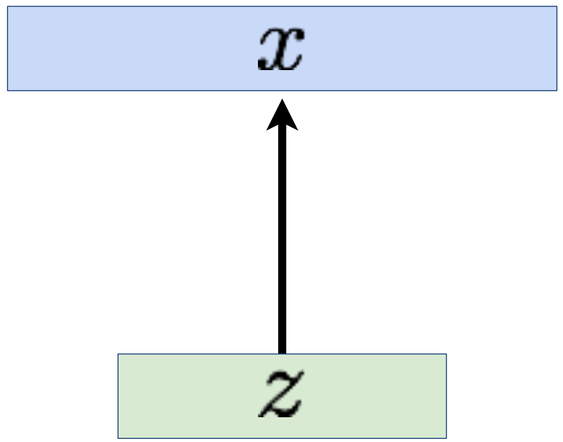
Assume training data is generated from underlying unobserved (latent) representation z

Sample from true **conditional**

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

Sample from true **prior**

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



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

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

Variational Autoencoders

[Kingma and Welling, 2014]

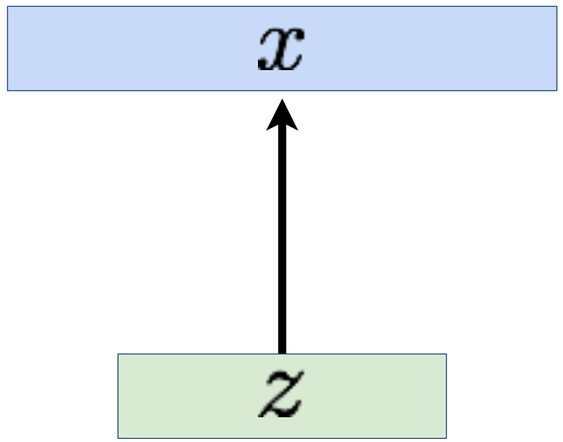
We want to **estimate the true parameters** θ^* of this generative model

Sample from true **conditional**

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

Sample from true **prior**

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



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

Variational Autoencoders

[Kingma and Welling, 2014]

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

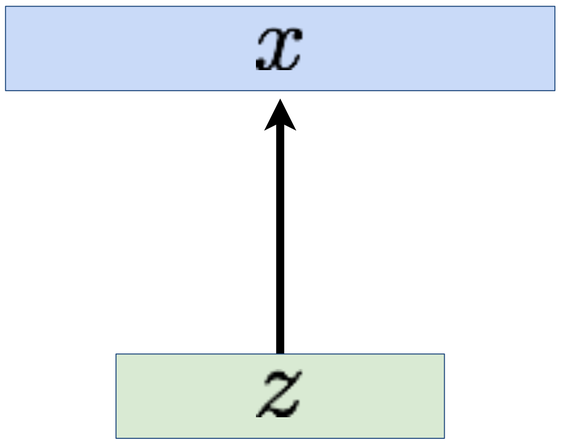
How do we **represent** this model?

Sample from true **conditional**

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

Sample from true **prior**

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



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

Variational Autoencoders

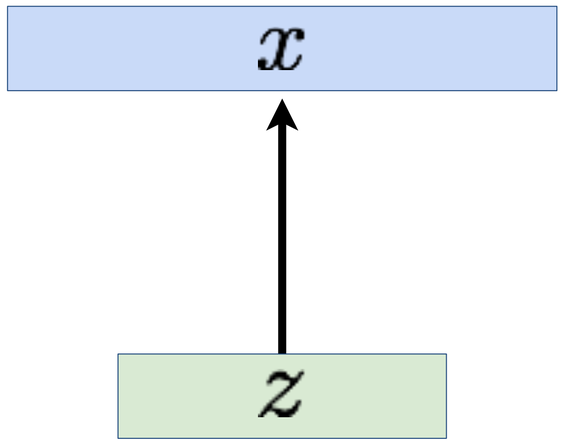
[Kingma and Welling, 2014]

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**

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

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

Reasonable for latent attributes, e.g., pose, amount of smile

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

Variational Autoencoders

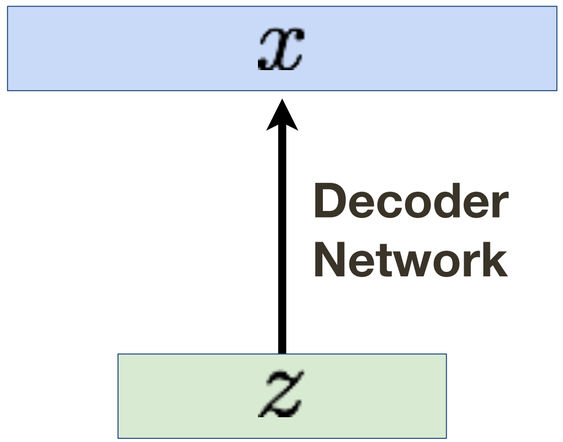
[Kingma and Welling, 2014]

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**

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

Choose prior $p(\boldsymbol{z})$ to be simple, e.g., Gaussian

Reasonable for latent attributes, e.g., pose, amount of smile

Conditional $p(\boldsymbol{x}|\boldsymbol{z})$ is complex (generates image)

Represent with Neural Network

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

Variational Autoencoders

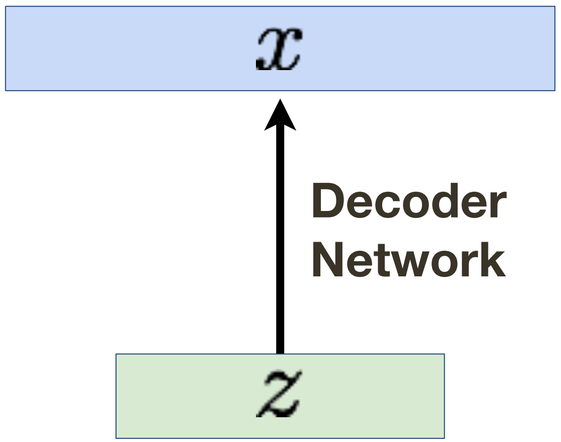
[Kingma and Welling, 2014]

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**

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

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

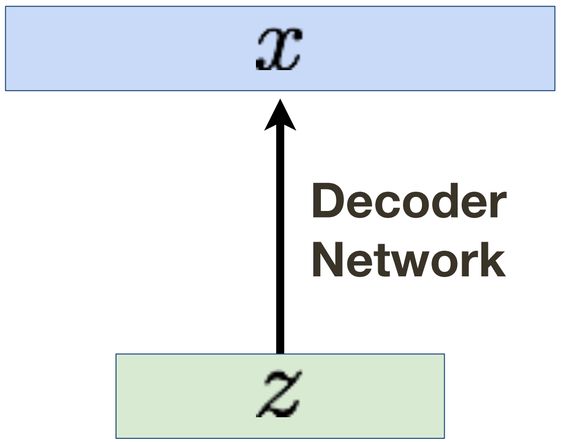
Variational Autoencoders

[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)

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

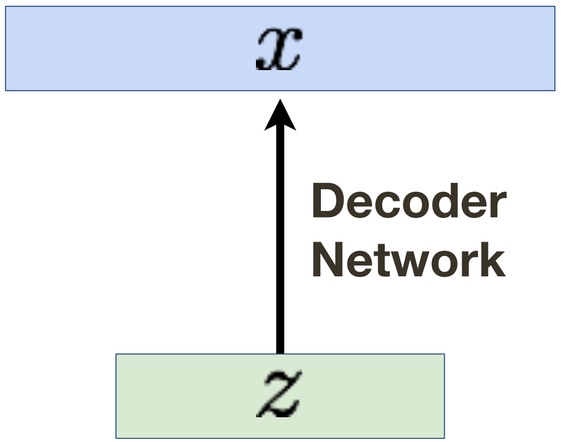
Variational Autoencoders

[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?

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

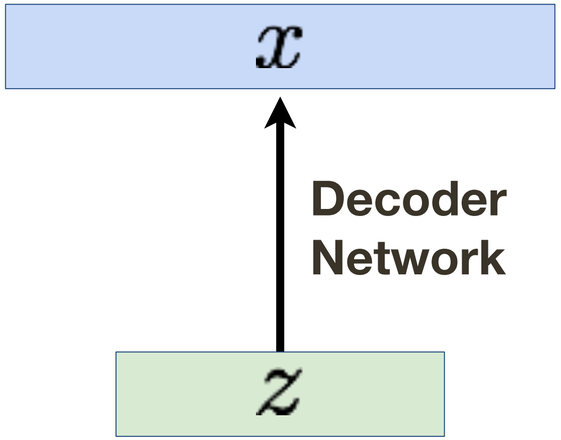
Variational Autoencoders

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

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

Intractability in Variational Autoencoder

[Kingma and Welling, 2014]

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

🙂

Simple **Gaussian** Prior

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

Intractability in Variational Autoencoder

[Kingma and Welling, 2014]

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

Decoder Neural Network

Simple **Gaussian** Prior

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

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

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

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

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

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

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

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

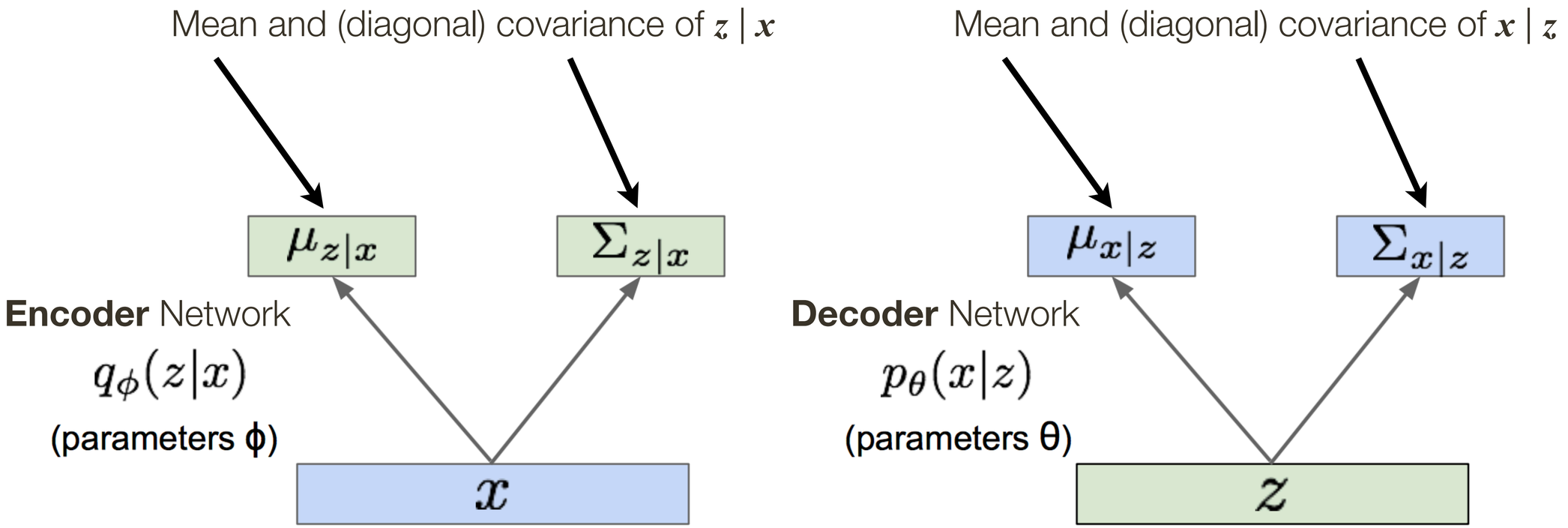
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Optional subtitle

Variational Autoencoder

[Kingma and Welling, 2014]

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



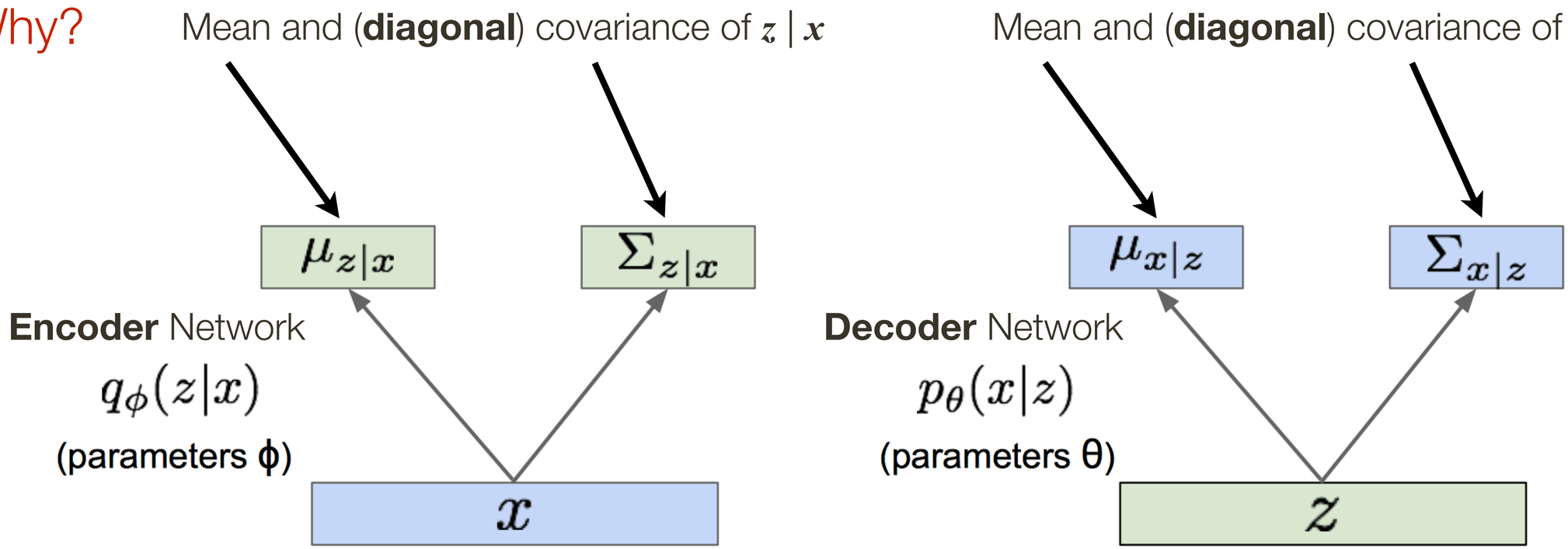
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder

[Kingma and Welling, 2014]

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

Why?

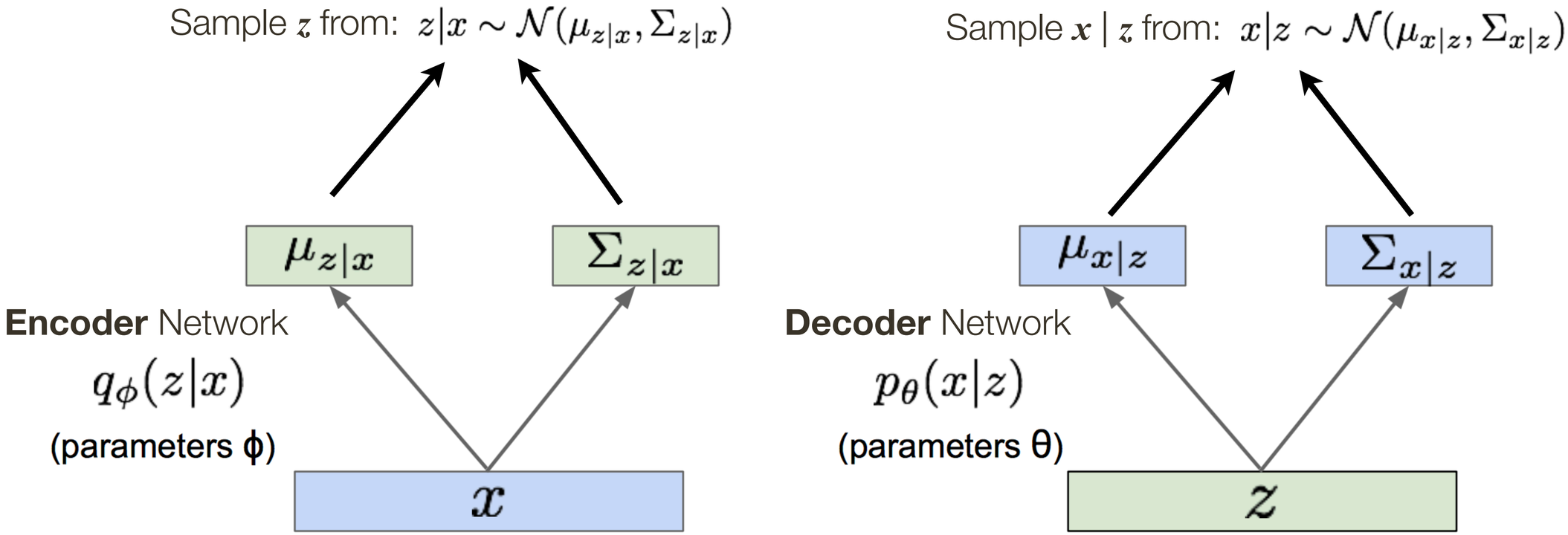


* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder

[Kingma and Welling, 2014]

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



* 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:

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

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

* 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:

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

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

* 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:

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

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

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

* 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:

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

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

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

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

* 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:

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

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

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

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

$$= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\text{Expectation with respect to z}} + \underbrace{D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))}_{\text{(using encoder network) leads to nice KL terms}}$$

* 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.

* 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:

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

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

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

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

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

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

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

Variational Autoencoder

[Kingma and Welling, 2014]

Derivation of lower bound of the data likelihood

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

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

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

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

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

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

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

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

[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:

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

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

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

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

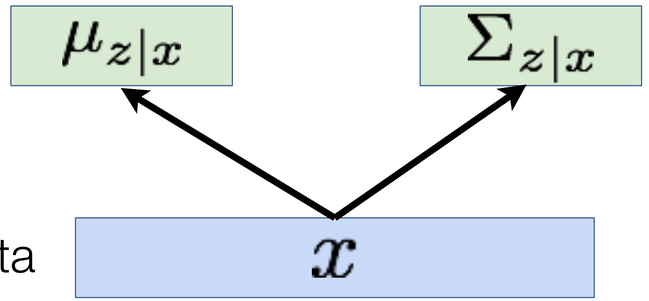
Putting it all together:
maximizing the likelihood lower bound

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

Encoder network

$q_\phi(z|x)$

Input Data



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

Variational Autoencoder: Learning

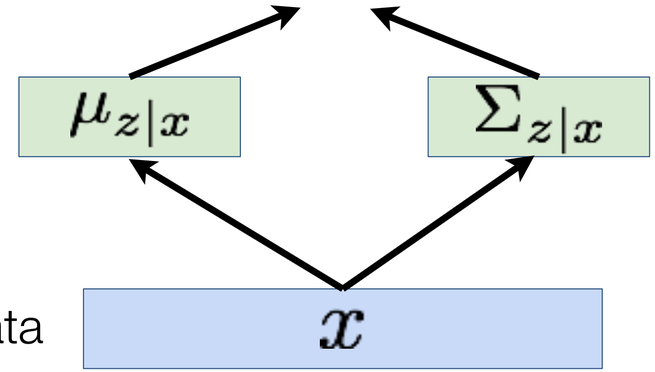
Putting it all together:
maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior

Encoder network

$q_\phi(z|x)$



Input Data

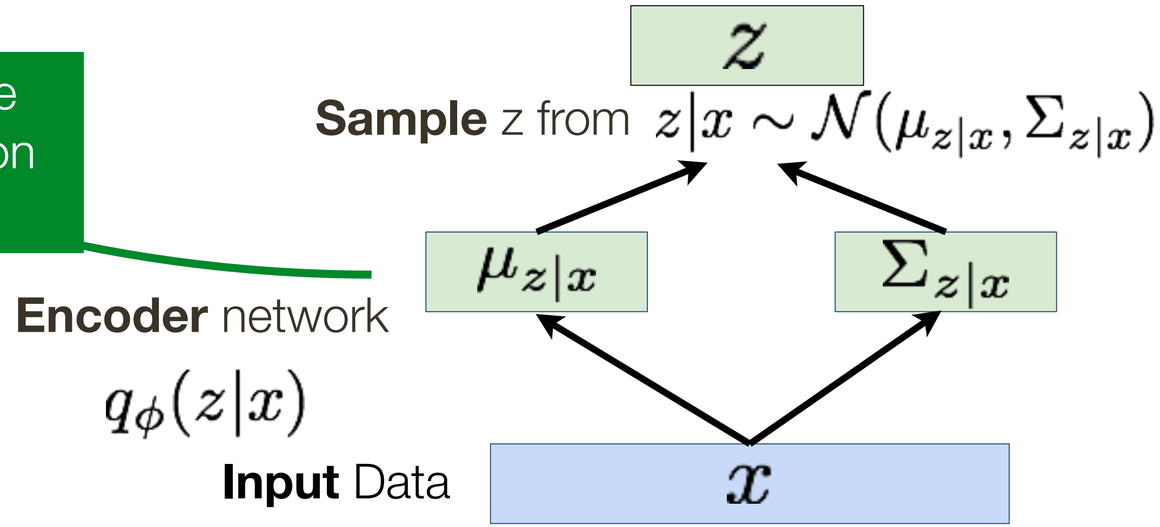
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior



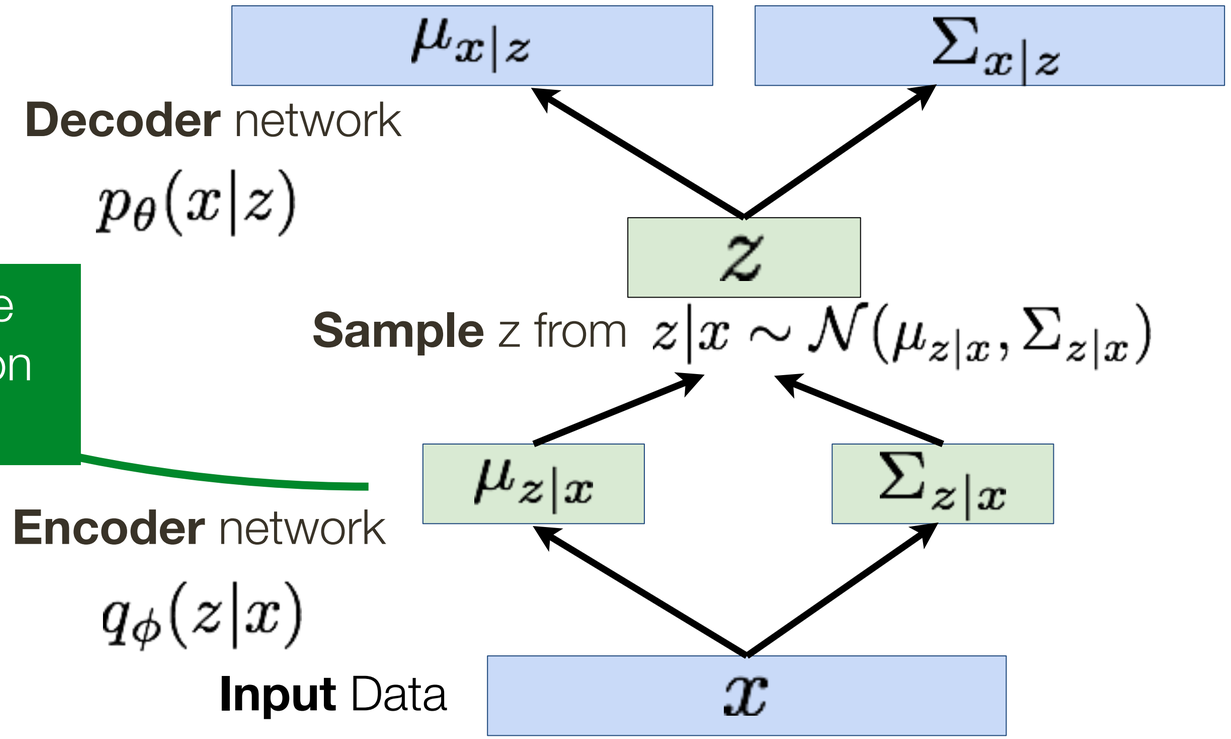
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder: Learning

Putting it all together:
maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior



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

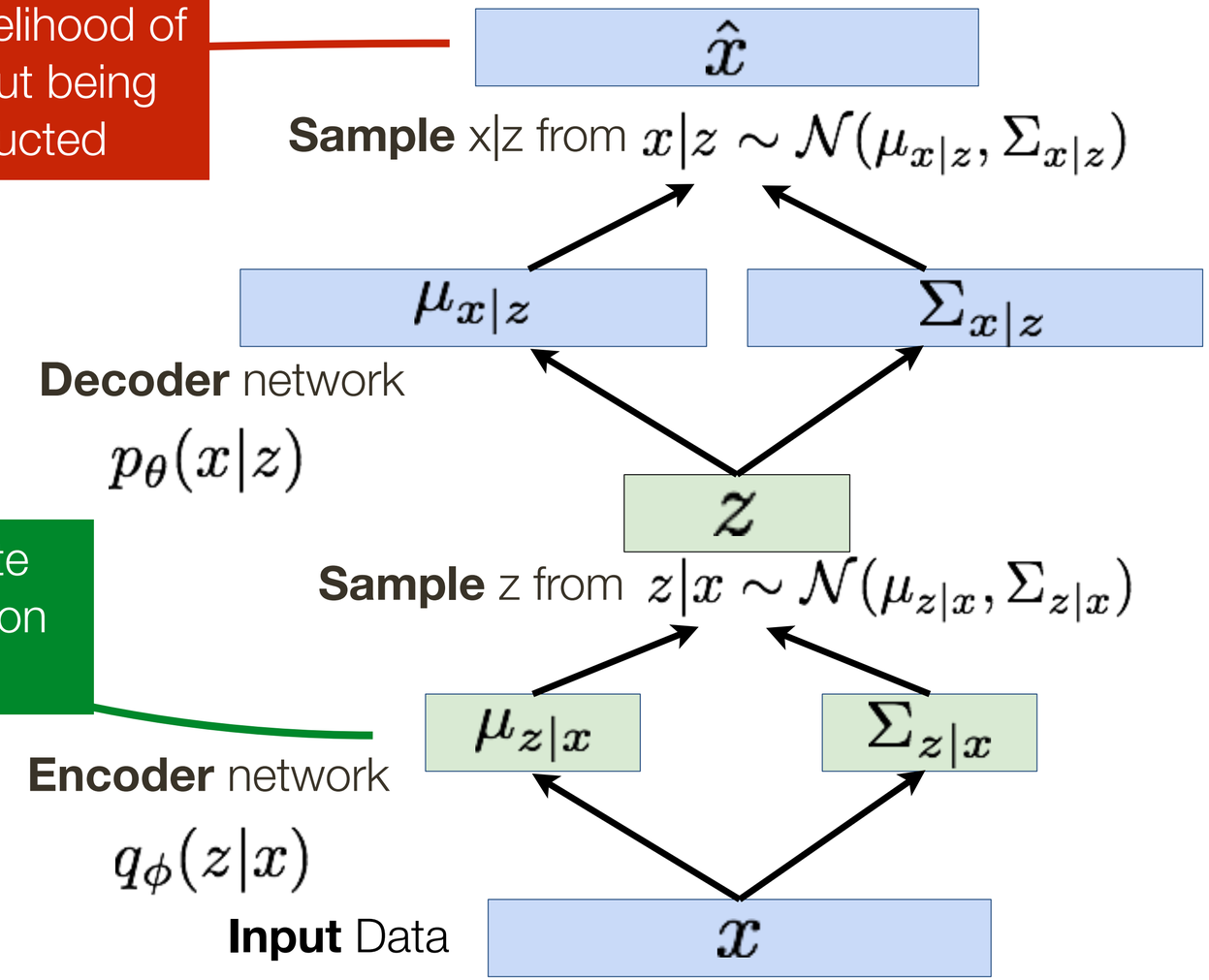
Variational Autoencoder: Learning

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**

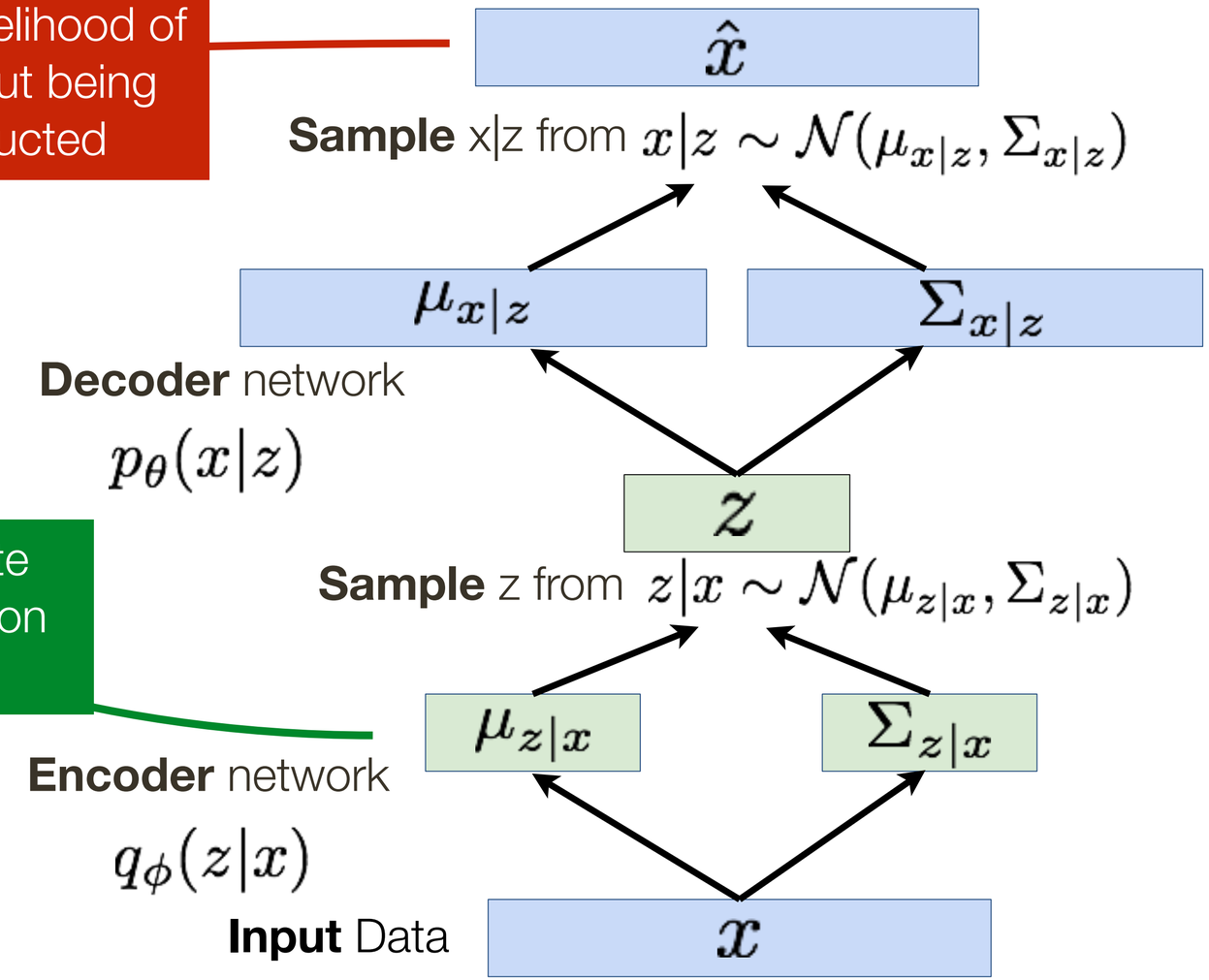
Variational Autoencoder: Learning

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

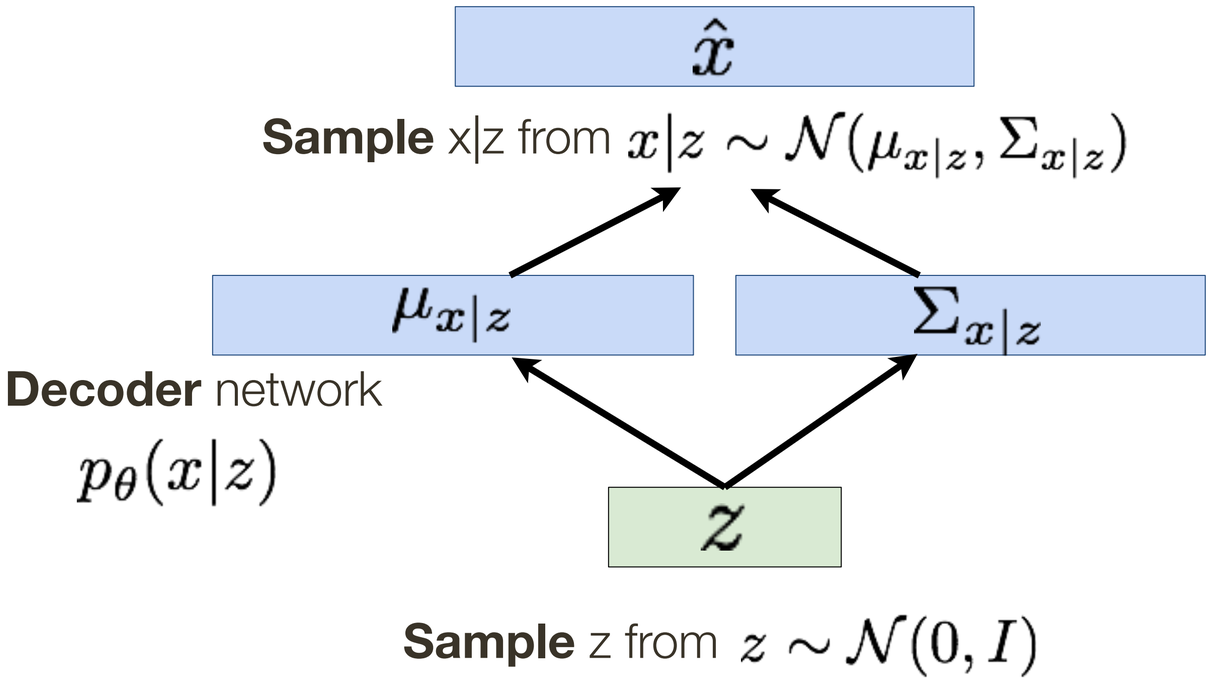


For every minibatch of input data: compute this forward pass, and then backprop!

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

Variational Autoencoder: Generating Data

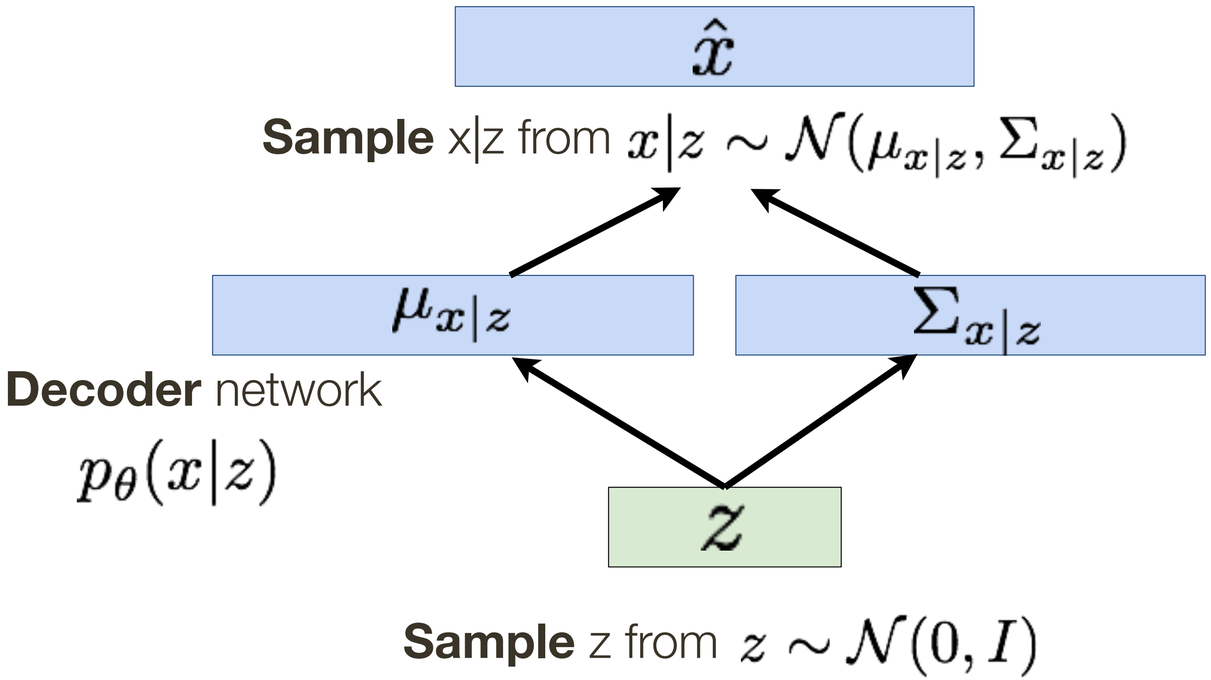
Use decoder network and sample z from **prior**



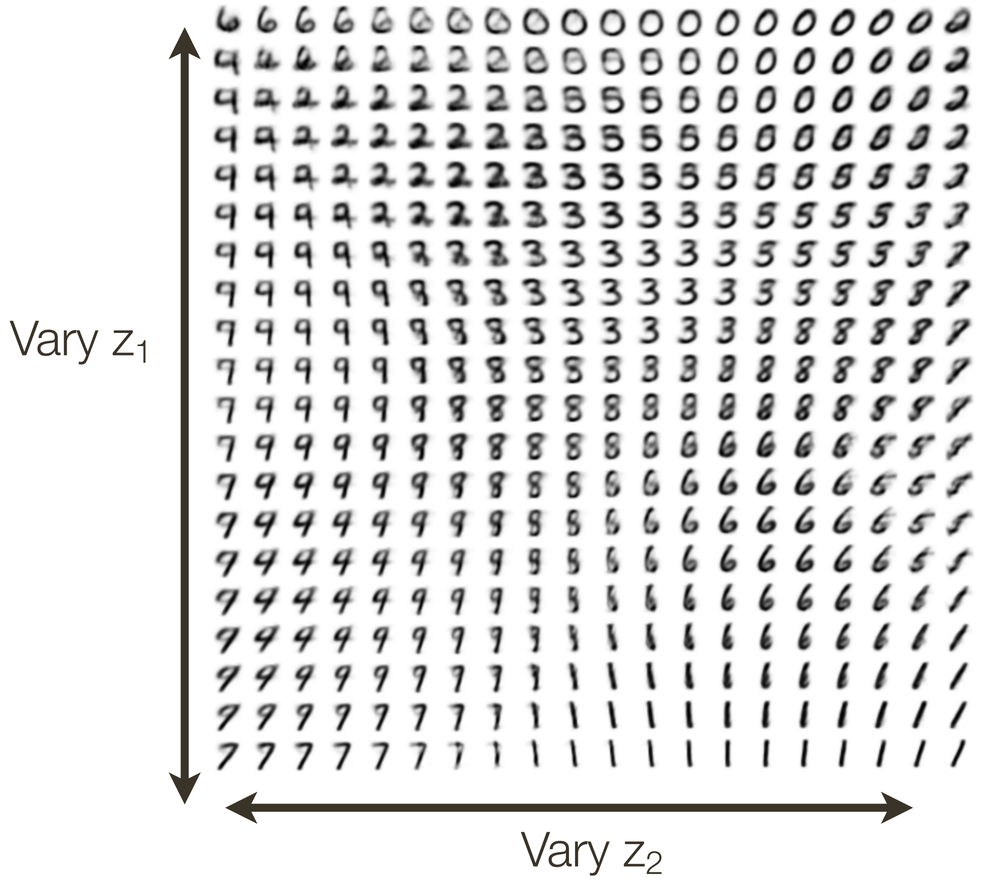
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder: Generating Data

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



Data manifold for 2-d z



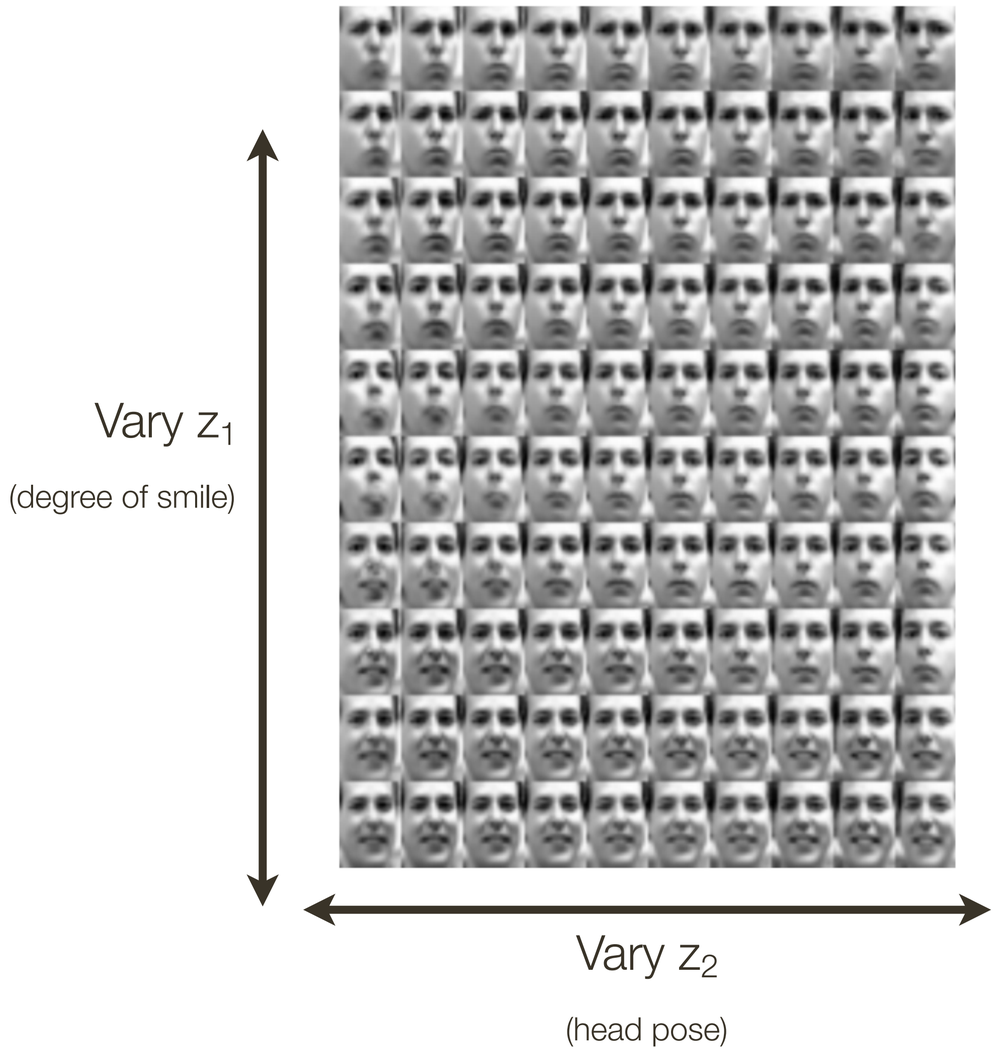
* slide from Fei-Fei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Variational Autoencoder: Generating Data

Diagonal prior on z =>
independent latent variables

Different dimensions of z encode
interpretable factors of variation

Data manifold for 2-d z



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

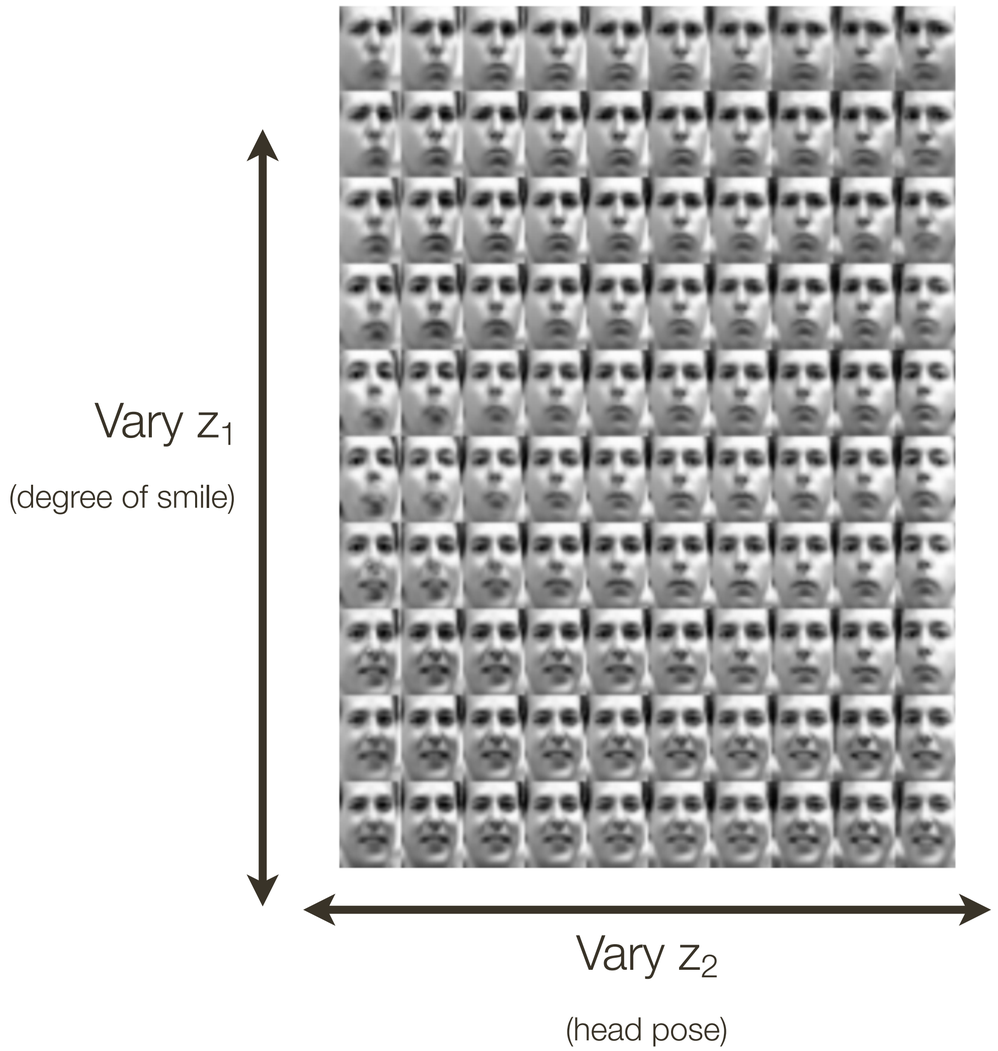
Variational Autoencoder: Generating Data

Diagonal prior on z =>
independent latent variables

Different dimensions of z encode
interpretable factors of variation

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

Data manifold for 2-d z



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

Variational Autoencoder: Generating Data

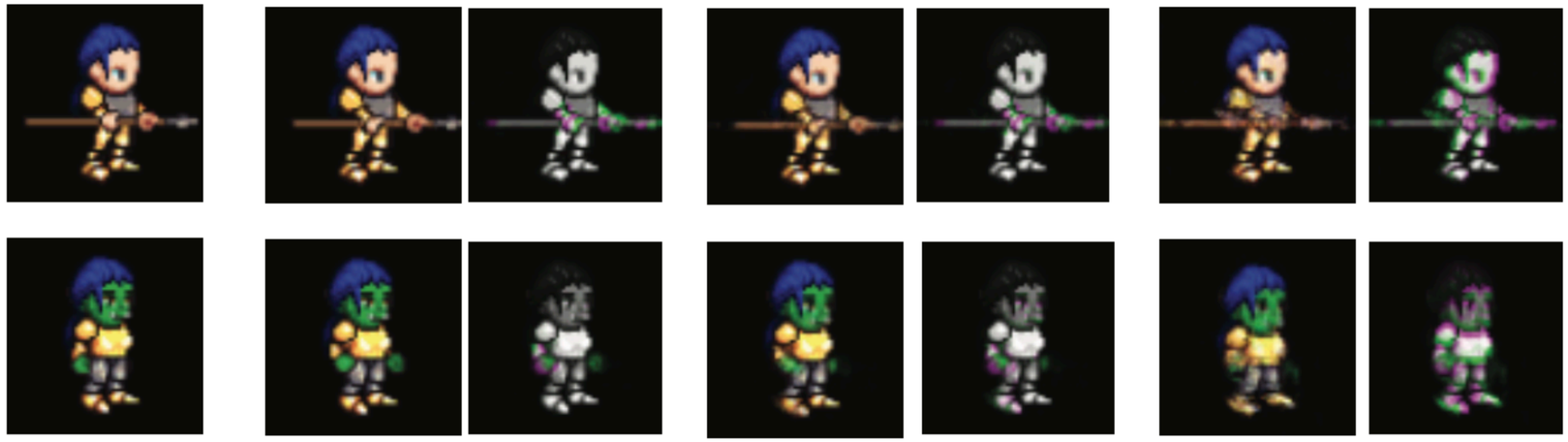
32x32 CIFAR-10

Labeled Faces in the Wild

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

Conditional VAEs

[Xue et al., 2016]



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

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)