

# COMP541

## DEEP LEARNING

Lecture #10 – Autoencoders and Autoregressive Models

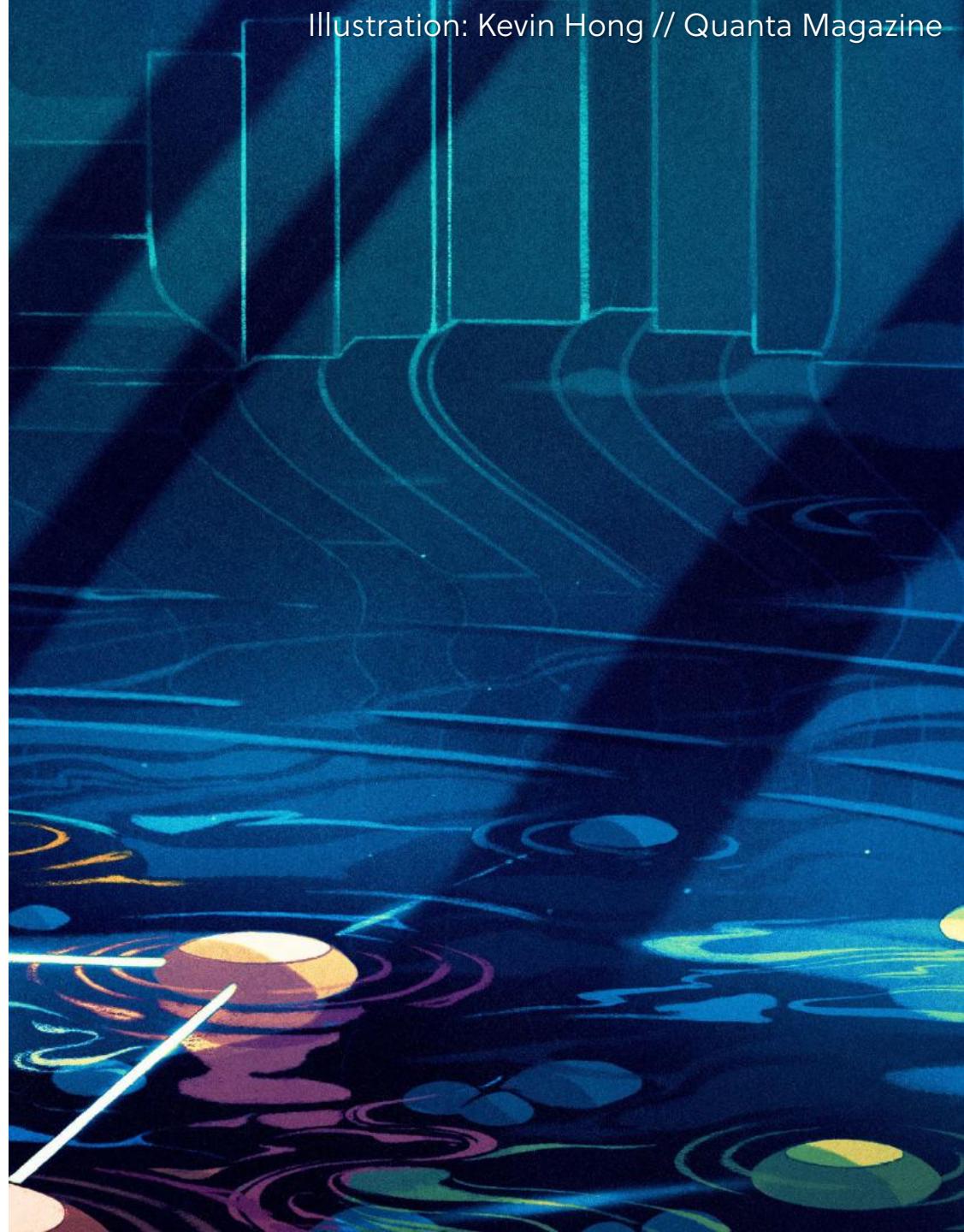


KOÇ  
UNIVERSITY

Aykut Erdem // Koç University // Fall 2022

# Previously on COMP541

- graph structured data
- graph neural nets (GNNs)
- GNNs for “classical” network problems



# Lecture overview

- supervised vs unsupervised learning
- generative modeling
- basic foundations
  - sparse coding
  - autoencoders
- autoregressive generative models

**Disclaimer:** Much of the material and slides for this lecture were borrowed from

- Pieter Abbeel, Peter Chen, Jonathan Ho, Aravind Srinivas' Berkeley CS294-158 class
- Ruslan Salakhutdinov's talk titled "Unsupervised Learning: Learning Deep Generative Models"
- Yoshua Bengio's IDT6266 class
- Bill Freeman, Antonio Torralba and Phillip Isola's MIT 6.869 class
- Nal Kalchbrenner's talks on "Generative Modelling as Sequence Learning" and "Generative Models of Language and Images"
- Justin Johnson's EECS 498/598 class

# Supervised vs Unsupervised Learning

## Supervised Learning

**Data:**  $(x, y)$

$x$  is data,  $y$  is label

**Goal:** Learn a function to map  $x \rightarrow y$

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

Classification



Cat

# Supervised vs Unsupervised Learning

## Supervised Learning

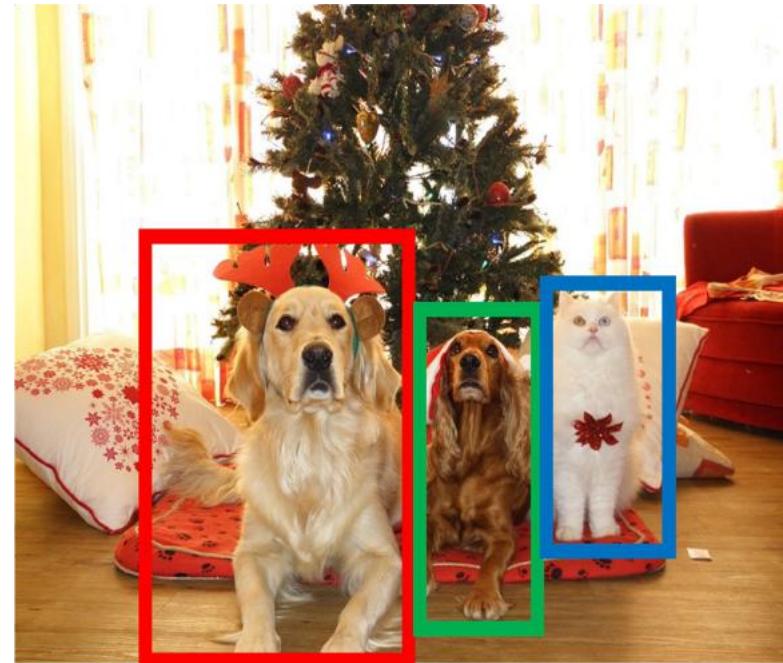
**Data:**  $(x, y)$

$x$  is data,  $y$  is label

**Goal:** Learn a function to map  $x \rightarrow y$

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

## Object Detection



**DOG, DOG, CAT**

# Supervised vs Unsupervised Learning

## Supervised Learning

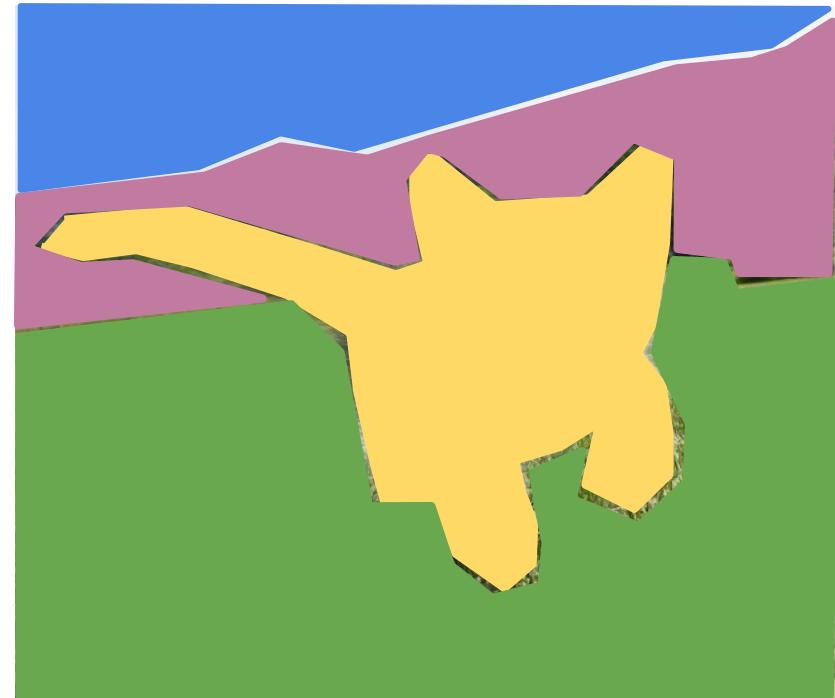
**Data:**  $(x, y)$

$x$  is data,  $y$  is label

**Goal:** Learn a function to map  $x \rightarrow y$

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

## Semantic Segmentation



GRASS, CAT, TREE, SKY

# Supervised vs Unsupervised Learning

## Supervised Learning

**Data:**  $(x, y)$

$x$  is data,  $y$  is label

**Goal:** Learn a function to map  $x \rightarrow y$

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

## Image captioning



*A cat sitting on a suitcase on the floor*

# Supervised vs Unsupervised Learning

## Supervised Learning

**Data:**  $(x, y)$

$x$  is data,  $y$  is label

**Goal:** Learn a function to map  $x \rightarrow y$

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

### Sentiment Analysis

“This Movie is amazing. It has a great plot and talented actors, and the supporting cast is really good as well.”



# Supervised vs Unsupervised Learning

## Supervised Learning

**Data:**  $(x, y)$

$x$  is data,  $y$  is label

**Goal:** Learn a function to map  $x \rightarrow y$

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

## Unsupervised Learning

**Data:**  $x$

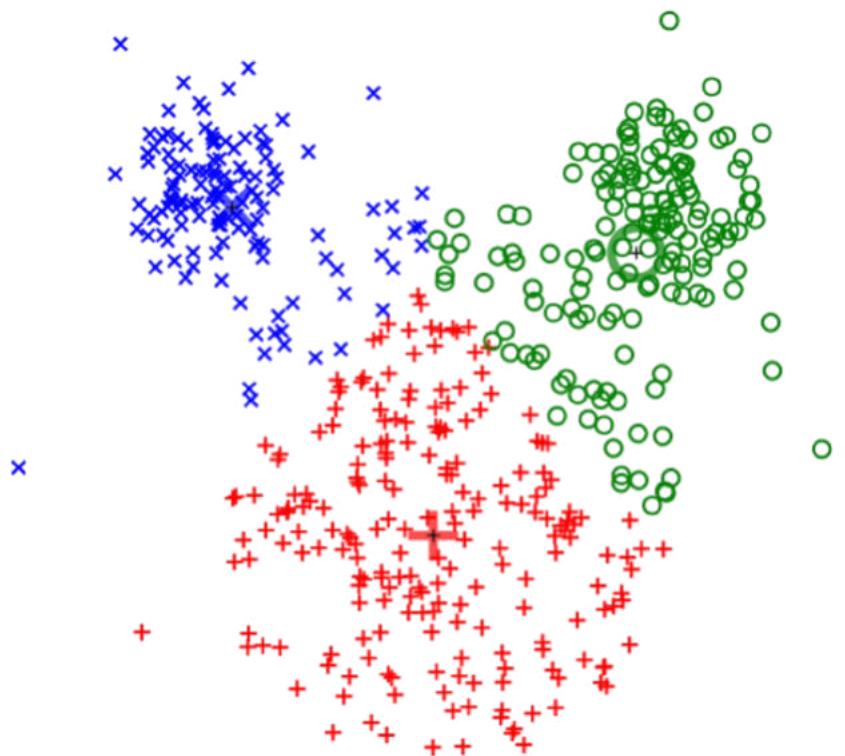
Just data, no labels!

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

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

# Supervised vs Unsupervised Learning

Clustering  
(e.g. K-Means)



**Unsupervised Learning**

**Data:** x

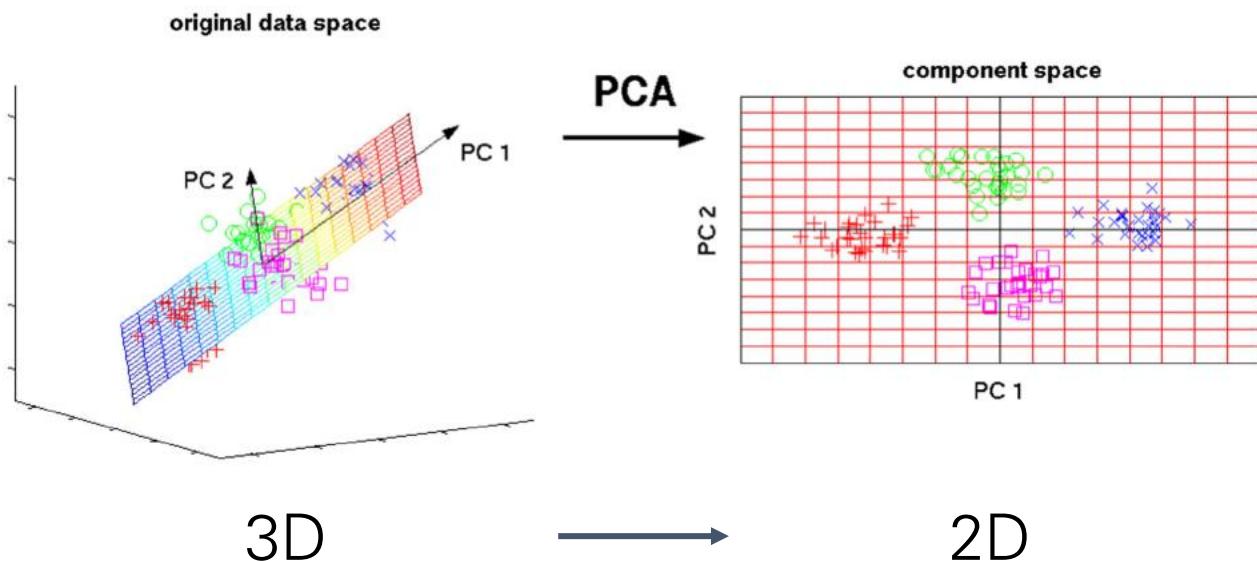
Just data, no labels!

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

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

# Supervised vs Unsupervised Learning

Dimensionality Reduction  
(e.g. Principal Components Analysis)



## Unsupervised Learning

**Data:**  $x$

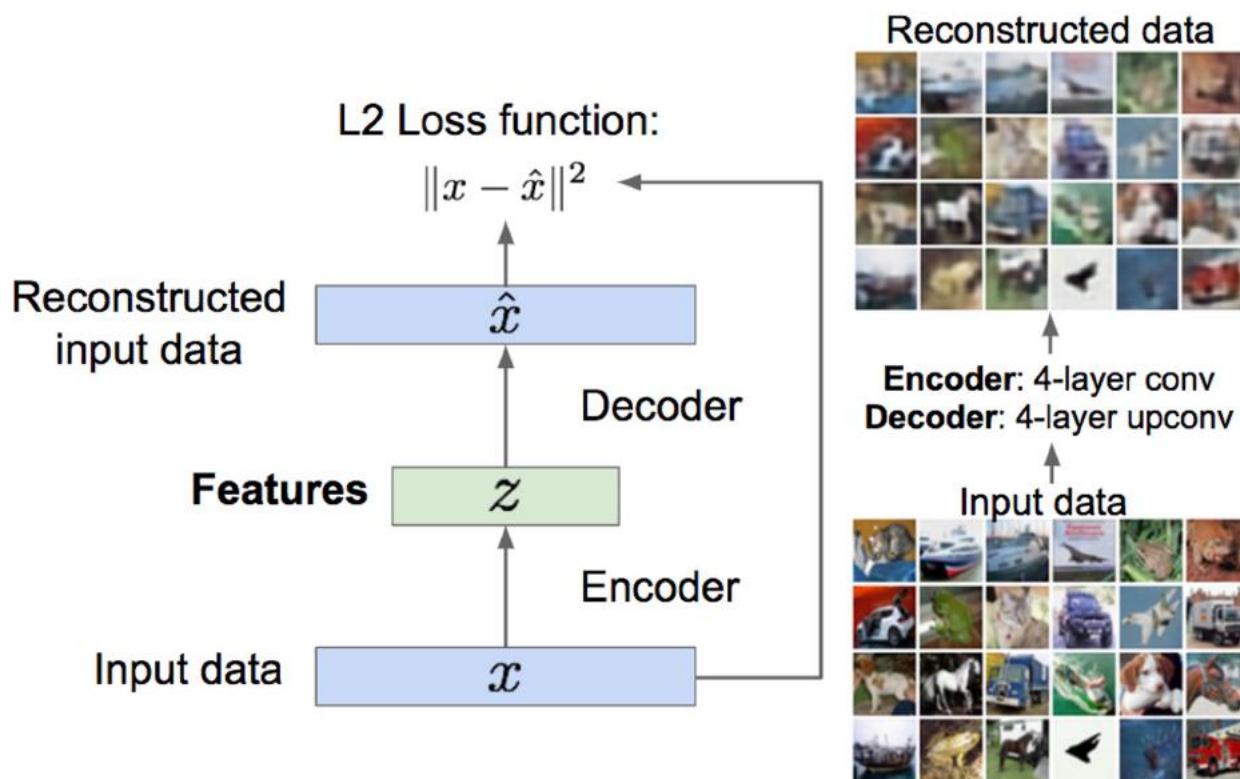
Just data, no labels!

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

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

# Supervised vs Unsupervised Learning

Feature Learning  
(e.g. autoencoders)



## Unsupervised Learning

**Data:**  $x$

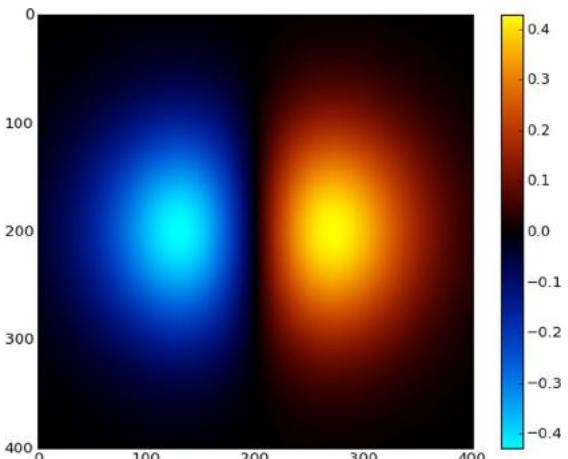
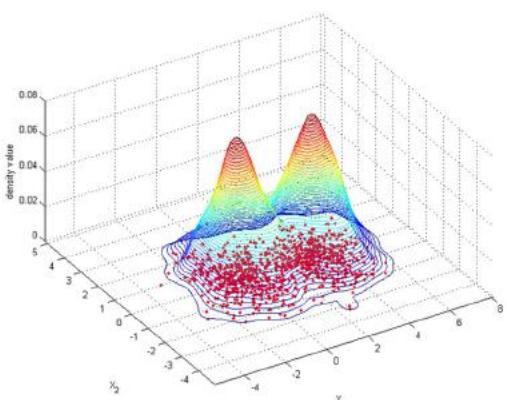
Just data, no labels!

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

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

# Supervised vs Unsupervised Learning

## Density Estimation



## 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.

# Supervised vs Unsupervised Learning

## Supervised Learning

**Data:**  $(x, y)$

$x$  is data,  $y$  is label

**Goal:** Learn a function to map  $x \rightarrow y$

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

## 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.

# Discriminative vs Generative Models

Discriminative Model:

Learn a probability distribution  $p(y|x)$

Generative Model:

Learn a probability distribution  $p(x)$

Conditional

Generative Model:

Learn  $p(x|y)$

**Data: x**



**Label: y**

Cat

# Discriminative vs Generative Models

Discriminative Model:

Learn a probability distribution  $p(y|x)$

Generative Model:

Learn a probability distribution  $p(x)$

Conditional Generative Model:  
Learn  $p(x|y)$

**Data:  $x$**



**Label:  $y$**

**Cat**

Probability Recap:

Density Function

$p(x)$  assigns a positive number to each possible  $x$ ; higher numbers mean  $x$  is more likely

Density functions are normalized:

$$\int_X p(x)dx = 1$$

Different values of  $x$  compete for density

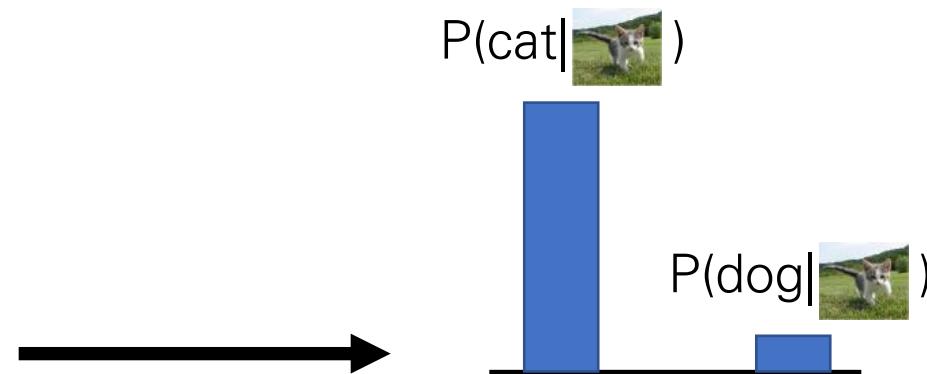
# Discriminative vs Generative Models

**Discriminative Model:**  
Learn a probability distribution  $p(y|x)$

**Generative Model:**  
Learn a probability distribution  $p(x)$

**Conditional Generative Model:**  
Learn  $p(x|y)$

Data:  $x$



Density functions are normalized:

$$\int_X p(x)dx = 1$$

**Density Function**  
 $p(x)$  assigns a positive number to each possible  $x$ ; higher numbers mean  $x$  is more likely

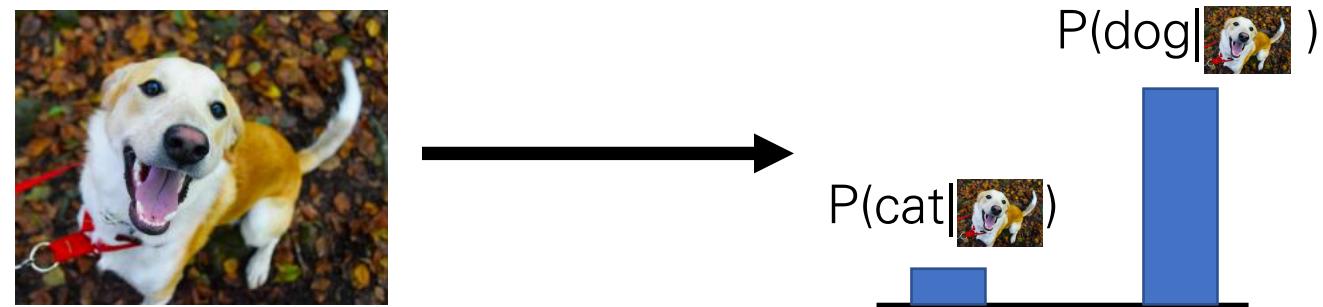
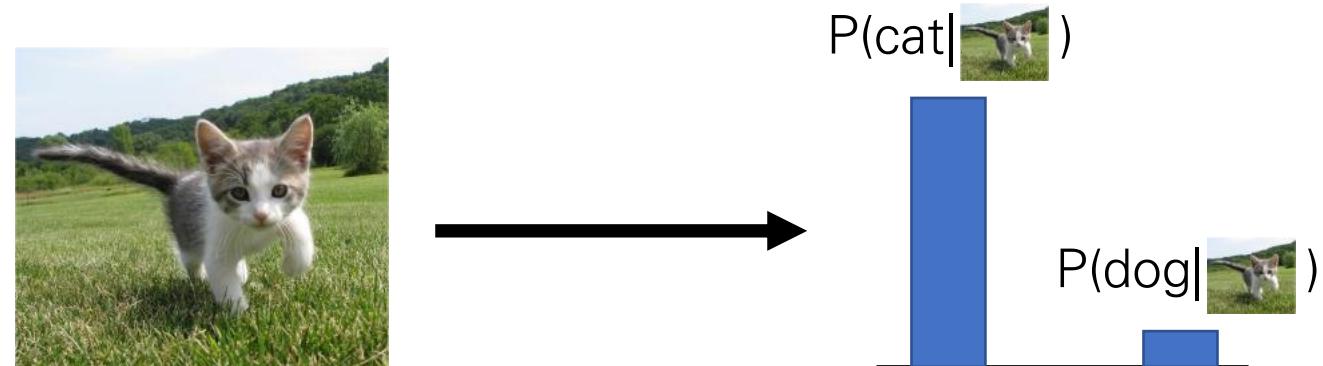
Different values of  $x$  compete for density

# Discriminative vs Generative Models

**Discriminative Model:**  
Learn a probability distribution  $p(y|x)$

**Generative Model:**  
Learn a probability distribution  $p(x)$

**Conditional Generative Model:**  
Learn  $p(x|y)$



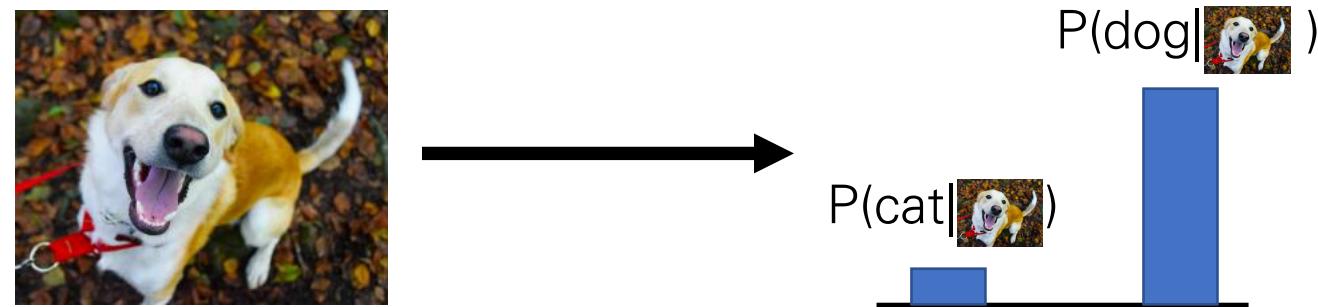
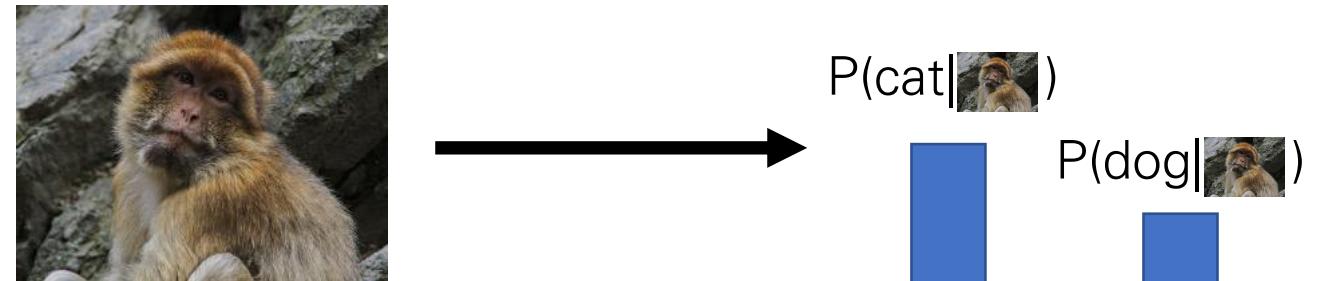
Discriminative model: the possible labels for each input “compete” for probability mass.  
But no competition between images

# Discriminative vs Generative Models

**Discriminative Model:**  
Learn a probability distribution  $p(y|x)$

**Generative Model:**  
Learn a probability distribution  $p(x)$

**Conditional Generative Model:**  
Learn  $p(x|y)$



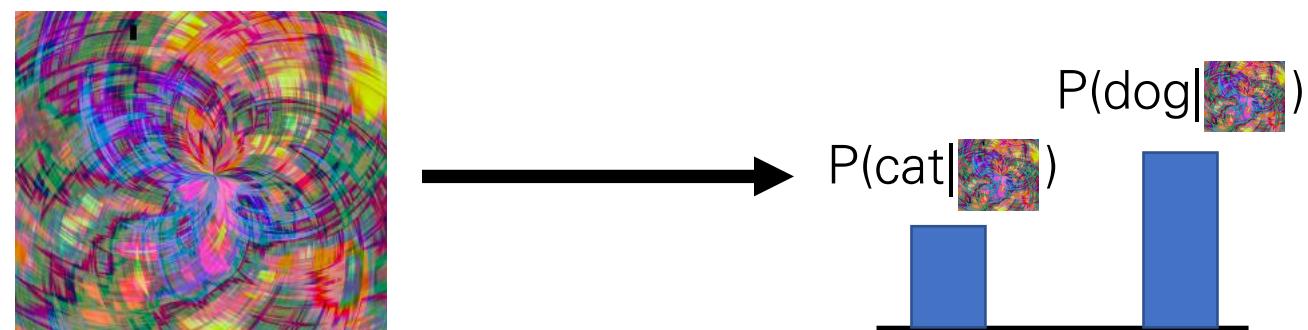
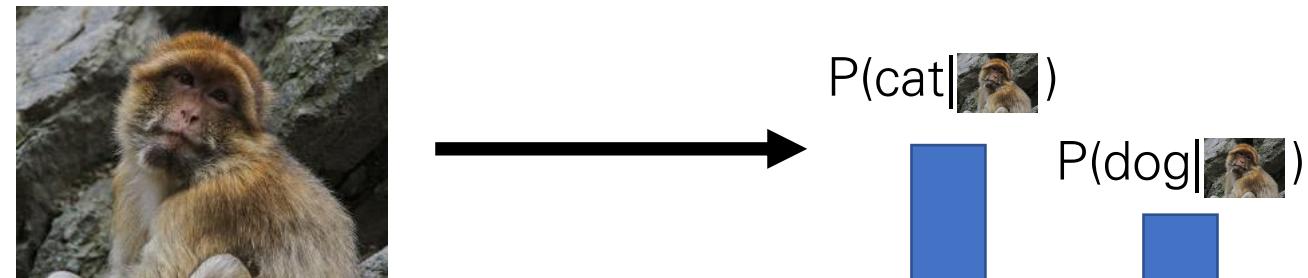
Discriminative model: No way for the model to handle unreasonable inputs; it must give label distributions for all images

# Discriminative vs Generative Models

**Discriminative Model:**  
Learn a probability distribution  $p(y|x)$

**Generative Model:**  
Learn a probability distribution  $p(x)$

**Conditional Generative Model:**  
Learn  $p(x|y)$



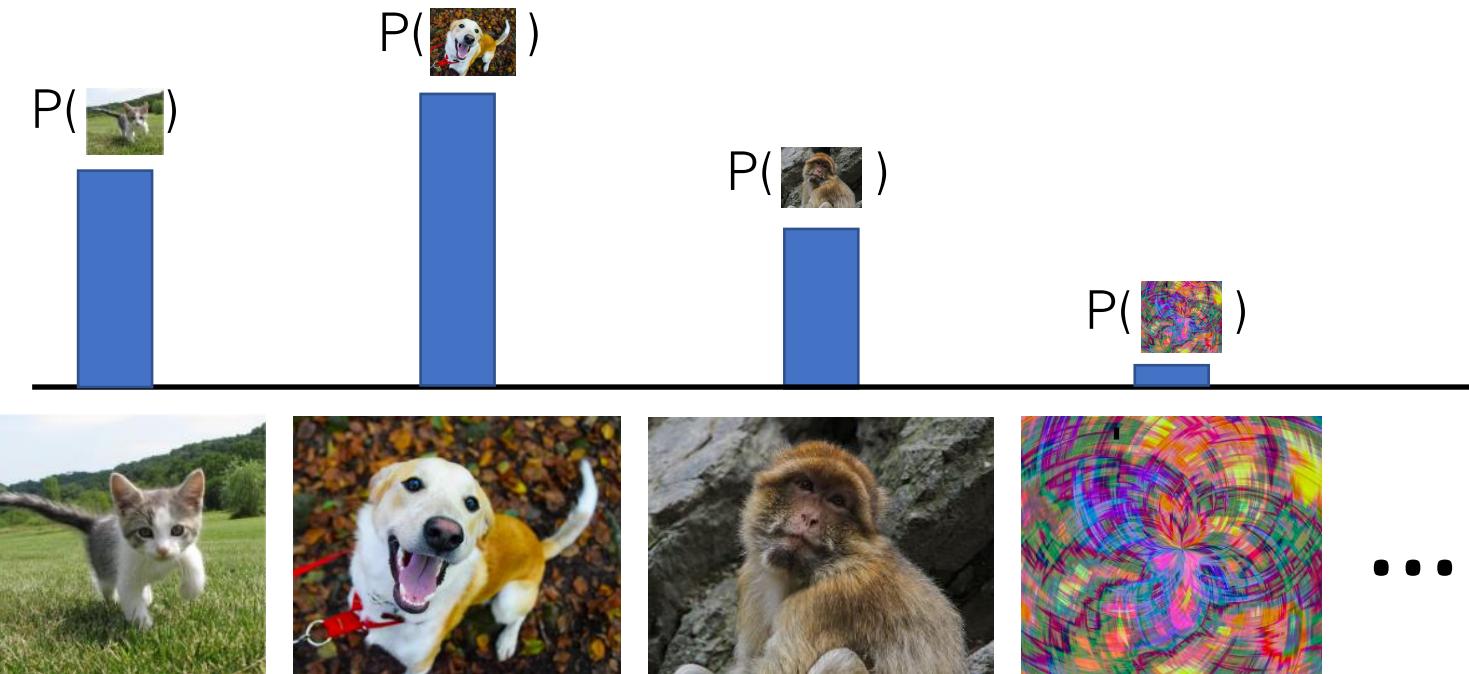
Discriminative model: No way for the model to handle unreasonable inputs; it must give label distributions for all images

# Discriminative vs Generative Models

**Discriminative Model:**  
Learn a probability distribution  $p(y|x)$

**Generative Model:**  
Learn a probability distribution  $p(x)$

**Conditional Generative Model:**  
Learn  $p(x|y)$



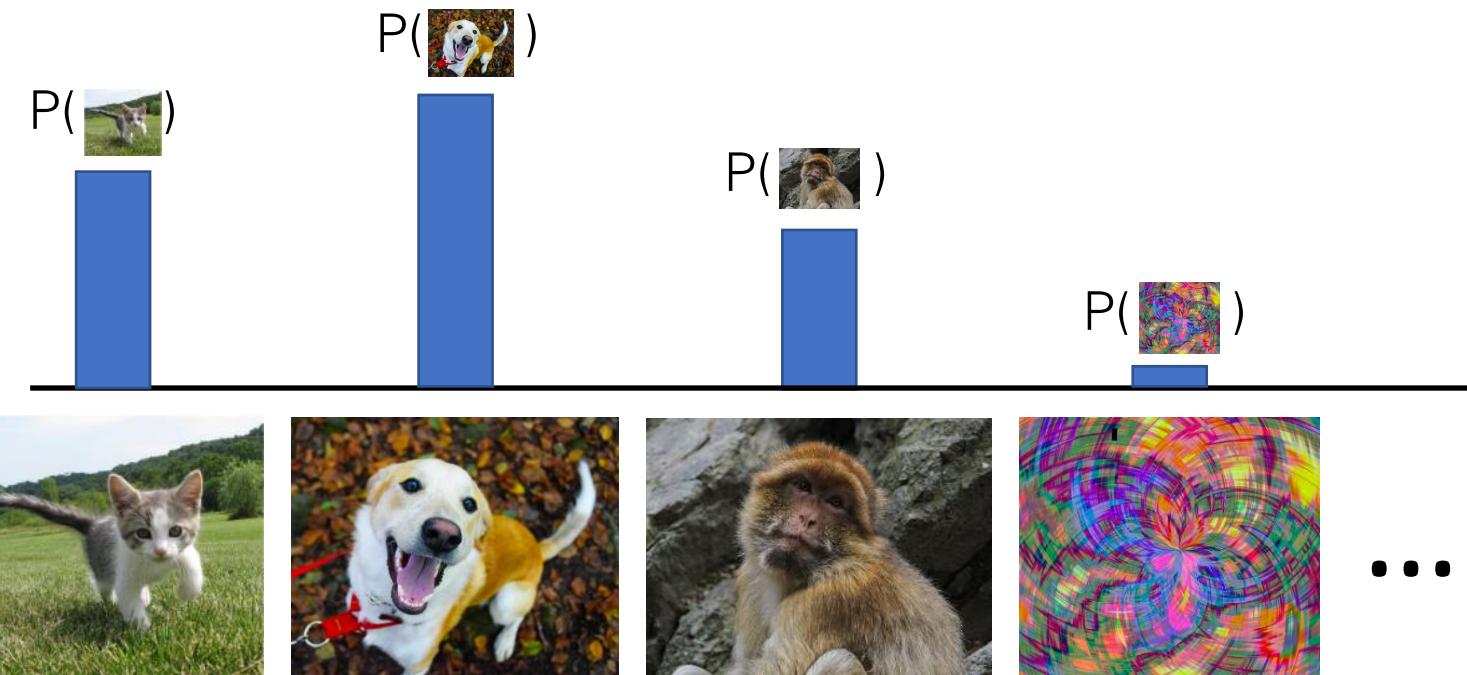
Generative model: All possible images compete with each other for probability mass

# Discriminative vs Generative Models

**Discriminative Model:**  
Learn a probability distribution  $p(y|x)$

**Generative Model:**  
Learn a probability distribution  $p(x)$

**Conditional Generative Model:**  
Learn  $p(x|y)$



Generative model: All possible images compete with each other for probability mass

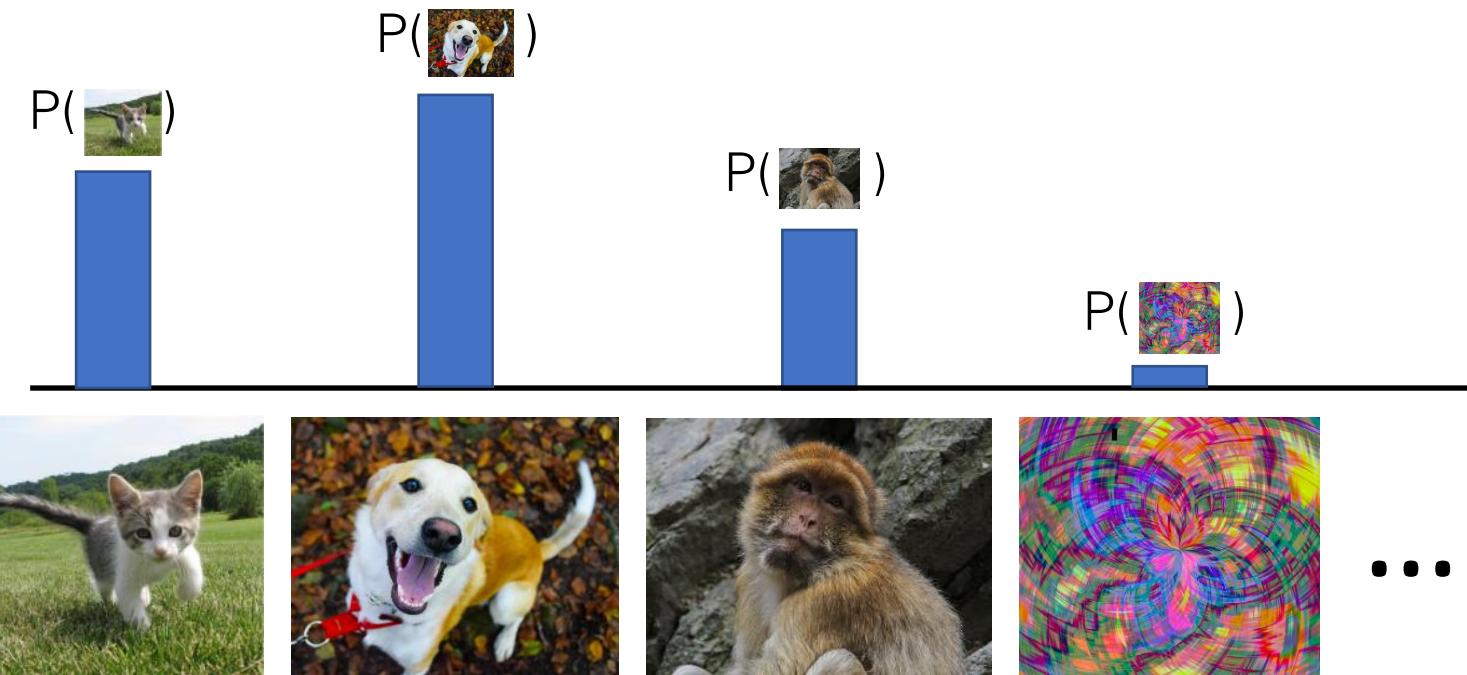
Requires deep image understanding! Is a dog more likely to sit or stand? How about 3-legged dog vs 3-armed monkey?

# Discriminative vs Generative Models

**Discriminative Model:**  
Learn a probability distribution  $p(y|x)$

**Generative Model:**  
Learn a probability distribution  $p(x)$

**Conditional Generative Model:**  
Learn  $p(x|y)$



Generative model: All possible images compete with each other for probability mass

Model can “reject” unreasonable inputs by assigning them small values

# Discriminative vs Generative Models

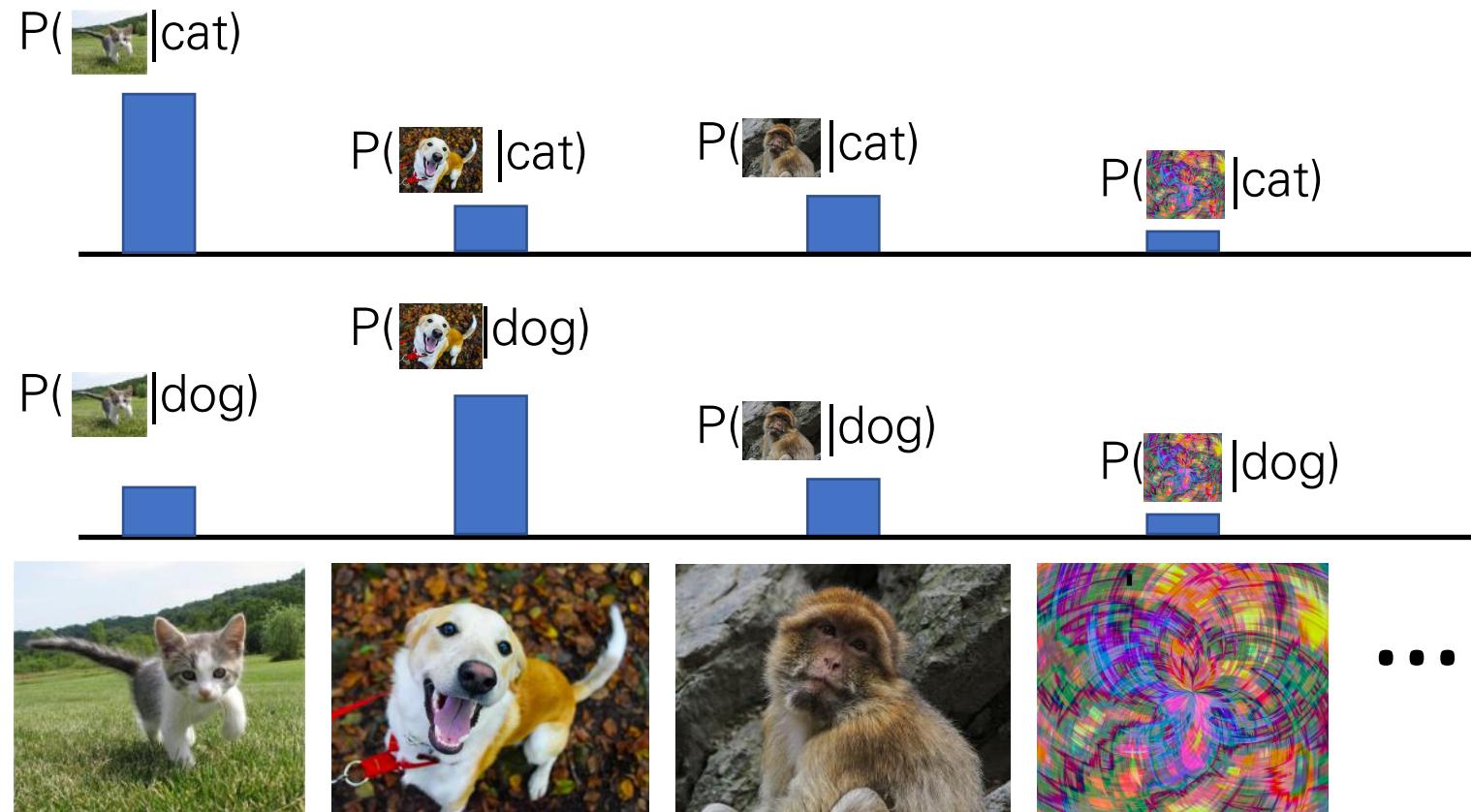
**Discriminative Model:**

Learn a probability distribution  $p(y|x)$

**Generative Model:**

Learn a probability distribution  $p(x)$

**Conditional Generative Model:**  
Learn  $p(x|y)$



Conditional Generative Model: Each possible label induces a competition among all images

# Discriminative vs Generative Models

Discriminative Model:

Learn a probability distribution  $p(y|x)$

Generative Model:

Learn a probability distribution  $p(x)$

Conditional Generative Model:  
Learn  $p(x|y)$

Recall Bayes' Rule:

$$P(x | y) = \frac{P(y | x)}{P(y)} P(x)$$

# Discriminative vs Generative Models

Discriminative Model:

Learn a probability distribution  $p(y|x)$

Generative Model:

Learn a probability distribution  $p(x)$

Conditional Generative Model:  
Learn  $p(x|y)$

Recall Bayes' Rule:

$$P(x | y) = \frac{P(y | x)}{P(y)} P(x)$$

Conditional Generative Model      Discriminative Model  
Prior over labels      (Unconditional) Generative Model

We can build a conditional generative model from other components!

# What can we do with a discriminative model?

**Discriminative Model:**

Learn a probability  
distribution  $p(y|x)$



Assign labels to data  
Feature learning (with labels)

**Generative Model:**

Learn a probability  
distribution  $p(x)$

**Conditional**

**Generative Model:**

Learn  $p(x|y)$

# What can we do with a discriminative model?

**Discriminative Model:**

Learn a probability distribution  $p(y|x)$



Assign labels to data  
Feature learning (with labels)

**Generative Model:**

Learn a probability distribution  $p(x)$



Detect outliers  
Feature learning (without labels)  
Sample to generate new data

**Conditional Generative Model:**

Learn  $p(x|y)$

# What can we do with a discriminative model?

**Discriminative Model:**

Learn a probability distribution  $p(y|x)$



Assign labels to data  
Feature learning (with labels)

**Generative Model:**

Learn a probability distribution  $p(x)$



Detect outliers  
Feature learning (without labels)  
Sample to **generate** new data

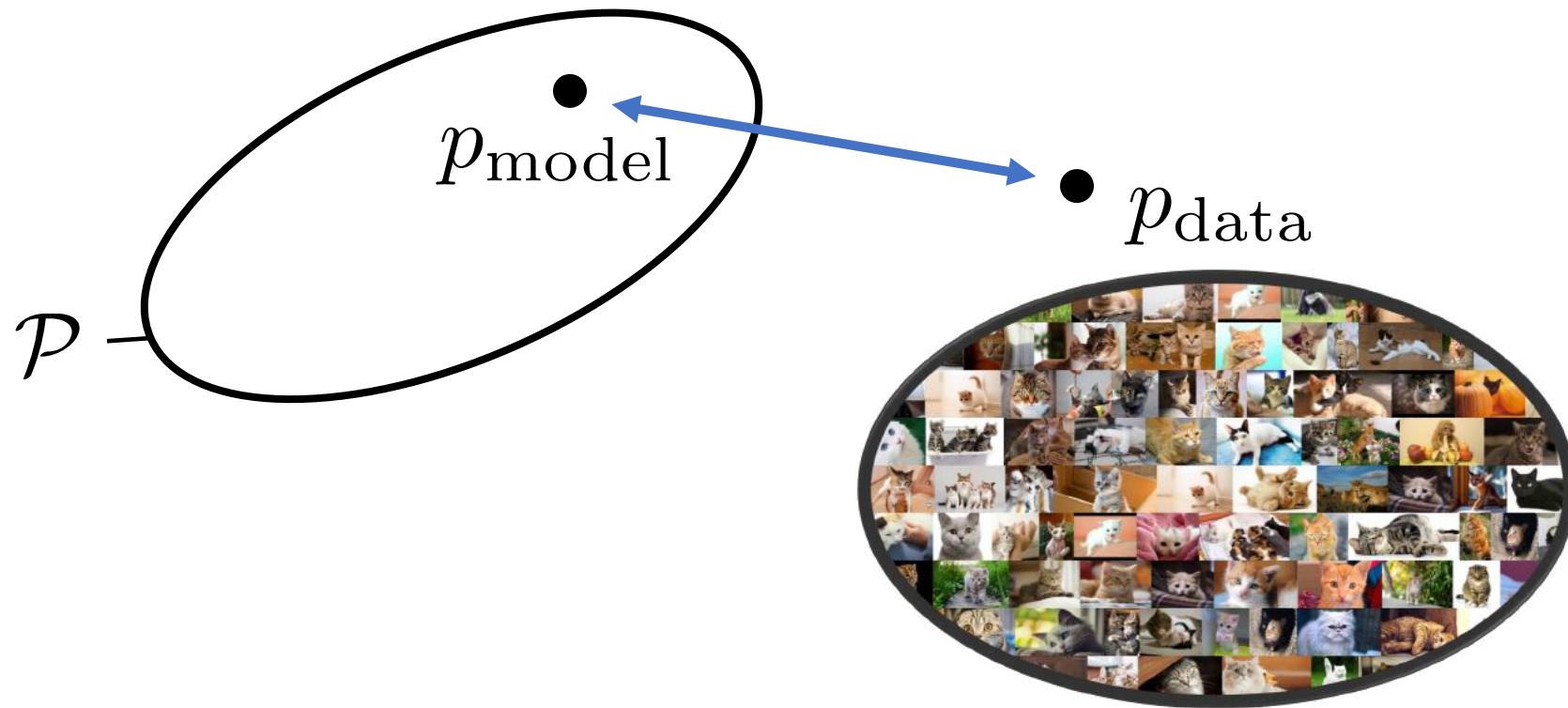
**Conditional Generative Model:**

Learn  $p(x|y)$



Assign labels, while rejecting outliers!  
Generate new data conditioned on input labels

# Generative Modeling



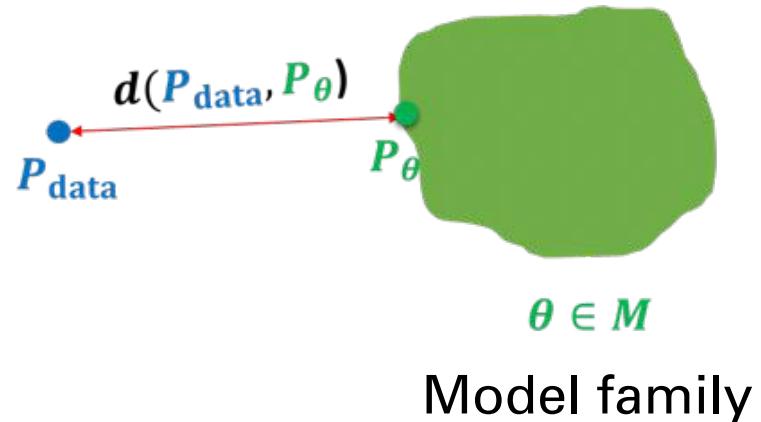
- Goal: Learn some underlying hidden structure of the training samples to generate novel samples from same data distribution

# Learning a generative model

- We are given a training set of examples, e.g., images of dogs



$$\begin{aligned} \mathbf{x}_i &\sim P_{\text{data}} \\ i &= 1, 2, \dots, n \end{aligned}$$



- We want to learn a probability distribution  $p(x)$  over images  $x$  s.t.
  - **Generation:** If we sample  $x_{\text{new}} \sim p(x)$ ,  $x_{\text{new}}$  should look like a dog (sampling)
  - **Density estimation:**  $p(x)$  should be high if  $x$  looks like a dog, and low otherwise (anomaly detection)
  - **Unsupervised representation learning:** We should be able to learn what these images have in common, e.g., ears, tail, etc. (features)

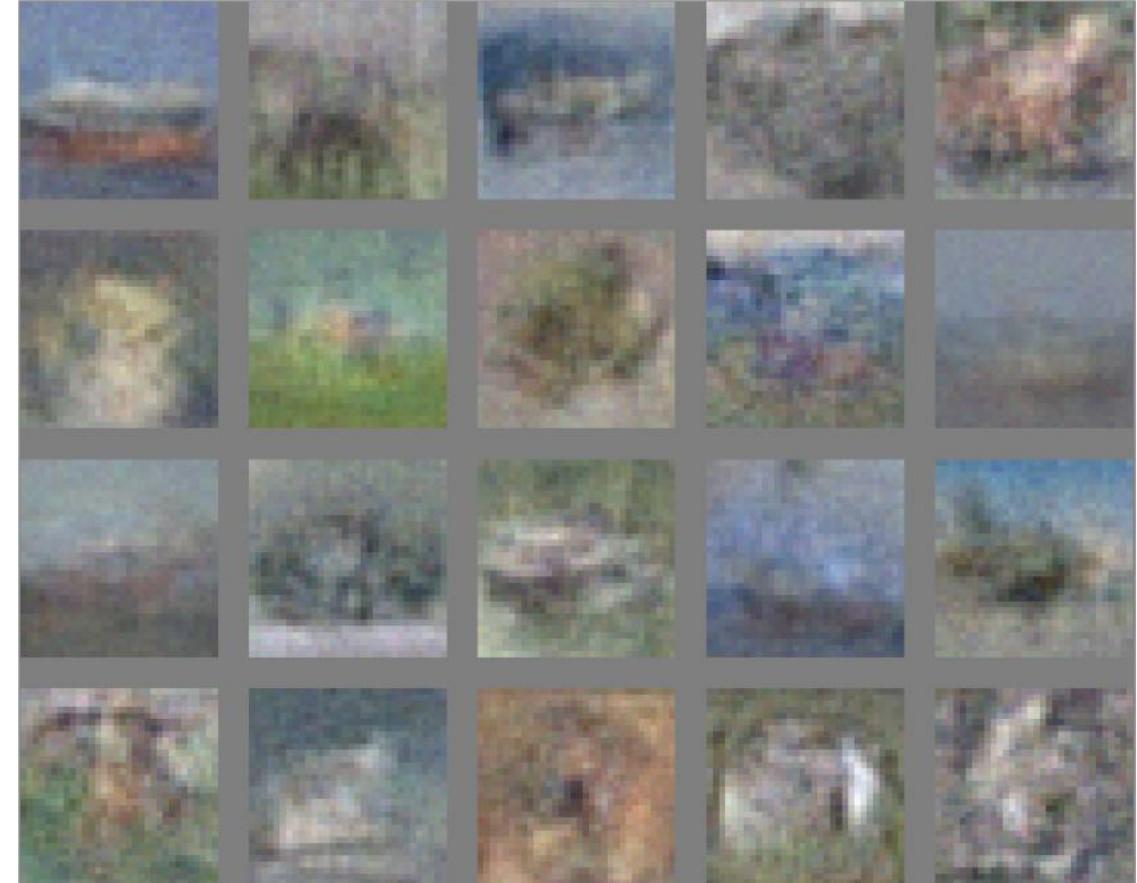
# Generate Images



# Generate Images

8 0 2 3 9 0 0  
9 1 1 7 1 9 4  
2 0 8 0 8 2 9  
6 3 8 7 0 6 1  
9 8 9 8 9 1 0  
4 3 4 8 2 8 1

# Generate Images



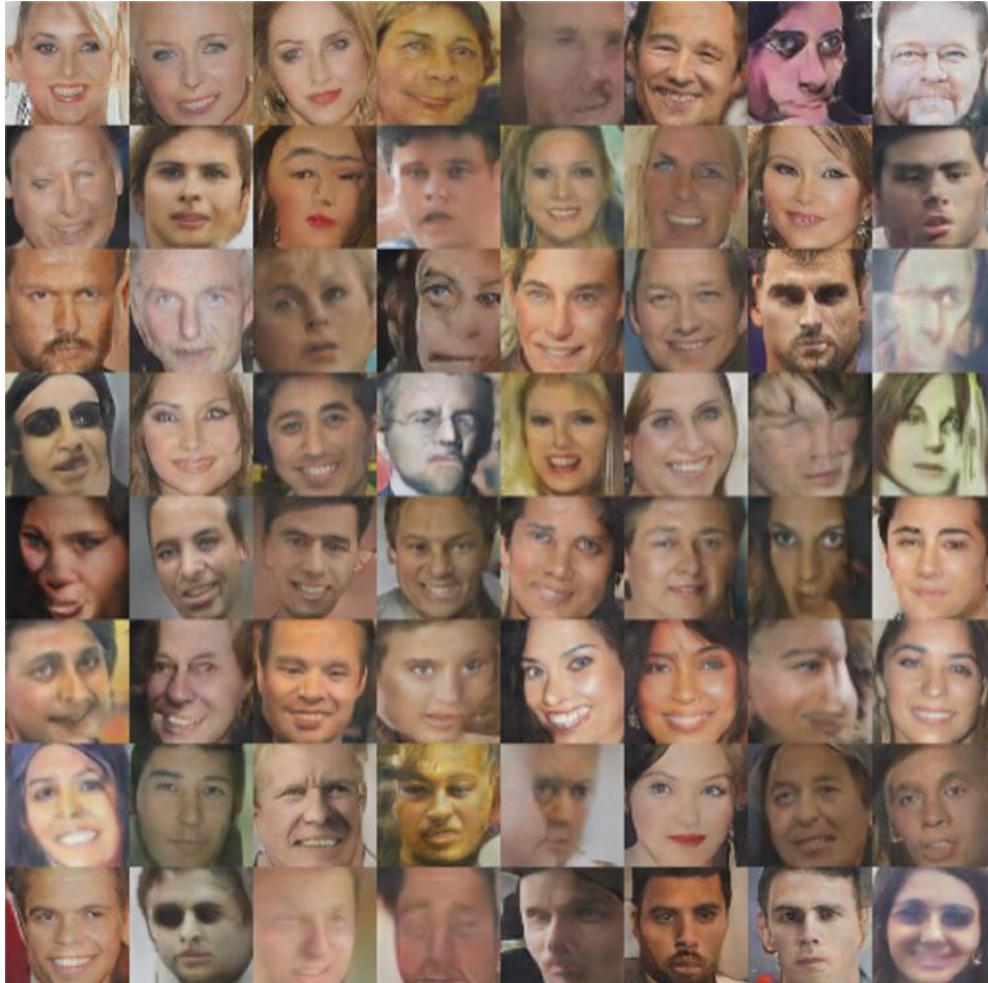
[GAN, Goodfellow et al. 2014]

# Generate Images



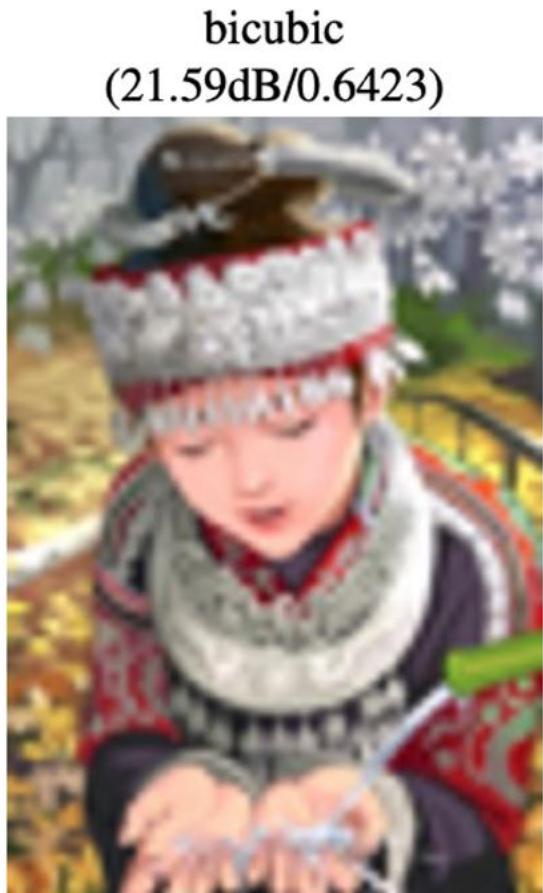
[DCGAN, Radford, Metz, Chintala 2015]

# Generate Images



[DCGAN, Radford, Metz, Chintala 2015]

# Generate Images



[Ledig, Theis, Huszar et al, 2017]

# Generate Images



# Generate Images



[BigGAN, Brock, Donahue, Simonyan, 2018]

# Generate Images



# Generate Images



[Latent Diffusion, Rombach, Blattmann, Lorenz, Esser, Ommer, 2022] 41

# Generate Images



[Latent Diffusion, Rombach, Blattmann, Lorenz, Esser, Ommer, 2022] 42

# Generate Audio



1 Second



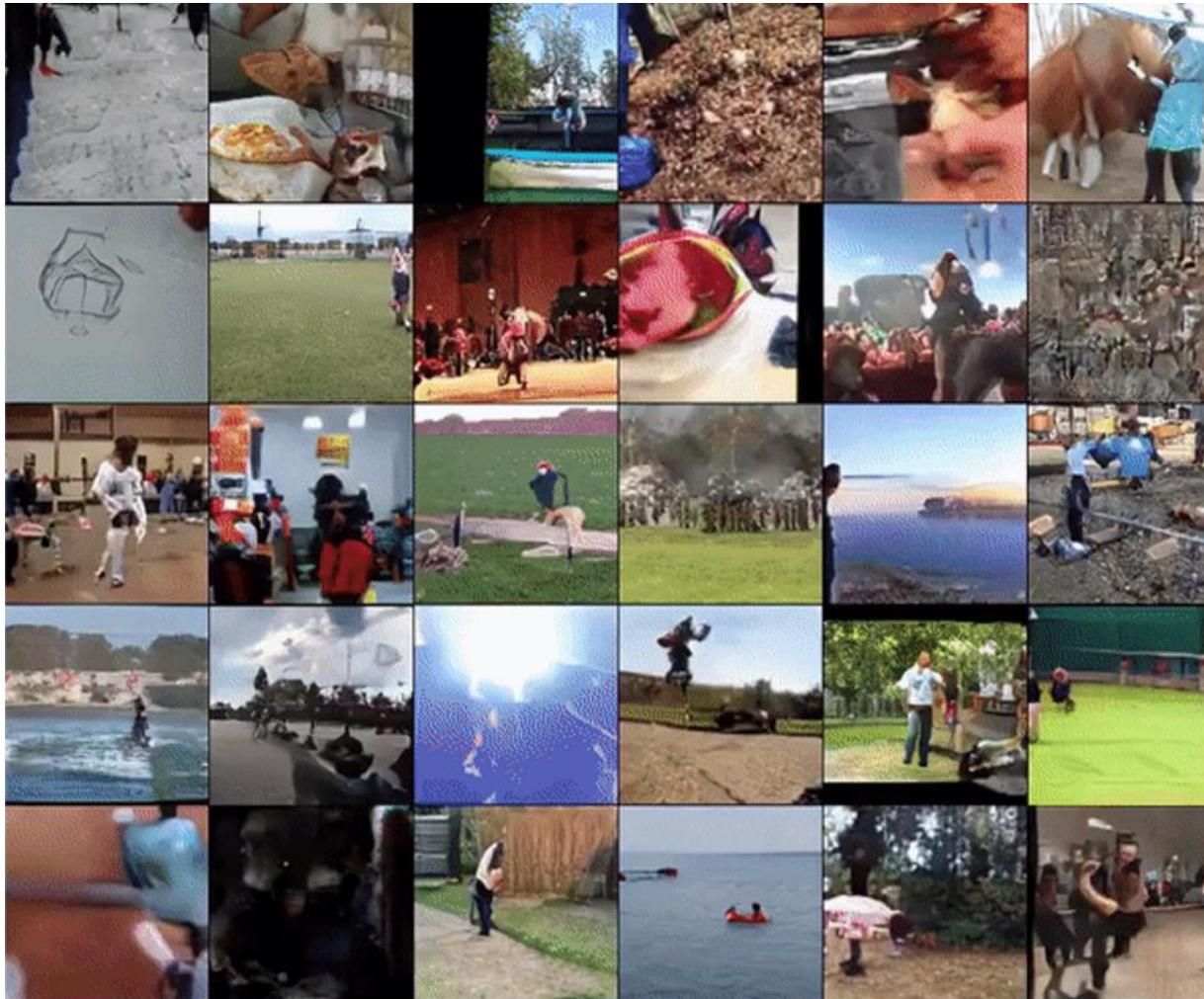
Parametric



WaveNet



# Generate Video

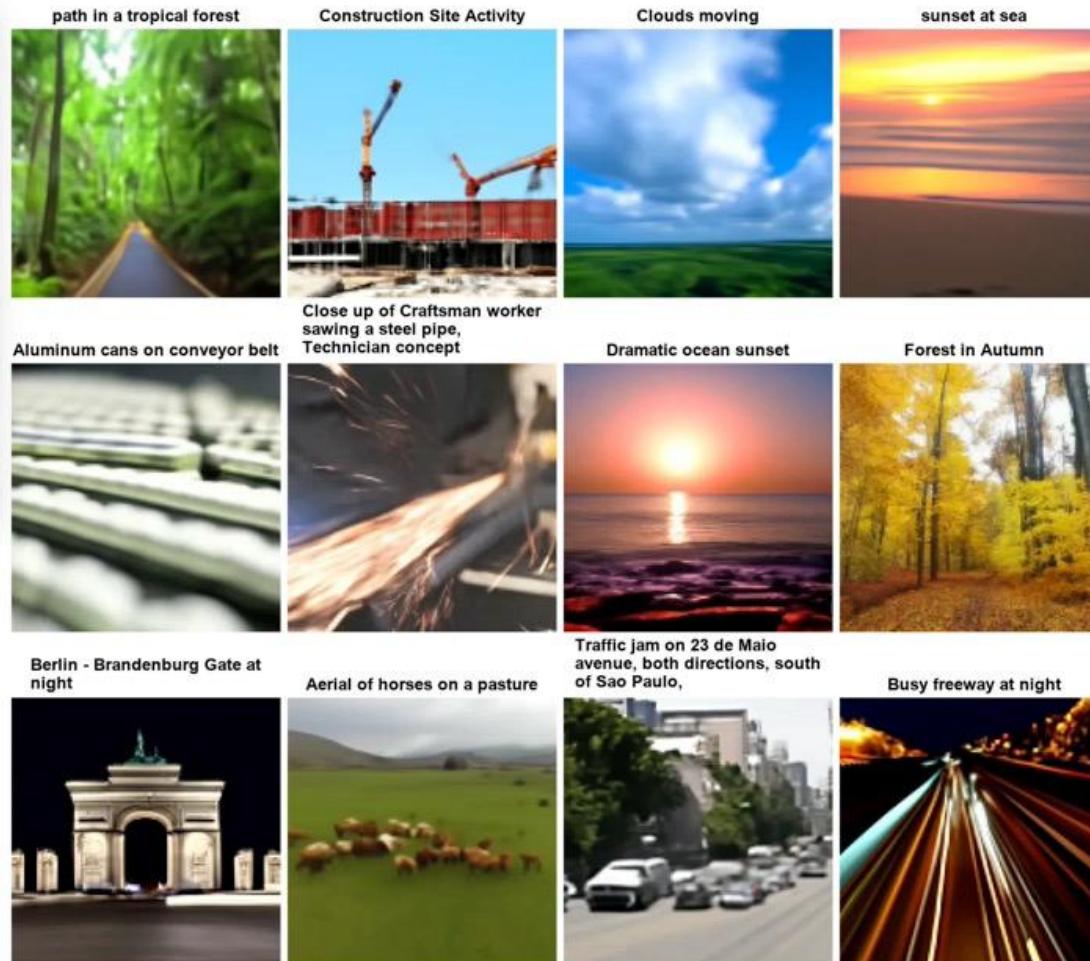


DVD-GAN: Adversarial Video Generation on Complex Datasets, Clark,  
Donahue, Simonyan, 2019

# Generate Video



# Generate Video



# Generate Text

PANDARUS:

Alas, I think he shall be come approached and the day  
When little strain would be attain'd into being never fed,  
And who is but a chain and subjects of his death,  
I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,  
Breaking and strongly should be buried, when I perish  
The earth and thoughts of many states.

DUKE VINCENTIO:

Well, your wit is in the care of side and that.

# Generate Math

```
\begin{proof}
```

We may assume that  $\mathcal{I}$  is an abelian sheaf on  $\mathcal{C}$ .

\item Given a morphism  $\Delta : \mathcal{F} \rightarrow \mathcal{I}$  is an injective and let  $\mathfrak{q}$  be an abelian sheaf on  $X$ .

Let  $\mathcal{F}$  be a fibered complex. Let  $\mathcal{F}$  be a category.

```
\begin{enumerate}
```

\item \hyperref[setain-construction-phantom]{Lemma}

\label{lemma-characterize-quasi-finite}

Let  $\mathcal{F}$  be an abelian quasi-coherent sheaf on  $\mathcal{C}$ .

Let  $\mathcal{F}$  be a coherent  $\mathcal{O}_X$ -module. Then  $\mathcal{F}$  is an abelian catenary over  $\mathcal{C}$ .

\item The following are equivalent

```
\begin{enumerate}
```

\item  $\mathcal{F}$  is an  $\mathcal{O}_X$ -module.

```
\end{lemma}
```

For  $\bigoplus_{n=1,\dots,m}$  where  $\mathcal{L}_{m_\bullet} = 0$ , hence we can find a closed subset  $\mathcal{H}$  in  $\mathcal{H}$  and any sets  $\mathcal{F}$  on  $X$ ,  $U$  is a closed immersion of  $S$ , then  $U \rightarrow T$  is a separated algebraic space.

*Proof.* Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by  $\coprod Z \times_U U \rightarrow V$ . Consider the maps  $M$  along the set of points  $\text{Sch}_{fppf}$  and  $U \rightarrow U$  is the fibre category of  $S$  in  $U$  in Section, ?? and the fact that any  $U$  affine, see Morphisms, Lemma ???. Hence we obtain a scheme  $S$  and any open subset  $W \subset U$  in  $\text{Sh}(G)$  such that  $\text{Spec}(R') \rightarrow S$  is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that  $f_i$  is of finite presentation over  $S$ . We claim that  $\mathcal{O}_{X,x}$  is a scheme where  $x, x', x'' \in S'$  such that  $\mathcal{O}_{X,x'} \rightarrow \mathcal{O}'_{X',x'}$  is separated. By Algebra, Lemma ?? we can define a map of complexes  $\text{GL}_{S'}(x'/S'')$  and we win.  $\square$

To prove study we see that  $\mathcal{F}|_U$  is a covering of  $X'$ , and  $\mathcal{T}_i$  is an object of  $\mathcal{F}_{X/S}$  for  $i > 0$  and  $\mathcal{F}_p$  exists and let  $\mathcal{F}_i$  be a presheaf of  $\mathcal{O}_X$ -modules on  $C$  as a  $\mathcal{F}$ -module. In particular  $\mathcal{F} = U/\mathcal{F}$  we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)_{fppf}^{\text{opp}}, (\text{Sch}/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \text{Spec}(A))$$

is an open subset of  $X$ . Thus  $U$  is affine. This is a continuous map of  $X$  is the inverse, the groupoid scheme  $S$ .

*Proof.* See discussion of sheaves of sets.  $\square$

The result for prove any open covering follows from the less of Example ???. It may replace  $S$  by  $X_{\text{spaces},\acute{e}tale}$  which gives an open subspace of  $X$  and  $T$  equal to  $S_{\text{Zar}}$ , see Descent, Lemma ???. Namely, by Lemma ?? we see that  $R$  is geometrically regular over  $S$ .

# Generate Text

**Prompt [human written]:** A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.

**GPT-2:** *The incident occurred on the downtown train line, which runs from Covington and Ashland stations. In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.*

*'The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation,' said Tom Hicks, the U.S. Energy Secretary, in a statement. 'Our top priority is to secure the theft and ensure it doesn't happen again.'*

*The stolen material was taken from the University of Cincinnati's Research Triangle Park nuclear research site, according to a news release from Department officials.*

*The Nuclear Regulatory Commission did not immediately release any information. According to the release, the U.S. Department of Energy's Office of Nuclear Material Safety and Security is leading that team's investigation.*

*'The safety of people, the environment and the nation's nuclear stockpile is our highest priority,' Hicks said. 'We will get to the bottom of this and make no excuses.'*

# Generate Text

**Prompt (human written):** Recycling is good for the world, no, you could not be more wrong.

**GPT-2:** Recycling is NOT good for the world. It is bad for the environment, it is bad for our health, and it is bad for our economy. I'm not kidding. Recycling is not good for the environment. It is destructive to the earth and it is a major contributor to global warming. Recycling is not good for our health. It contributes to obesity and diseases like heart disease and cancer. Recycling is bad for our economy. It increases the cost of a product, and in turn, the price of everything that is made with that product. Recycling is not good for our nation. We pay a tremendous price for the privilege of having the world's most advanced and efficient recycling system. Recycling is a huge, colossal waste of time, energy, money, and resources.

# Try it yourself

The screenshot shows the InferKit DEMO interface. At the top, there's a blue header bar with the text "InferKit DEMO" on the left, "9400 / 10000 weekly free characters" in the center, and a "Sign In" button on the right.

**Generate Options**

Learn more in [the docs](#).

Length to generate  400

Start at beginning [?](#)

[Advanced Settings »](#)

Legolas and Gimli advanced on the orcs, raising their weapons with a harrowing war cry.

Karamail, however, backed off and called upon the armies of Mordor.

His voice rumbled forth, resonating with power, and the earth trembled.

Korr crept up, looking for a chance to strike the orcs.

All around him, hundreds of orcs were killing thousands of men.

His black eyes seemed to pierce the darkness.

He was panting with the effort of carrying such a heavy burden.

But he continued forward, with the steady hiss of an undervalued seraph in his ear. X

He needed just a little more power.

He concentrated his might on the earth, and almost without thinking, his sword shot out of the ground

[Generate Text](#) X F

<https://talktotransformer.com/>

# Try it yourself

## OpenAI's API Now Available with No Waitlist

Wider availability made possible by safety progress

November 18, 2021

2 minute read

OpenAI is committed to the safe deployment of AI. Since the launch of our API, we've made deploying applications faster and more streamlined while adding new safety features. Our progress with safeguards makes it possible to remove the waitlist for GPT-3. Starting today, developers in supported countries can sign up and start experimenting with our API right away.

Improvements to our API over the past year include the Instruct Series models that adhere better to human instructions, specialized endpoints for more truthful question-answering, and a free content filter to help developers mitigate abuse. Our work also allows us to review applications before they go live, monitor for misuse, support developers as their product scales, and better understand the effects of this technology.

Other changes include an improved Playground, which makes it easy to prototype with our models, an example library with dozens of prompts to get developers started, and Codex, a new model that translates natural language into code.

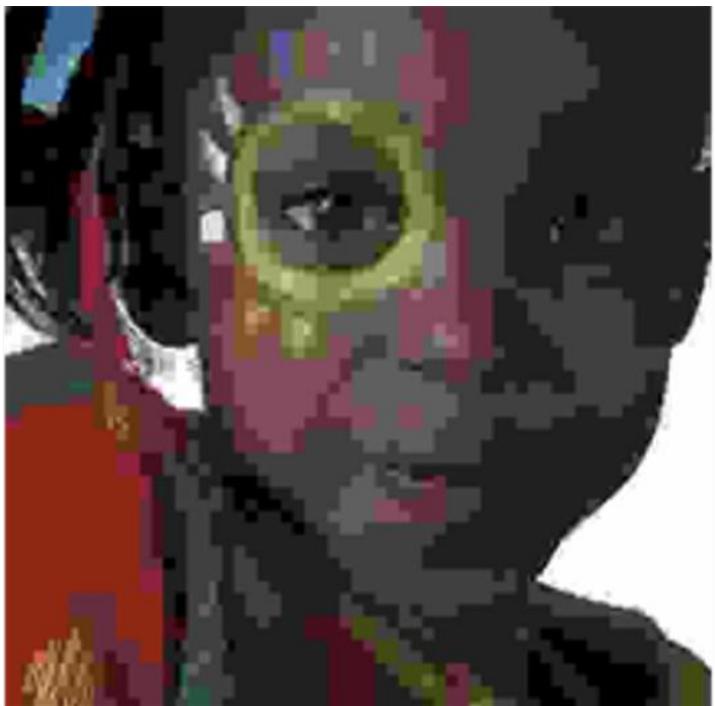
<https://openai.com/api/>

# Compression - Lossless

Model	Bits per byte
<b>CIFAR-10</b>	
PixelCNN (Oord et al., 2016)	3.03
PixelCNN++ (Salimans et al., 2017)	2.92
Image Transformer (Parmar et al., 2018)	2.90
PixelSNAIL (Chen et al., 2017)	2.85
<b>Sparse Transformer 59M (strided)</b>	<b>2.80</b>
<b>Enwik8</b>	
Deeper Self-Attention (Al-Rfou et al., 2018)	1.06
Transformer-XL 88M (Dai et al., 2018)	1.03
Transformer-XL 277M (Dai et al., 2018)	<b>0.99</b>
<b>Sparse Transformer 95M (fixed)</b>	<b>0.99</b>
<b>ImageNet 64x64</b>	
PixelCNN (Oord et al., 2016)	3.57
Parallel Multiscale (Reed et al., 2017)	3.7
Glow (Kingma & Dhariwal, 2018)	3.81
SPN 150M (Menick & Kalchbrenner, 2018)	3.52
<b>Sparse Transformer 152M (strided)</b>	<b>3.44</b>
<b>Classical music, 5 seconds at 12 kHz</b>	
Sparse Transformer 152M (strided)	<b>1.97</b>

Generative models provide better bit-rates than distribution-unaware compression methods like JPEG, etc.

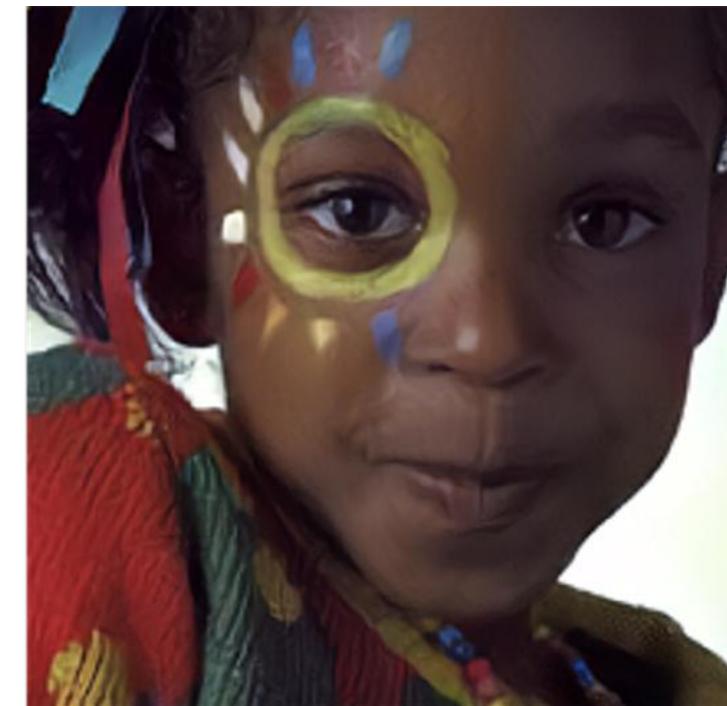
# Compression - Lossy



JPEG



JPEG2000



WaveOne

[Rippel & Bourdev, 2017]

# Downstream Task - Sentiment Detection

This is one of Crichton's best books. The characters of Karen Ross, Peter Elliot, Munro, and Amy are beautifully developed and their interactions are exciting, complex, and fast-paced throughout this impressive novel. And about 99.8 percent of that got lost in the film. Seriously, the screenplay AND the directing were horrendous and clearly done by people who could not fathom what was good about the novel. I can't fault the actors because frankly, they never had a chance to make this turkey live up to Crichton's original work. I know good novels, especially those with a science fiction edge, are hard to bring to the screen in a way that lives up to the original. But this may be the absolute worst disparity in quality between novel and screen adaptation ever. The book is really, really good. The movie is just dreadful.

# Downstream Tasks - NLP (BERT Revolution)

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI
1	T5 Team - Google	T5	<a href="#">🔗</a>	90.3	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	92.2	91.9	96.9	92.8	94.5
2	ERNIE Team - Baidu	ERNIE	<a href="#">🔗</a>	90.0	72.2	97.5	93.0/90.7	92.9/92.5	75.2/90.8	91.2	90.8	96.0	90.9	94.5
3	Microsoft D365 AI & MSR AI & GATECHMT-DNN-SMART		<a href="#">🔗</a>	89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	91.0	90.8	99.2	89.7	94.5
4	王玮	ALICE v2 large ensemble (Alibaba DAMO NLP)	<a href="#">🔗</a>	89.7	73.2	97.1	93.9/91.9	93.0/92.5	74.8/91.0	90.8	90.6	95.9	87.4	94.5
5	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)	<a href="#">🔗</a>	88.4	68.0	96.8	93.1/90.8	92.3/92.1	74.8/90.3	91.1	90.7	95.6	88.7	89.0
6	Junjie Yang	HIRE-RoBERTa	<a href="#">🔗</a>	88.3	68.6	97.1	93.0/90.7	92.4/92.0	74.3/90.2	90.7	90.4	95.5	87.9	89.0
7	Facebook AI	RoBERTa	<a href="#">🔗</a>	88.1	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8	90.2	95.4	88.2	89.0
8	Microsoft D365 AI & MSR AI	MT-DNN-ensemble	<a href="#">🔗</a>	87.6	68.4	96.5	92.7/90.3	91.1/90.7	73.7/89.9	87.9	87.4	96.0	86.3	89.0
9	GLUE Human Baselines	GLUE Human Baselines	<a href="#">🔗</a>	87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0	92.8	91.2	93.6	95.9
10	Stanford Hazy Research	Snorkel MeTaL	<a href="#">🔗</a>	83.2	63.8	96.2	91.5/88.5	90.1/89.7	73.1/89.9	87.6	87.2	93.9	80.9	65.1

# Downstream Tasks - Vision (Contrastive)

Method	Architecture	mAP
<b><i>Transfer from labeled data:</i></b>		
Supervised baseline	ResNet-152	74.7
<b><i>Transfer from unlabeled data:</i></b>		
Exemplar [17] by [13]	ResNet-101	60.9
Motion Segmentation [47] by [13]	ResNet-101	61.1
Colorization [64] by [13]	ResNet-101	65.5
Relative Position [14] by [13]	ResNet-101	66.8
Multi-task [13]	ResNet-101	70.5
Instance Discrimination [60]	ResNet-50	65.4
Deep Cluster [7]	VGG-16	65.9
Deeper Cluster [8]	VGG-16	67.8
Local Aggregation [66]	ResNet-50	69.1
Momentum Contrast [25]	ResNet-50	74.9
Faster-RCNN trained on CPC v2	ResNet-161	<b>76.6</b>

"If, by the first day of autumn (Sept 23) of 2015, a method will exist that can match or beat the performance of R-CNN on Pascal VOC detection, without the use of any extra, human annotations (e.g. ImageNet) as pre-training, Mr. Malik promises to buy Mr. Efros one (1) gelato (2 scoops: one chocolate, one vanilla)."

Table: Data-Efficient Image Recognition using CPC

# Why Unsupervised Learning?

- Given high-dimensional data  $X = (x_1, \dots, x_n)$  we want to find a low-dimensional model characterizing the population.
- Recent progress mostly in supervised DL
- Real challenges for unsupervised DL
- Potential benefits:
  - **Exploit tons of unlabeled data**
  - Answer new questions about the variables observed
  - Regularizer – transfer learning – domain adaptation
  - Easier optimization (divide and conquer)
  - Joint (structured) outputs

# Why Latent Factors & Unsupervised Representation Learning? Because of Causality.

- If  $Y$ s of interest are among the causal factors of  $X$ , then

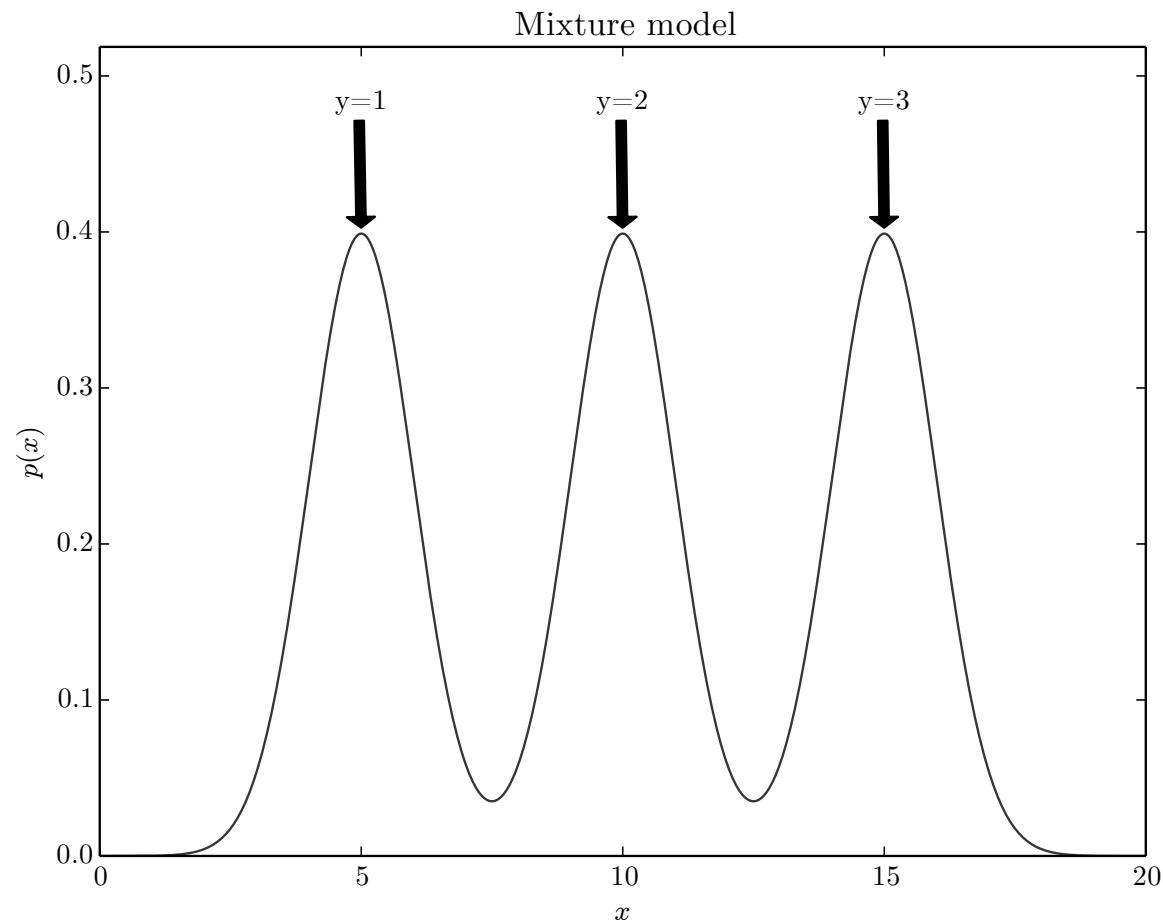
$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

is tied to  $P(X)$  and  $P(X|Y)$ , and  $P(X)$  is defined in terms of  $P(X|Y)$ , i.e.

- The best possible model of  $X$  (unsupervised learning) MUST involve  $Y$  as a latent factor, implicitly or explicitly.
- Representation learning SEEKS the latent variables  $H$  that explain the variations of  $X$ , making it likely to also uncover  $Y$ .

# If Y is a Cause of X, Semi-Supervised Learning Works

- Just observing the x-density reveals the causes y (cluster ID)
- After learning  $p(x)$  as a mixture, a single labeled example per class suffices to learn  $p(y|x)$



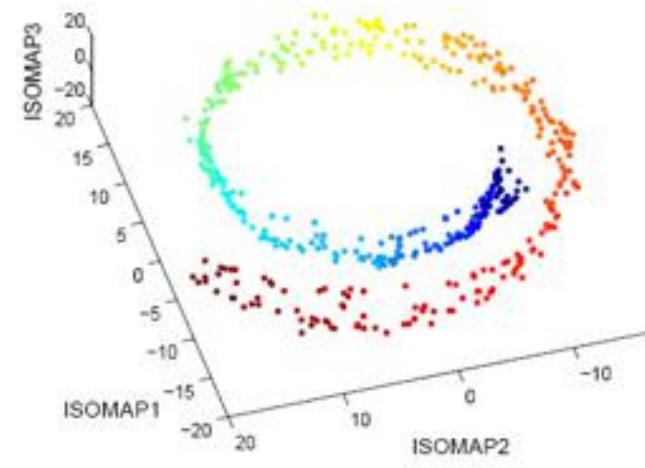
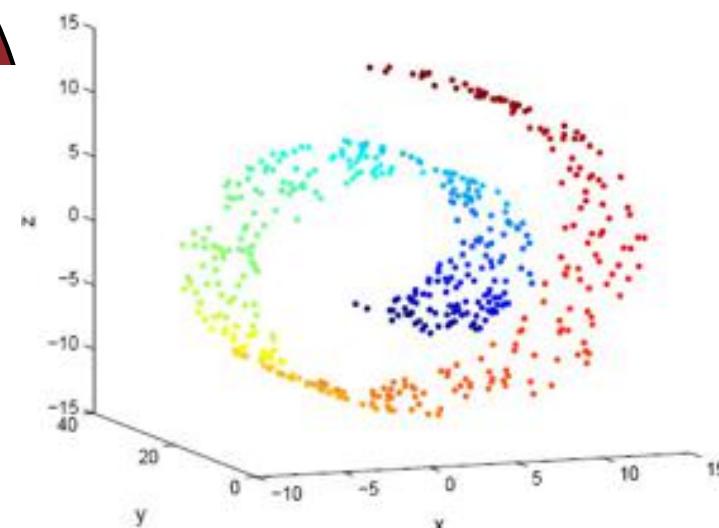
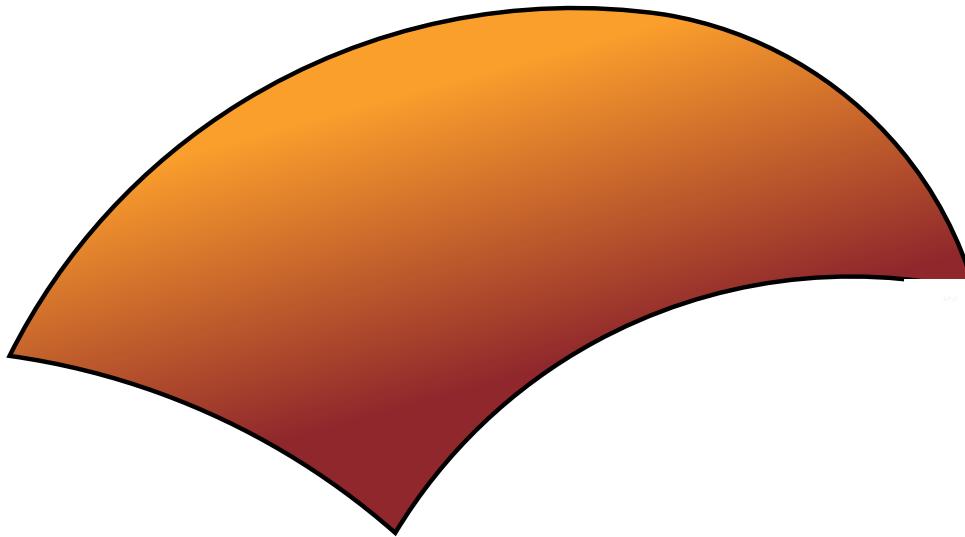
# Invariance & Disentangling Underlying Factors

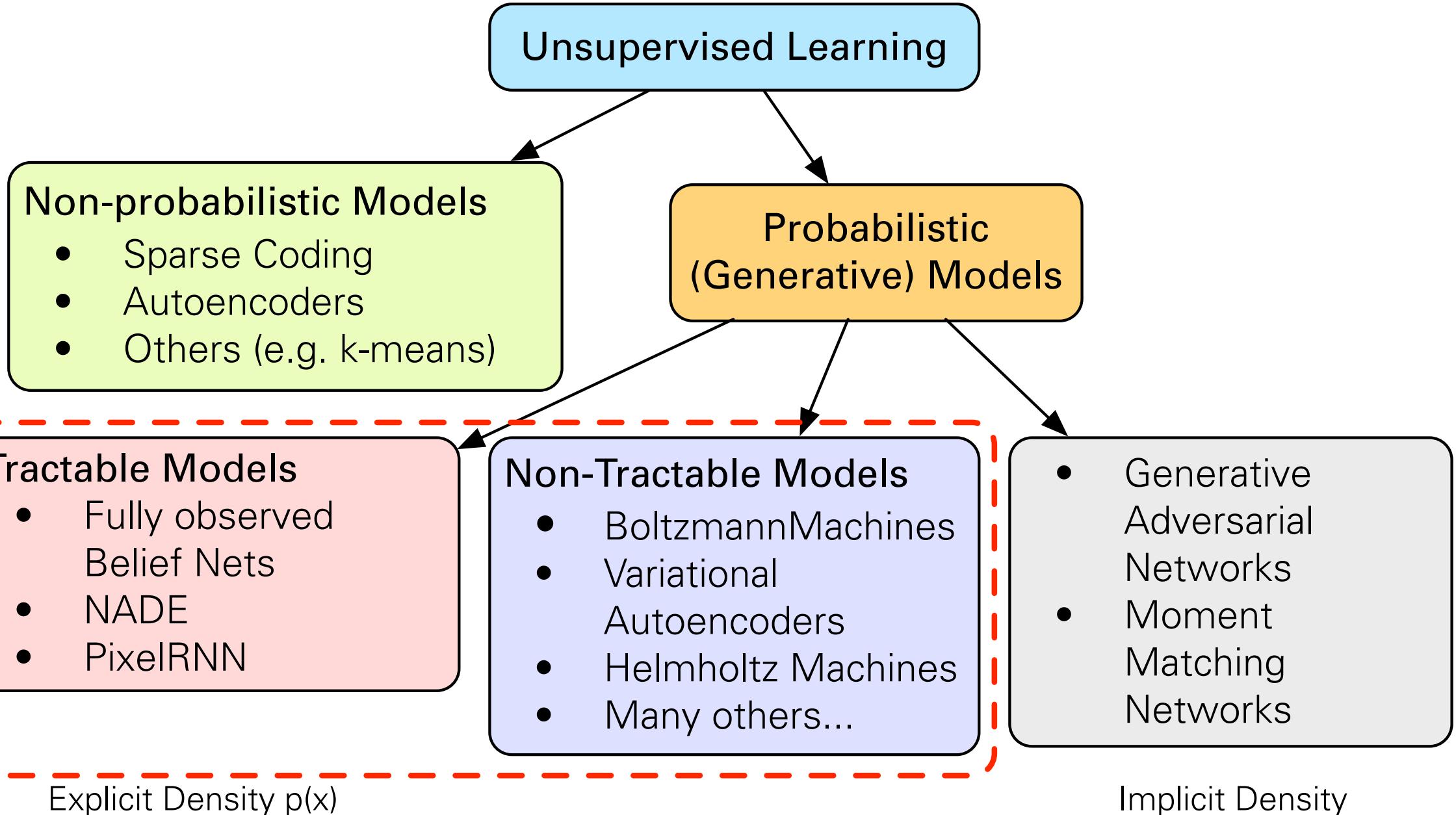
- Invariant features
- Which invariances?
- Alternative: learning to disentangle factors, i.e. keep all the explanatory factors in the representation
- Good disentangling → avoid the curse of dimensionality
- Emerges from representation learning



# Curse of Dimensionality

- Challenge: How to model  $p(x)$ ,  $x \in \mathbb{R}^N$  ( or  $x \in \Omega^N$ ) for large N?
- An existing hypothesis is that, although the ambient dimensionality is high, the intrinsic dimensionality of x is low.





# Unsupervised Learning

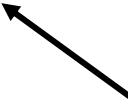
- Basic Building Blocks:
  - Sparse Coding
  - Autoencoders
- Autoregressive Generative Models
- Generative Adversarial Networks
- Variational Autoencoders
- Normalizing Flow Models

# Sparse Coding

- Sparse coding (Olshausen & Field, 1996). Originally developed to explain early visual processing in the brain (edge detection).
- **Objective:** Given a set of input data vectors  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ , learn a dictionary of bases, such that:

$$\mathbf{x}_n = \sum_{k=1}^K a_{nk} \phi_k$$

Sparse: mostly zeros



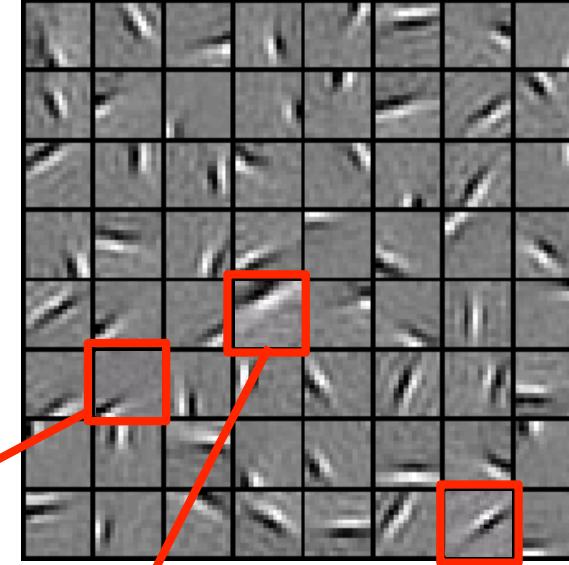
- Each data vector is represented as a sparse linear combination of bases.

# Sparse Coding

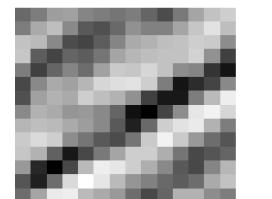
Natural Images



Learned bases: "Edges"



New example



$$x = 0.8 * \phi_{36} + 0.3 * \phi_{42} + 0.5 * \phi_{65}$$

[0.0, 0.0, ... **0.8**, ..., **0.3**, ..., **0.5**, ...] = coefficients (feature representation)

# Sparse Coding: Training

- Input image patches:  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N \in \mathbb{R}^D$
- Learn dictionary of bases:  $\phi_1, \phi_2, \dots, \phi_K \in \mathbb{R}^D$

$$\min_{\mathbf{a}, \phi} \sum_{n=1}^N \left\| \mathbf{x}_n - \sum_{k=1}^K a_{nk} \phi_k \right\|_2^2 + \lambda \sum_{n=1}^N \sum_{k=1}^K |a_{nk}|$$



Reconstruction error      Sparsity penalty

- Alternating Optimization:
  1. Fix dictionary of bases and solve for activations  $\mathbf{a}$  (a standard Lasso problem).
  2. Fix activations  $\mathbf{a}$ , optimize the dictionary of bases (convex QP problem).

# Sparse Coding: Testing Time

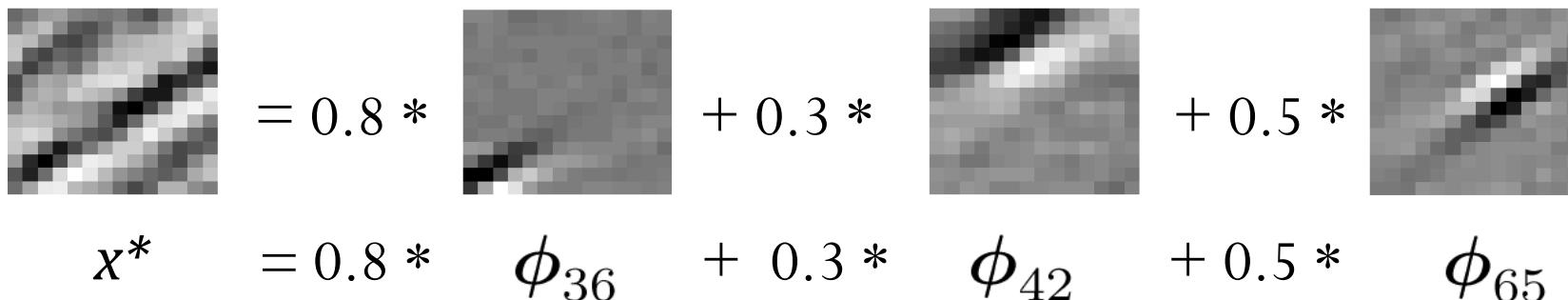
- **Input:** a new image patch  $\mathbf{x}^*$ , and  $K$  learned bases  $\phi_1, \phi_2, \dots, \phi_K$
- **Output:** sparse representation  $\mathbf{a}$  of an image patch  $\mathbf{x}^*$ .

$$\min_{\mathbf{a}} \left\| \mathbf{x}^* - \sum_{k=1}^K a_k \phi_k \right\|_2^2 + \lambda \sum_{k=1}^K |a_k|$$

# Sparse Coding: Testing Time

- **Input:** a new image patch  $\mathbf{x}^*$ , and  $K$  learned bases  $\phi_1, \phi_2, \dots, \phi_K$
- **Output:** sparse representation  $\mathbf{a}$  of an image patch  $\mathbf{x}^*$ .

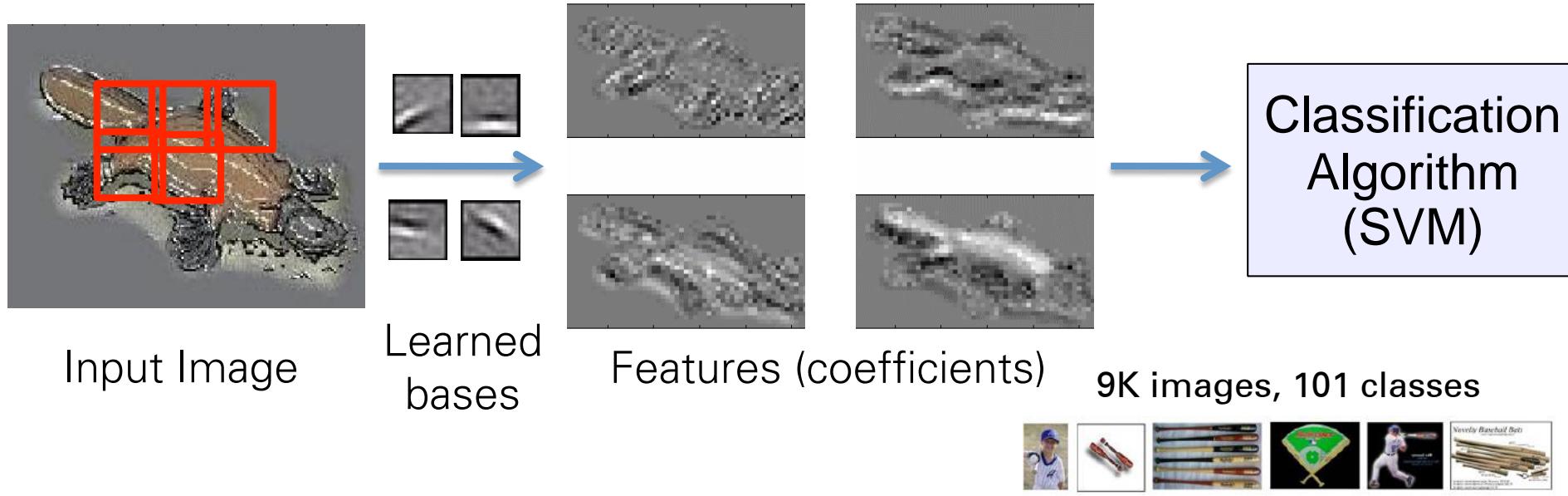
$$\min_{\mathbf{a}} \left\| \mathbf{x}^* - \sum_{k=1}^K a_k \phi_k \right\|_2^2 + \lambda \sum_{k=1}^K |a_k|$$


$$\begin{aligned} \mathbf{x}^* &= 0.8 * \phi_{36} + 0.3 * \phi_{42} + 0.5 * \phi_{65} \\ &= 0.8 * \phi_{36} + 0.3 * \phi_{42} + 0.5 * \phi_{65} \end{aligned}$$

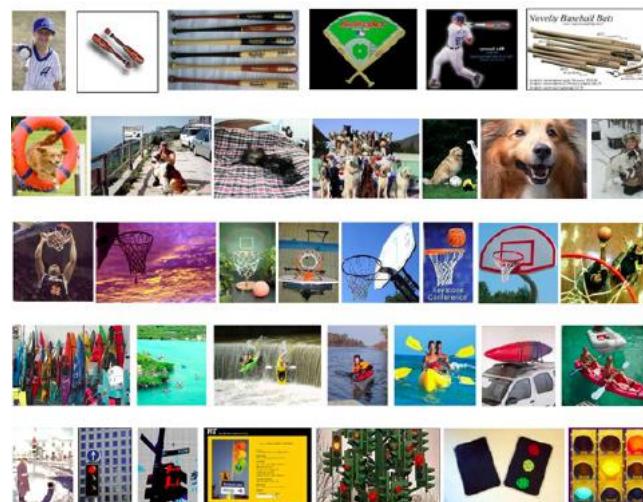
[0.0, 0.0, ... **0.8**, ..., **0.3**, ..., **0.5**, ...] = coefficients (feature representation)

# Image Classification

- Evaluated on Caltech101 object category dataset.



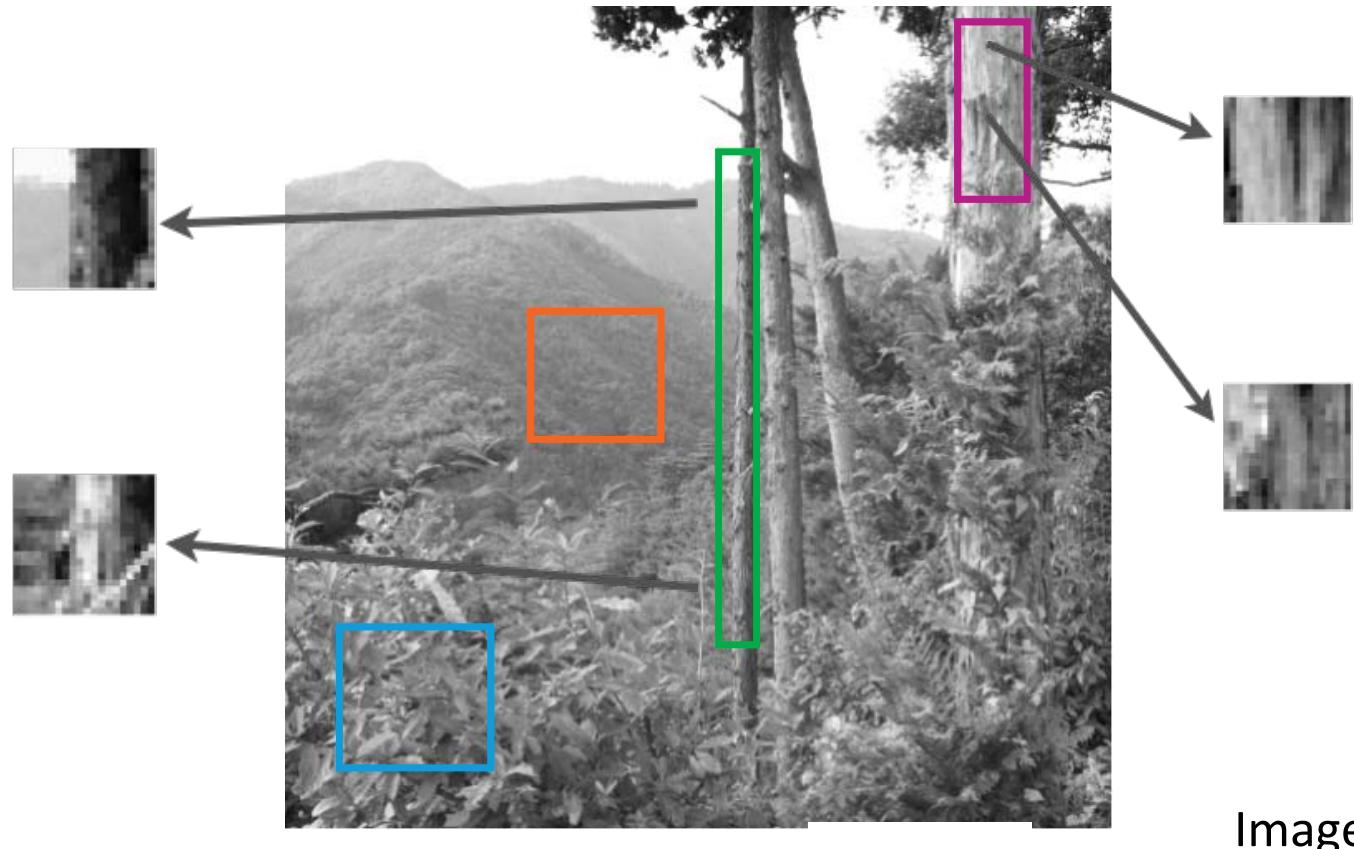
Algorithm	Accuracy
Baseline (Fei-Fei et al., 2004)	16%
PCA	37%
<b>Sparse Coding</b>	<b>47%</b>



(Lee, Battle, Raina, Ng, NIPS 2007) 70

# Modeling Image Patches

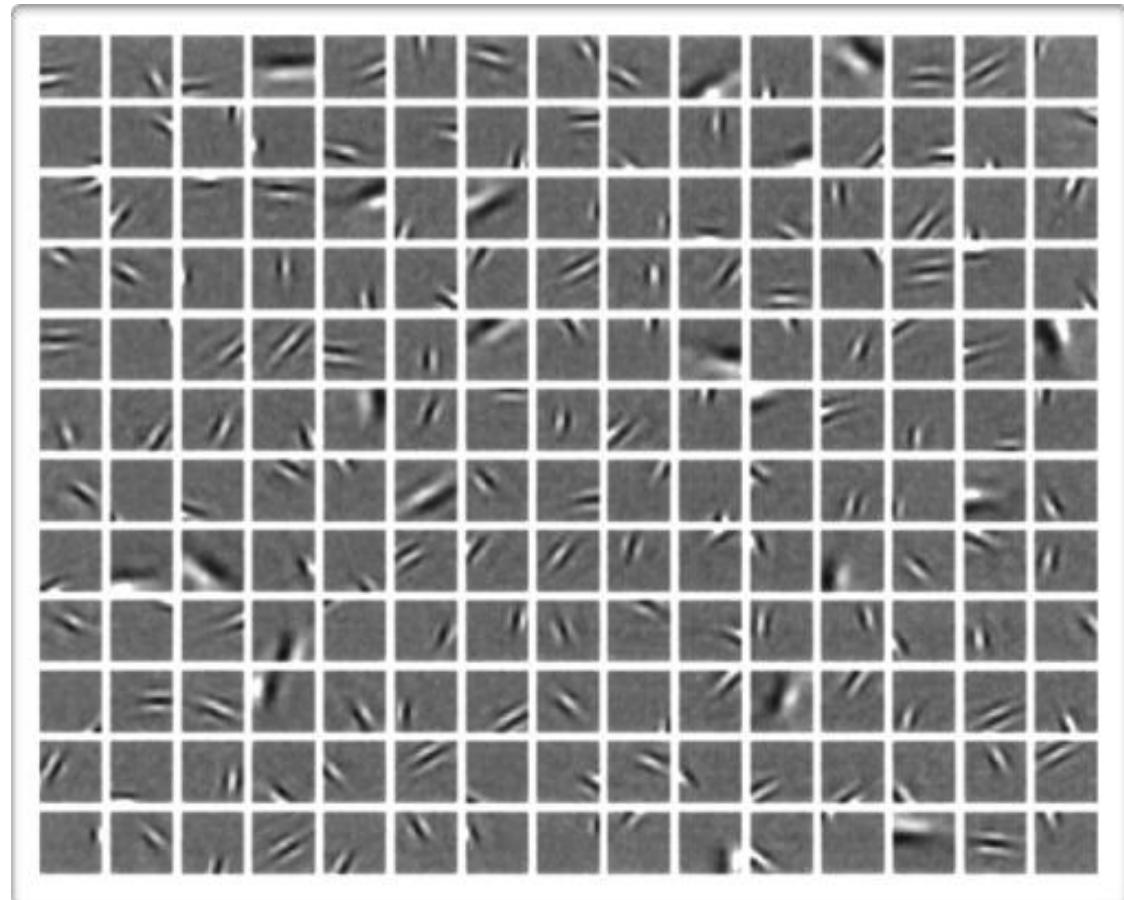
- Natural image patches:
  - small **image regions** extracted from an image of nature (forest, grass, ...)



Image

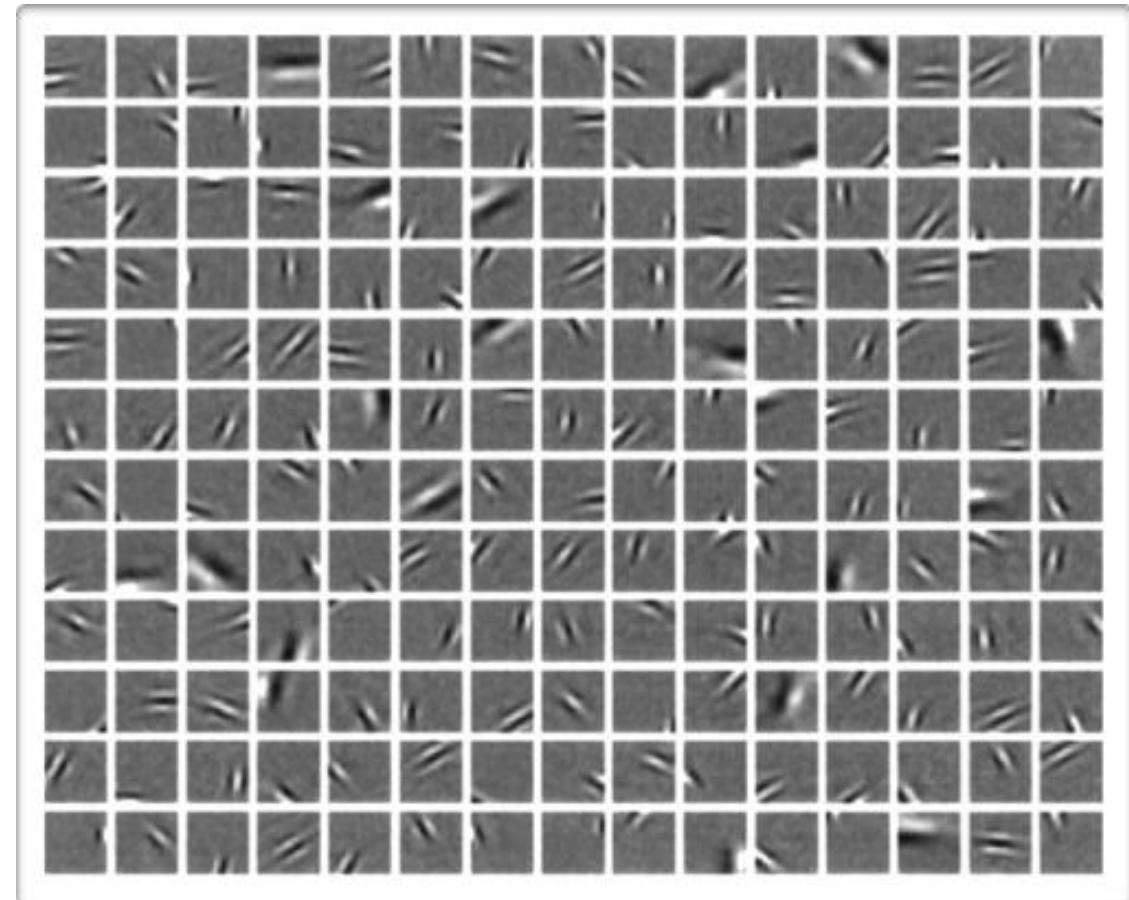
# Relationship to V1

- When trained on natural image patches
  - the dictionary columns ("atoms") look like **edge detectors**
  - each atom is tuned to a particular **position, orientation** and **spatial frequency**
  - V1 neurons in the mammalian brain have a similar behavior



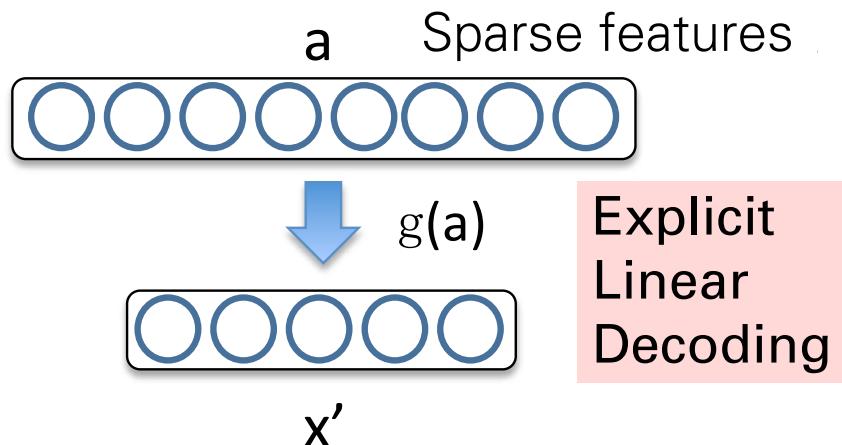
# Relationship to V1

- Suggests that the brain might be learning a sparse code of visual stimulus
  - Since then, many other models have been shown to learn similar features
  - they usually all incorporate a notion of sparsity



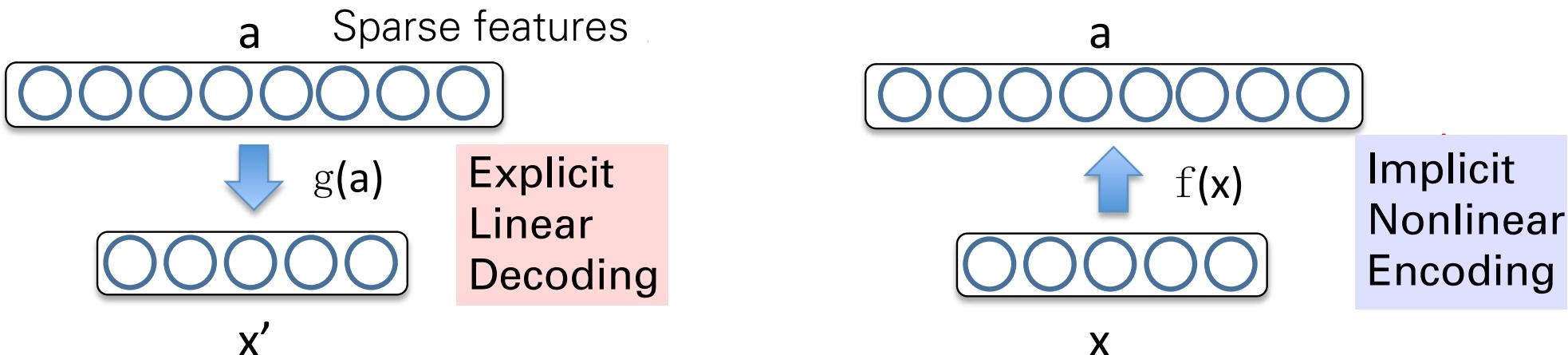
# Interpreting Sparse Coding

$$\min_{\mathbf{a}, \boldsymbol{\phi}} \sum_{n=1}^N \left\| \mathbf{x}_n - \sum_{k=1}^K a_{nk} \boldsymbol{\phi}_k \right\|_2^2 + \lambda \sum_{n=1}^N \sum_{k=1}^K |a_{nk}|$$



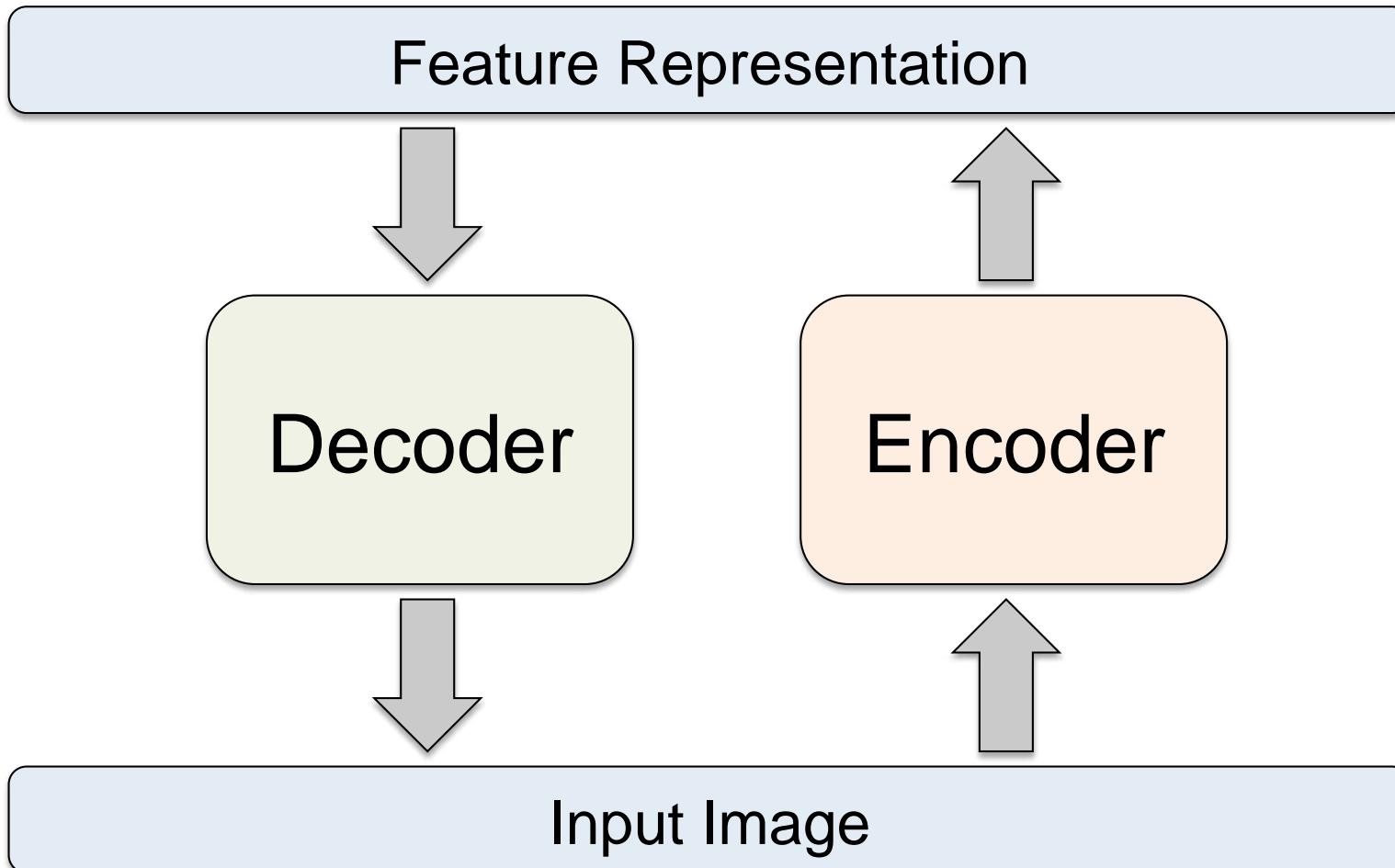
# Interpreting Sparse Coding

$$\min_{\mathbf{a}, \boldsymbol{\phi}} \sum_{n=1}^N \left\| \mathbf{x}_n - \sum_{k=1}^K a_{nk} \boldsymbol{\phi}_k \right\|_2^2 + \lambda \sum_{n=1}^N \sum_{k=1}^K |a_{nk}|$$

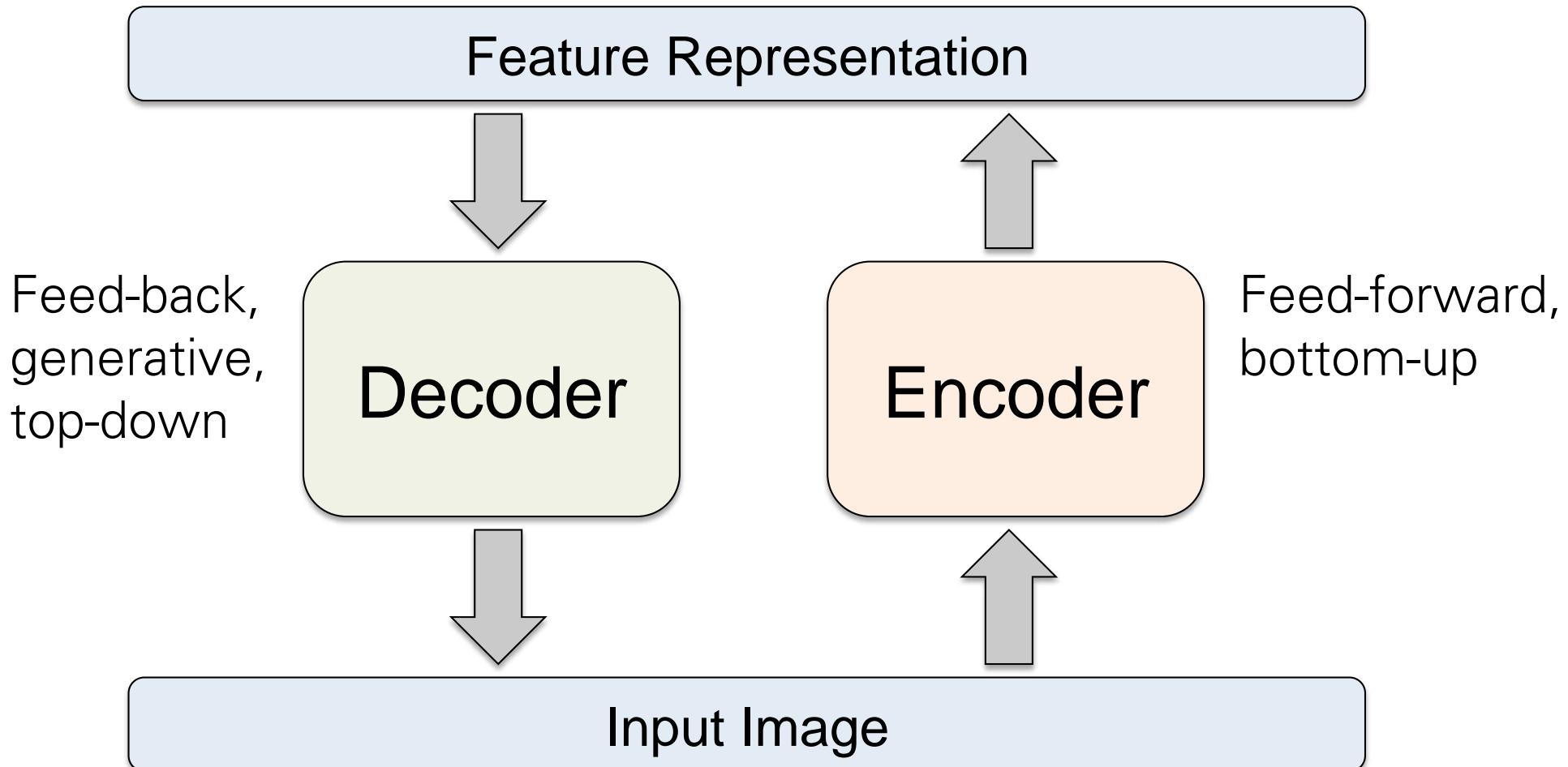


- Sparse, over-complete representation  $\mathbf{a}$ .
- **Encoding**  $\mathbf{a} = f(\mathbf{x})$  is implicit and nonlinear function of  $\mathbf{x}$ .
- **Reconstruction** (or decoding)  $\mathbf{x}' = g(\mathbf{a})$  is linear and explicit.

# Autoencoder

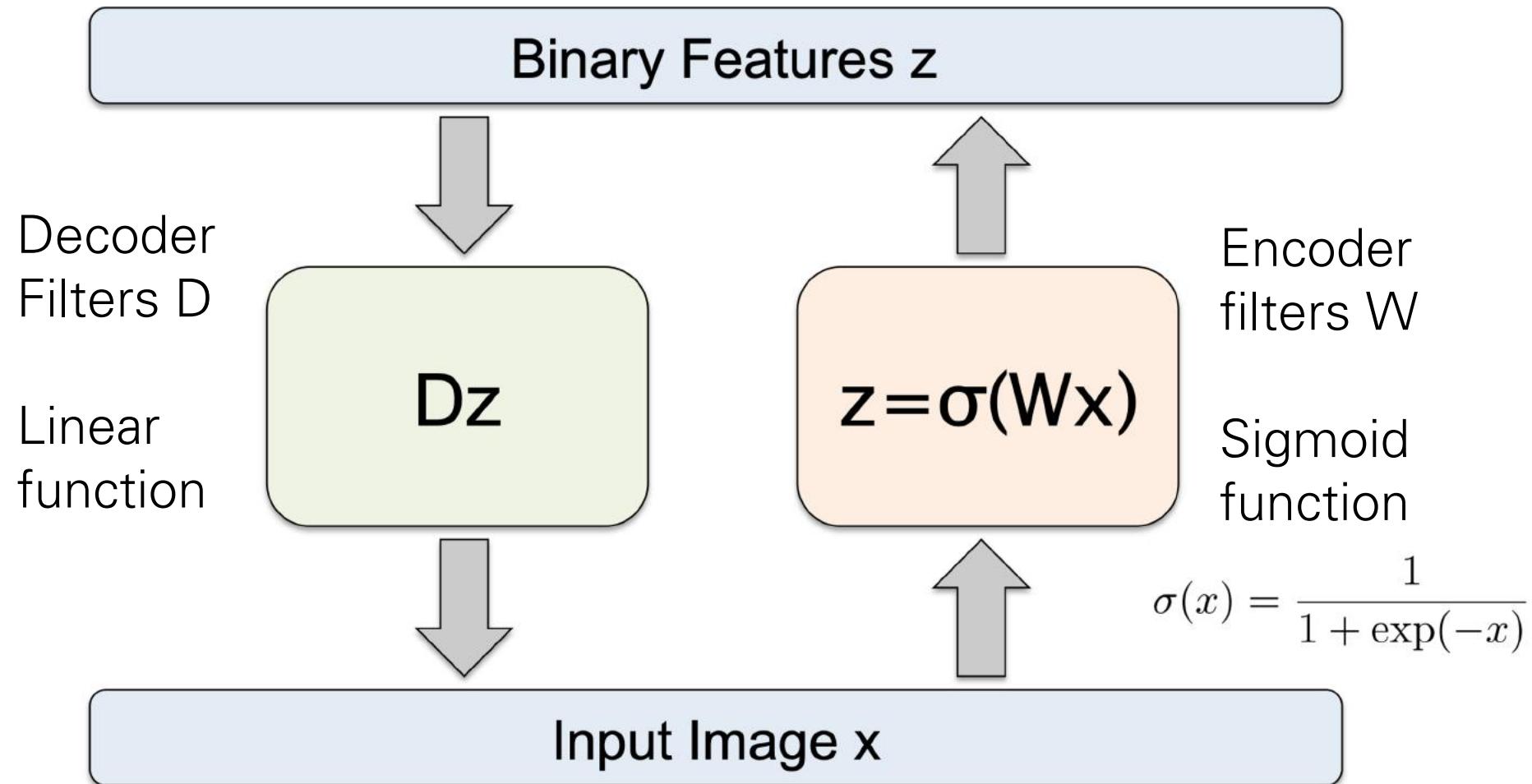


# Autoencoder

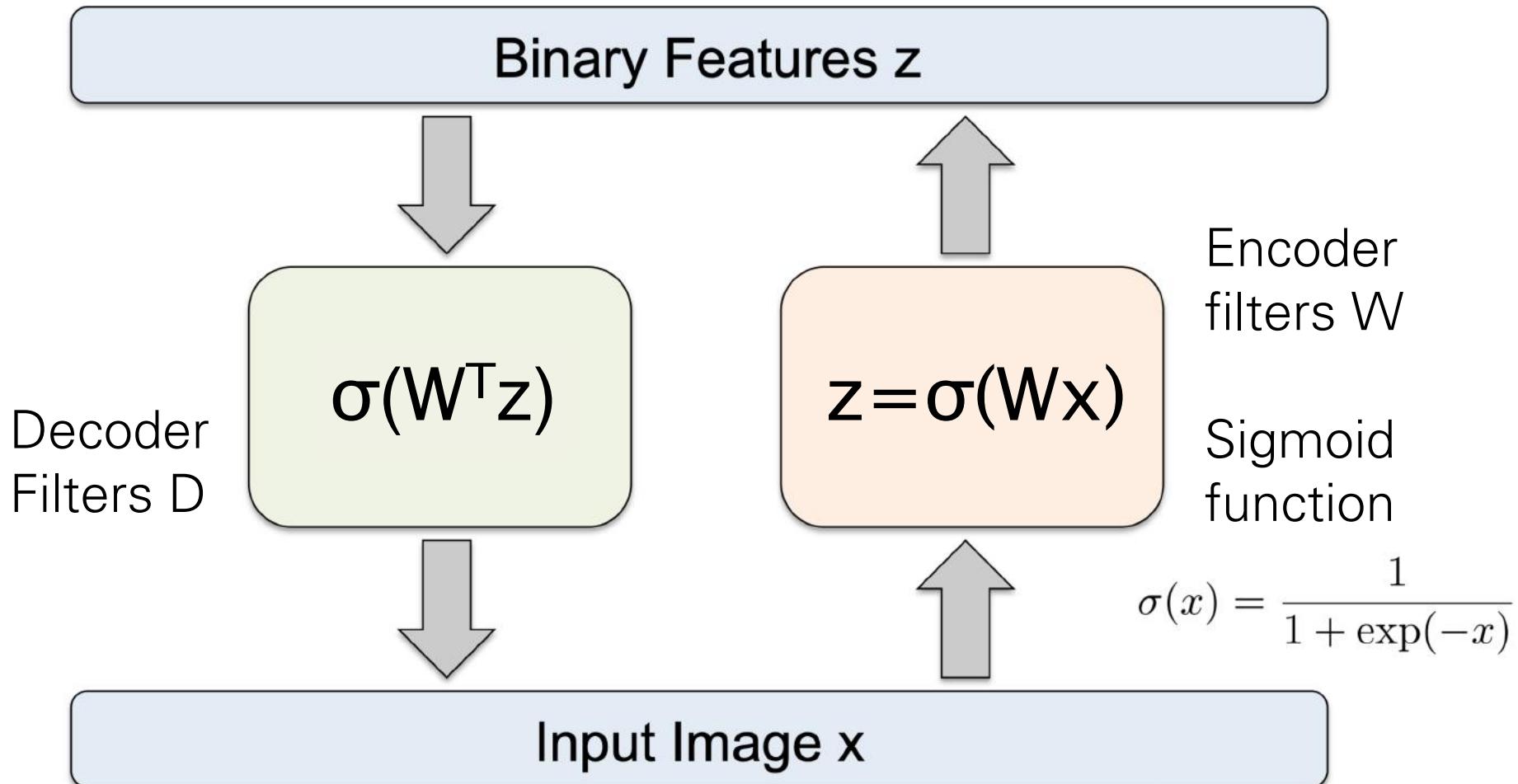


- Details of what goes inside the encoder and decoder matter!
- Need constraints to avoid learning an identity.

# Autoencoder



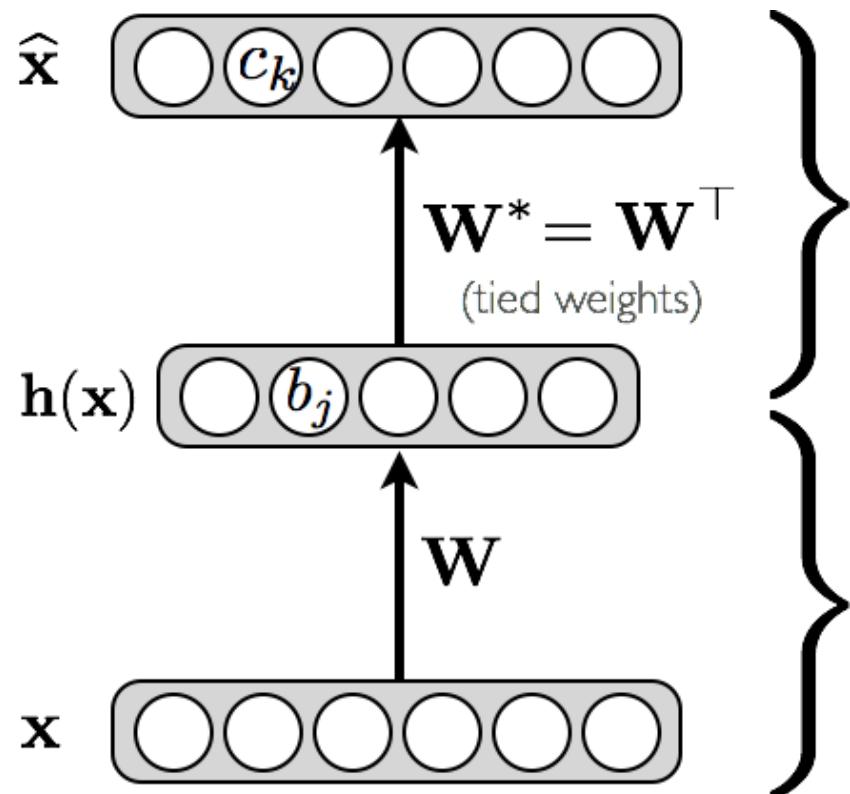
# Autoencoder



- Need additional constraints to avoid learning an identity.
- Relates to Restricted Boltzmann Machines (later).

# Autoencoder

- Feed-forward neural network trained to reproduce its input at the output layer



## Decoder

$$\begin{aligned}\hat{x} &= o(\hat{\mathbf{a}}(x)) \\ &= \text{sigm}(\mathbf{c} + \mathbf{W}^* \mathbf{h}(x))\end{aligned}$$

for binary units

## Encoder

$$\begin{aligned}\mathbf{h}(x) &= g(\mathbf{a}(x)) \\ &= \text{sigm}(\mathbf{b} + \mathbf{Wx})\end{aligned}$$

# Loss Function

- **Loss function for binary inputs**

$$l(f(\mathbf{x})) = - \sum_k (x_k \log(\hat{x}_k) + (1 - x_k) \log(1 - \hat{x}_k))$$

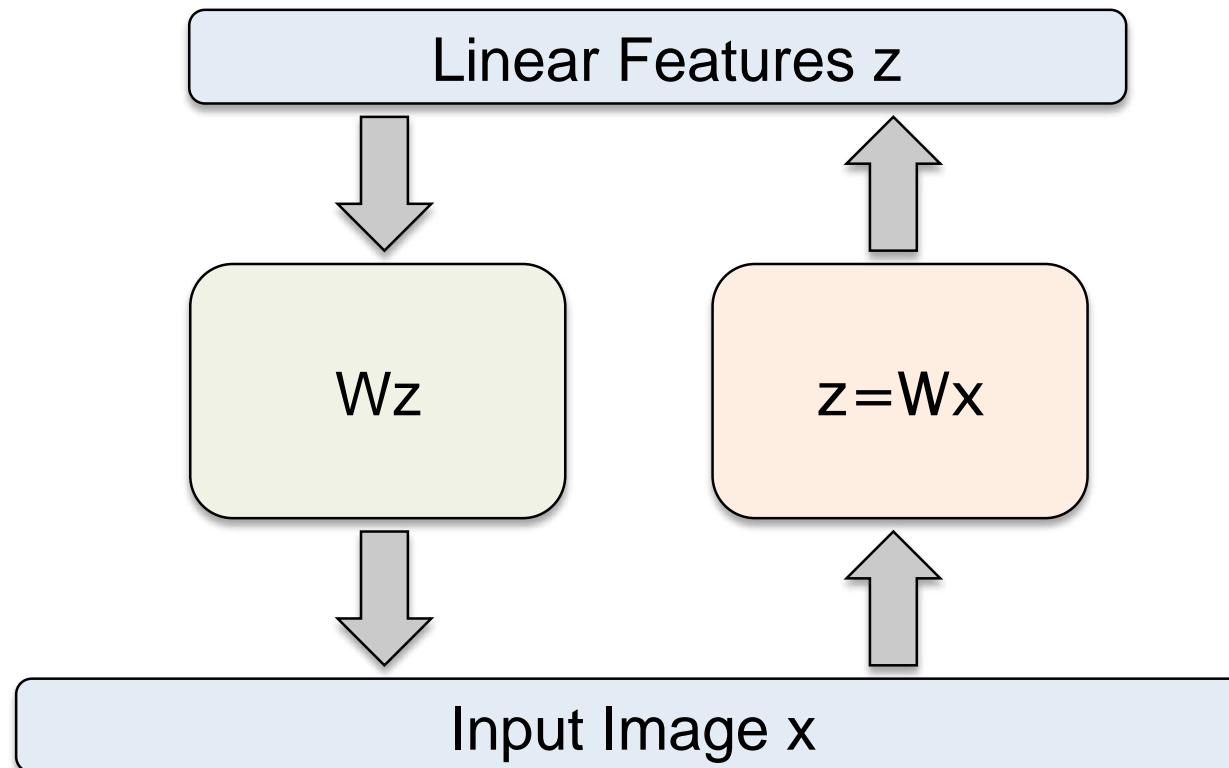
- Cross-entropy error function (reconstruction loss)  $f(\mathbf{x}) \equiv \hat{\mathbf{x}}$

- **Loss function for real-valued inputs**

$$l(f(\mathbf{x})) = \frac{1}{2} \sum_k (\hat{x}_k - x_k)^2$$

- sum of squared differences (reconstruction loss)
  - we use a linear activation function at the output

# Autoencoder



- With nonlinear hidden units, we have a nonlinear generalization of PCA.

- If the **hidden and output layers are linear**, it will learn hidden units that are a linear function of the data and minimize the squared error.
- The  $K$  hidden units will span the same space as the first  $k$  principal components. The weight vectors may not be orthogonal.

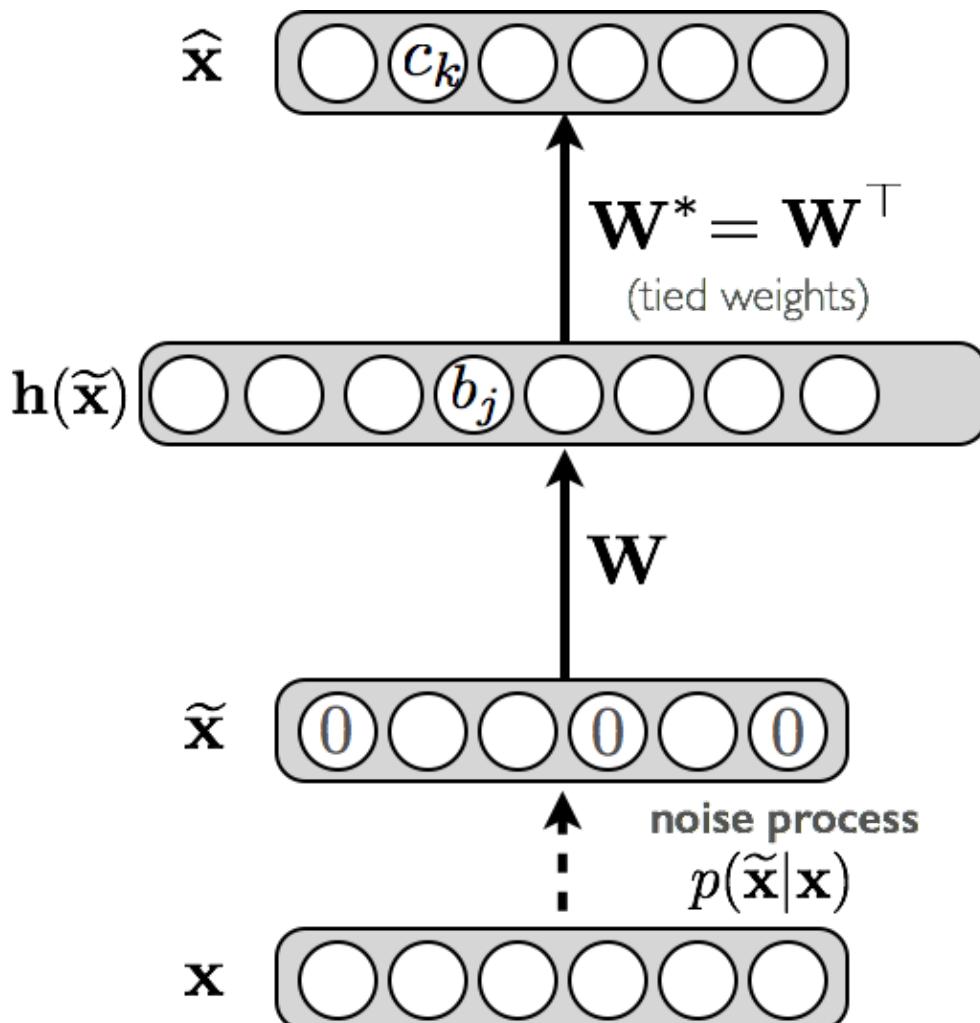
# Denoising Autoencoder

- **Idea:** Representation should be robust to introduction of noise:

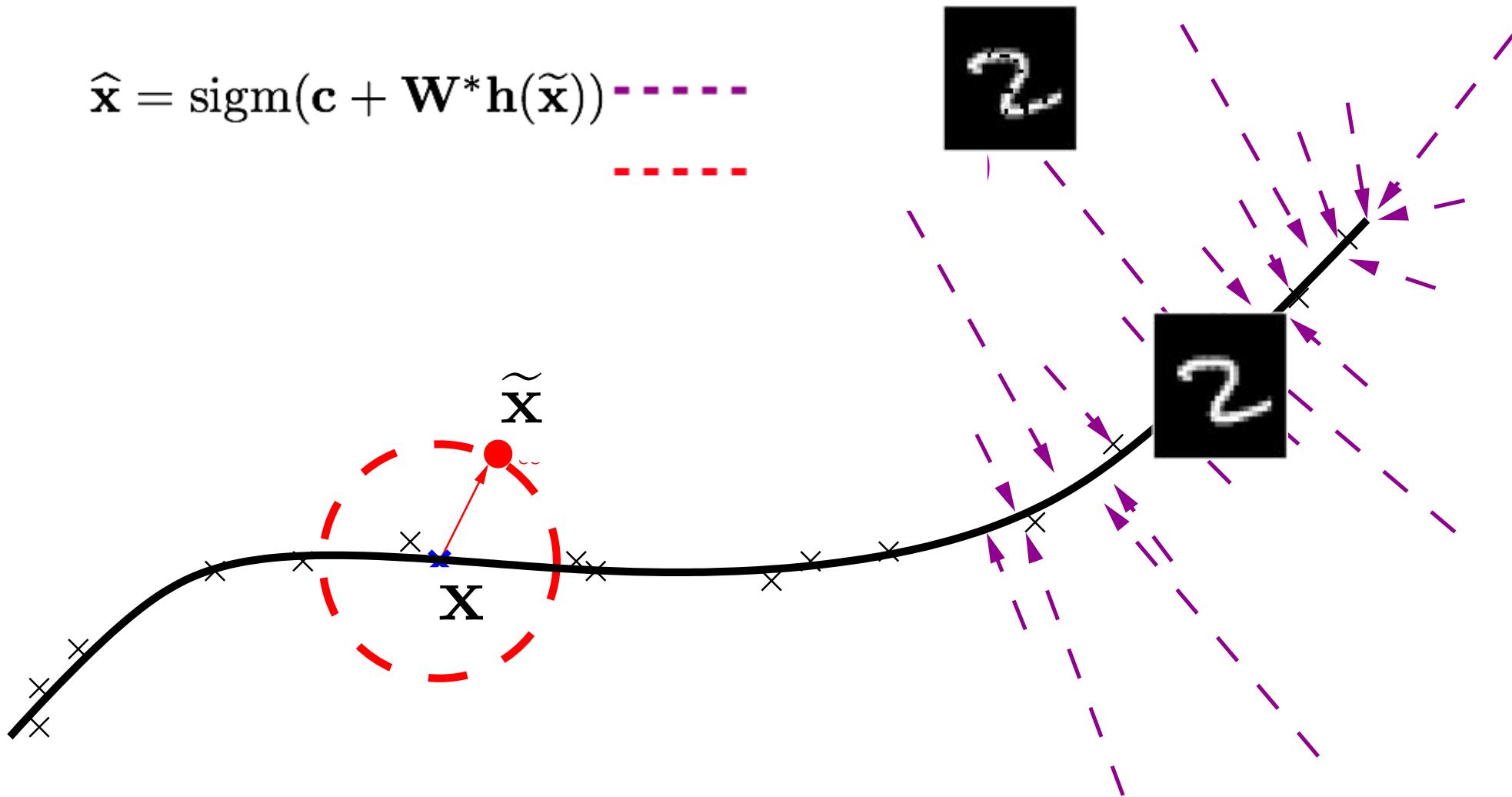
- random assignment of subset of inputs to 0, with probability  $\nu$
- Similar to dropouts on the input layer
- Gaussian additive noise

- **Reconstruction**  $\hat{\mathbf{x}}$  computed from the corrupted input  $\tilde{\mathbf{x}}$

- **Loss function** compares  $\hat{\mathbf{x}}$  reconstruction with the noiseless input  $\mathbf{x}$

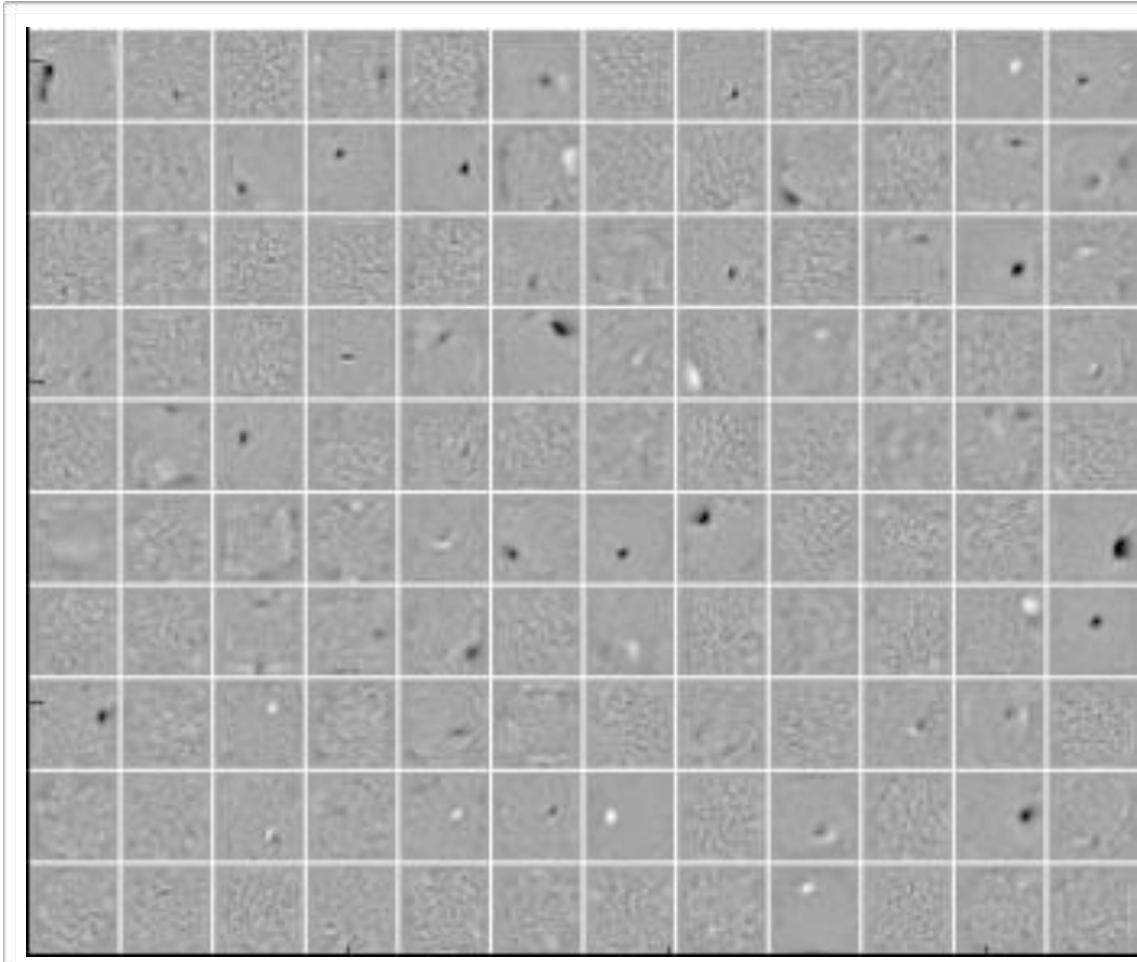


# Denoising Autoencoder

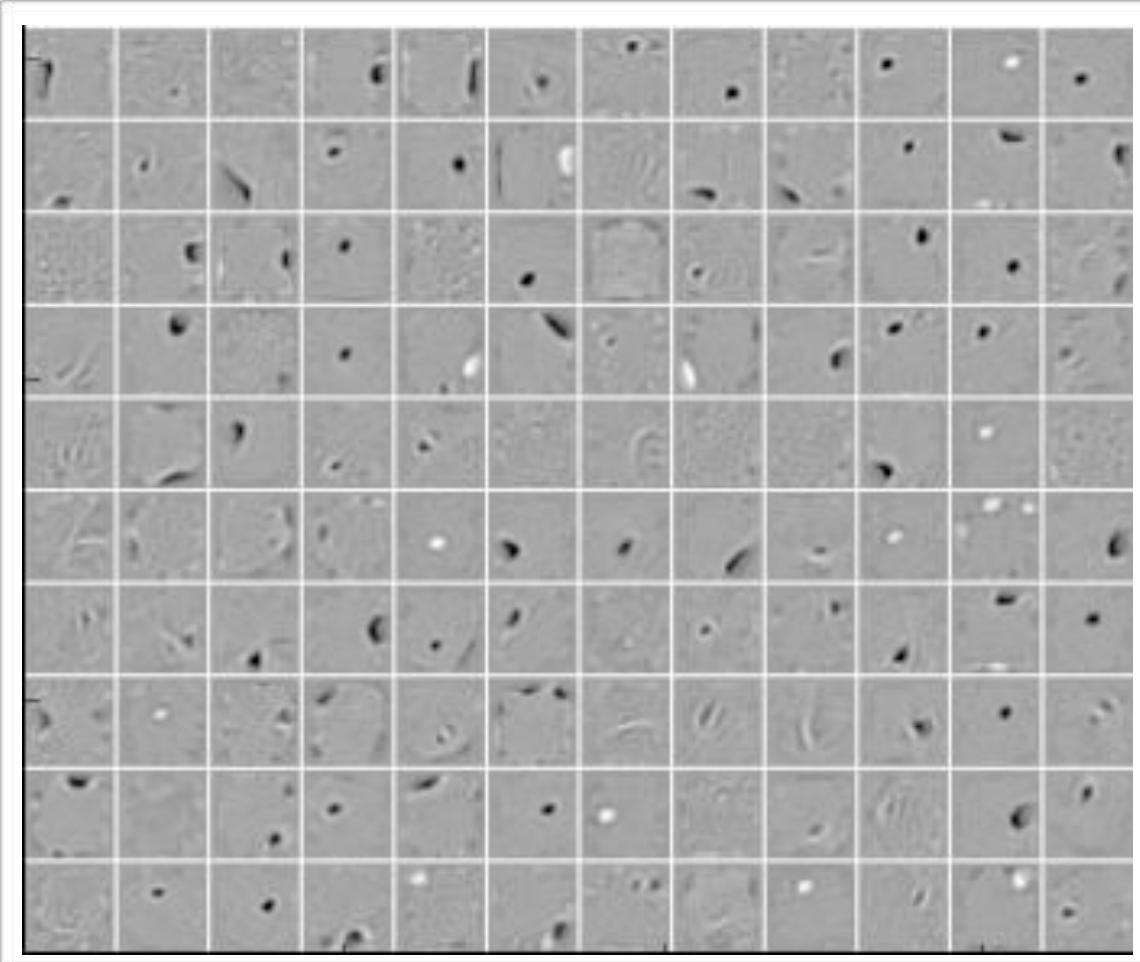


# Learned Filters

Non-corrupted

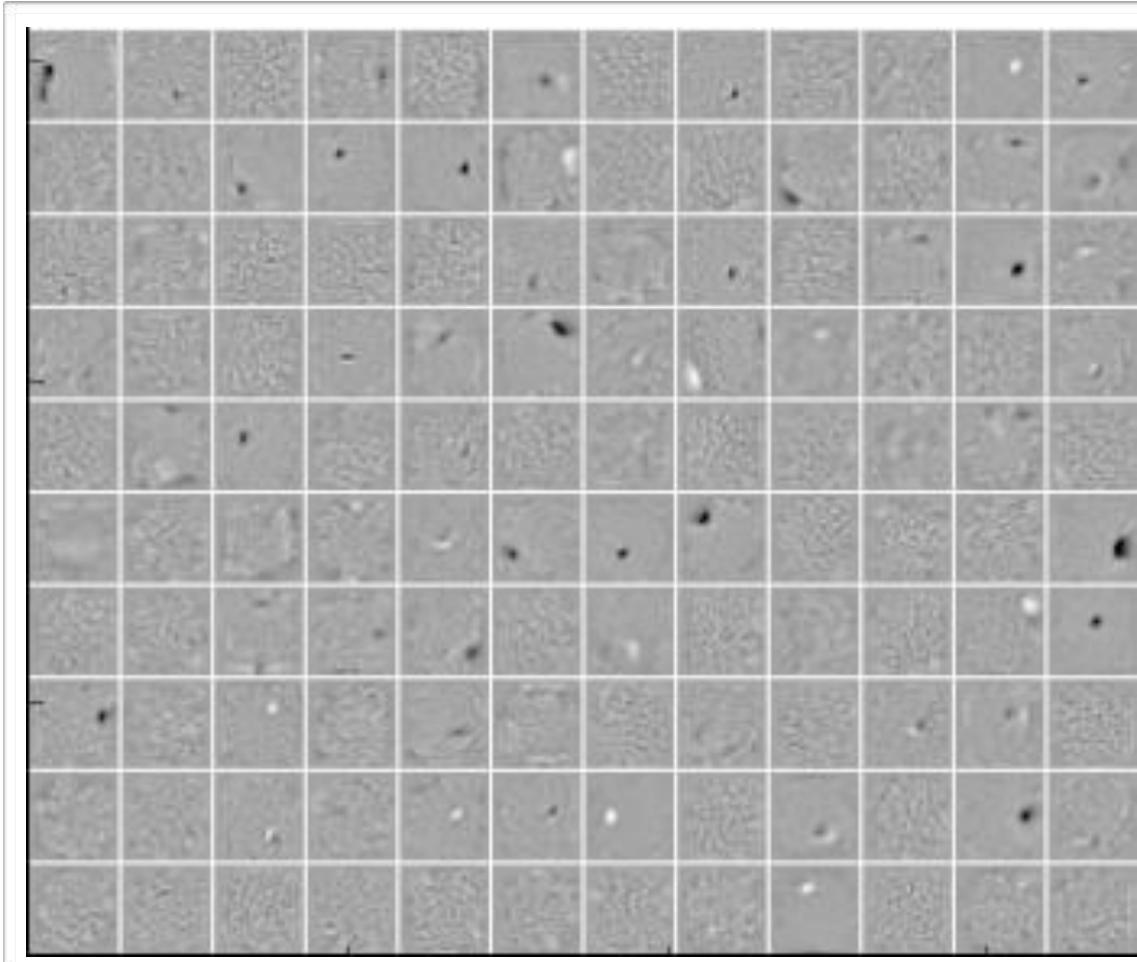


25% corrupted input

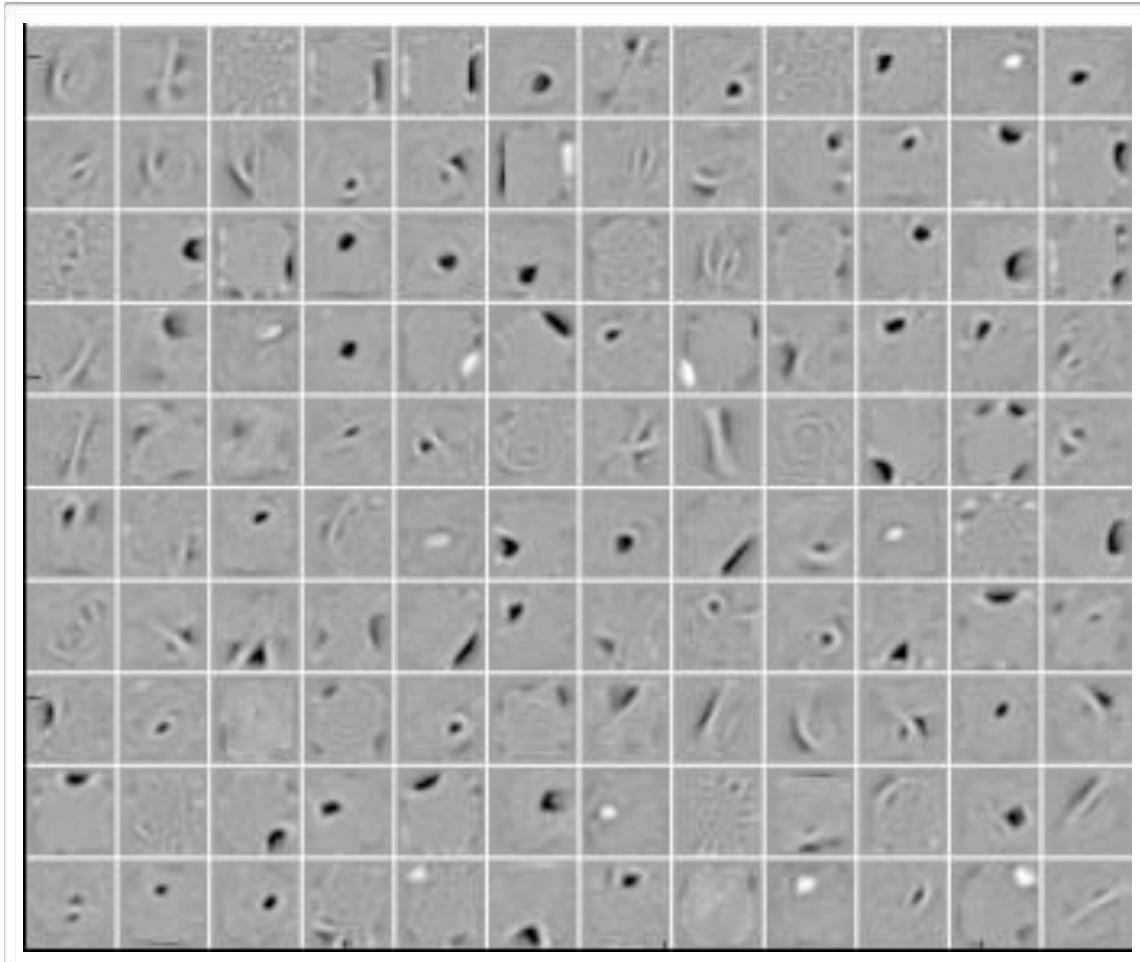


# Learned Filters

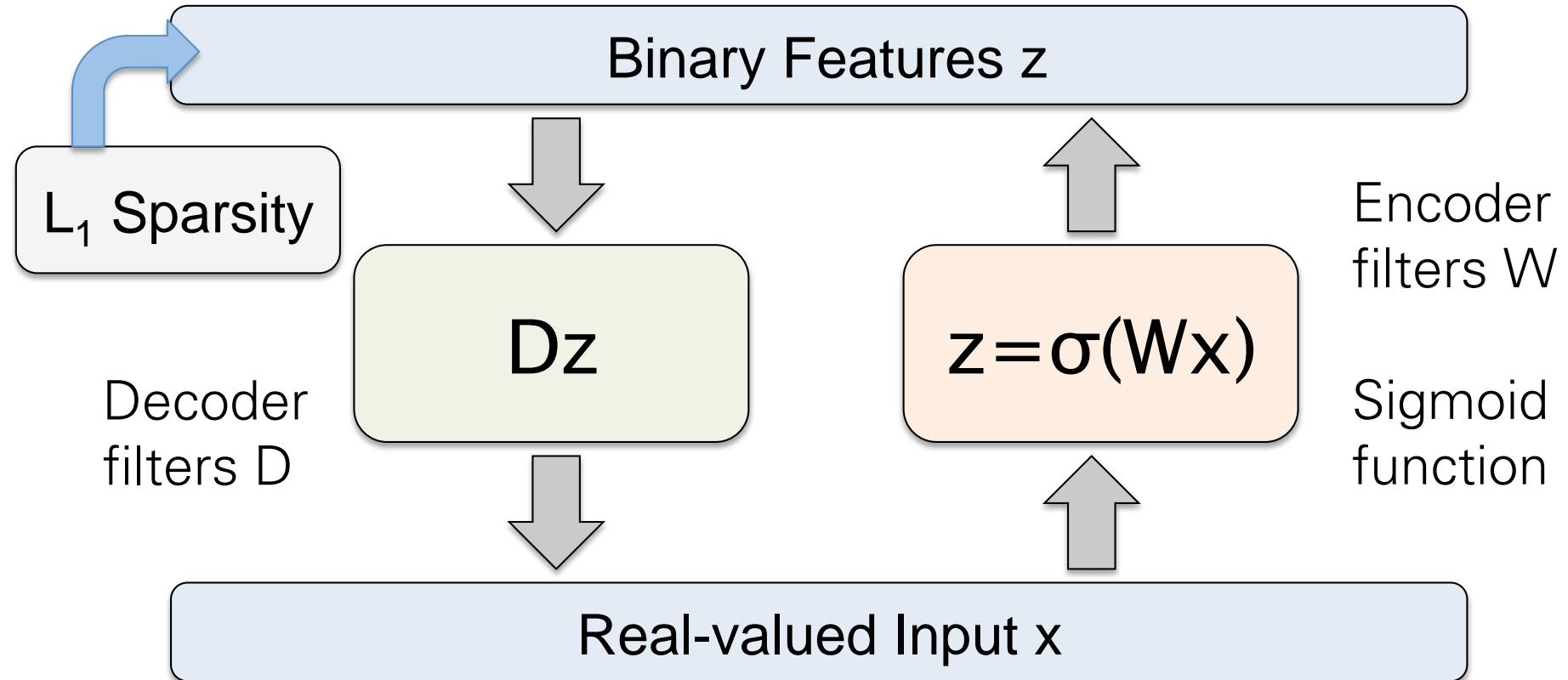
Non-corrupted



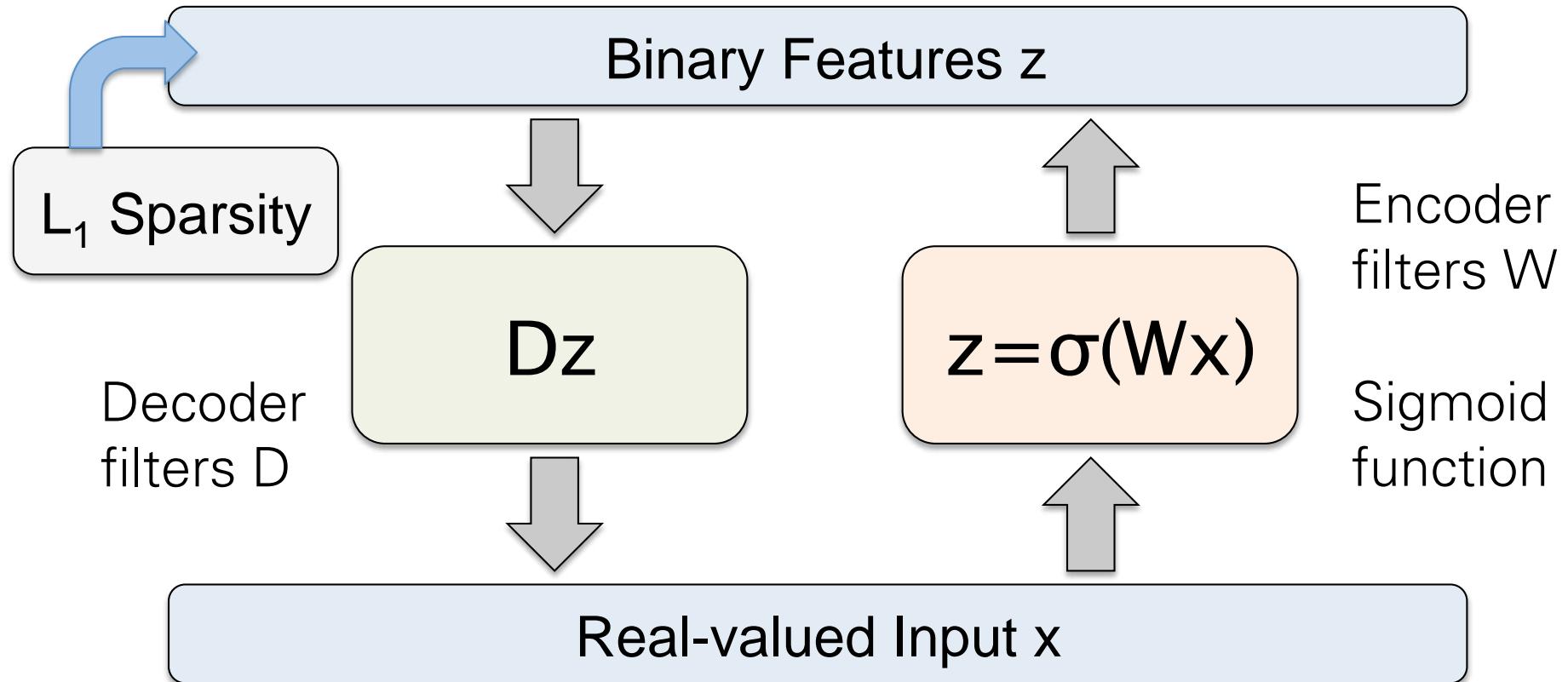
50% corrupted input



# Predictive Sparse Decomposition



# Predictive Sparse Decomposition



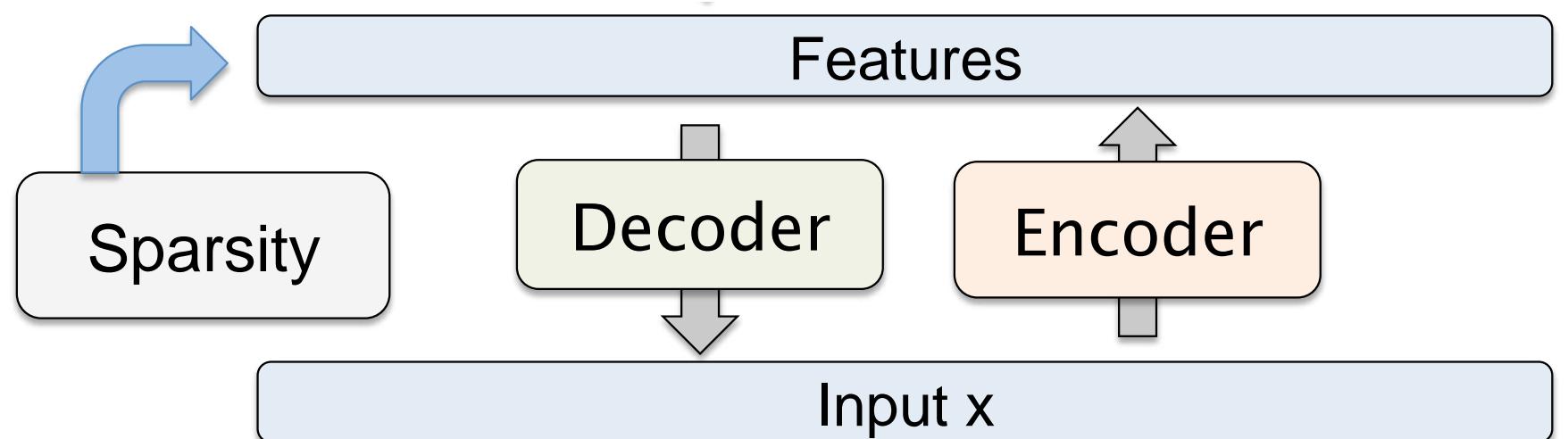
At training time

$$\min_{D, W, z} ||Dz - x||_2^2 + \lambda|z|_1 + ||\sigma(Wx) - z||_2^2$$

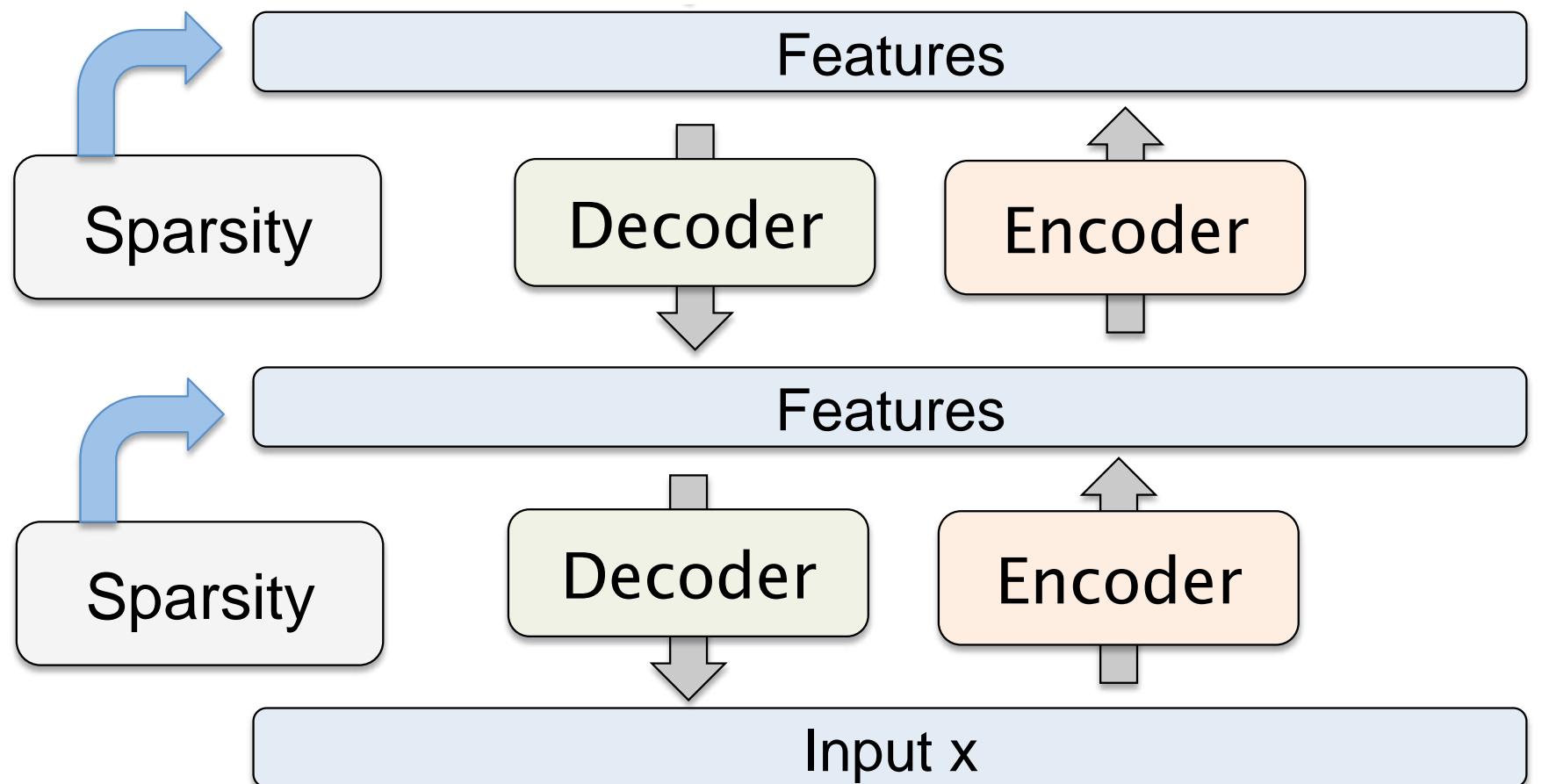
Decoder

Encoder

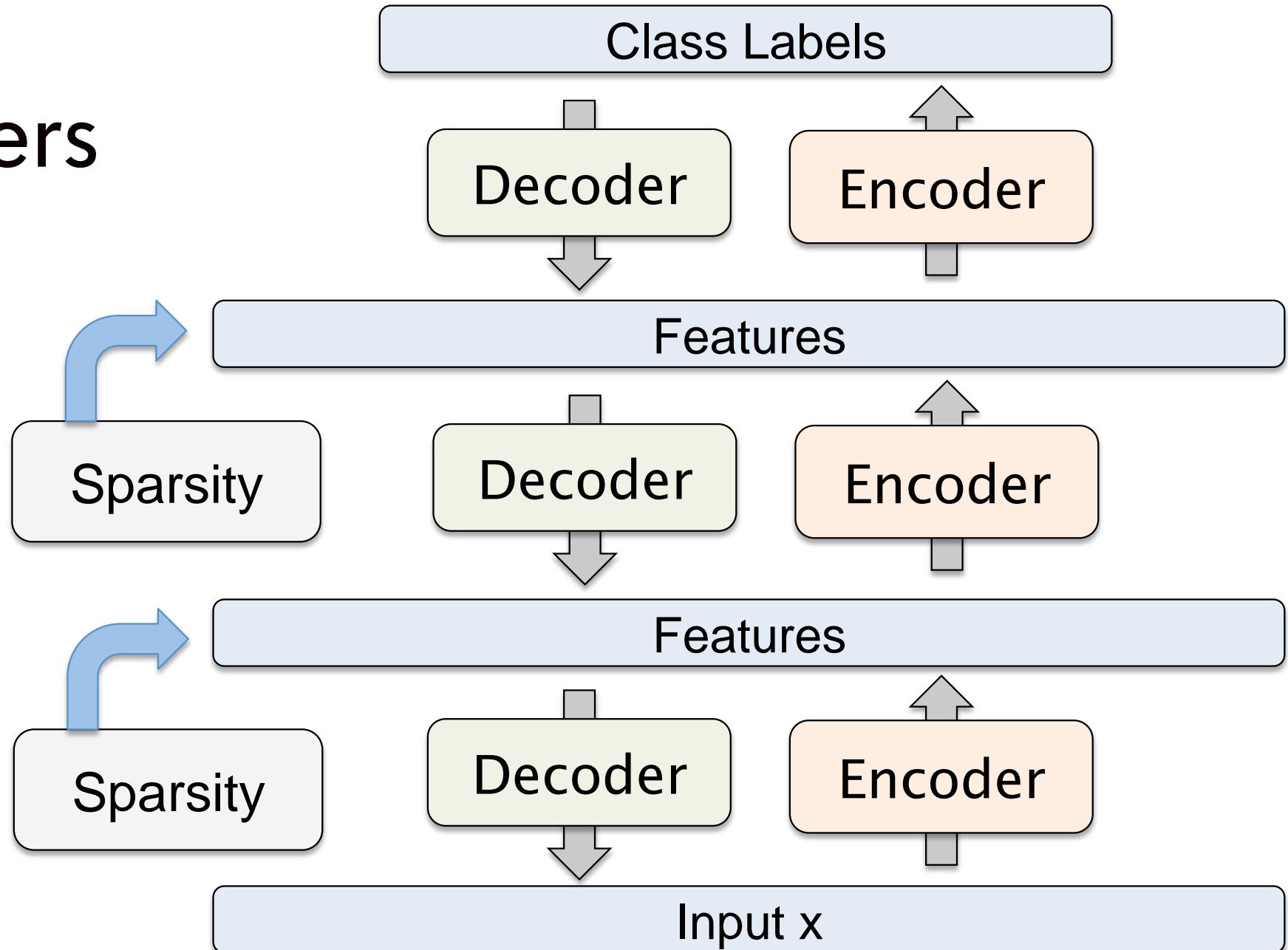
# Stacked Autoencoders



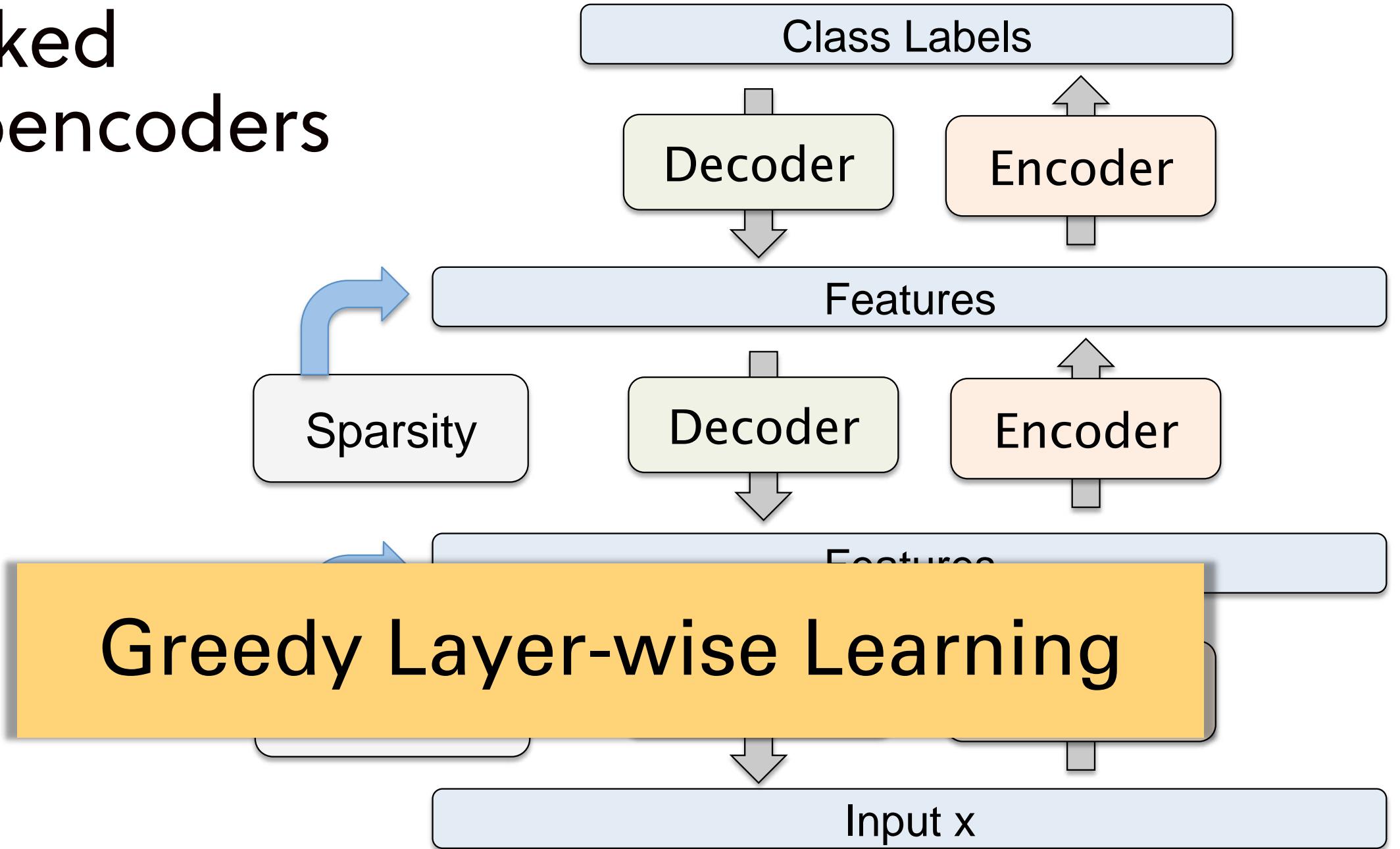
# Stacked Autoencoders



# Stacked Autoencoders

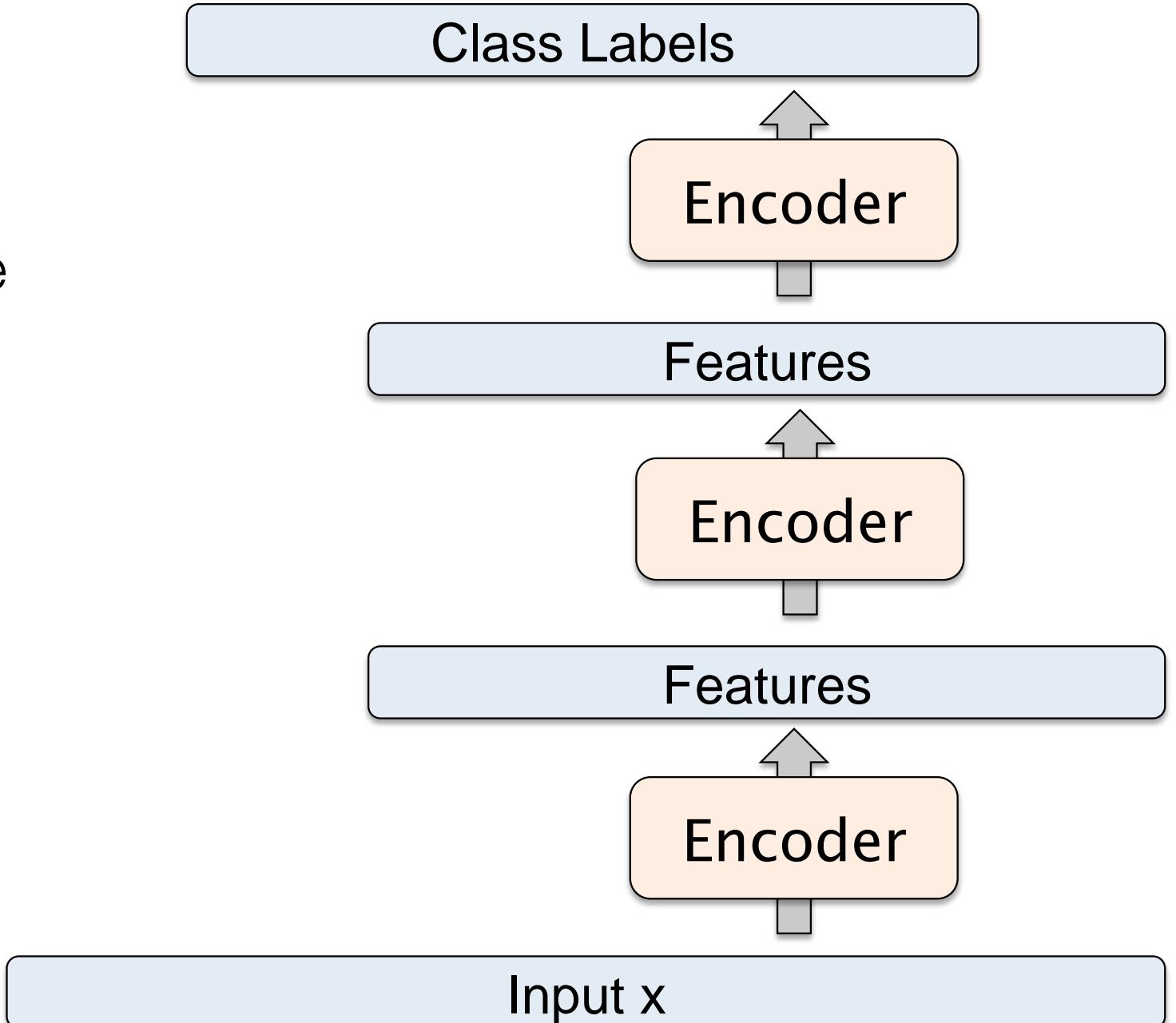


# Stacked Autoencoders



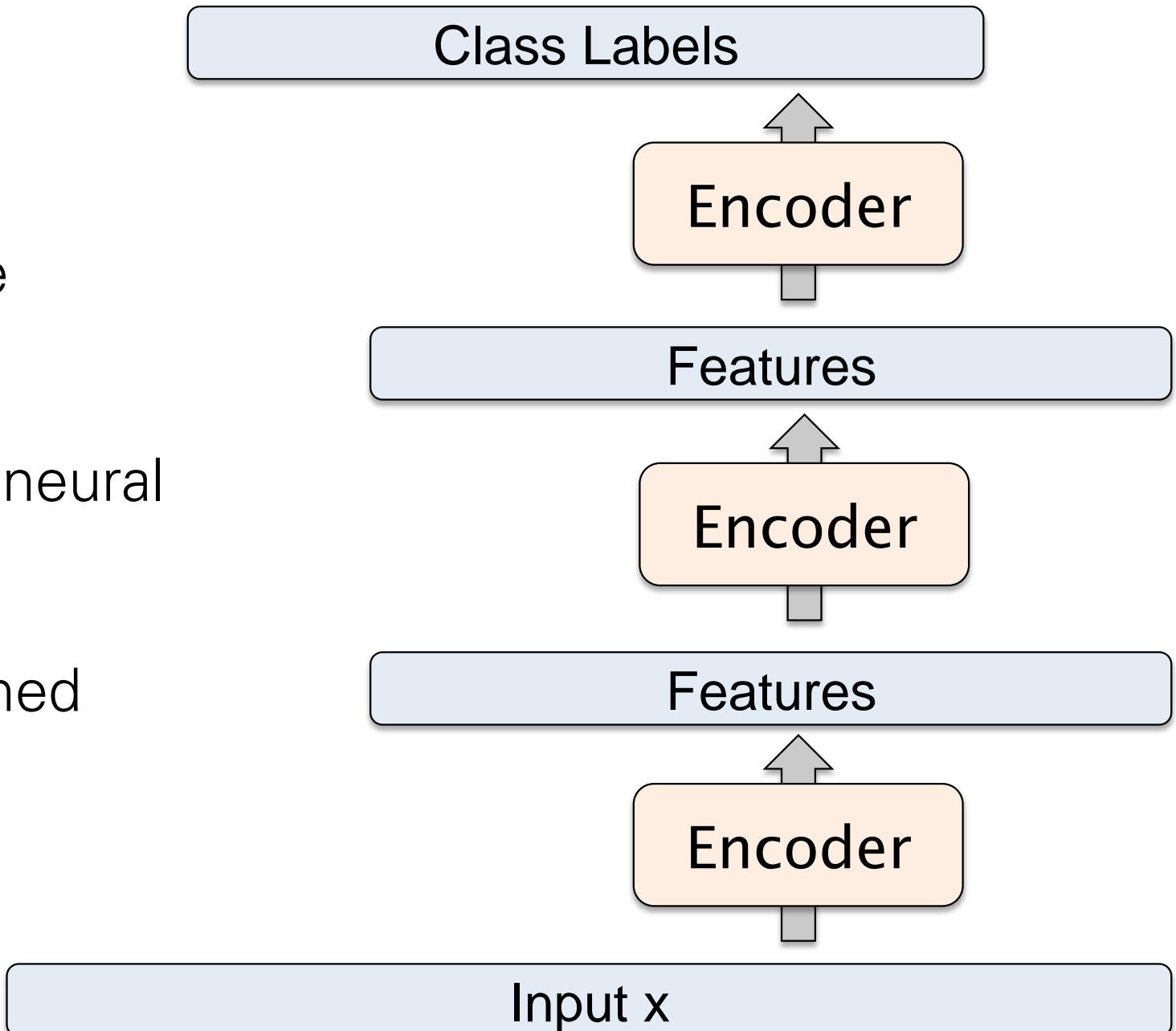
# Stacked Autoencoders

- Remove decoders and use feed-forward part.

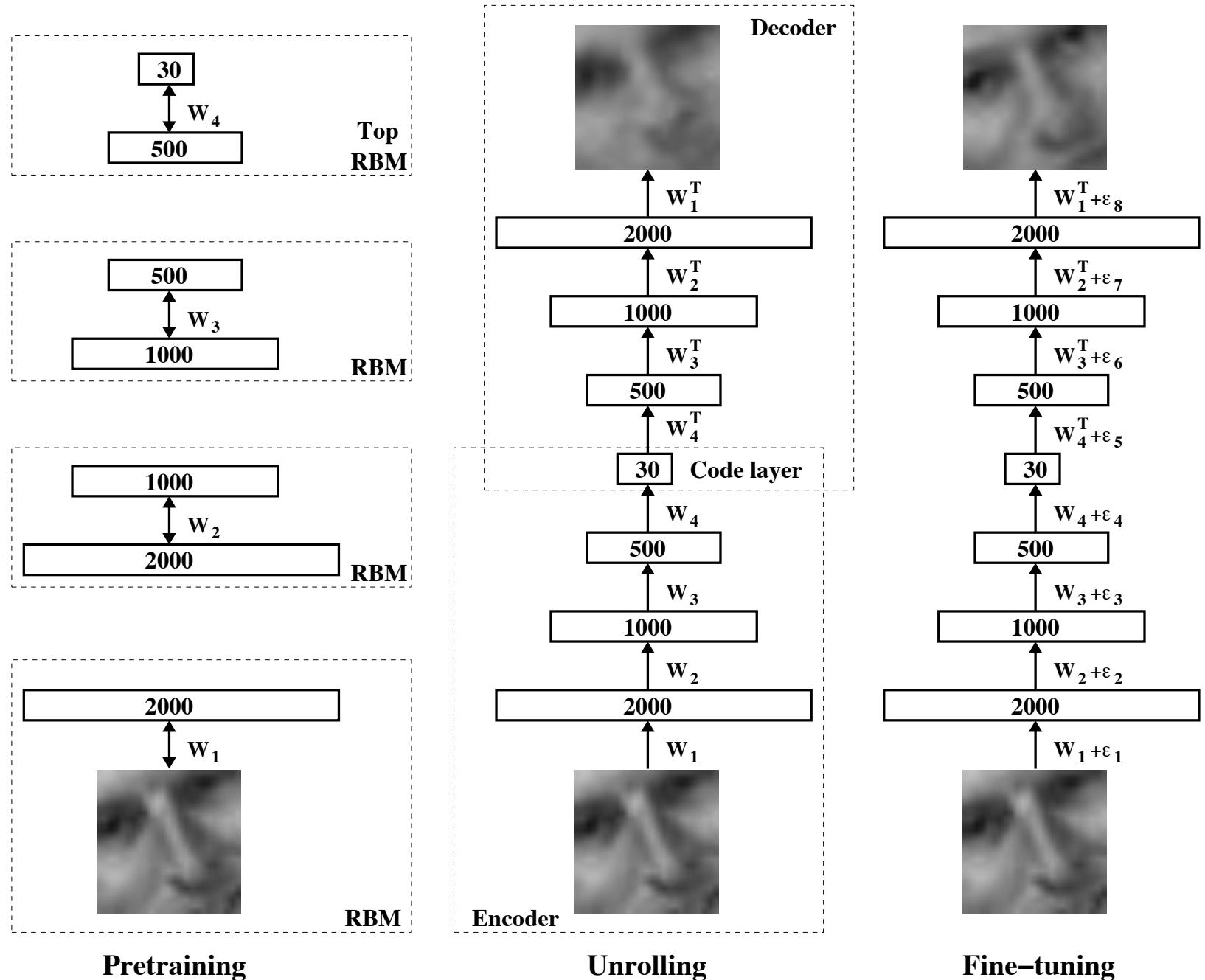


# Stacked Autoencoders

- Remove decoders and use feed-forward part.
- Standard, or convolutional neural network architecture.
- Parameters can be fine-tuned using backpropagation.

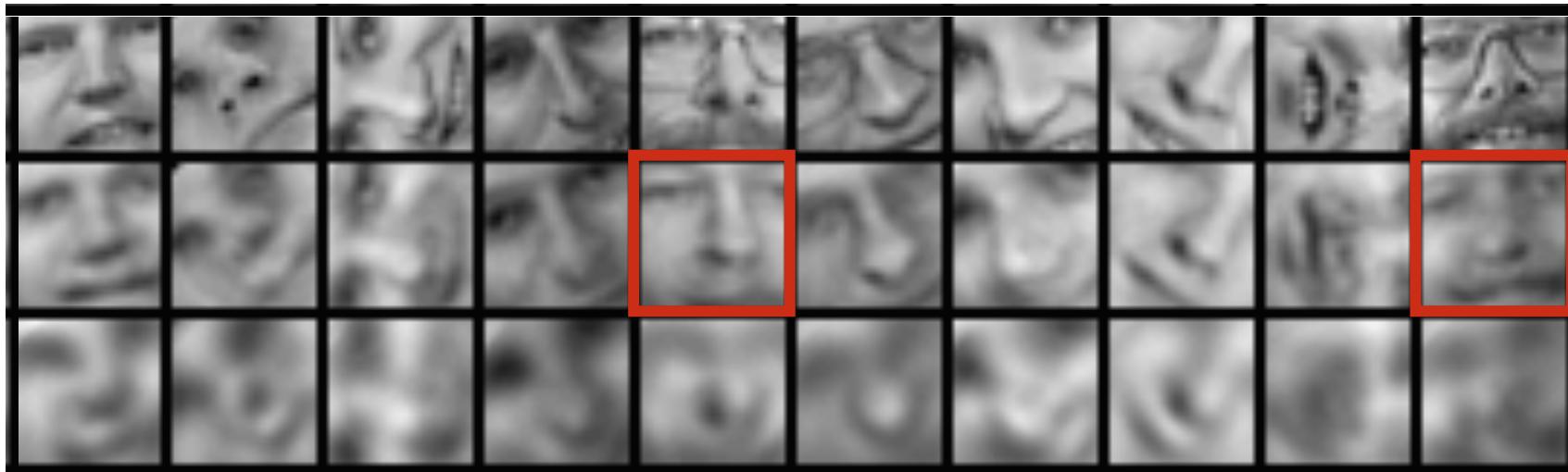


# Deep Autoencoder



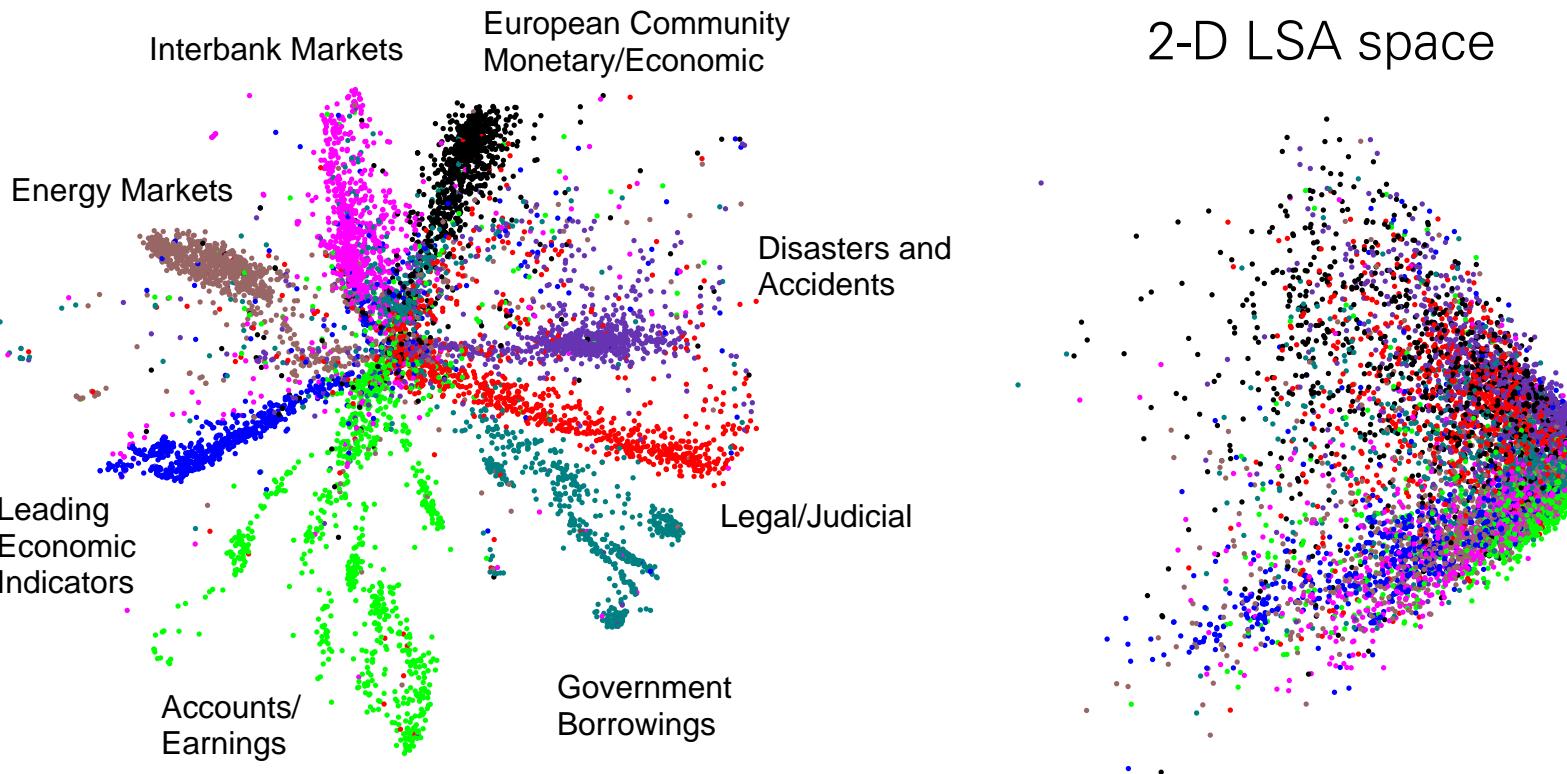
# Deep Autoencoders

- 25x25 – 2000 – 1000 – 500 – 30 autoencoder to extract 30-D real-valued codes for Oliver face patches.



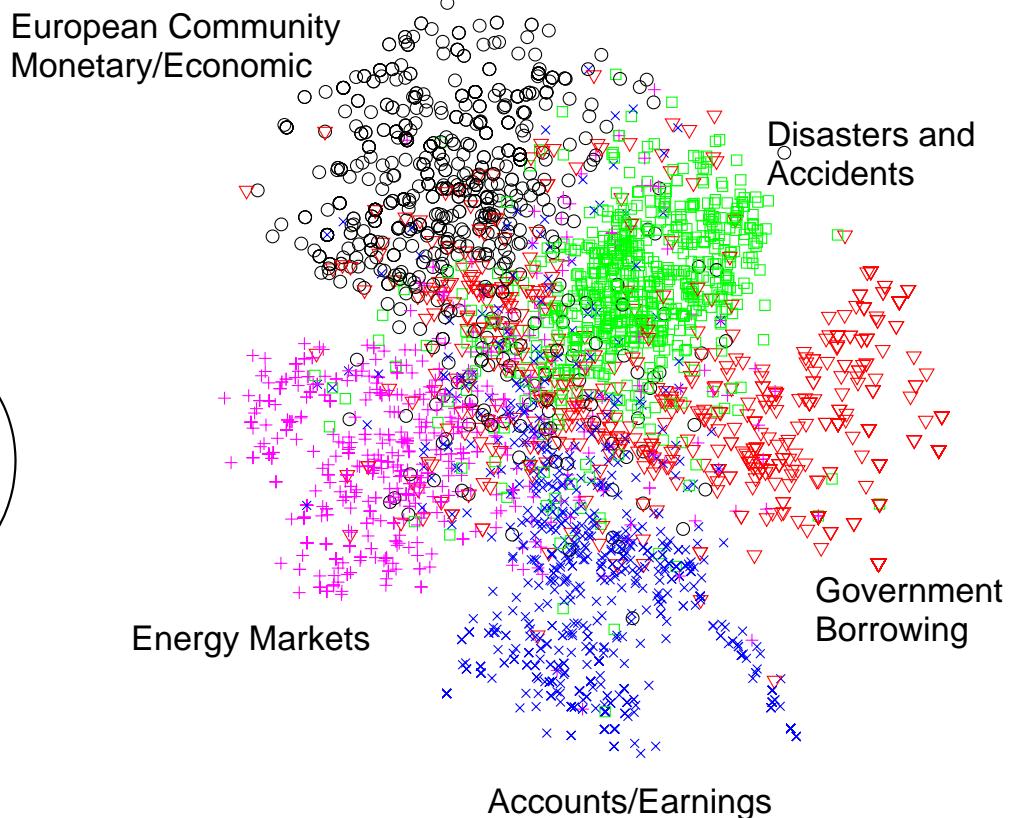
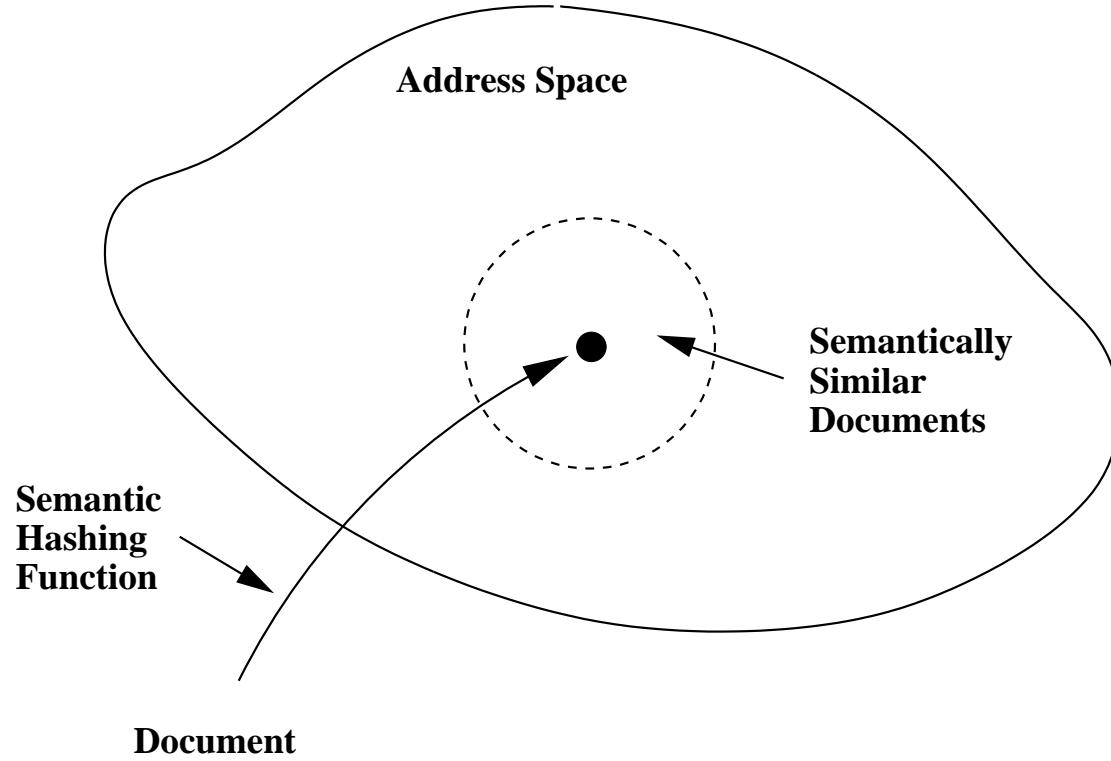
- **Top:** Random samples from the test dataset.
- **Middle:** Reconstructions by the 30-dimensional deep autoencoder.
- **Bottom:** Reconstructions by the 30-dimensional PCA.

# Information Retrieval



- The Reuters Corpus Volume II contains 804,414 newswire stories (randomly split into **402,207 training** and **402,207 test**).
- “Bag-of-words” representation: each article is represented as a vector containing the counts of the most frequently used 2000 words in the training set.

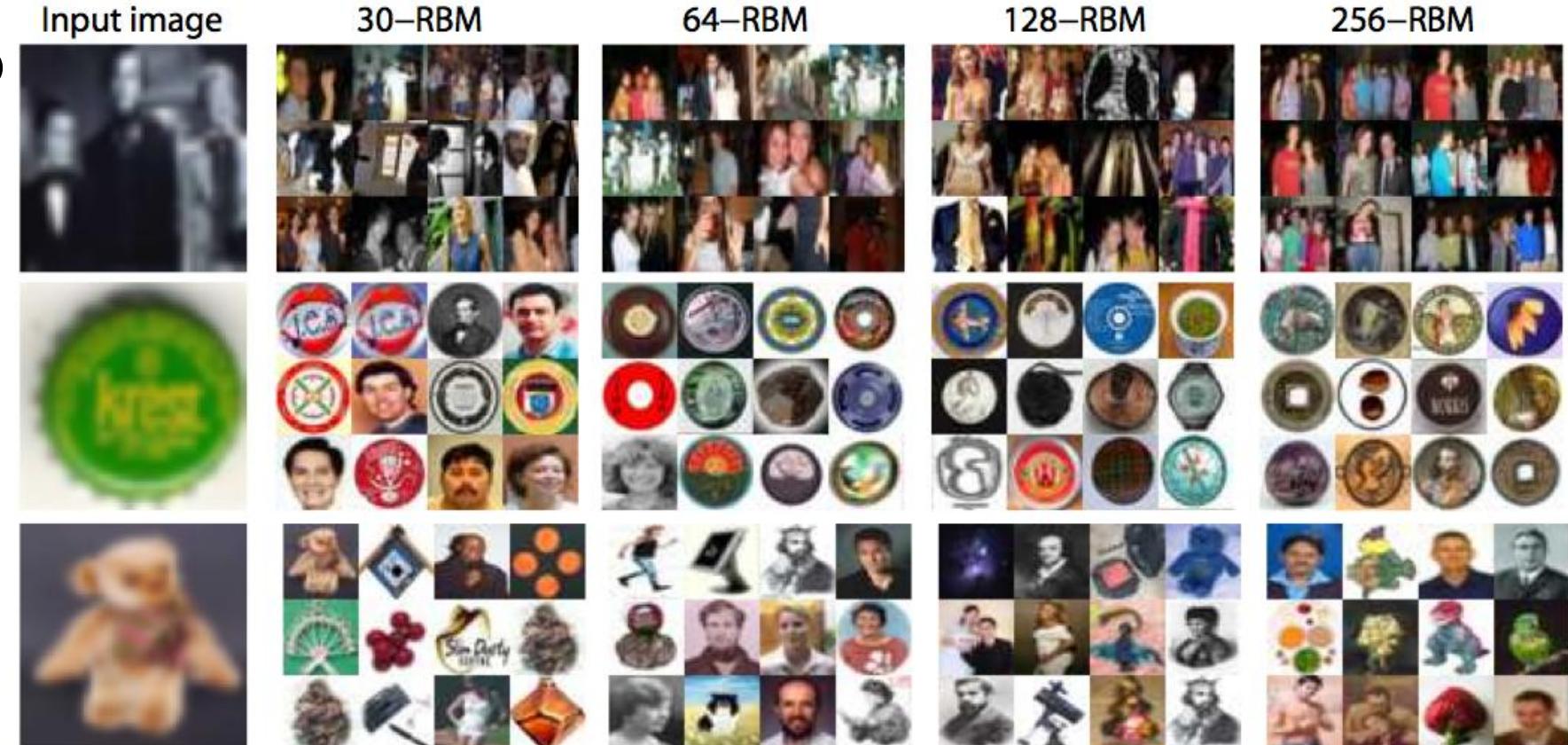
# Semantic Hashing



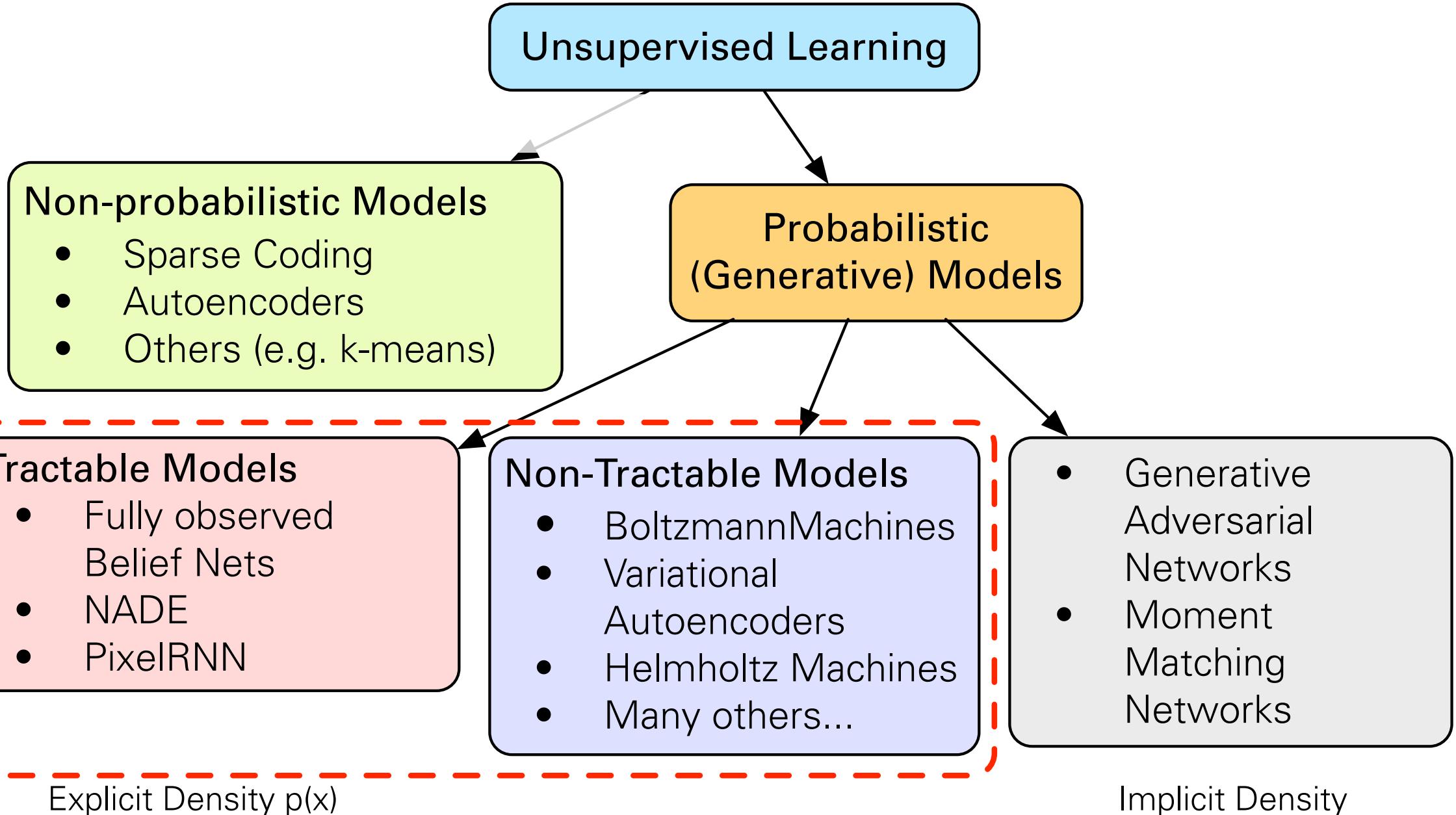
- Learn to map documents into **semantic 20-D binary codes**.
- Retrieve similar documents stored at the nearby addresses **with no search at all**.

# Searching Large Image Database using Binary Codes

- Map images into binary codes for fast retrieval.

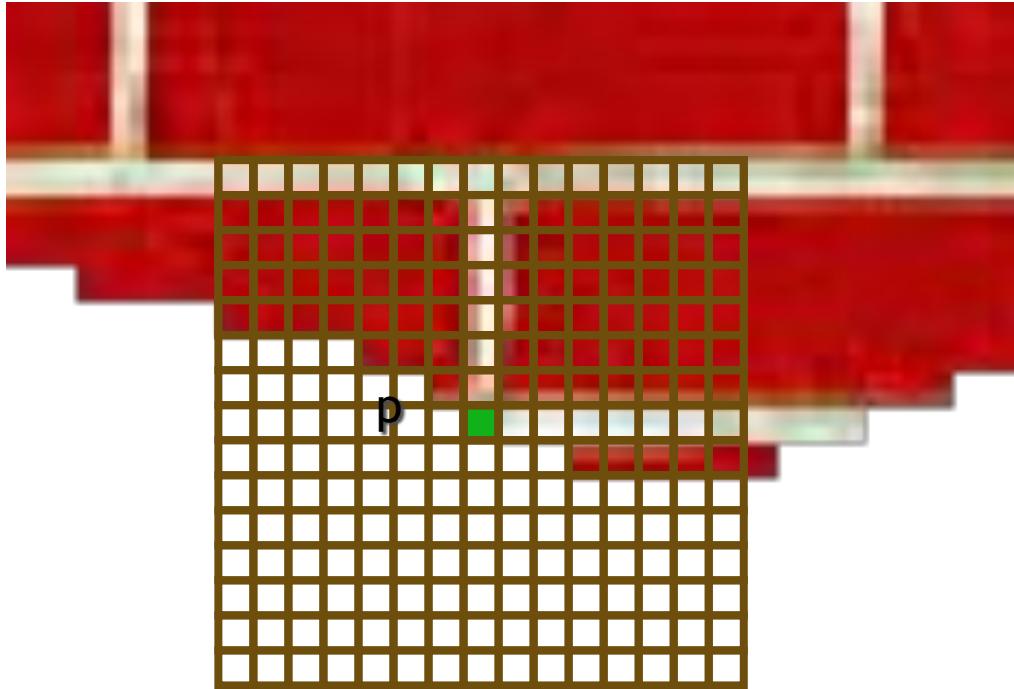


- Small Codes, Torralba, Fergus, Weiss, CVPR 2008
- Spectral Hashing, Y. Weiss, A. Torralba, R. Fergus, NIPS 2008
- Kulis and Darrell, NIPS 2009, Gong and Lazebnik, CVPR 2011
- Norouzi and Fleet, ICML 2011



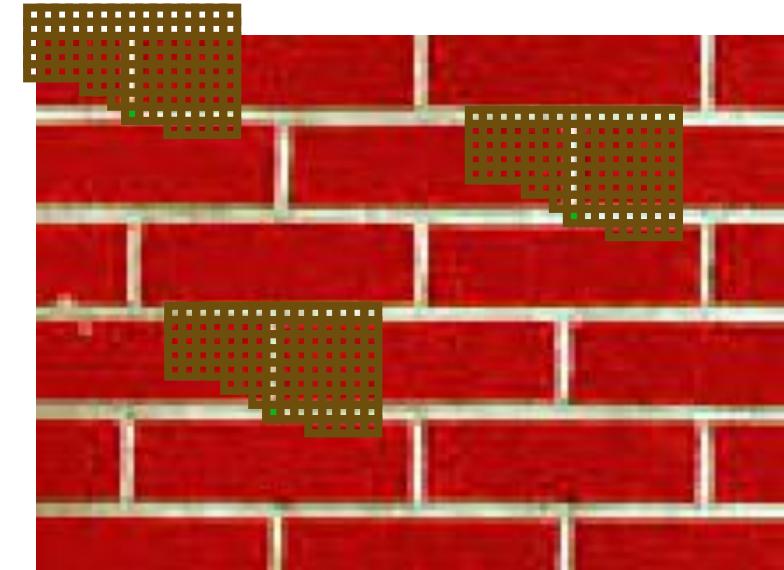
# Autoregressive Generative Models

# Texture synthesis by non-parametric sampling



Synthesizing a pixel

non-parametric  
sampling

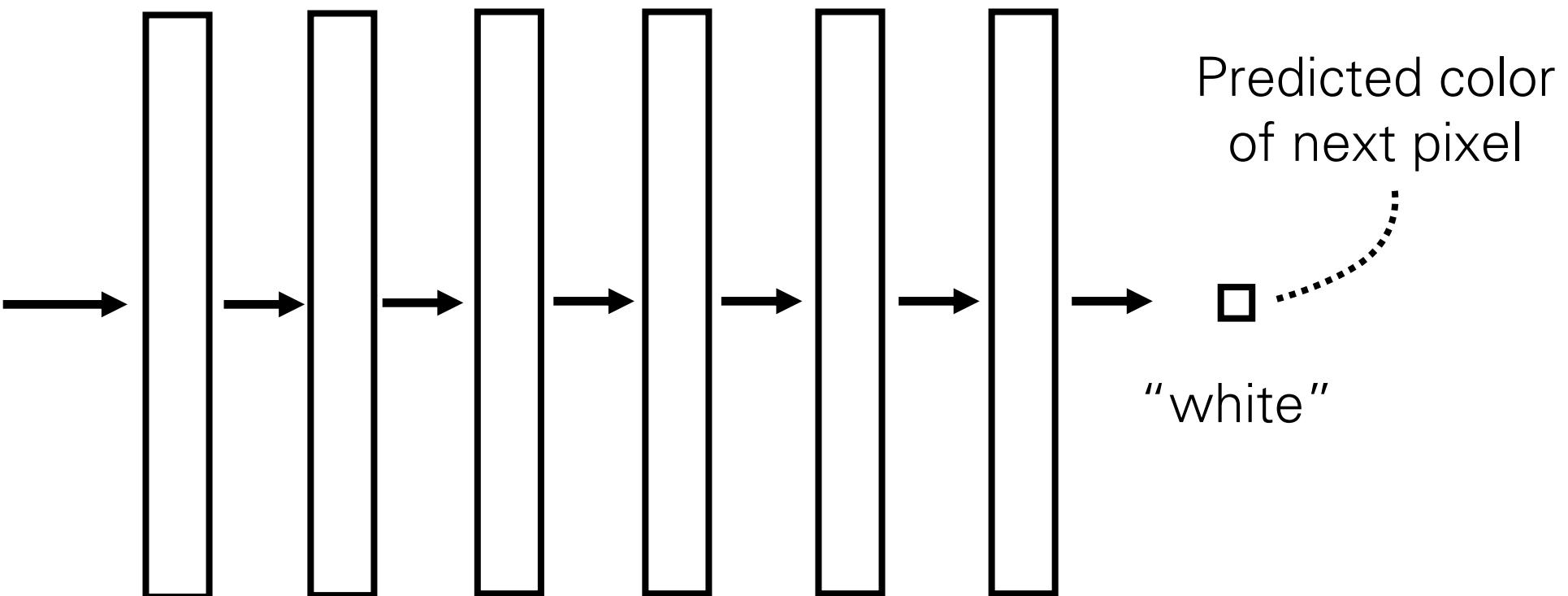
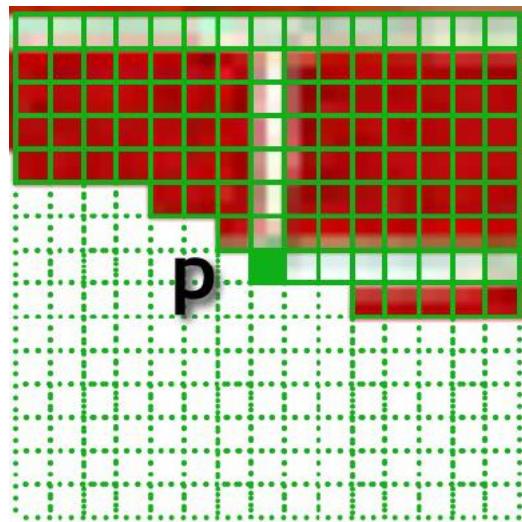


Input image

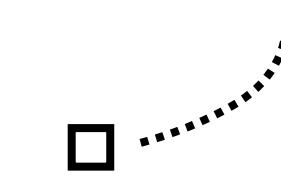
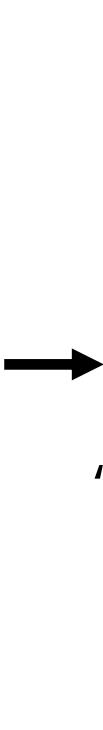
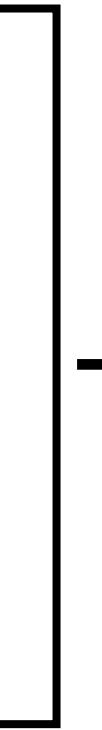
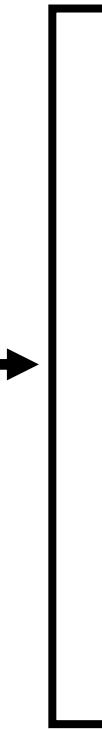
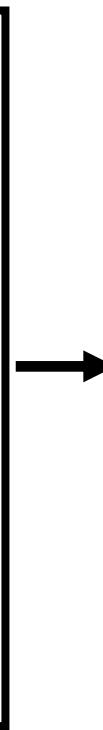
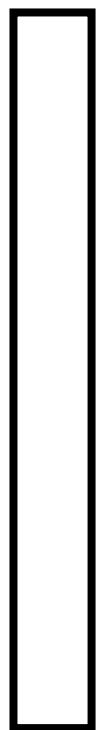
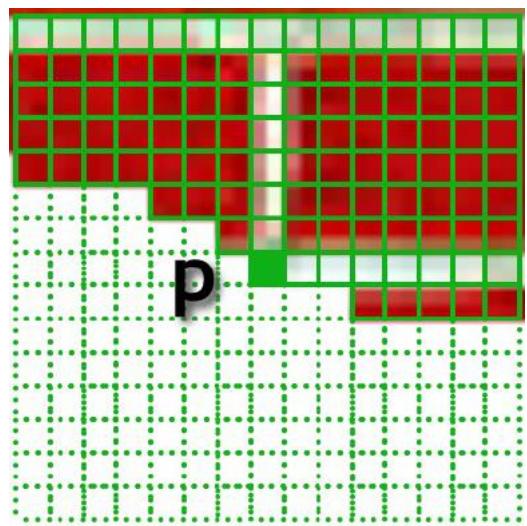
Models  $P(p|N(p))$

# Texture synthesis with a deep net

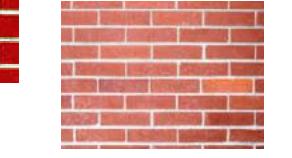
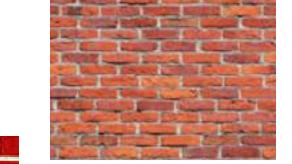
Input partial  
image



Input partial  
image



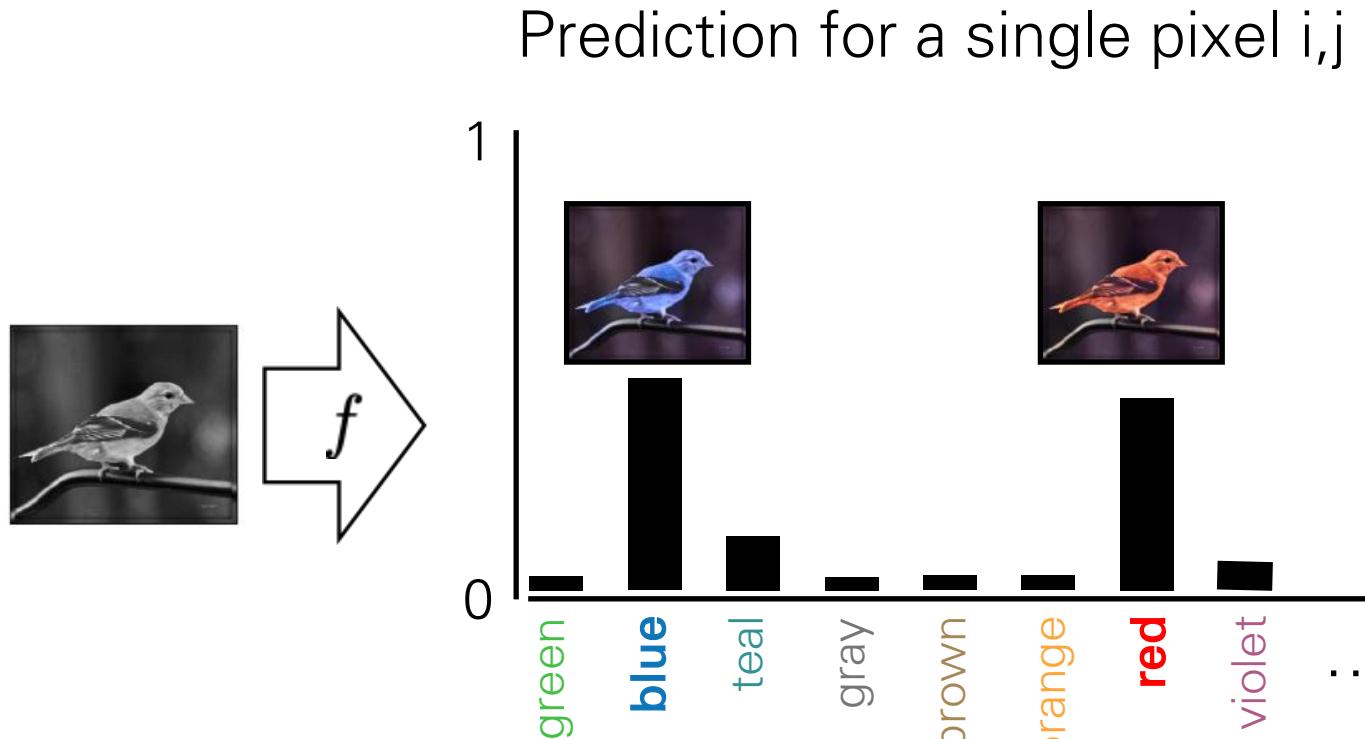
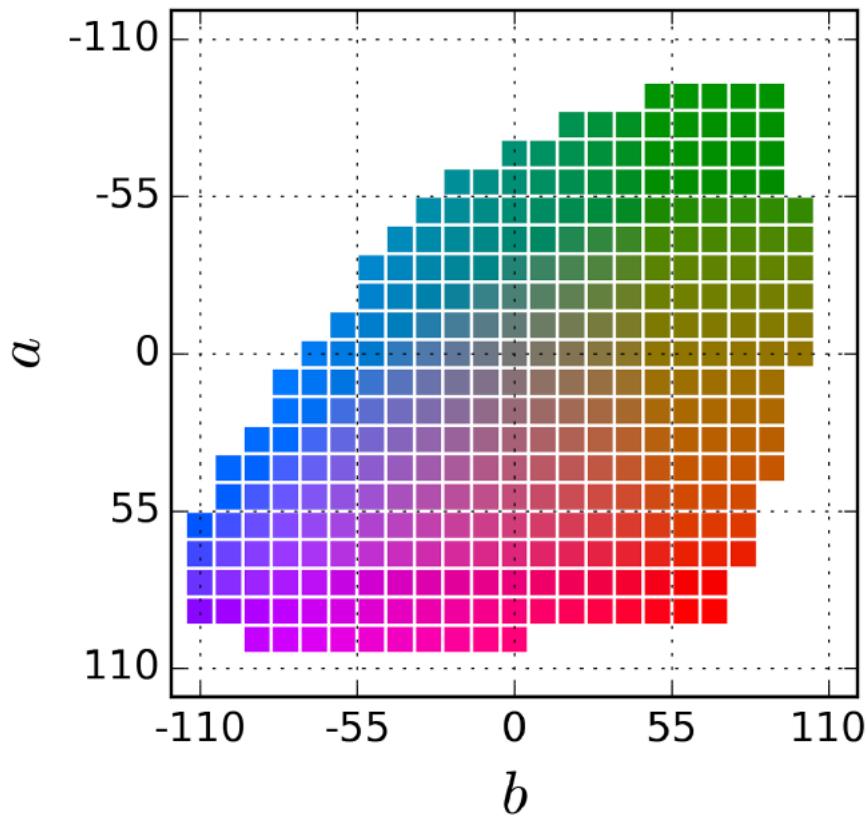
Predicted color  
of next pixel



...

# Idea: We can represent colors as discrete classes

$$\mathbf{y} \in \mathbb{R}^{H \times W \times K}$$



$$\mathcal{L}(\mathbf{y}, f_{\theta}(\mathbf{x})) = H(\mathbf{y}, \text{softmax}(f_{\theta}(\mathbf{x})))$$

And we can interpret the learner as modeling  $P(\text{next pixel} \mid \text{previous pixels})$ :

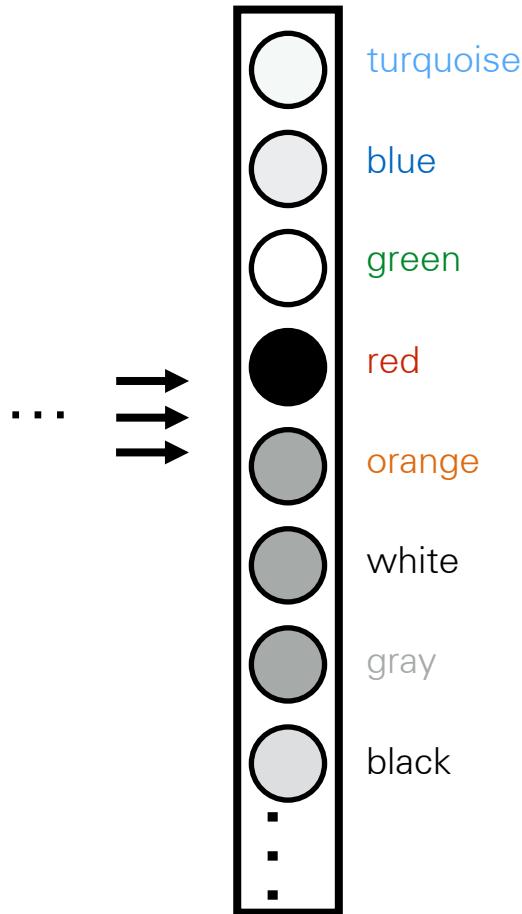
**Softmax regression** (a.k.a. multinomial logistic regression)

$$\hat{\mathbf{y}} \equiv [P_{\theta}(Y = 1|X = \mathbf{x}), \dots, P_{\theta}(Y = K|X = \mathbf{x})] \quad \leftarrow \text{predicted probability of each class given input } \mathbf{x}$$

$$H(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_{k=1}^K y_k \log \hat{y}_k \quad \leftarrow \text{picks out the -log likelihood of the ground truth class } \mathbf{y \text{ under the model prediction } \hat{\mathbf{y}}}$$

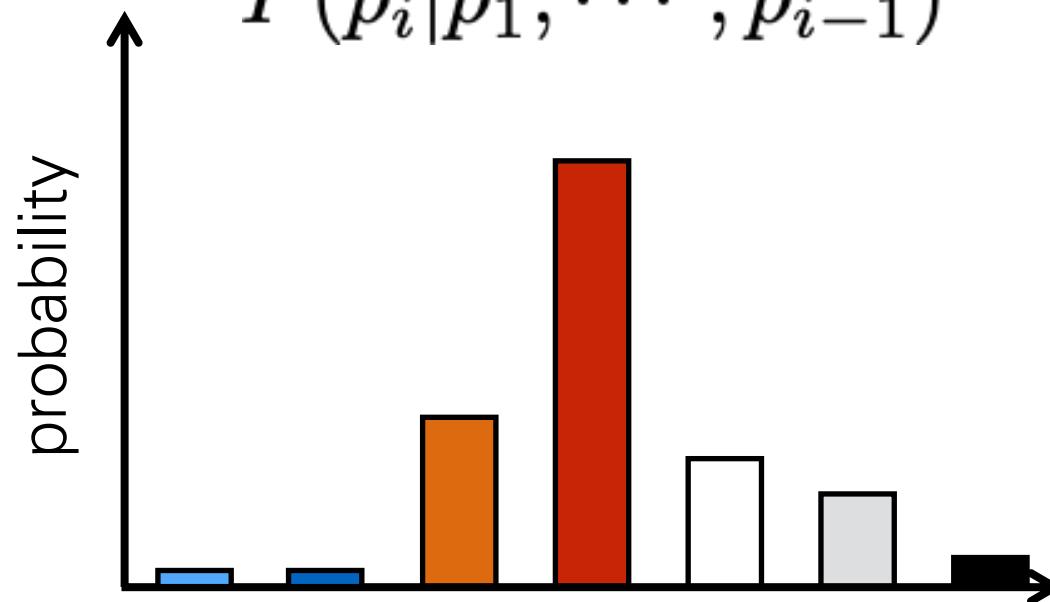
$$f^* = \arg \min_{f \in \mathcal{F}} \sum_{i=1}^N H(\mathbf{y}_i, \hat{\mathbf{y}}_i) \quad \leftarrow \text{max likelihood learner!}$$

## Network output

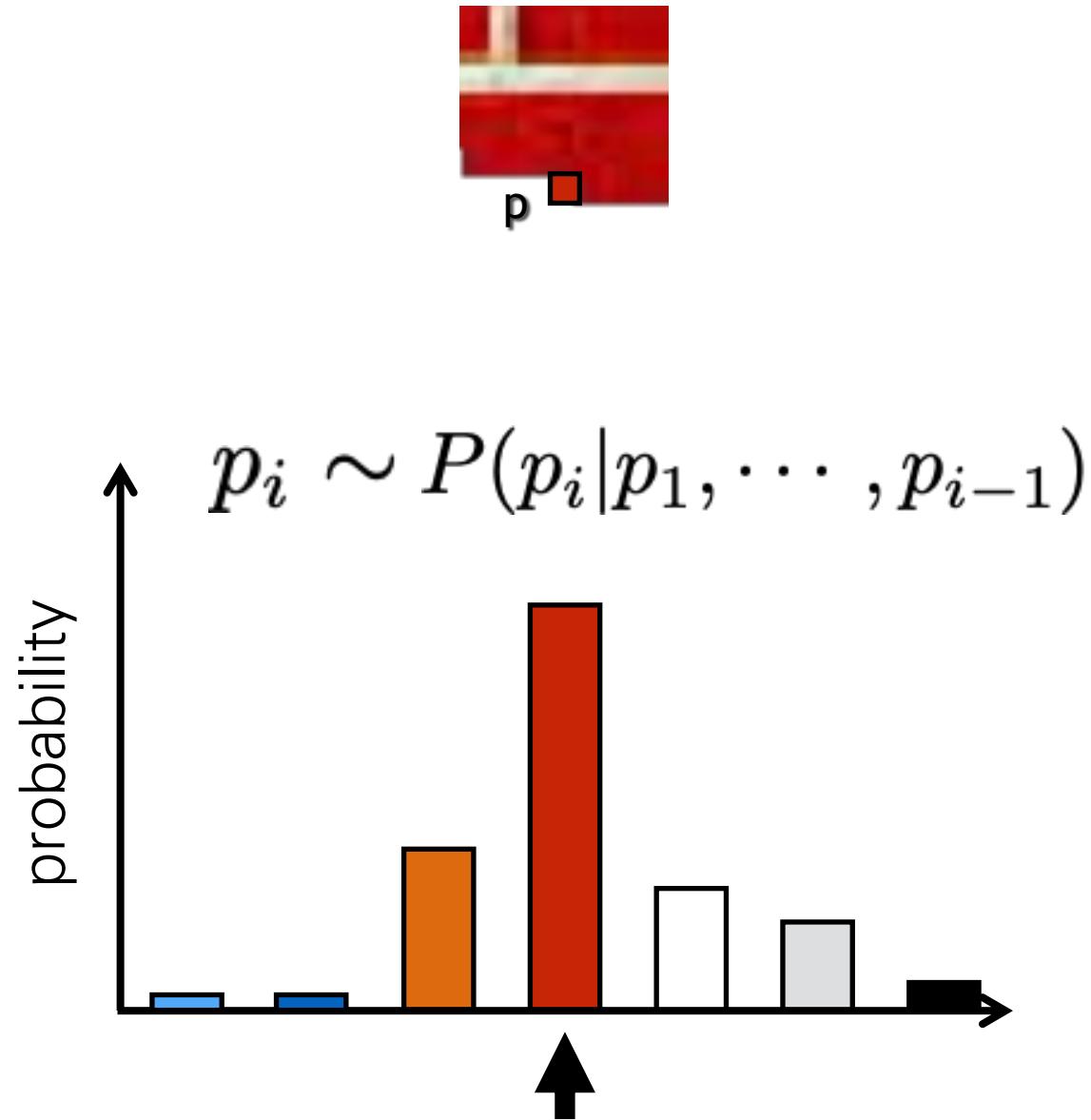
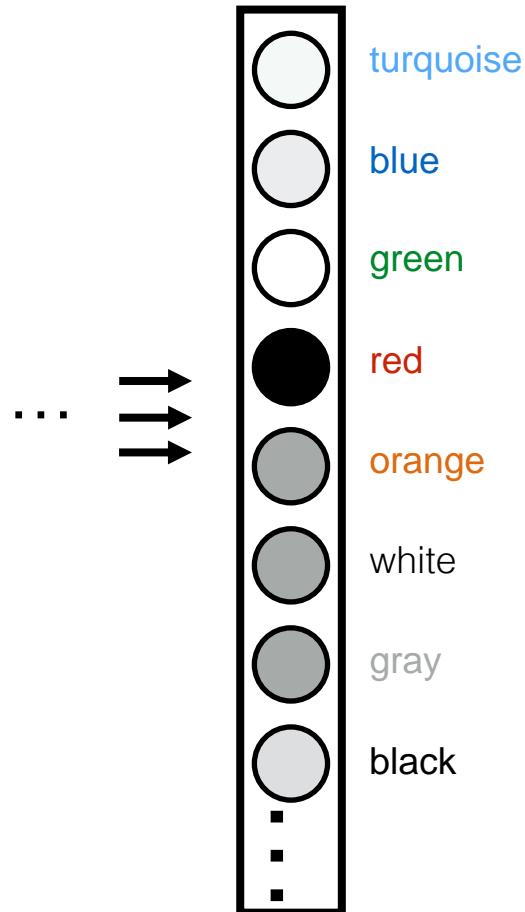


$P(\text{next pixel} \mid \text{previous pixels})$

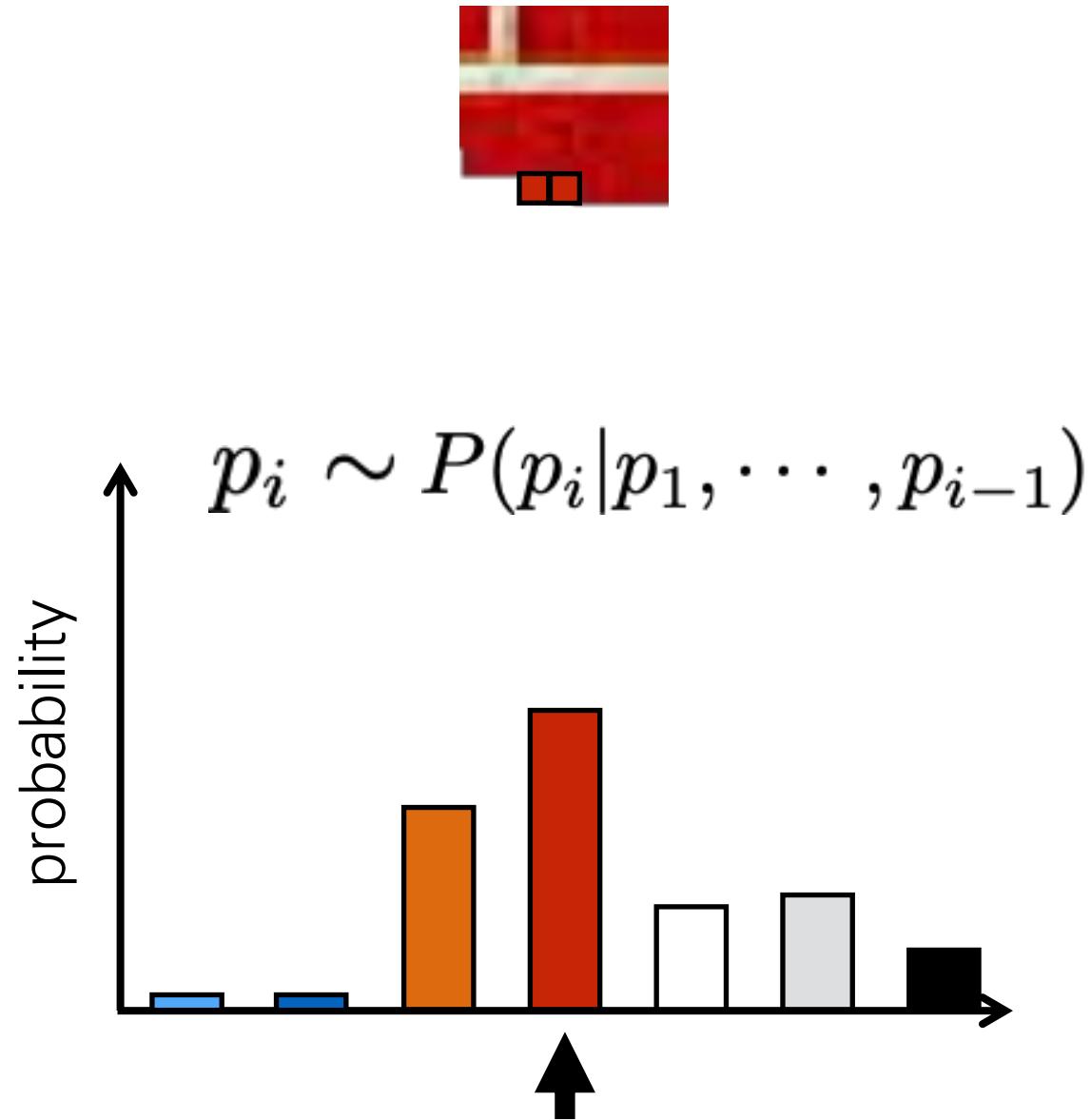
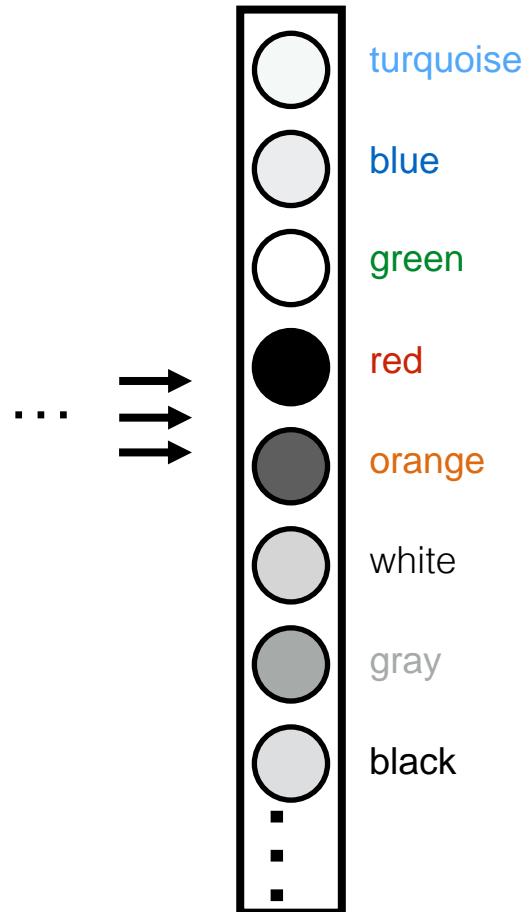
$$P(p_i | p_1, \dots, p_{i-1})$$



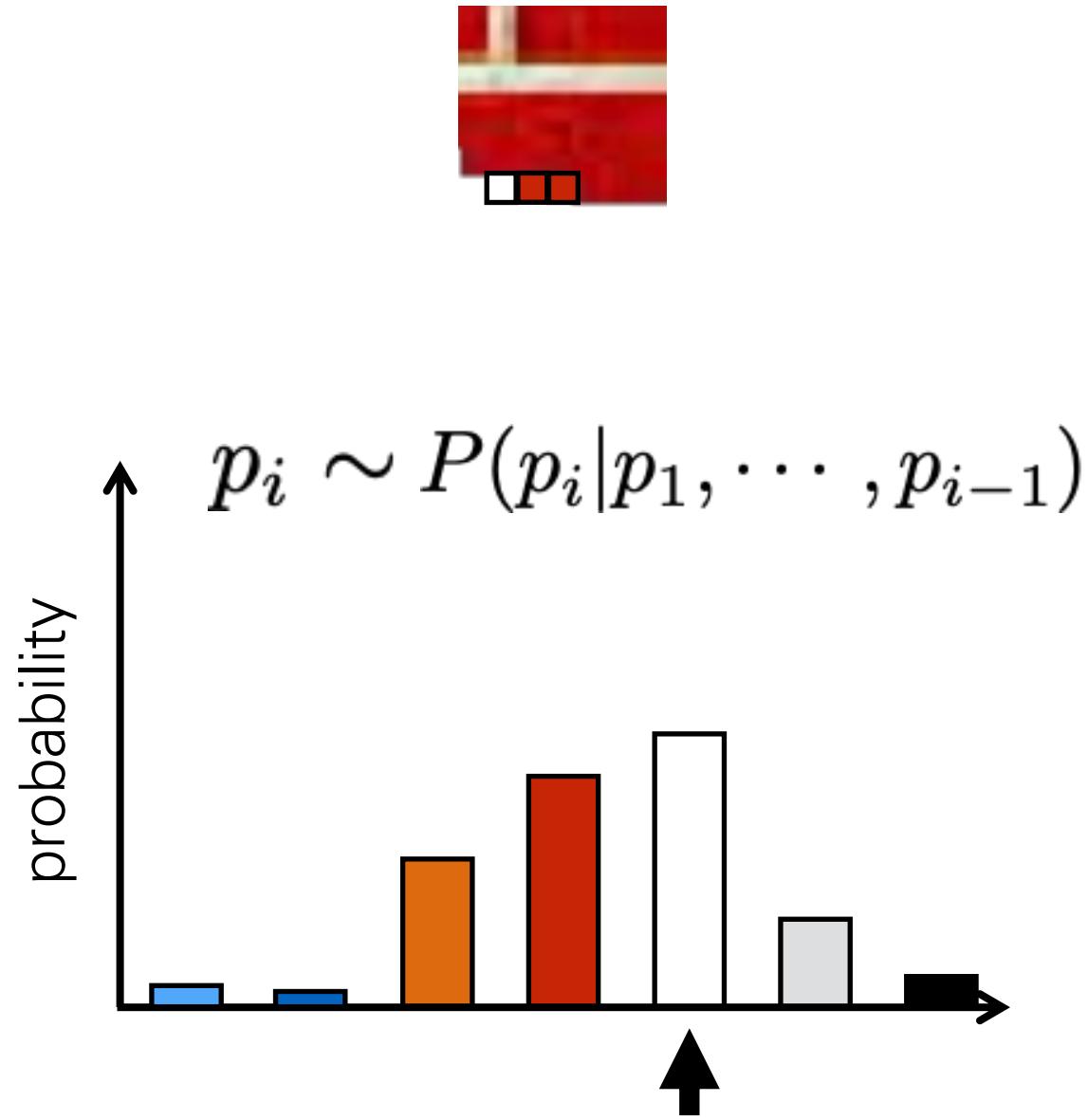
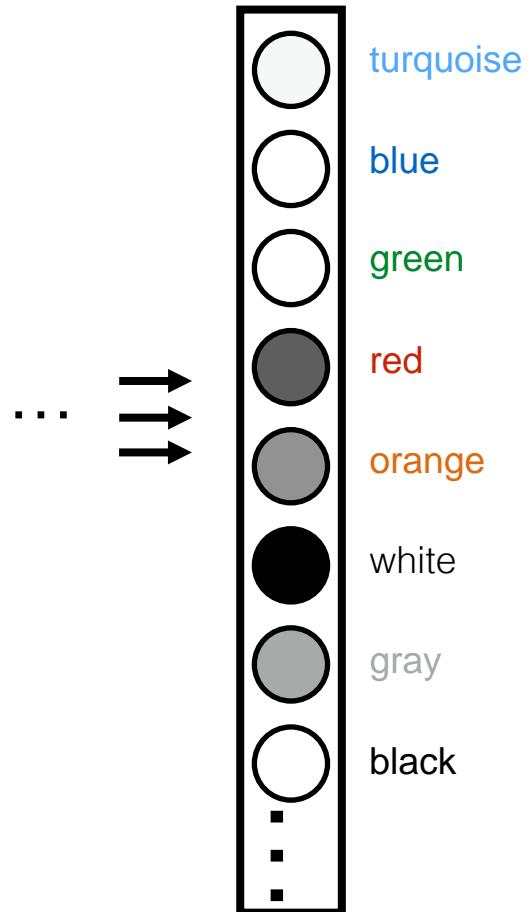
## Network output



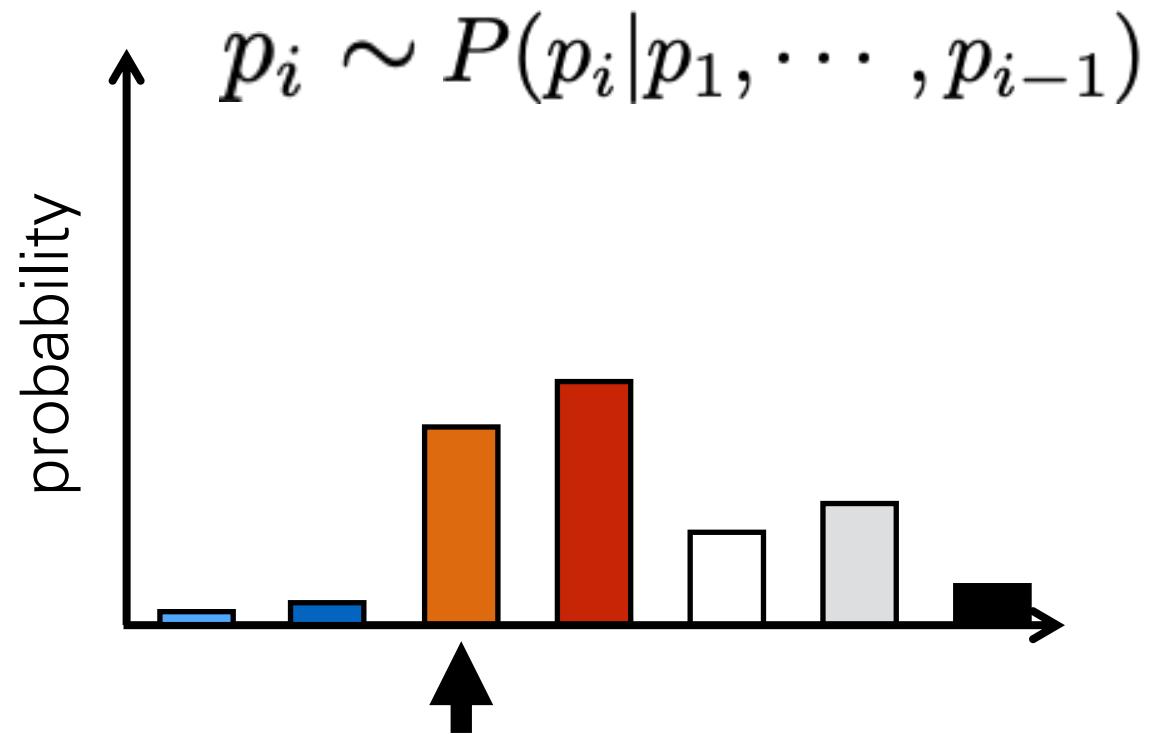
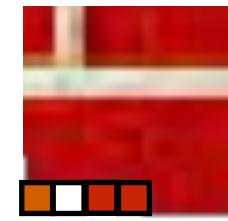
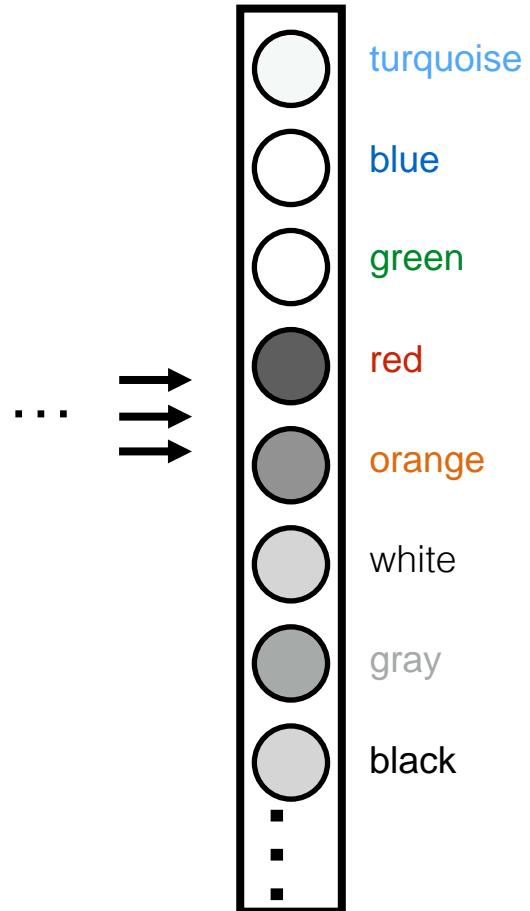
## Network output



## Network output



## Network output



$$p_1 \sim P(p_1)$$

$$p_2 \sim P(p_2|p_1)$$

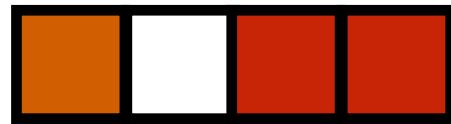
$$p_3 \sim P(p_3|p_1, p_2)$$

$$p_4 \sim P(p_4|p_1, p_2, p_3)$$

$$\{p_1, p_2, p_3, p_4\} \sim P(p_4|p_1, p_2, p_3)P(p_3|p_1, p_2)P(p_2|p_1)P(p_1)$$

$$p_i \sim P(p_i|p_1, \dots, p_{i-1})$$

$$p_3 \ p_4 \ p_2 \ p_1$$



$$\boxed{\mathbf{p} \sim \prod_{i=1}^N P(p_i|p_1, \dots, p_{i-1})}$$

# Autoregressive probability model

$$\mathbf{p} \sim \prod_{i=1}^N P(p_i | p_1, \dots, p_{i-1})$$

$$P(\mathbf{p}) = \prod_{i=1}^N P(p_i | p_1, \dots, p_{i-1}) \quad \leftarrow \text{General product rule}$$

The sampling procedure we defined above takes exact samples from the learned probability distribution (pmf).

Multiplying all conditionals evaluates the probability of a full joint configuration of pixels.

# Learning the Distribution of Natural Data

$$p(\mathbf{x}) = \prod_i p(x_i | \mathbf{x}_{<})$$

1D sequences such as text or sound

$$p(\mathbf{x}) = \prod_j \prod_i p(x_{i,j} | \mathbf{x}_{<})$$

2D tensors such as images

$$p(\mathbf{x}) = \prod_k \prod_j \prod_i p(x_{i,j,k} | \mathbf{x}_{<})$$

3D tensors such as videos

- Fully visible belief networks [Frey et al., 1996] [Frey, 1998]
- NADE/MADE [Larochelle and Murray, 2011] [Germain et al., 2015]
- PixelRNN/PixelCNN (Images) [van den Oord, Kalchbrenner, Kavukcuoglu, 2016]  
[van den Oord, Kalchbrenner, Vinyals, et al., 2016]
- Video Pixel Nets (Videos) [Kalchbrenner, van den Oord, Simonyan, et al., 2016]
- ByteNet (Language/seq2seq) [Kalchbrenner, Espeholt, Simonyan, et al., 2016]
- WaveNet (Audio) [van den Oord, Dieleman, Zen, et al., 2016]

# PixelCNN

$P($



)

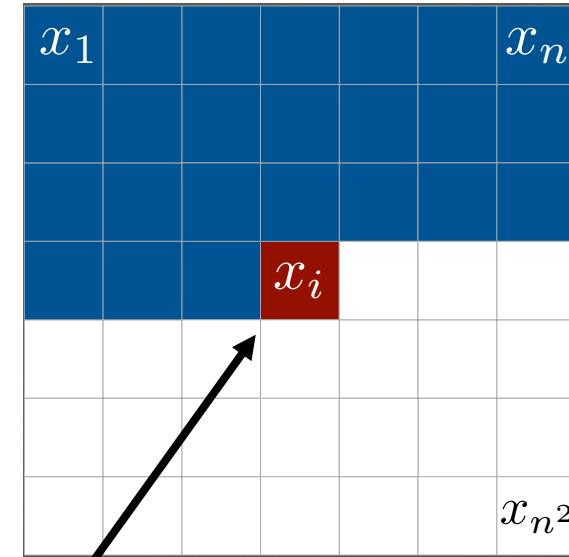
- approach the generation process as sequence modeling problem
- an explicit density model

# PixelCNN

$P($



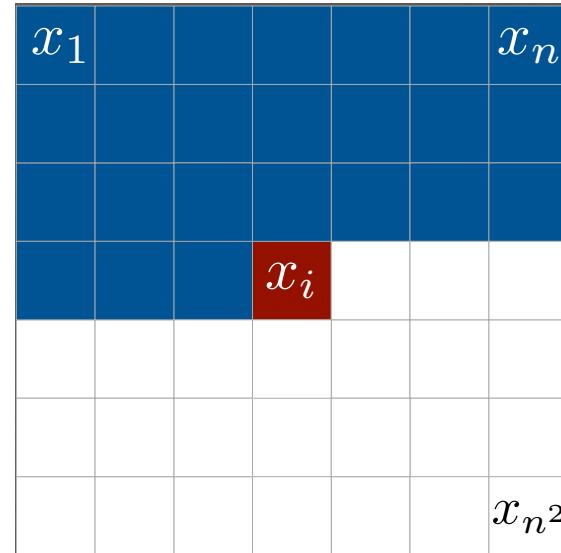
)



$$p(x_i | \mathbf{x}_{<})$$

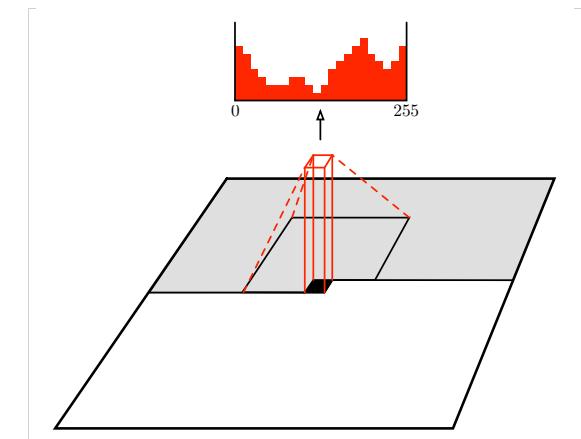
# PixelCNN

$P($  )

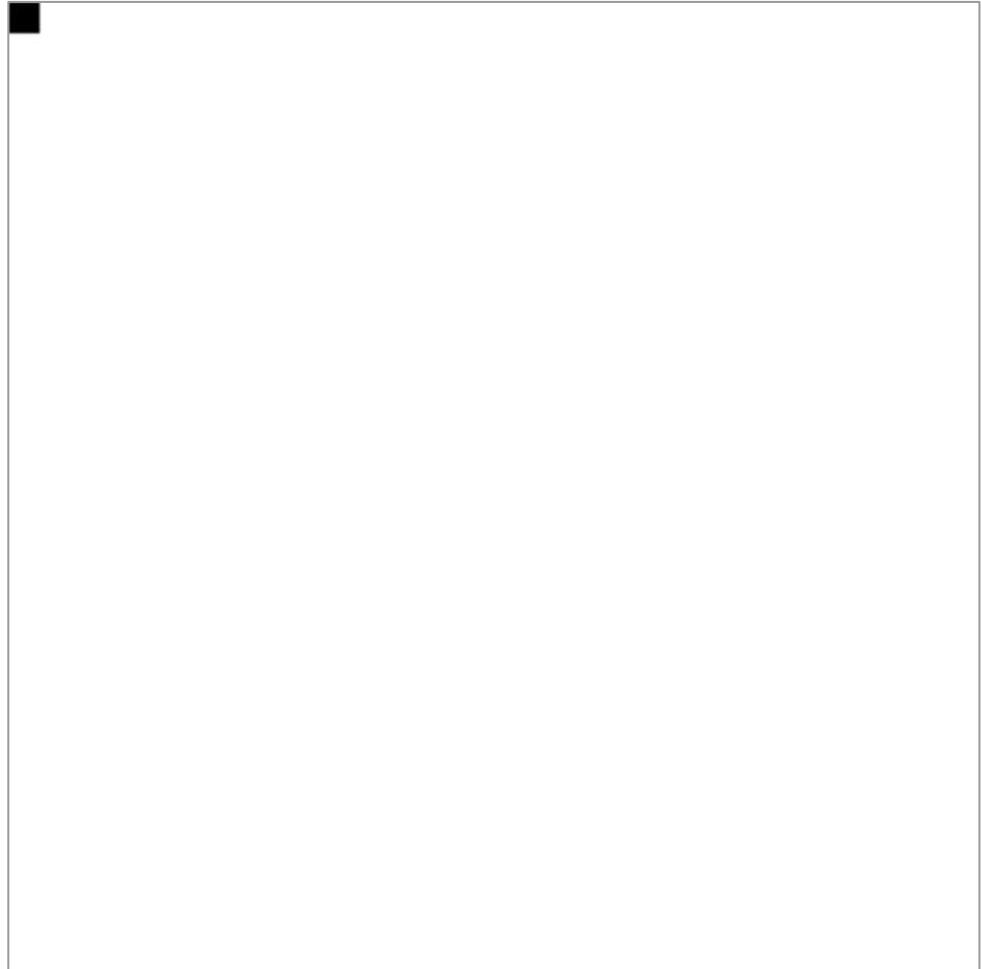


By chain rule and using **pixels** as variables,

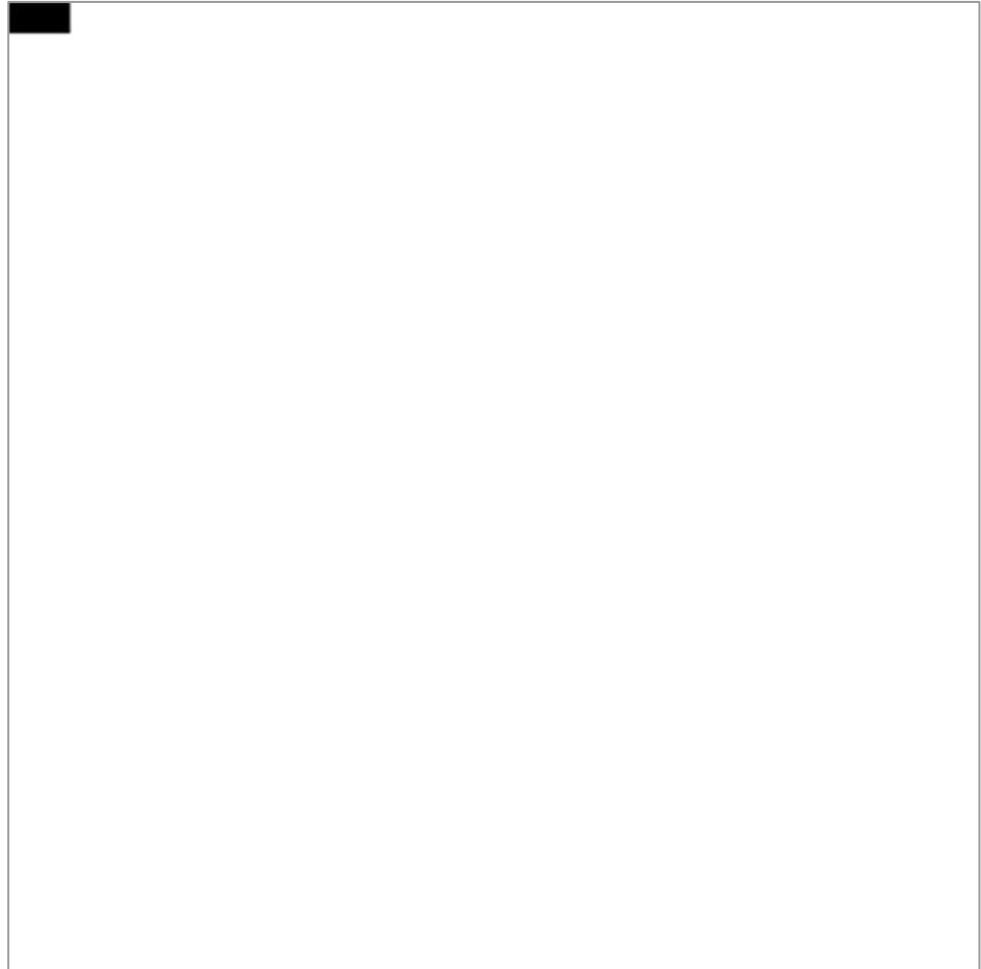
$$P(X) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)$$



# PixelCNN



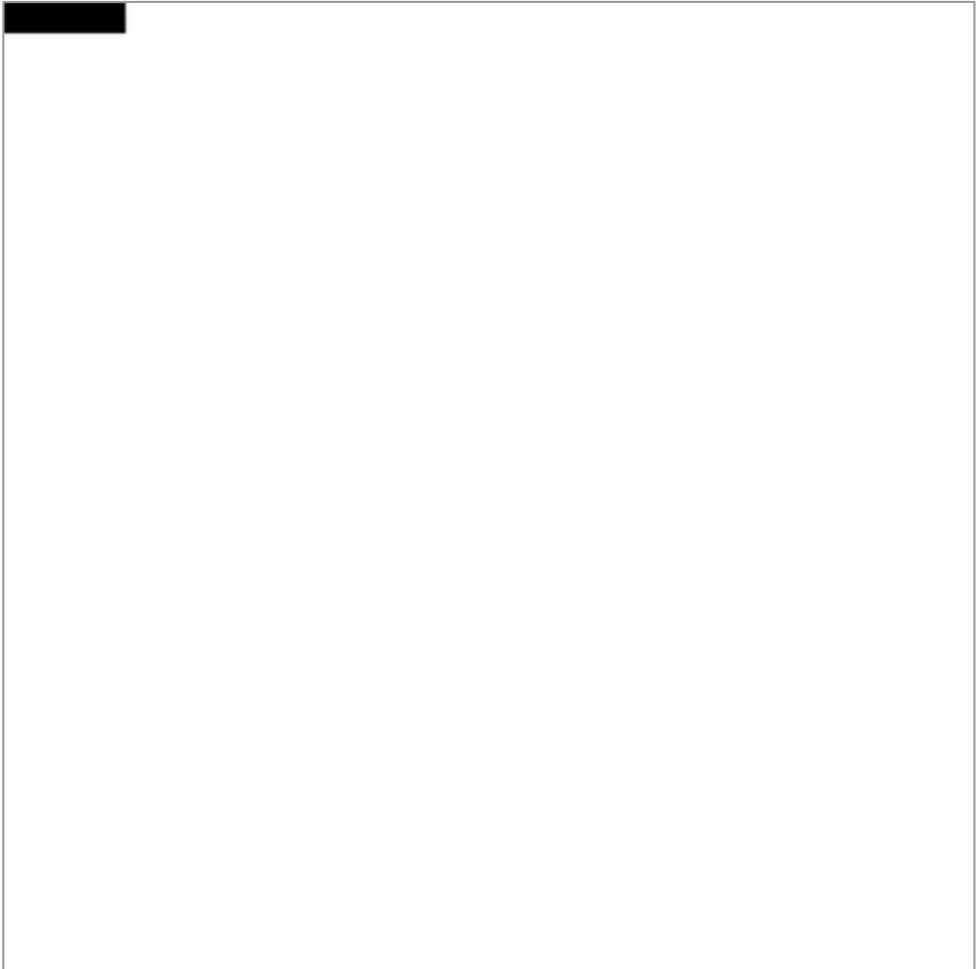
# PixelCNN



# PixelCNN



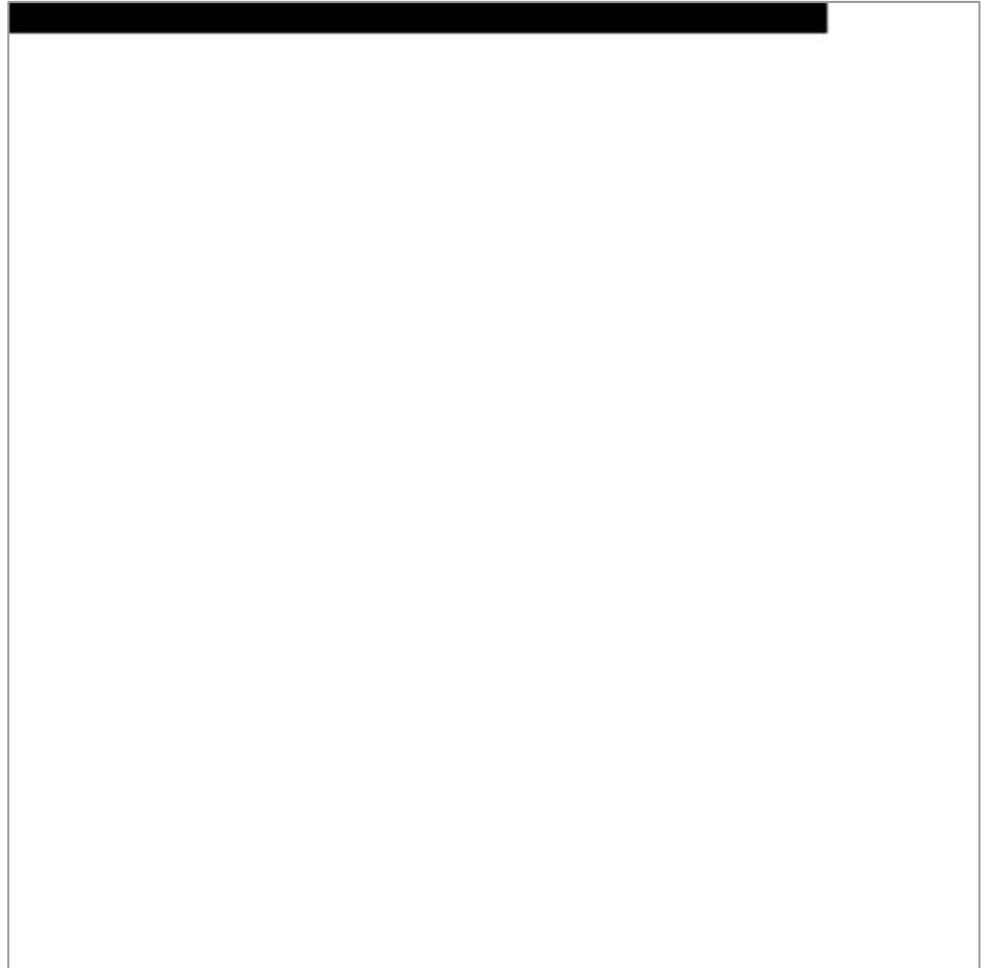
# PixelCNN



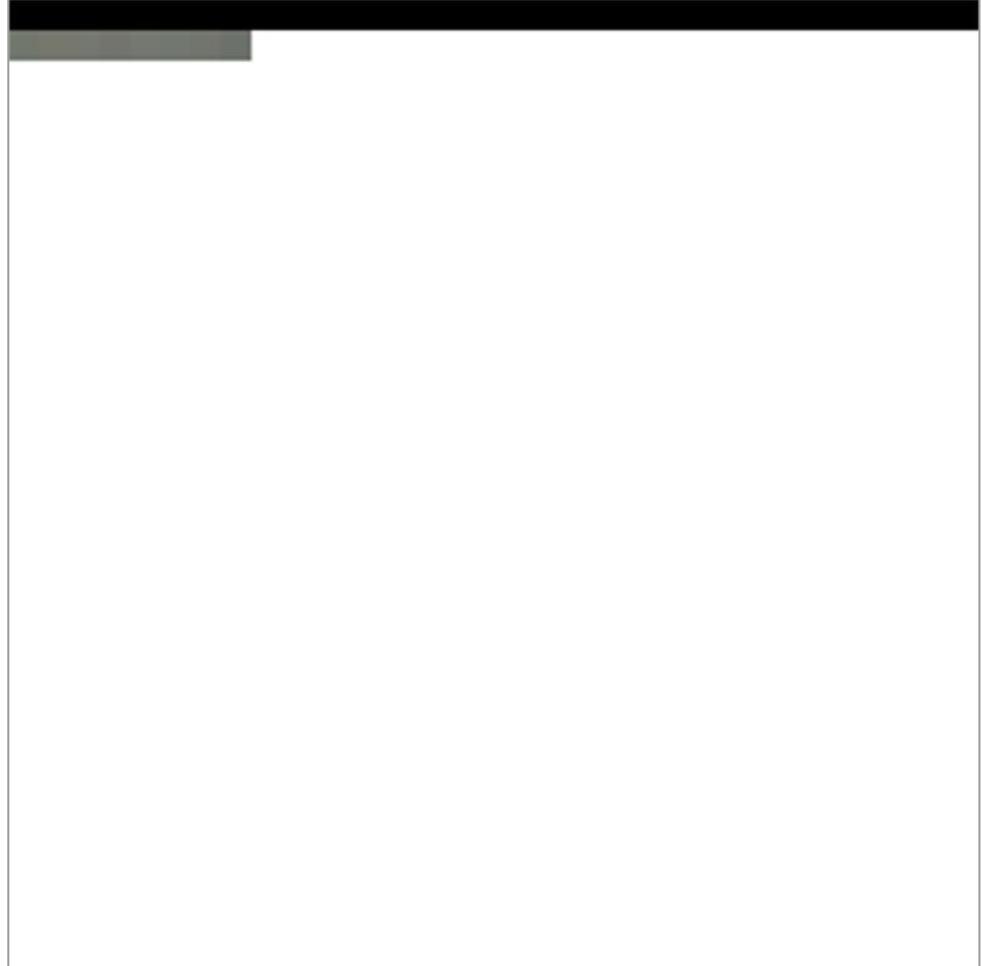
# PixelCNN



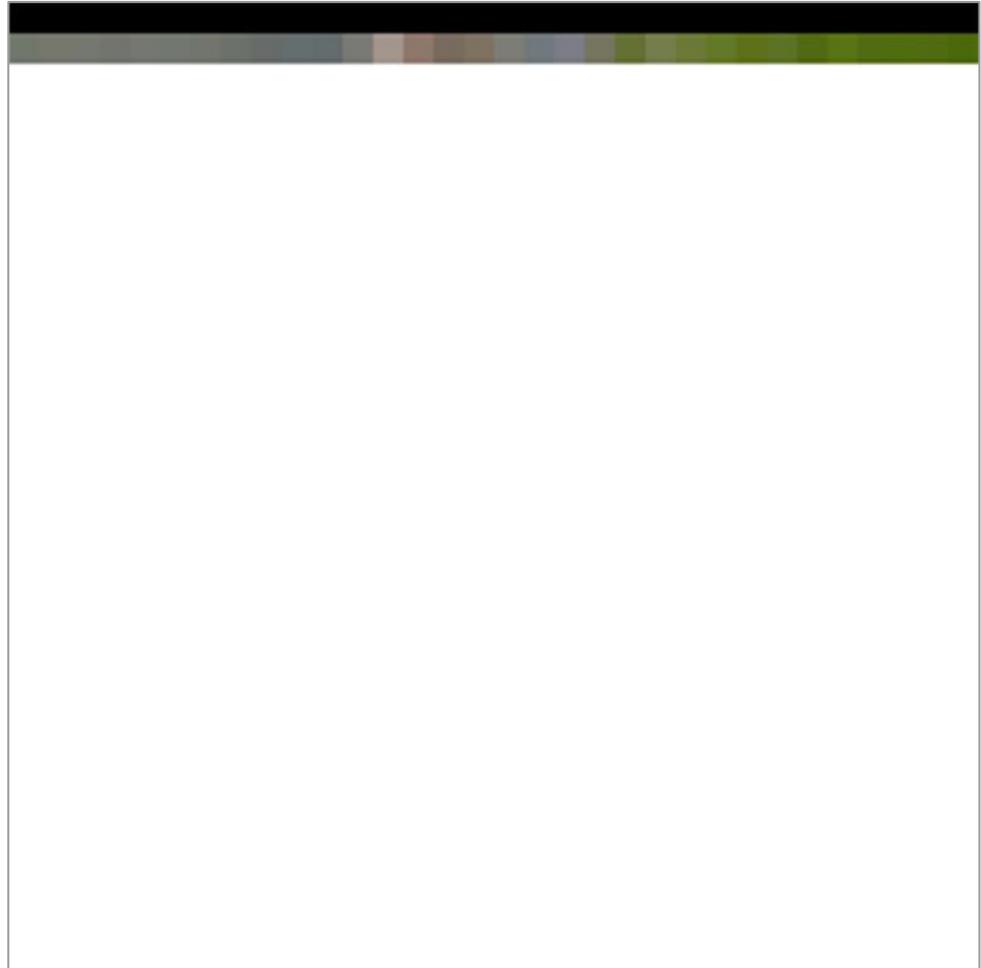
# PixelCNN



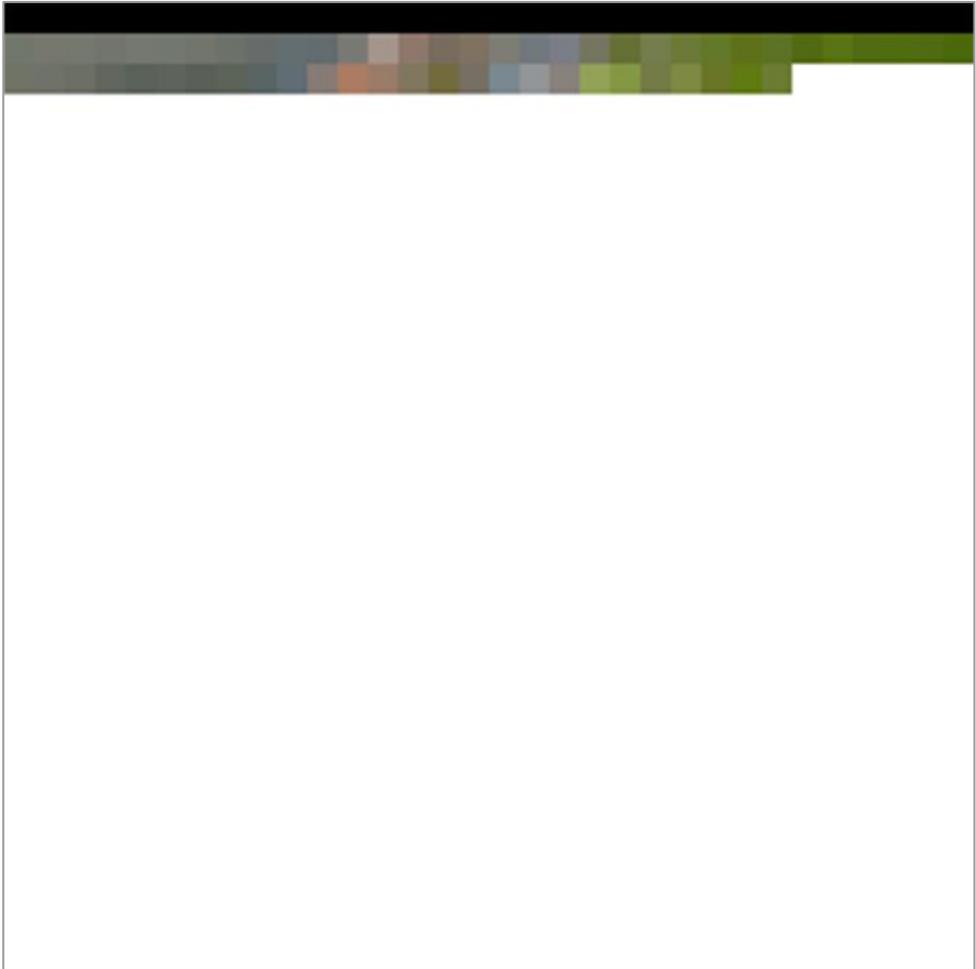
# PixelCNN



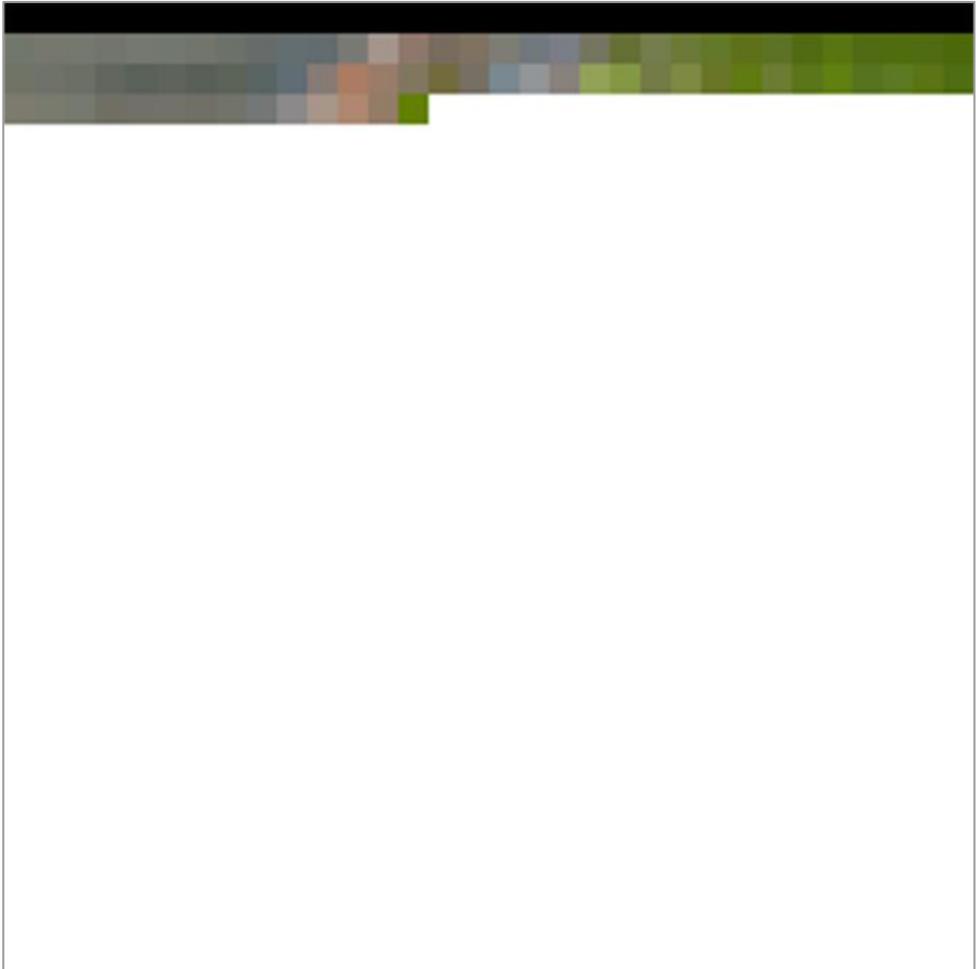
# PixelCNN



# PixelCNN



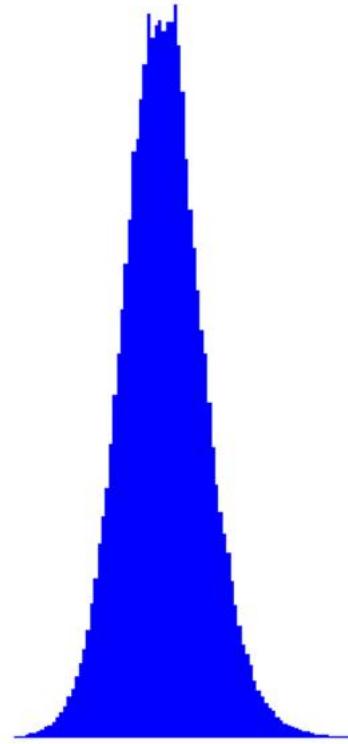
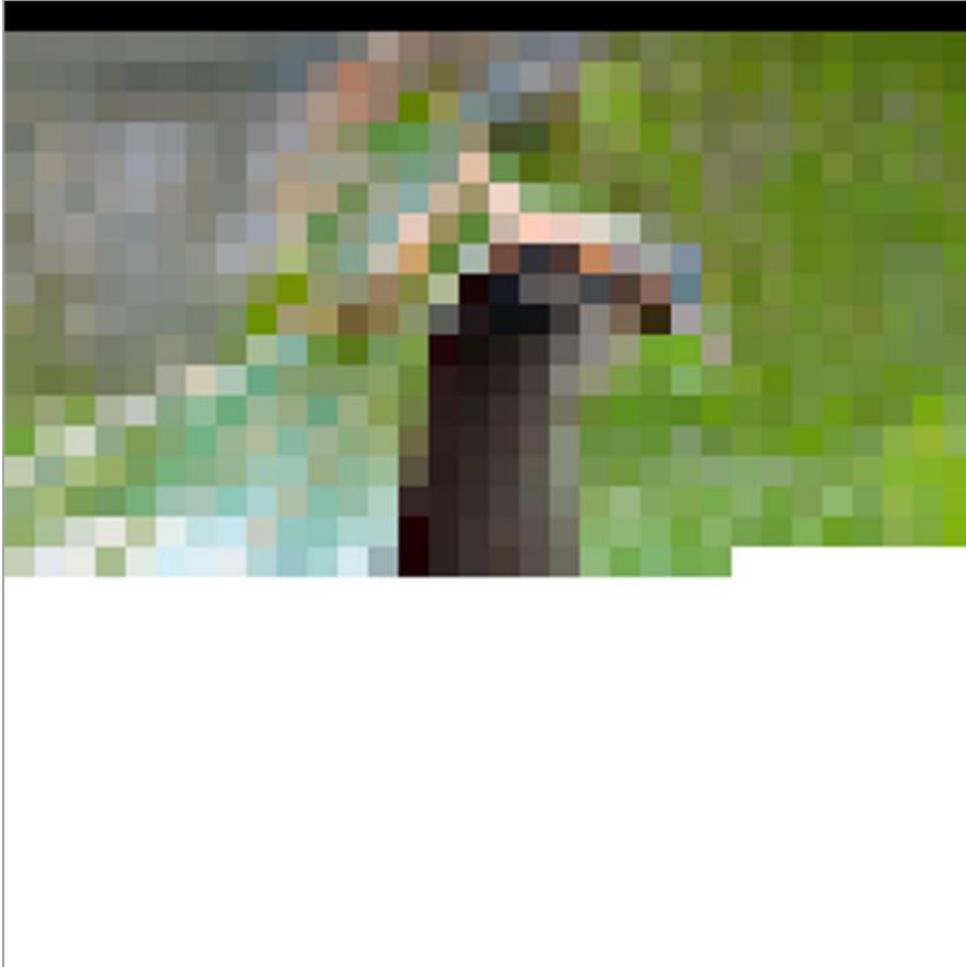
# PixelCNN



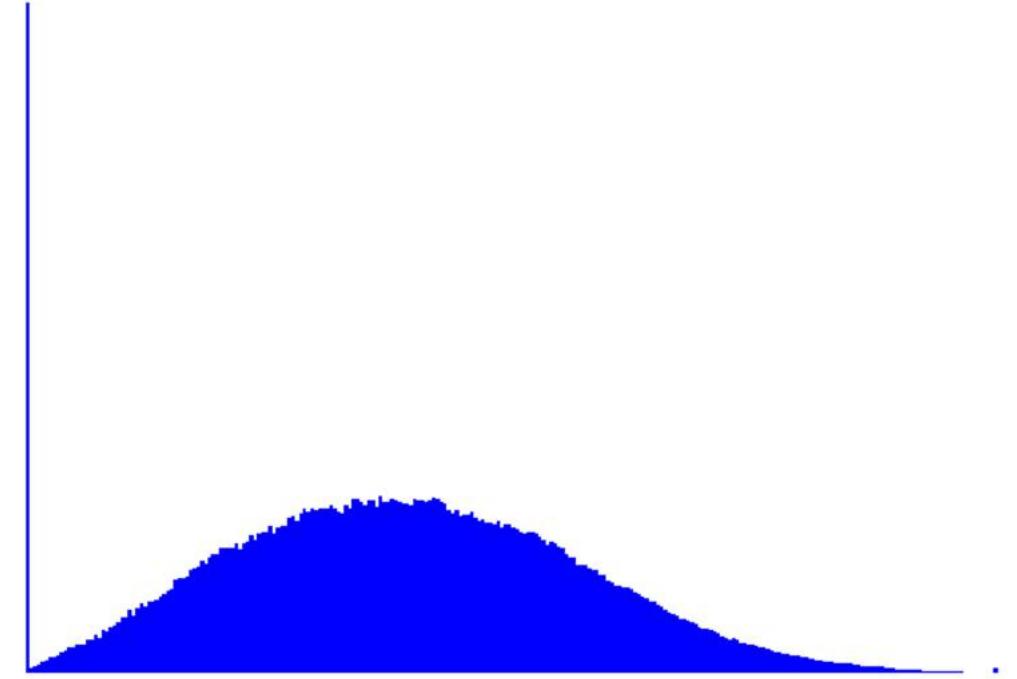
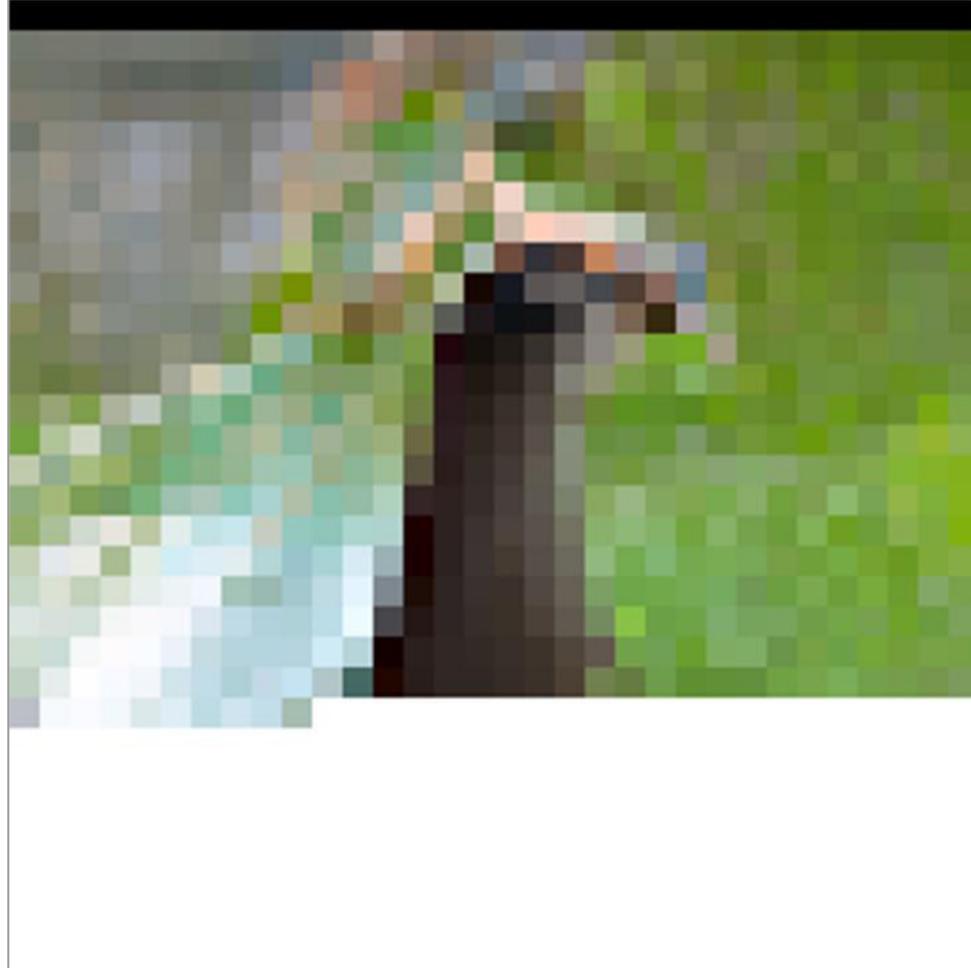
# PixelCNN



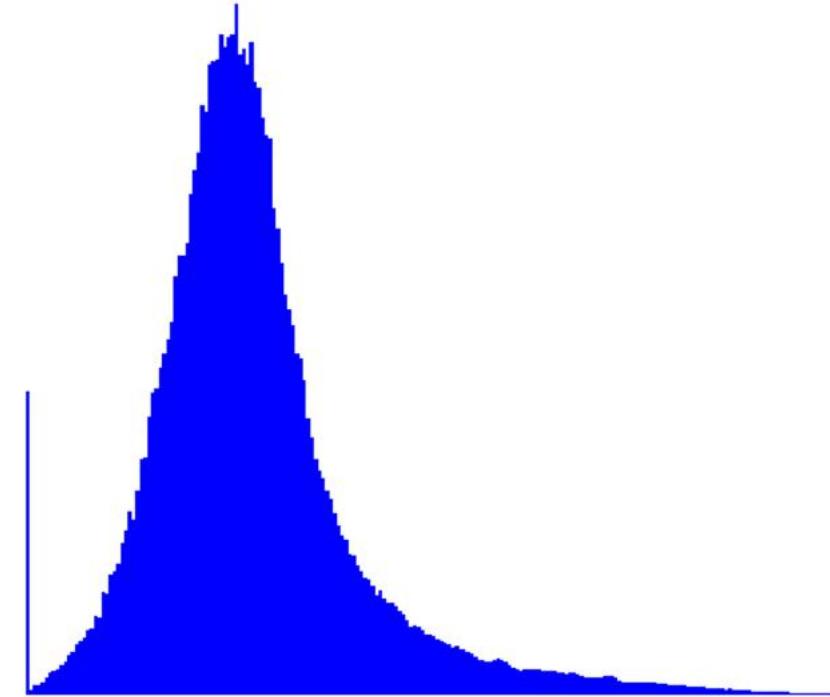
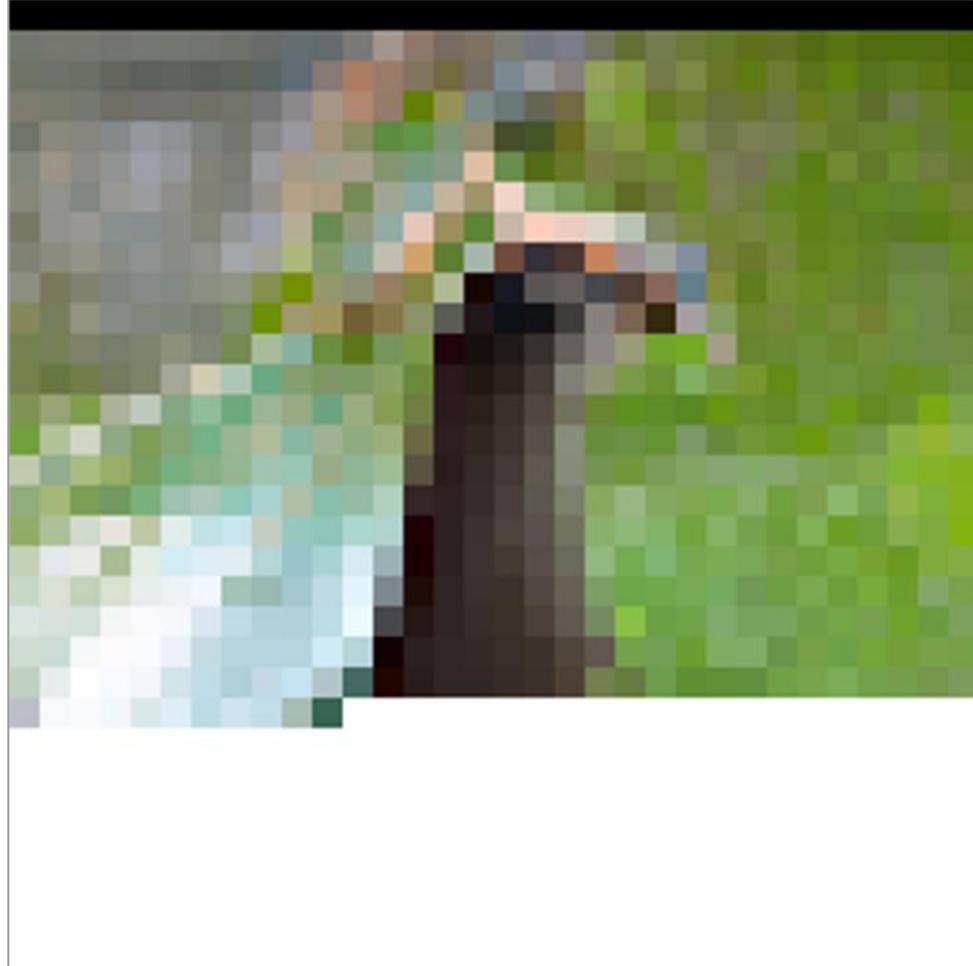
# PixelCNN – Softmax Sampling



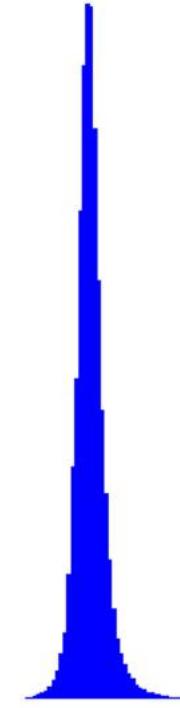
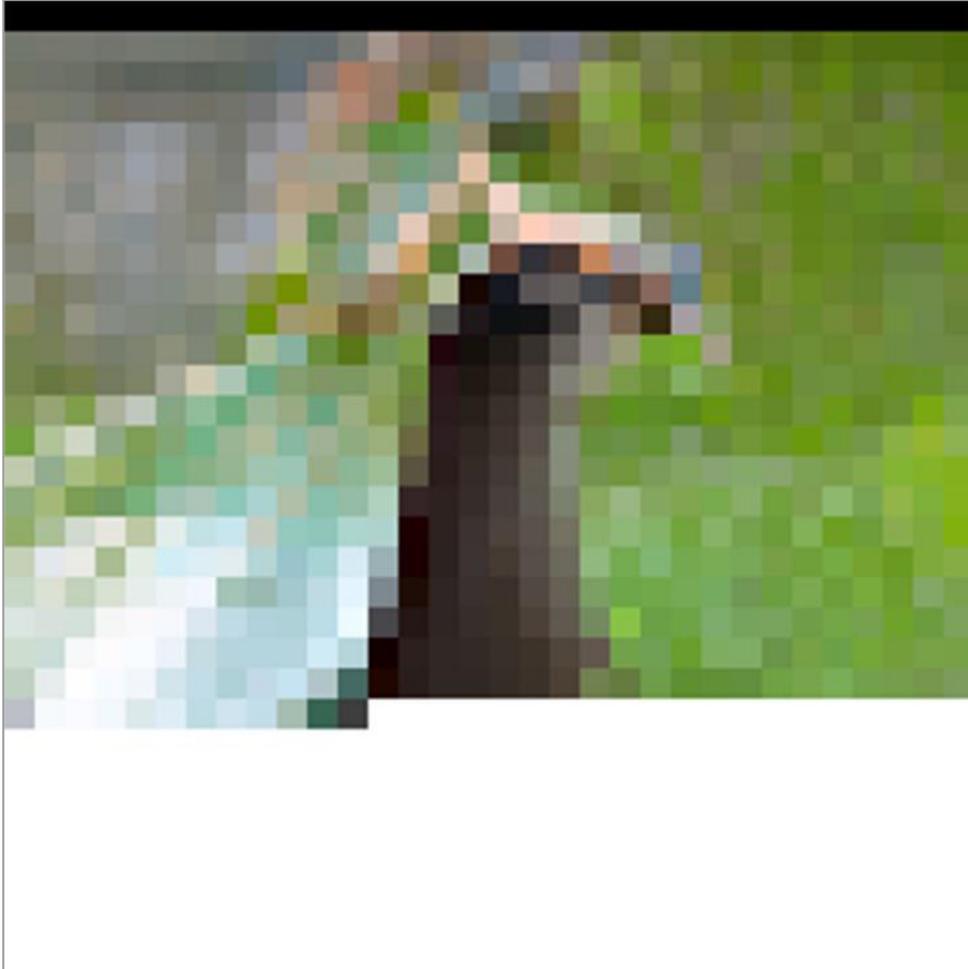
# PixelCNN – Softmax Sampling



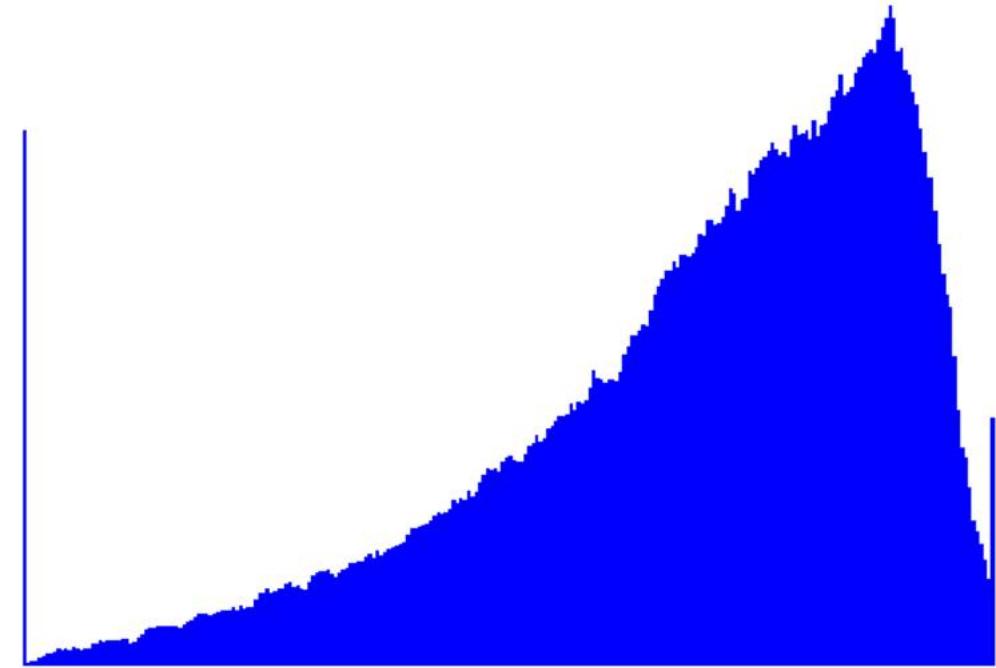
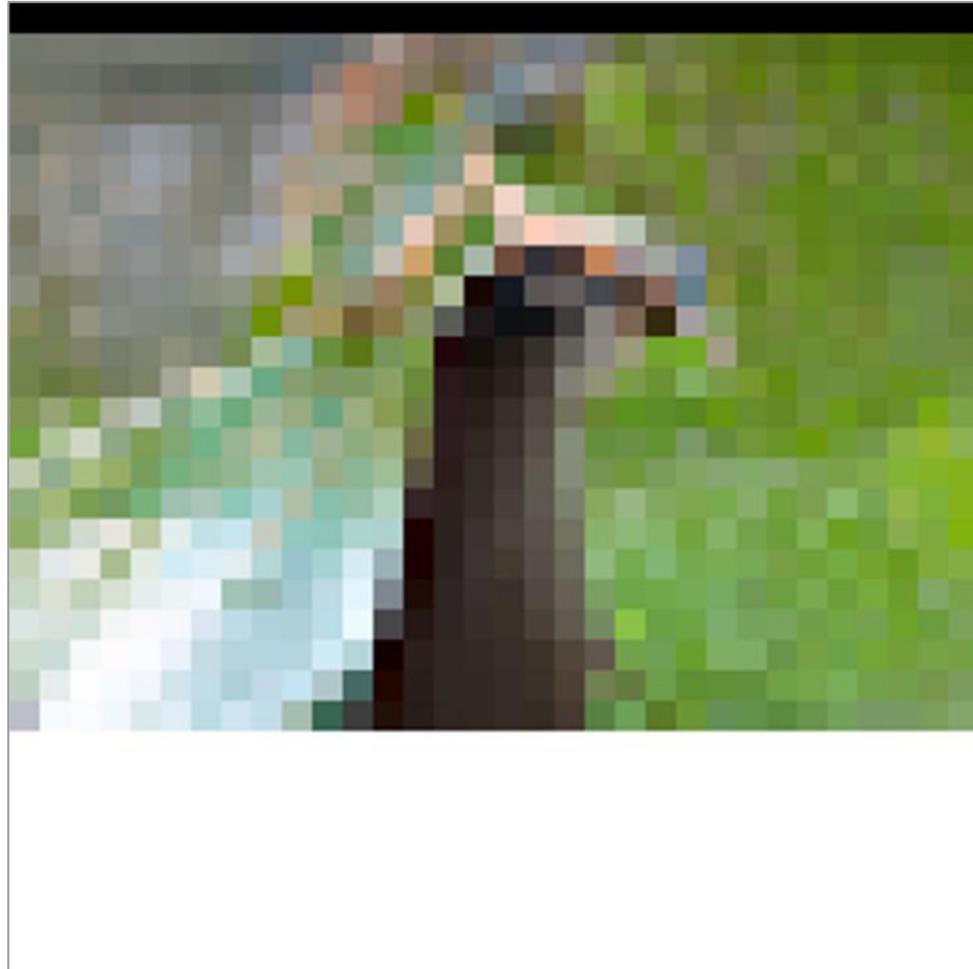
# PixelCNN – Softmax Sampling



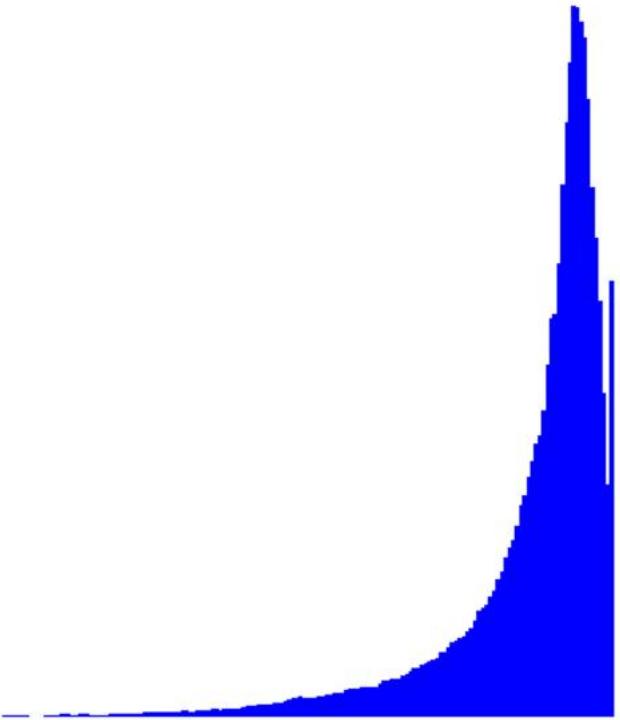
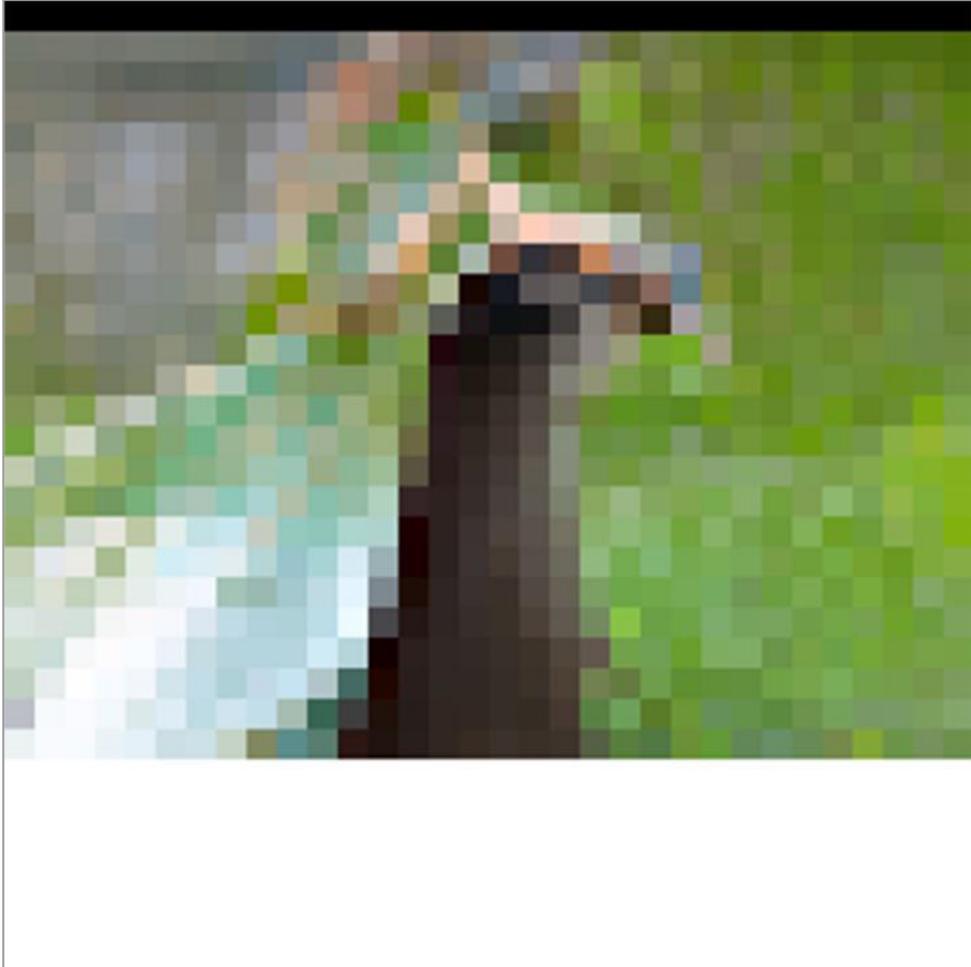
# PixelCNN – Softmax Sampling



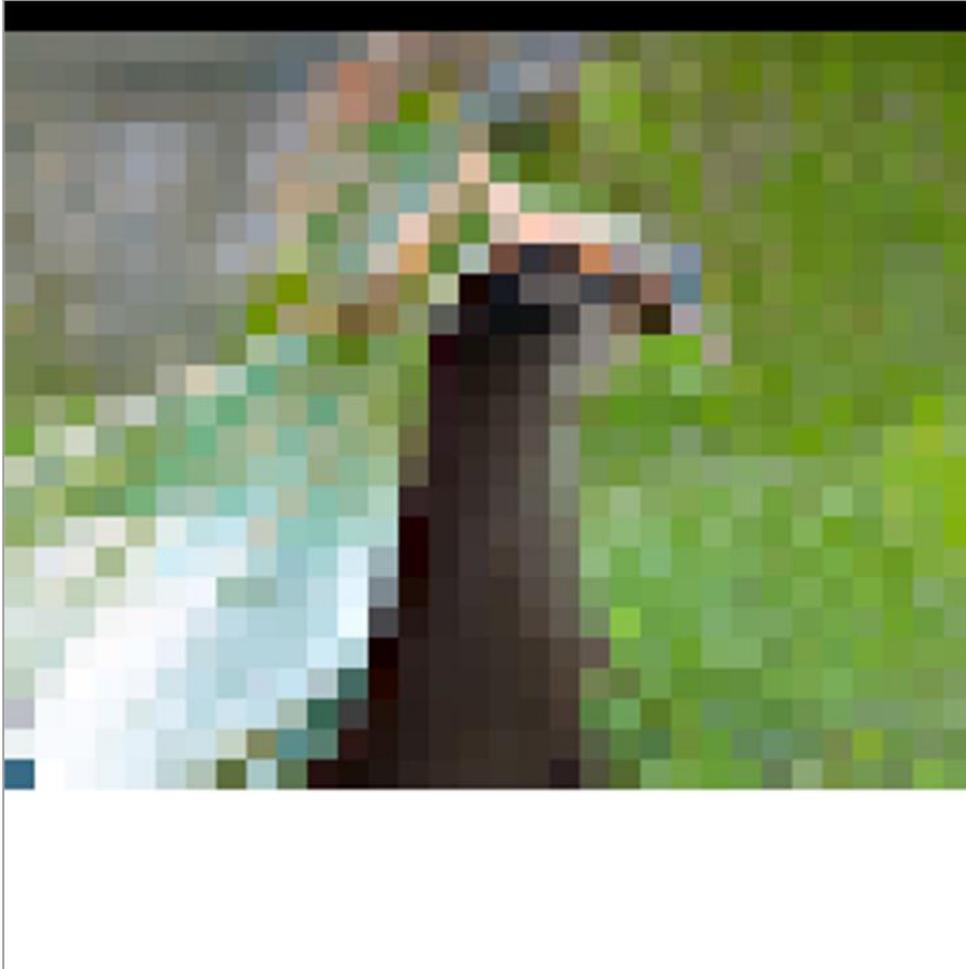
# PixelCNN – Softmax Sampling



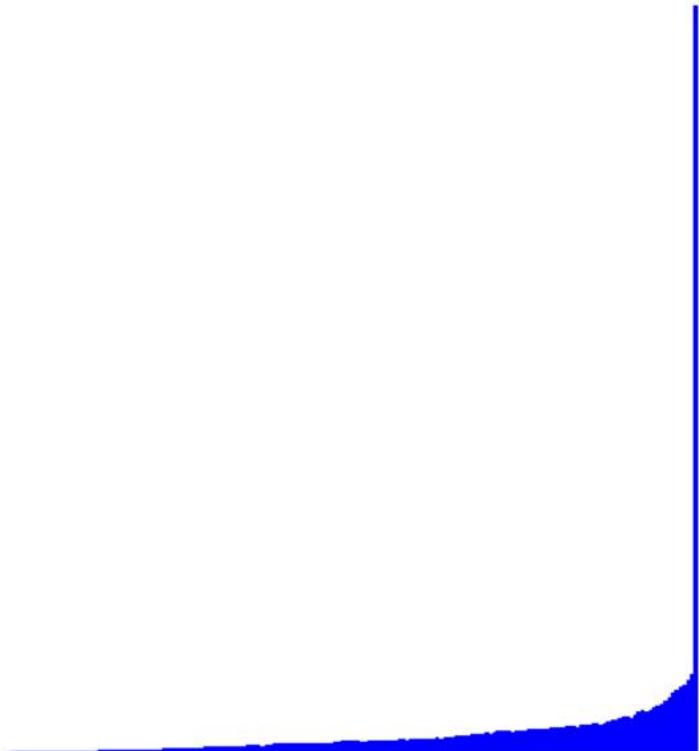
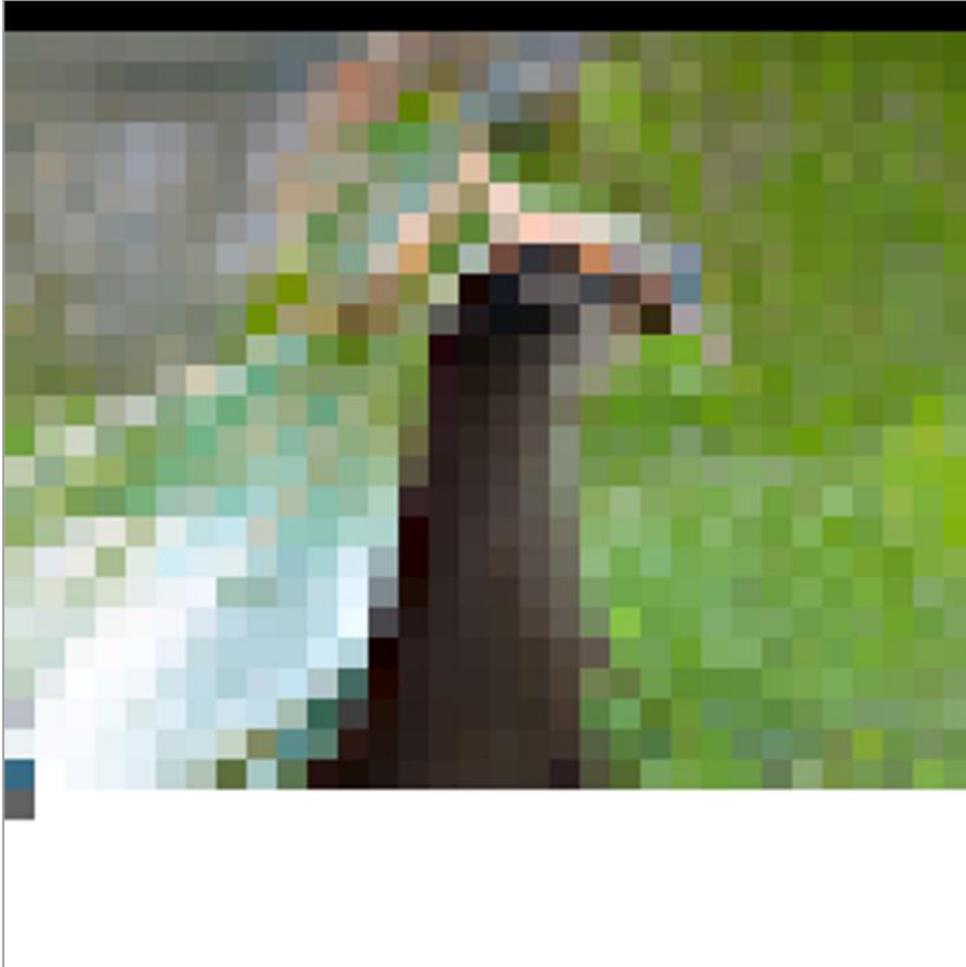
# PixelCNN – Softmax Sampling



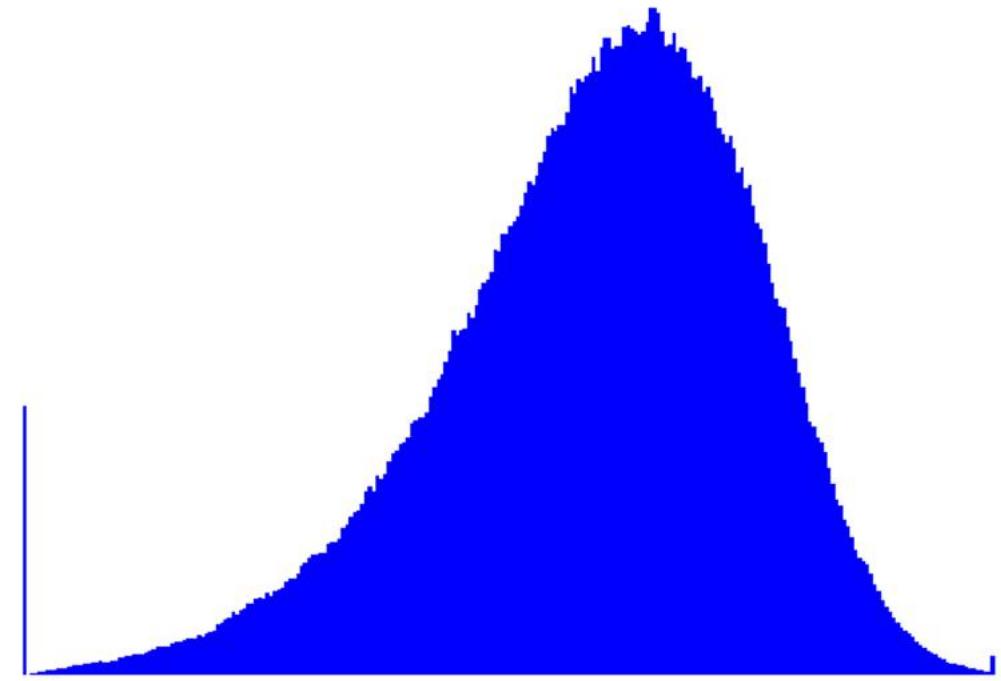
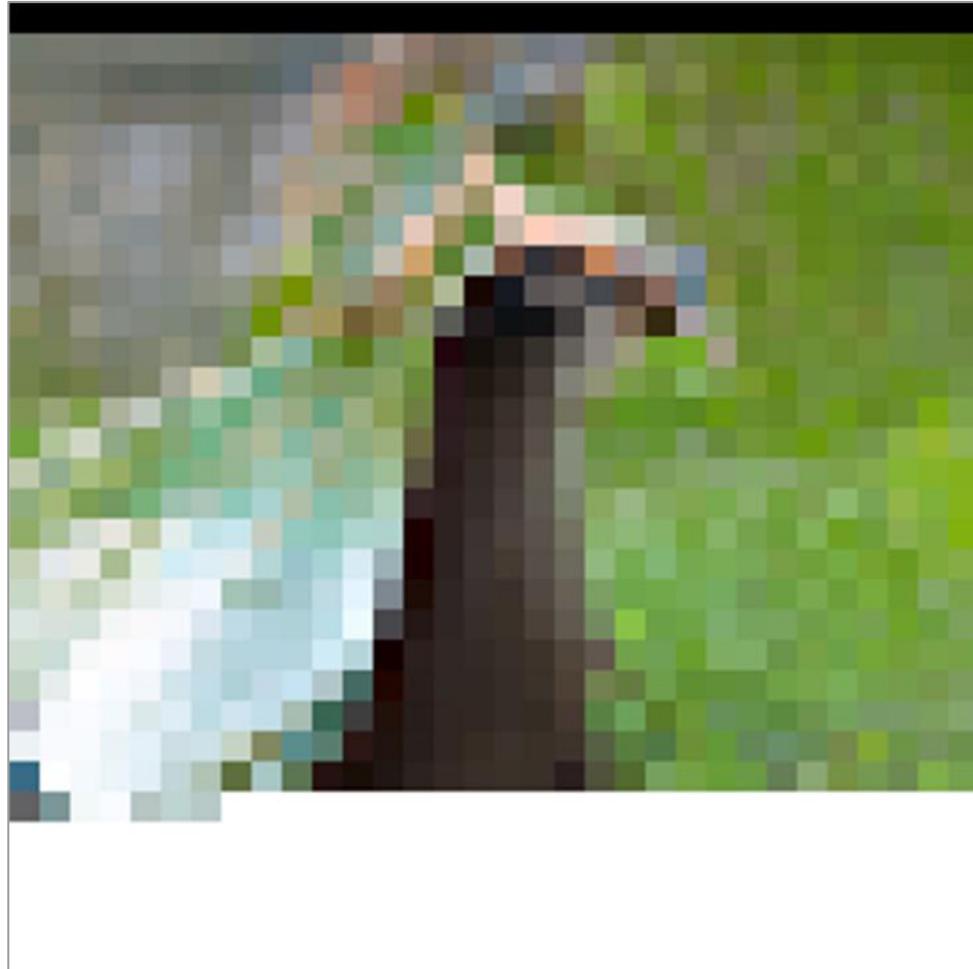
# PixelCNN – Softmax Sampling



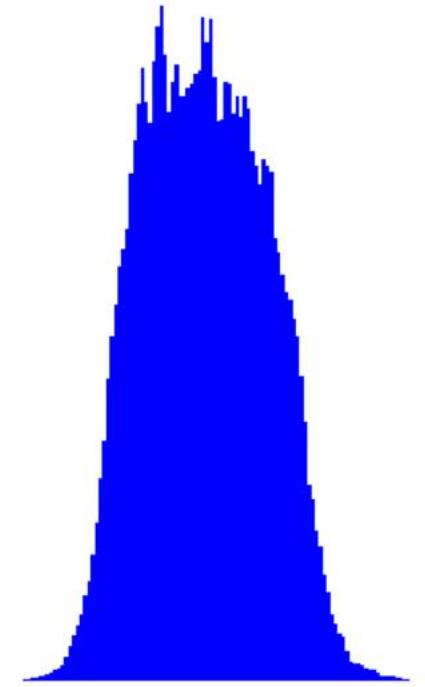
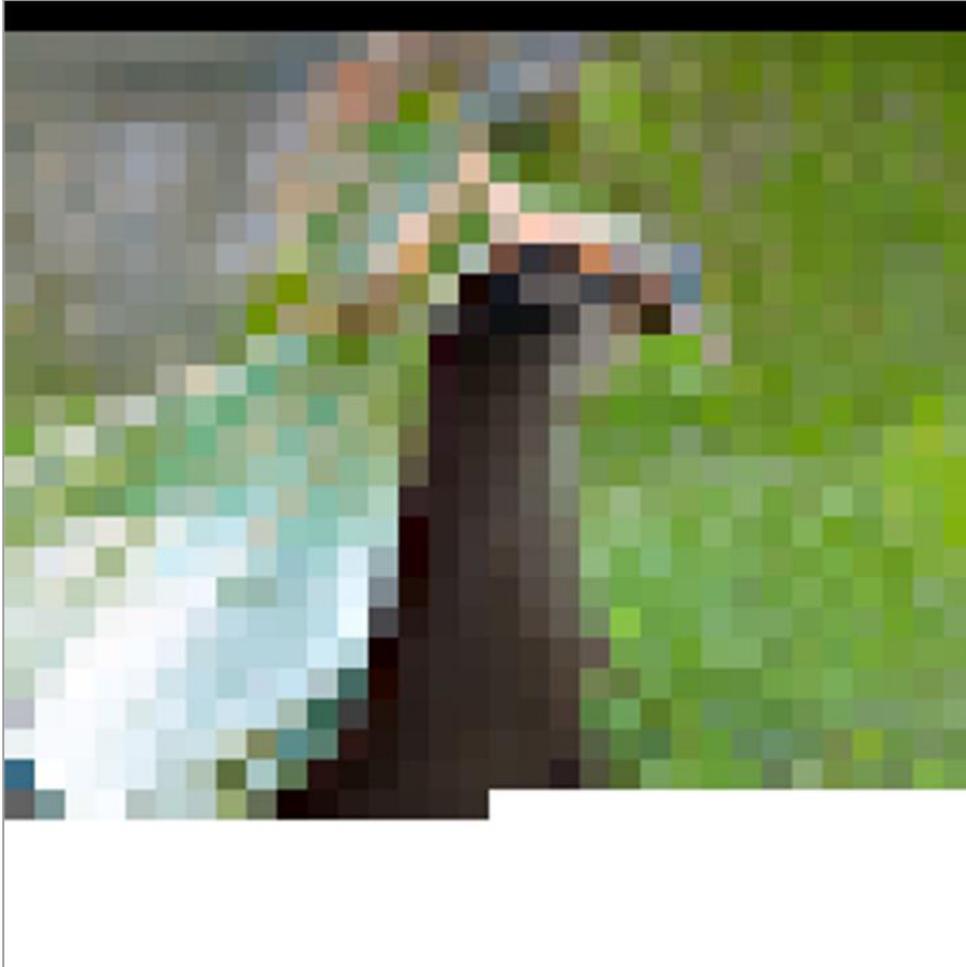
# PixelCNN – Softmax Sampling



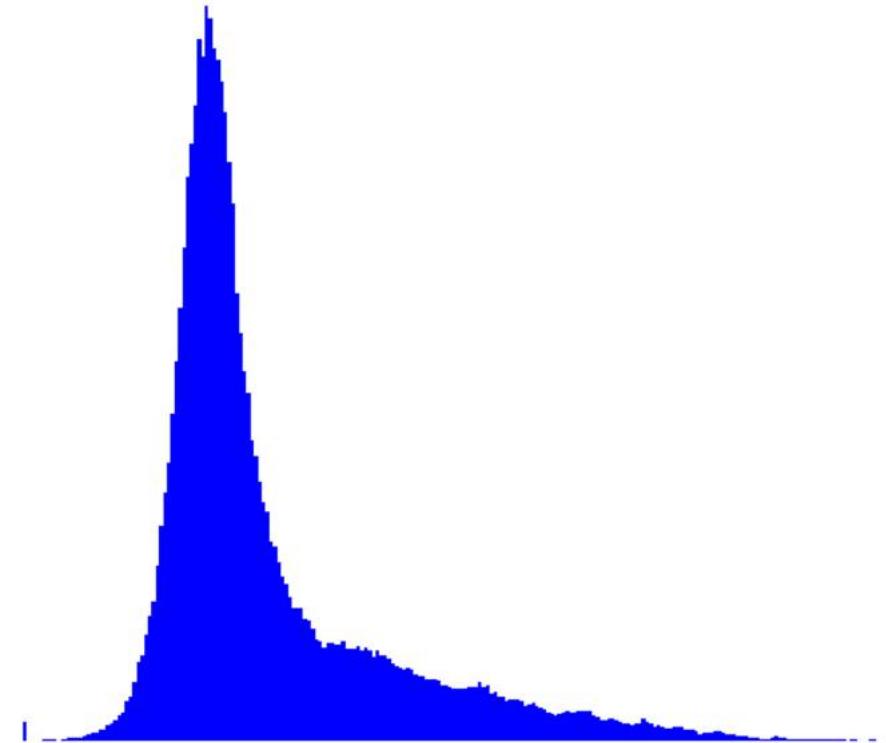
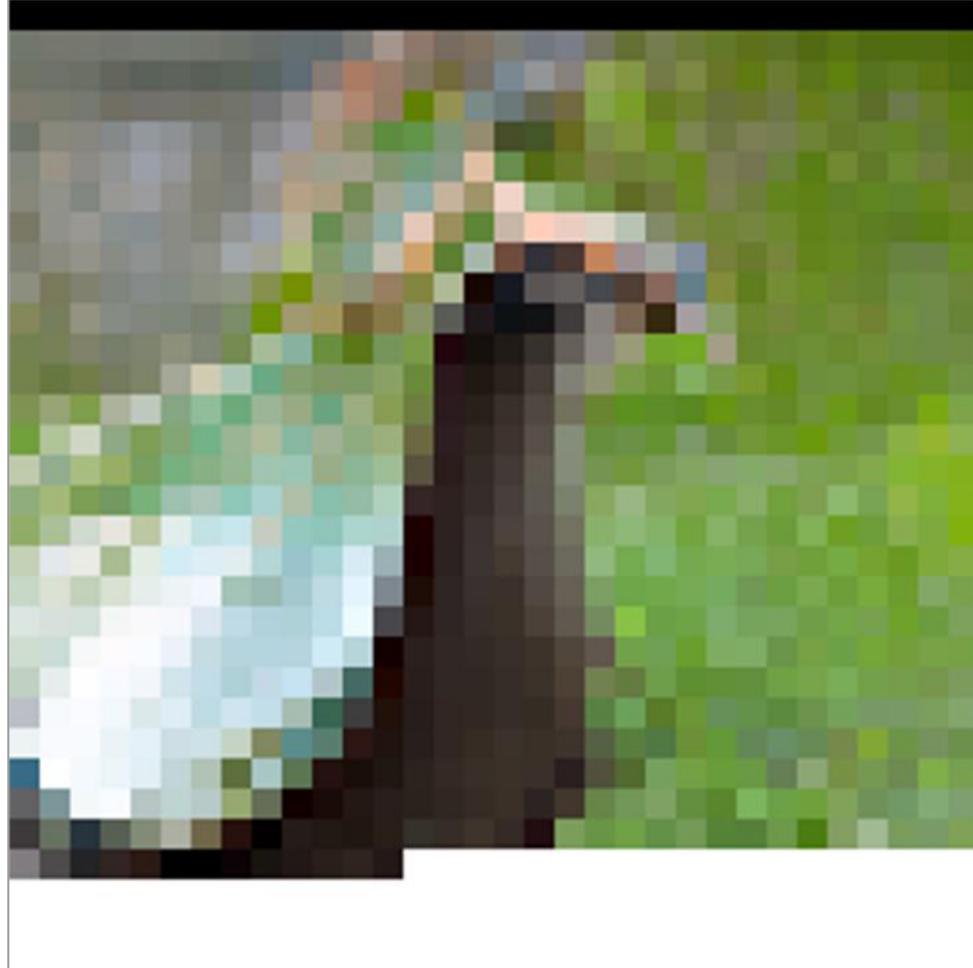
# PixelCNN – Softmax Sampling



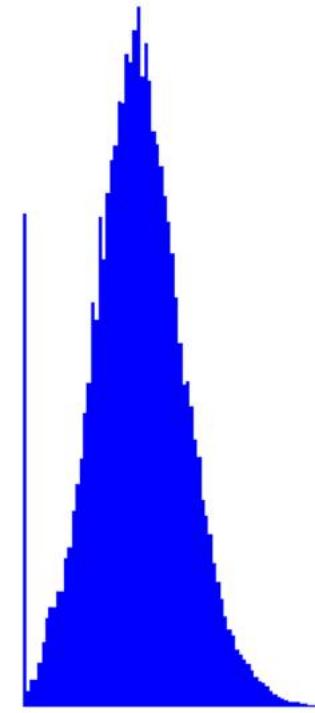
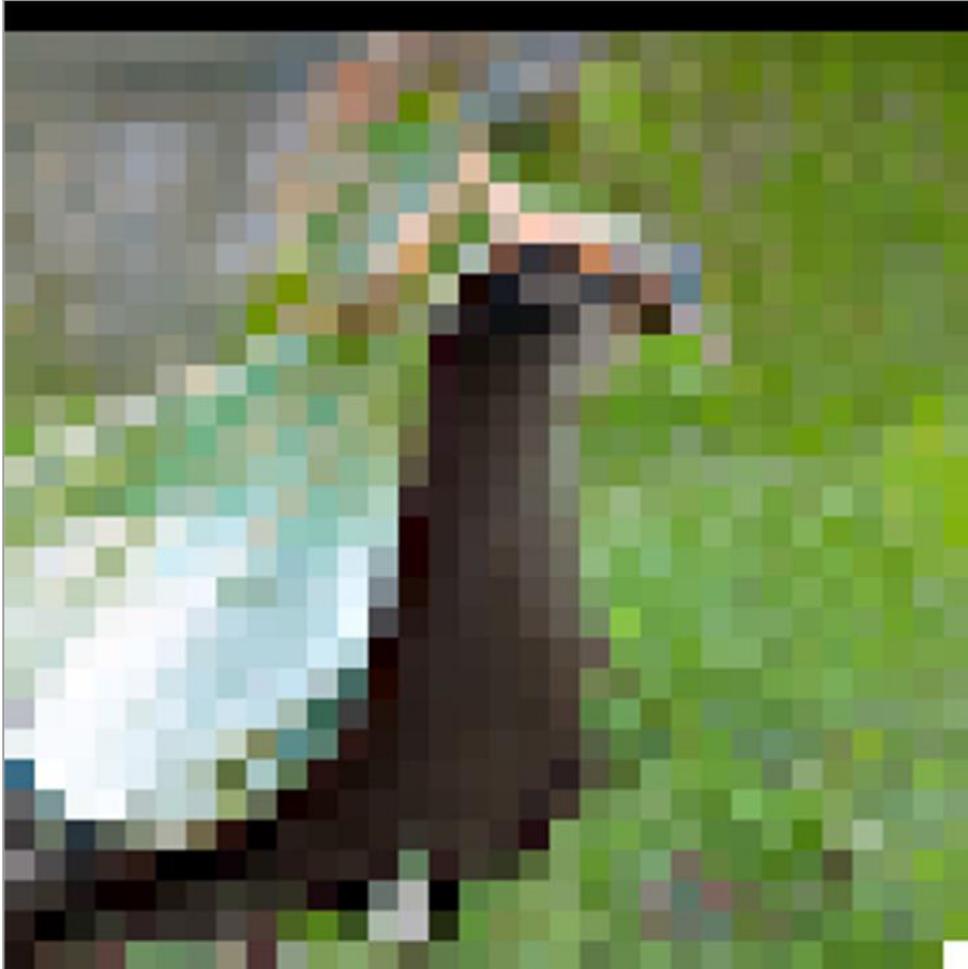
# PixelCNN – Softmax Sampling



# PixelCNN – Softmax Sampling

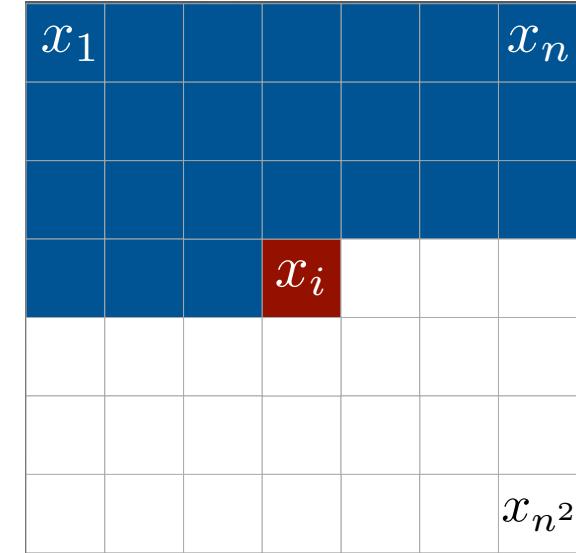
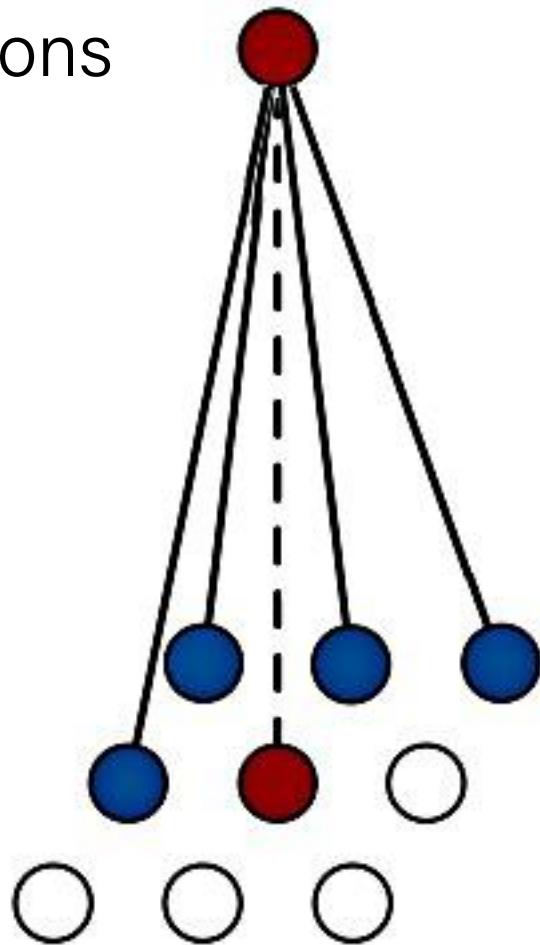


# PixelCNN – Softmax Sampling



# PixelCNN

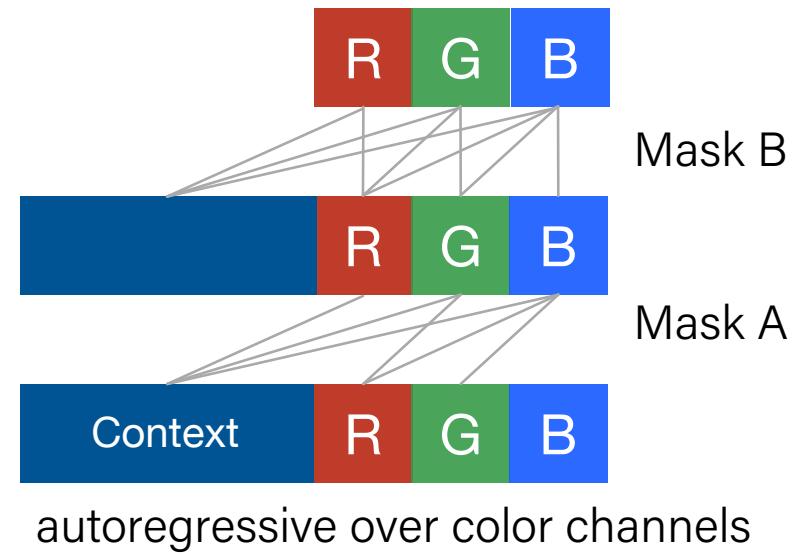
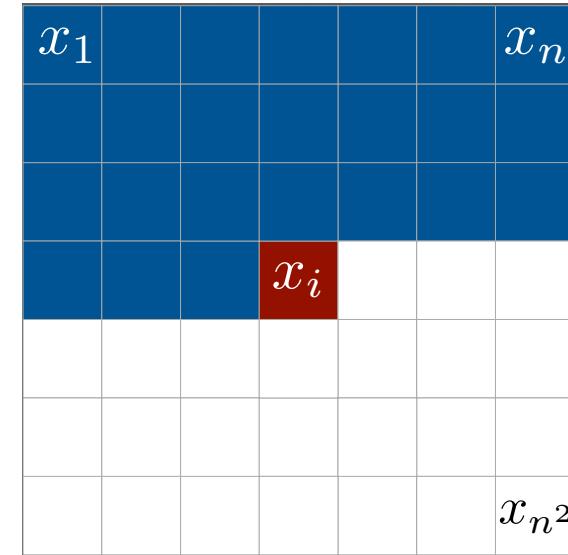
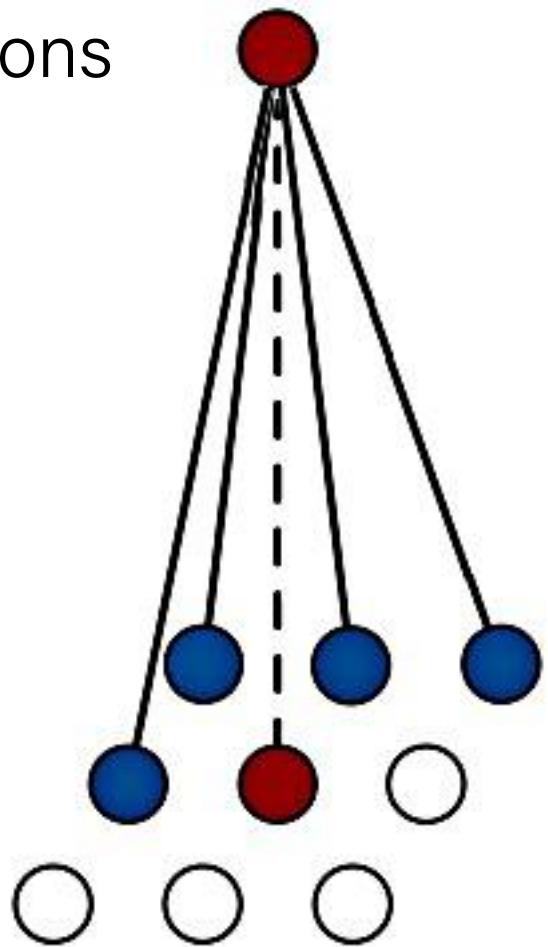
use masked convolutions  
to enforce the  
autoregressive  
relationship



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

# PixelCNN

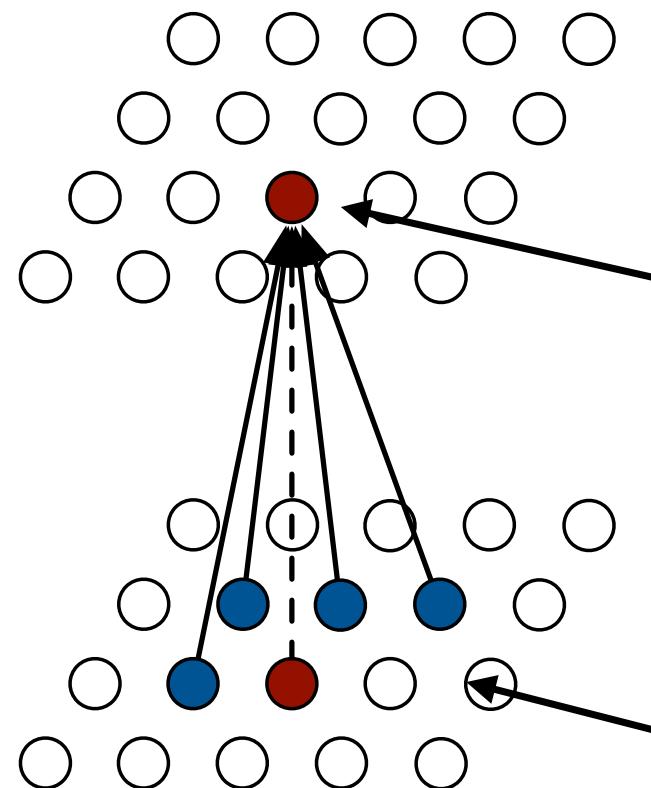
use masked convolutions  
to enforce the  
autoregressive  
relationship



$$p(x_i \mid \mathbf{x}_{<i}) = p(x_{i,R} \mid \mathbf{x}_{<i})p(x_{i,G} \mid x_{i,R}, \mathbf{x}_{<i})p(x_{i,B} \mid x_{i,R}, x_{i,G}, \mathbf{x}_{<i})$$

# PixelCNN

Multiple layers of masked convolutions



masked convolution

composing multiple layers increases the context size

only depends on pixel above and to the left

# Samples from PixelCNN

Topics: CIFAR-10

- Samples from a class-conditioned PixelCNN



Coral Reef

# Samples from PixelCNN

Topics: CIFAR-10

- Samples from a class-conditioned PixelCNN



Sorrel horse

# Samples from PixelCNN

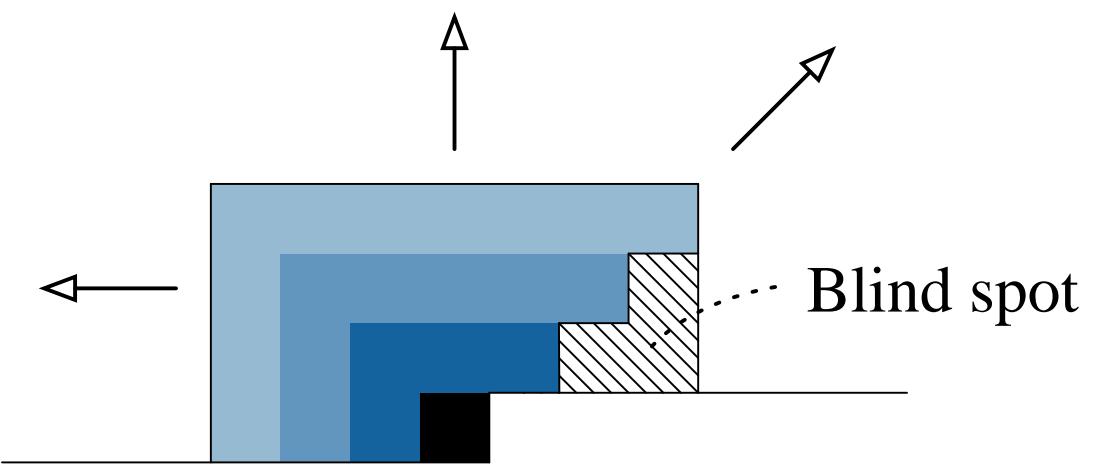
Topics: CIFAR-10

- Samples from a class-conditioned PixelCNN

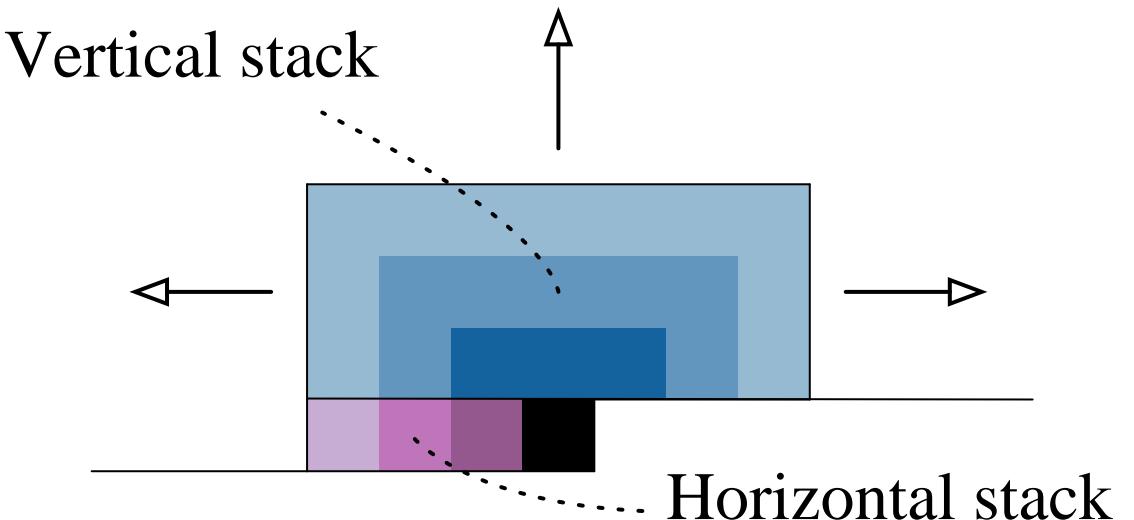


Sandbar

# Improving PixelCNN



Stacking layers of masked convolution creates a blindspot

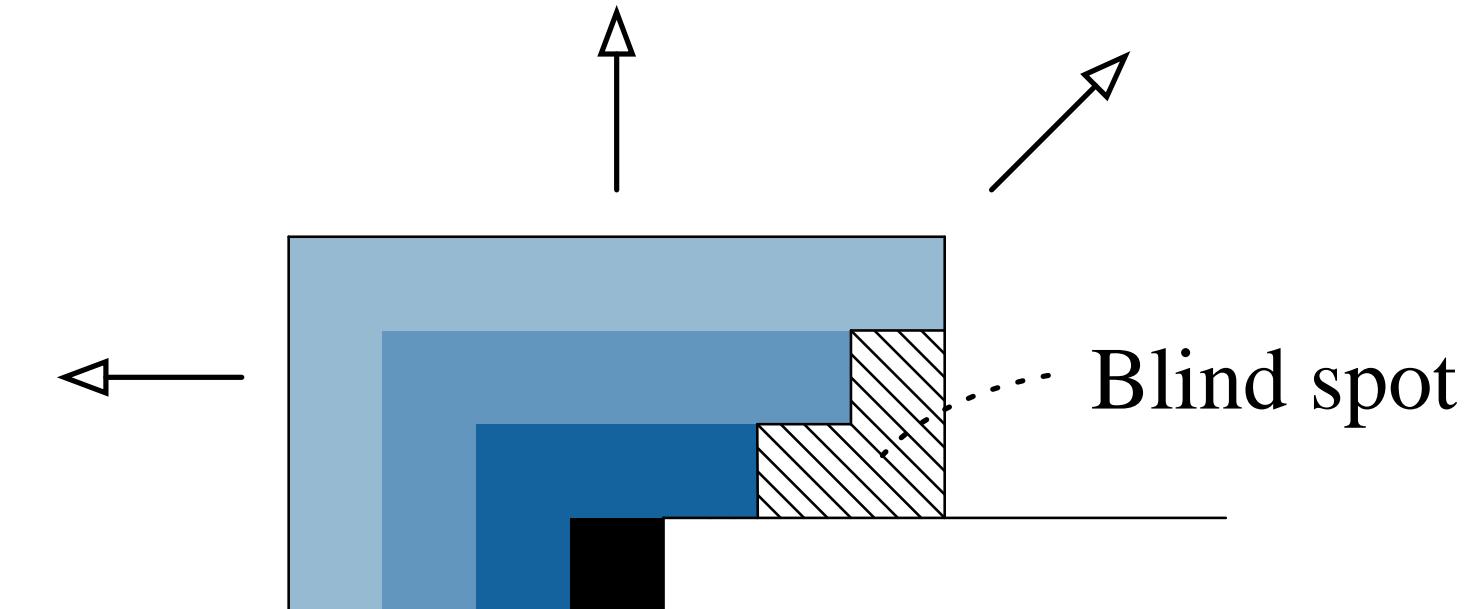


**Solution:** use two stacks of convolution, a vertical stack and a horizontal stack

# Improving PixelCNN I

There is a problem with this form of masked convolution.

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

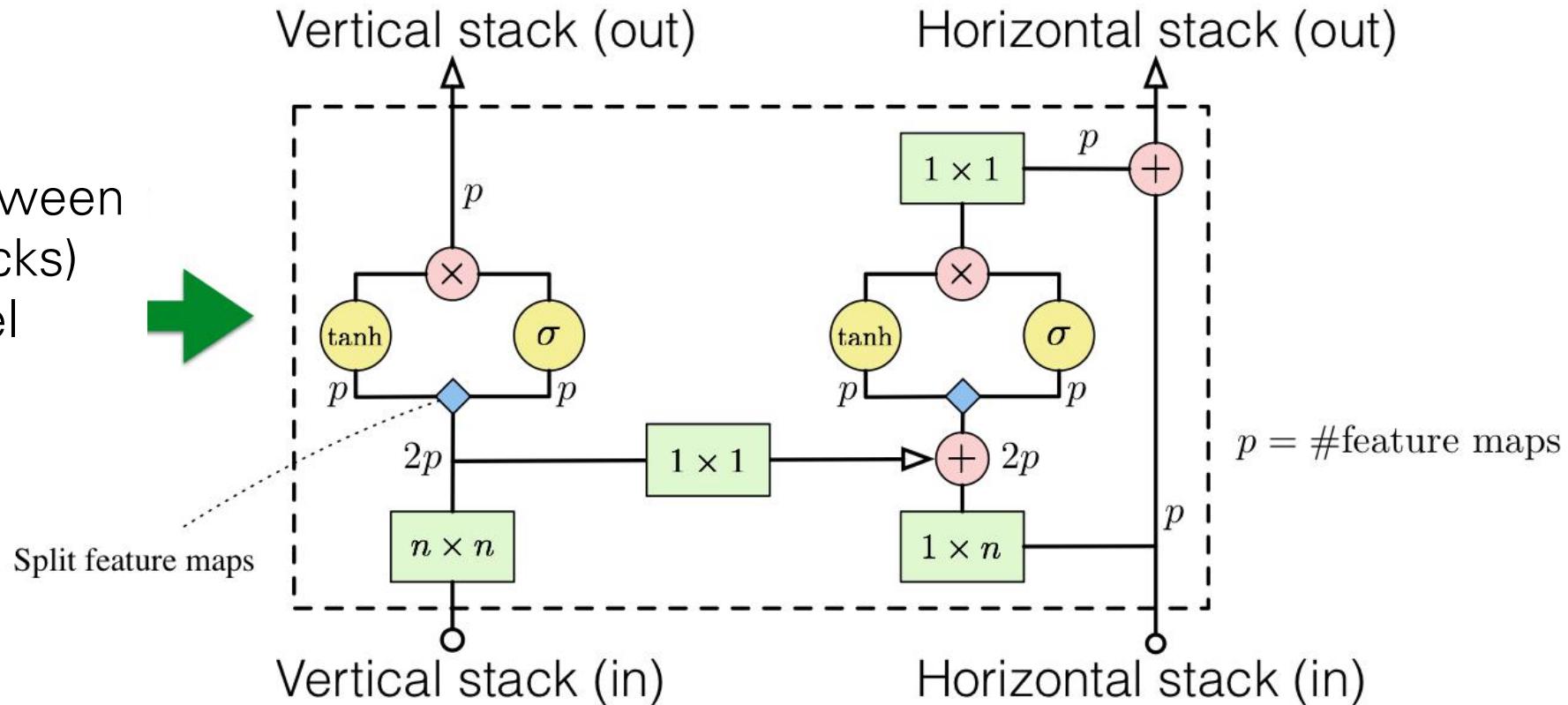


Stacking layers of masked convolution creates a blindspot

# Improving PixelCNN II

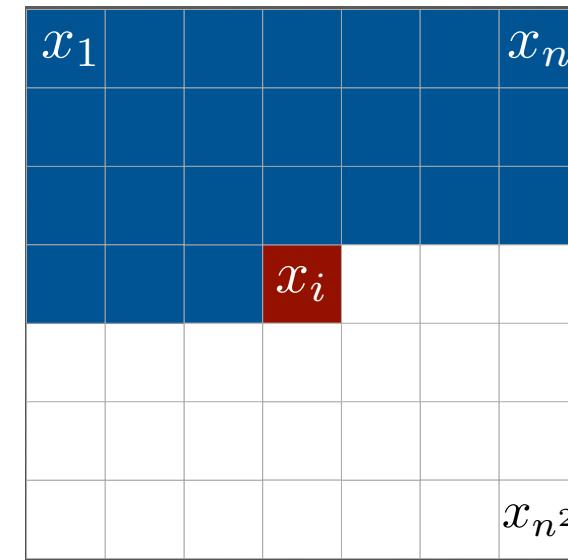
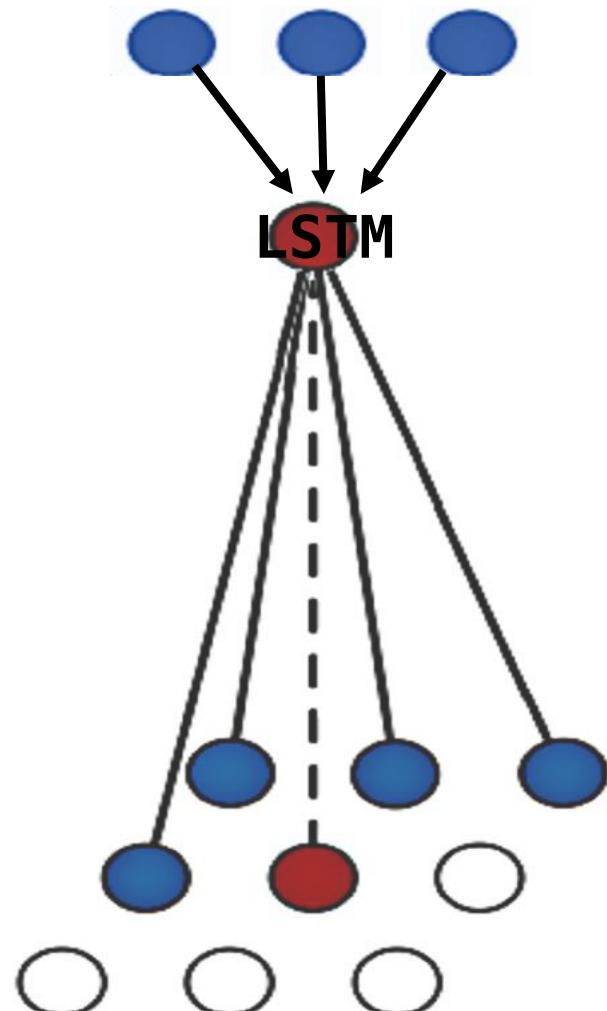
Use more expressive nonlinearity:  $\mathbf{h}_{k+1} = \tanh(W_{k,f} * \mathbf{h}_k) \odot \sigma(W_{k,g} * \mathbf{h}_k)$

This information flow (between vertical and horizontal stacks) preserves the correct pixel dependencies



# Convolutional Long Short-Term Memory

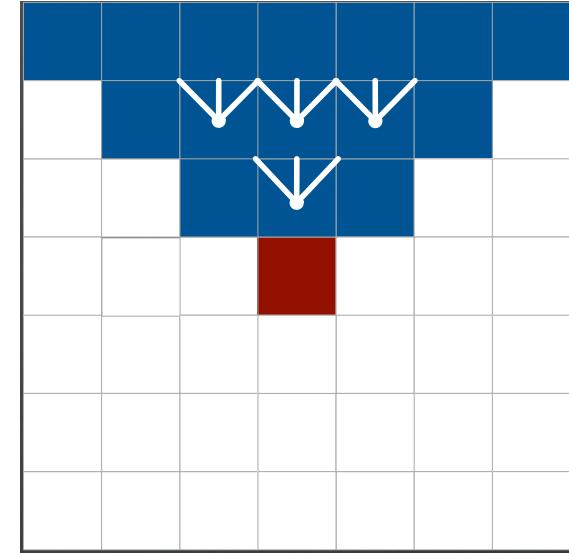
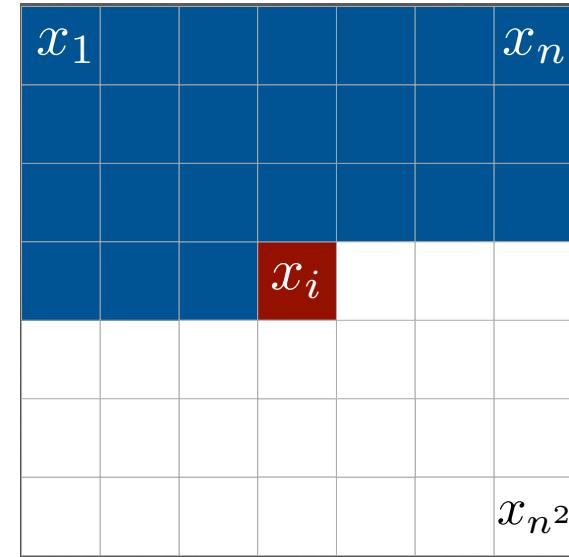
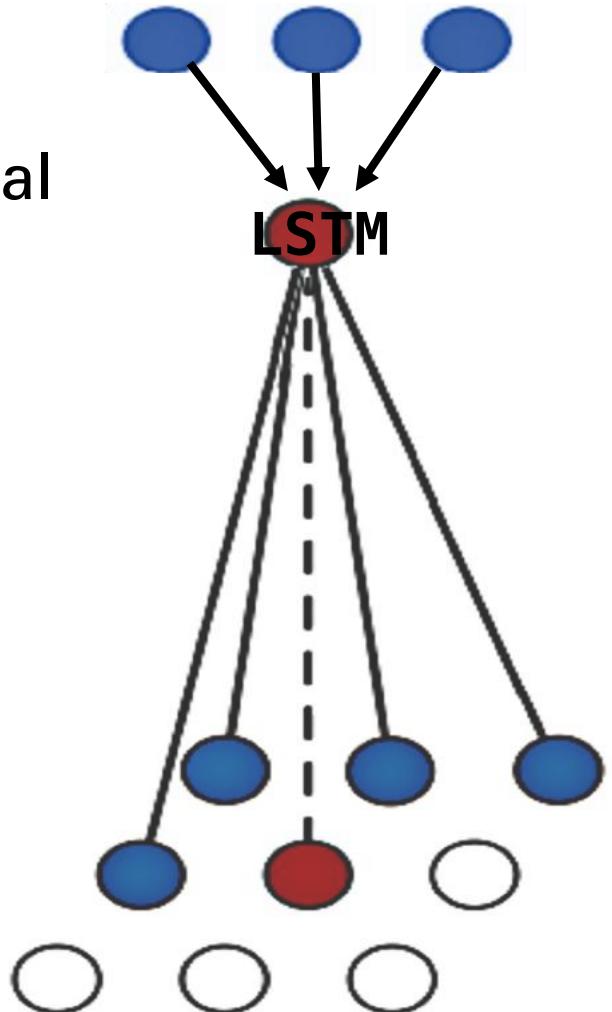
Row LSTM



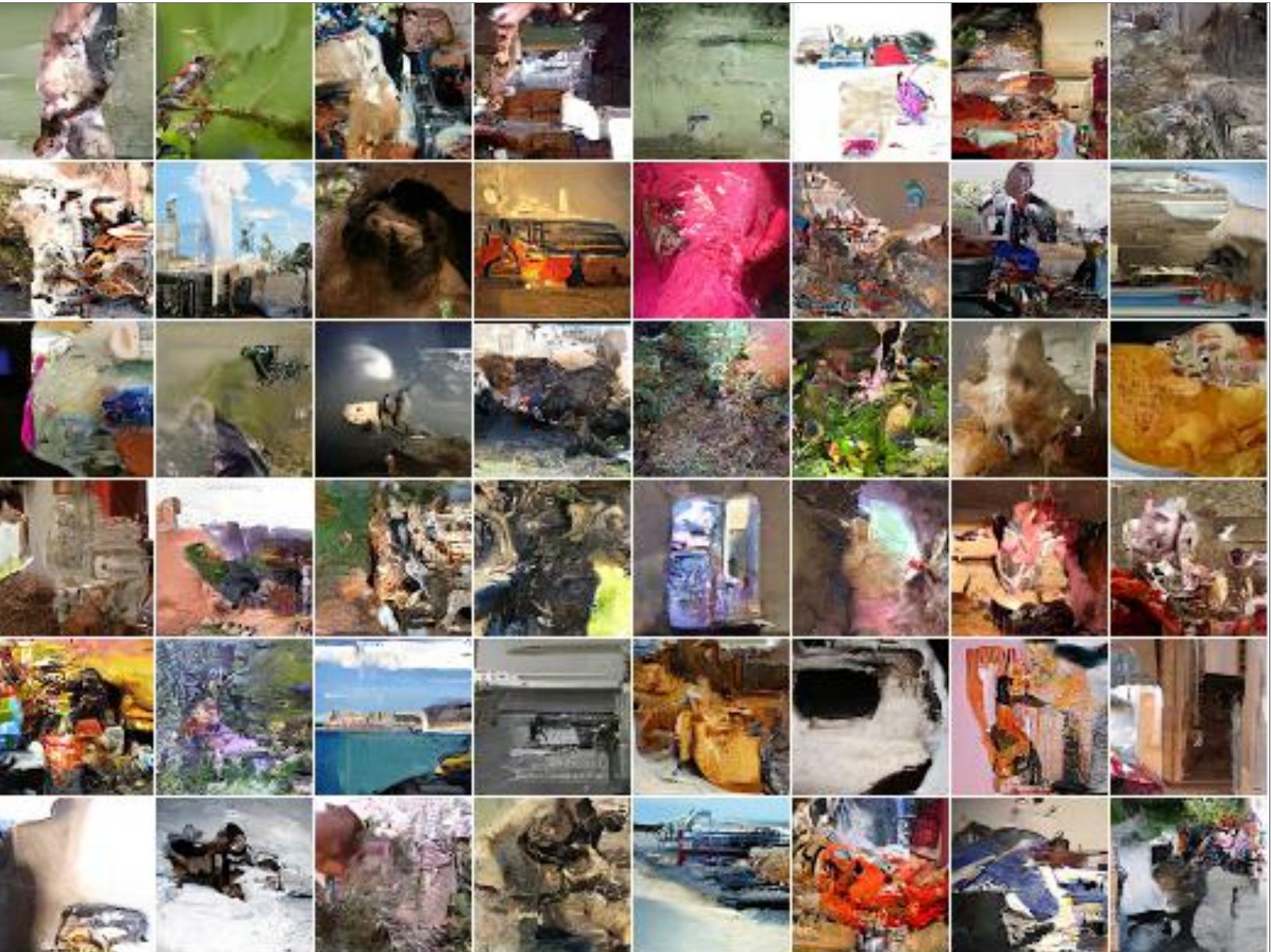
Stollenga et al, 2015  
Oord, Kalchbrenner, Kavukcuoglu, 2016

# Pixel RNN

Multiple layers of convolutional LSTM

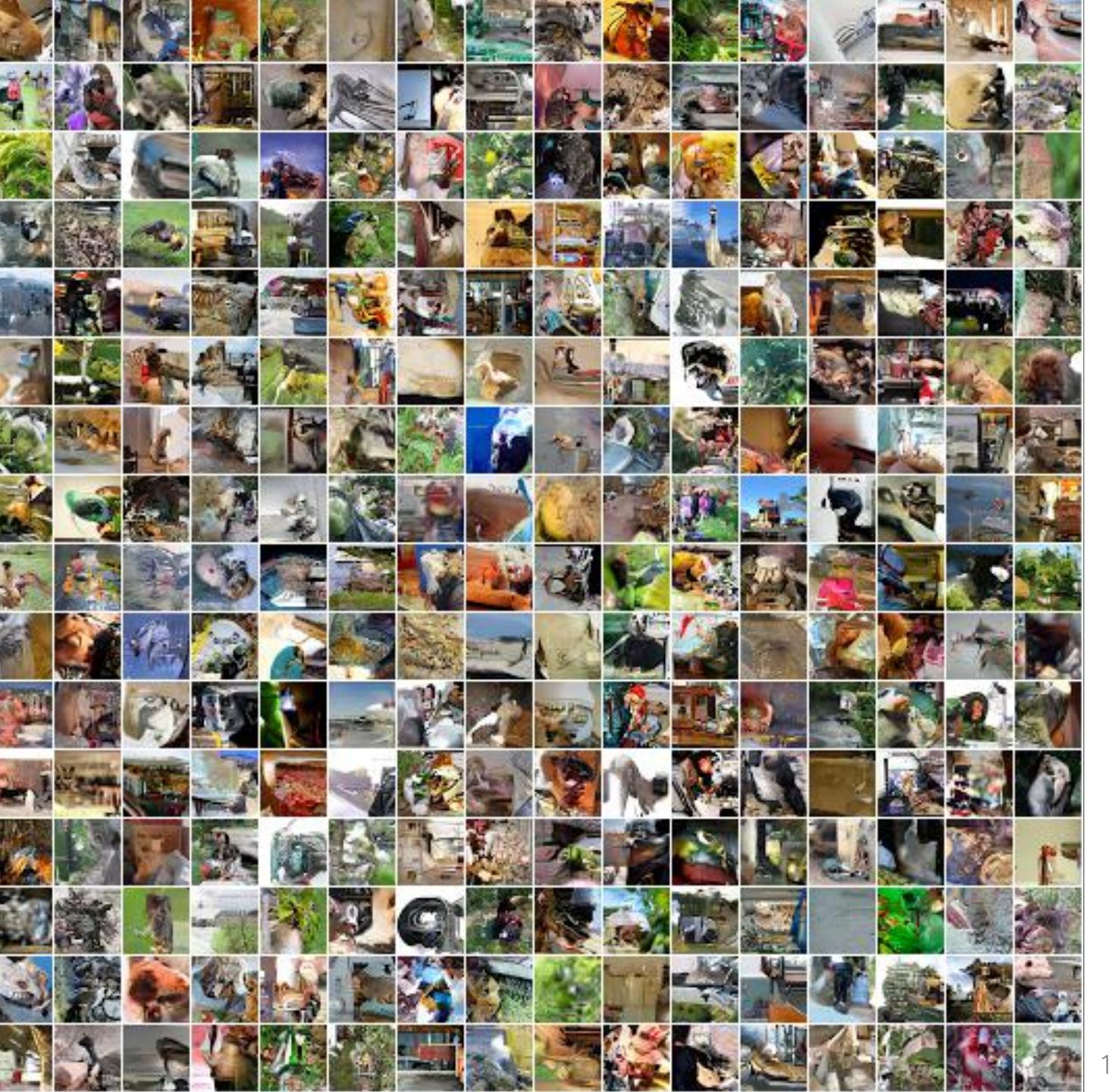


# Samples from PixelRNN



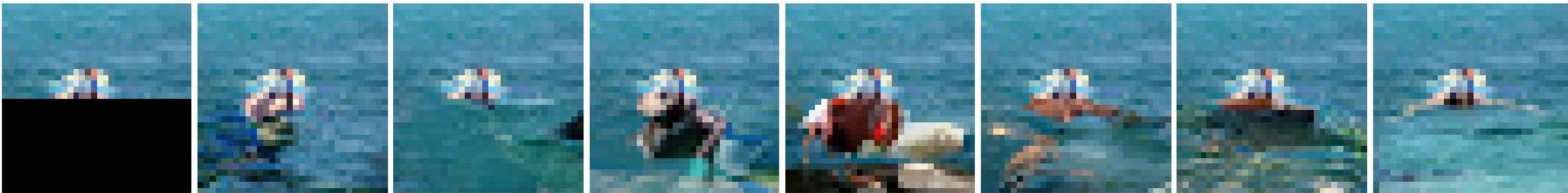
Slide credit:  
Nal Kalchbrenner

# Samples from PixelRNN

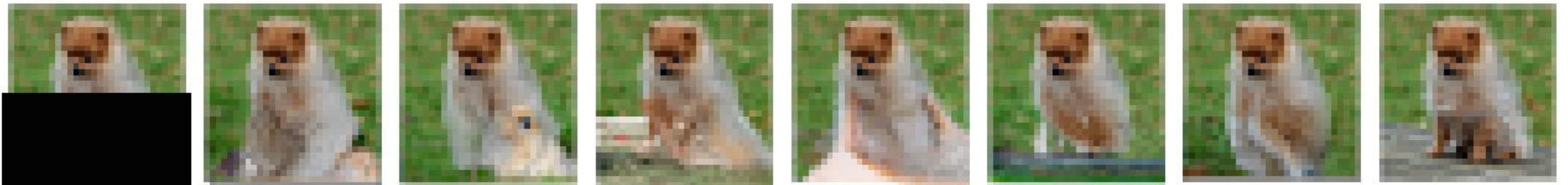


# Image completions (conditional samples) from PixelRNN

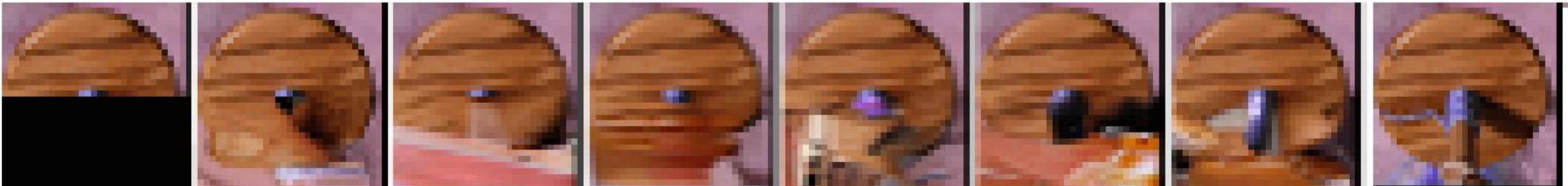
occluded



completions



original



[PixelRNN, van der Oord et al. 2016]

# Modeling Audio

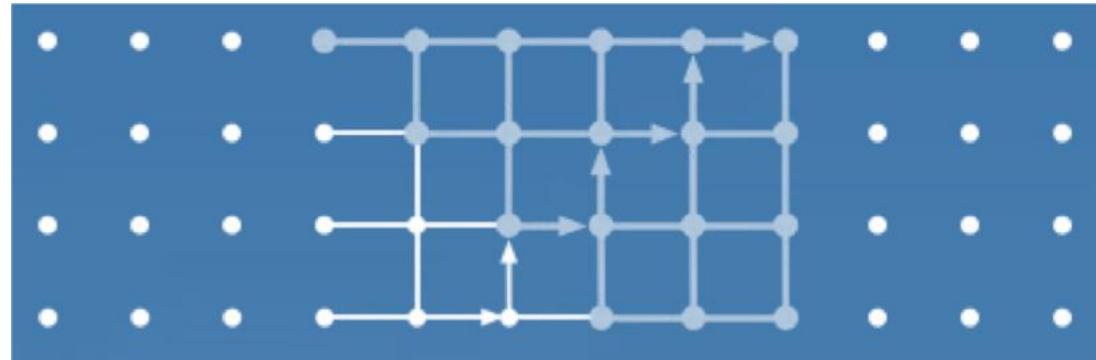


1 Second

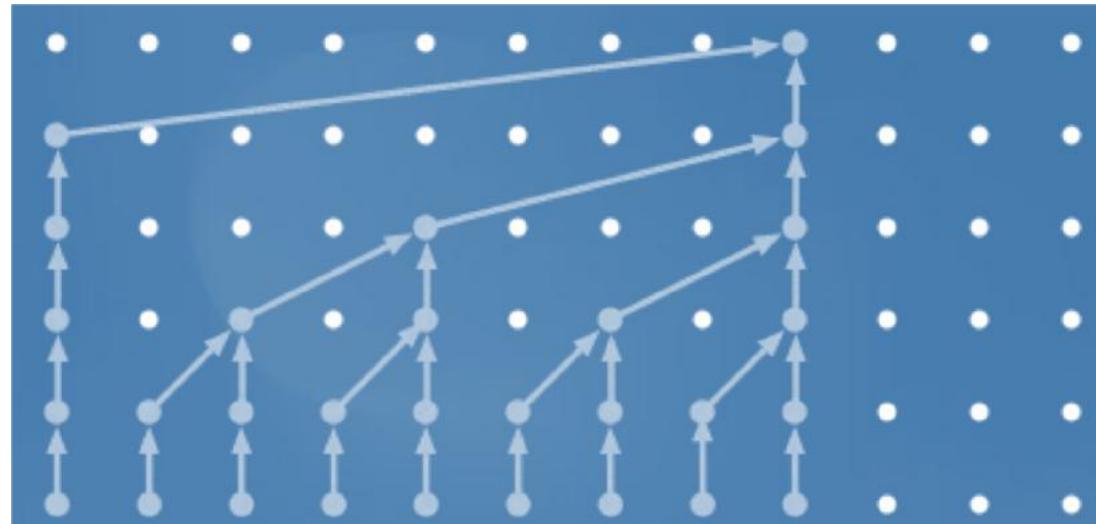


# Architecture for 1D sequences (Bytenet / Wavenet)

Deep RNN



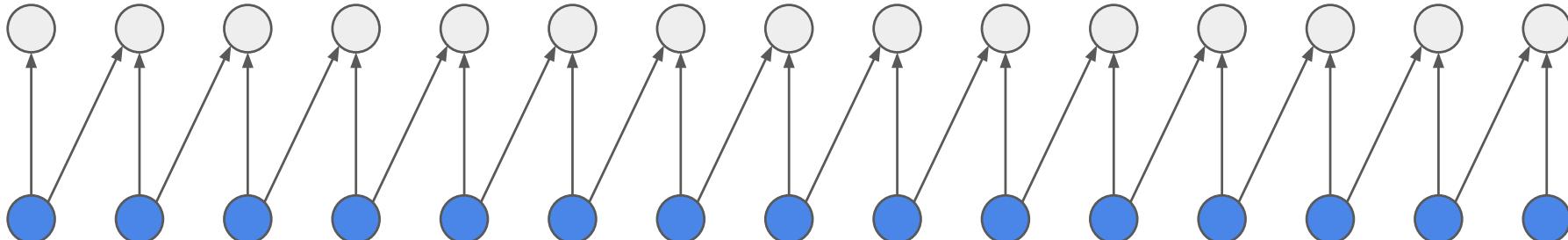
Bytenet decoder



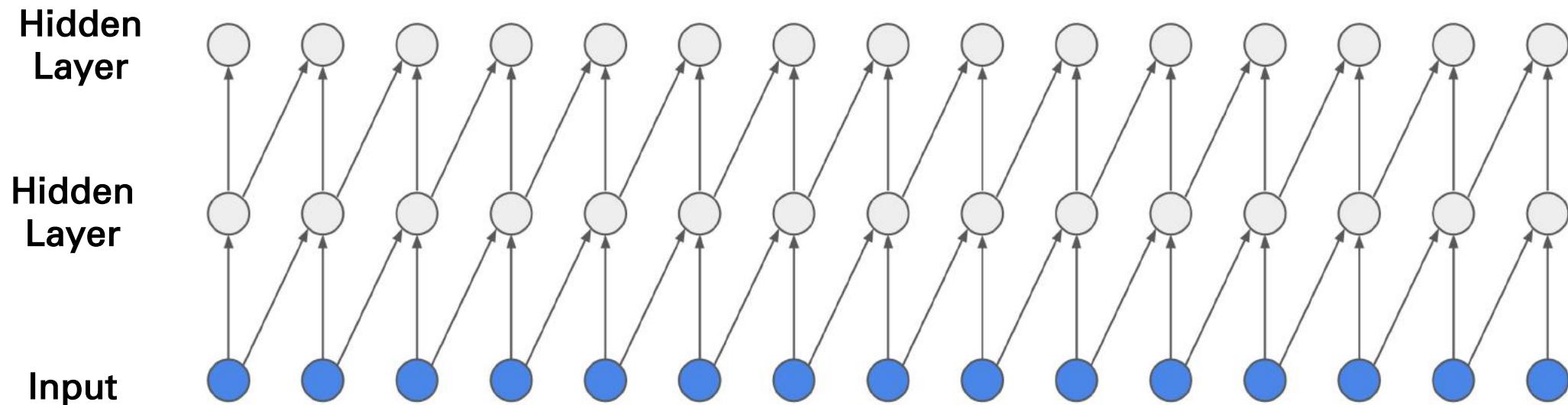
- Stack of **dilated, masked 1-D convolutions** in the decoder
- The architecture is **parallelizable** along the time dimension (during training or scoring)
- Easy access to **many states** from the past

# Causal Convolution

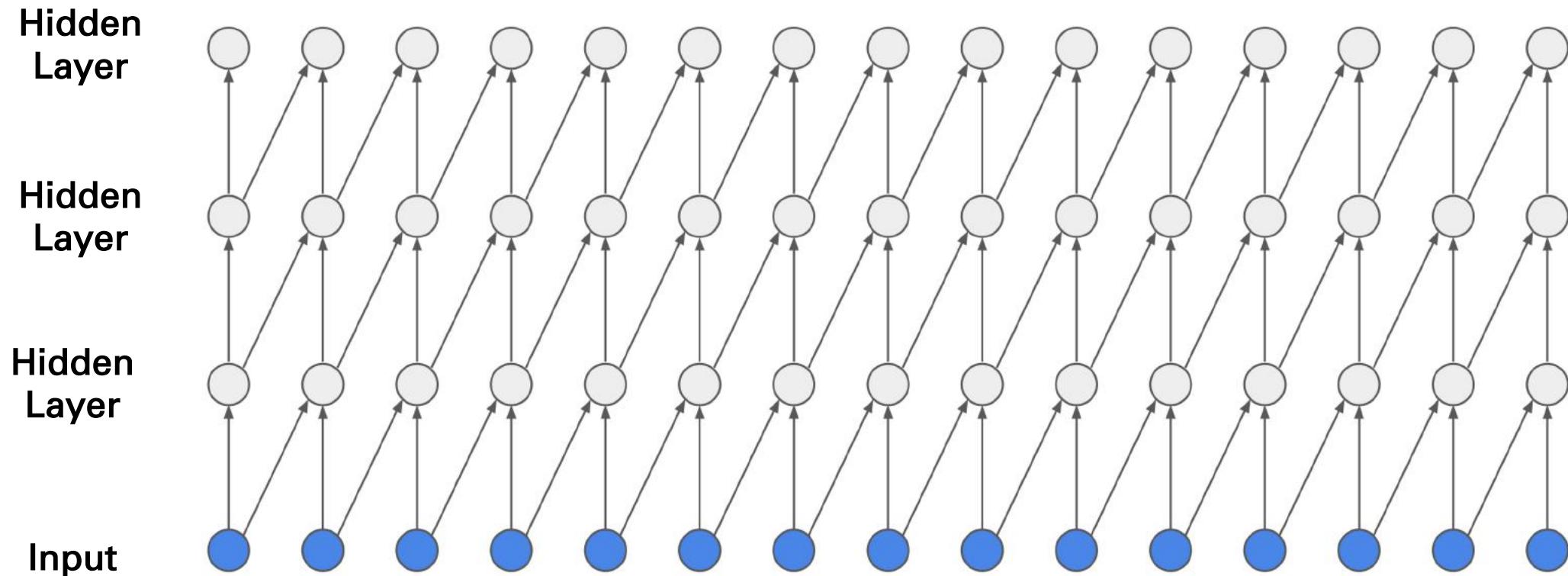
Hidden Layer  
Input



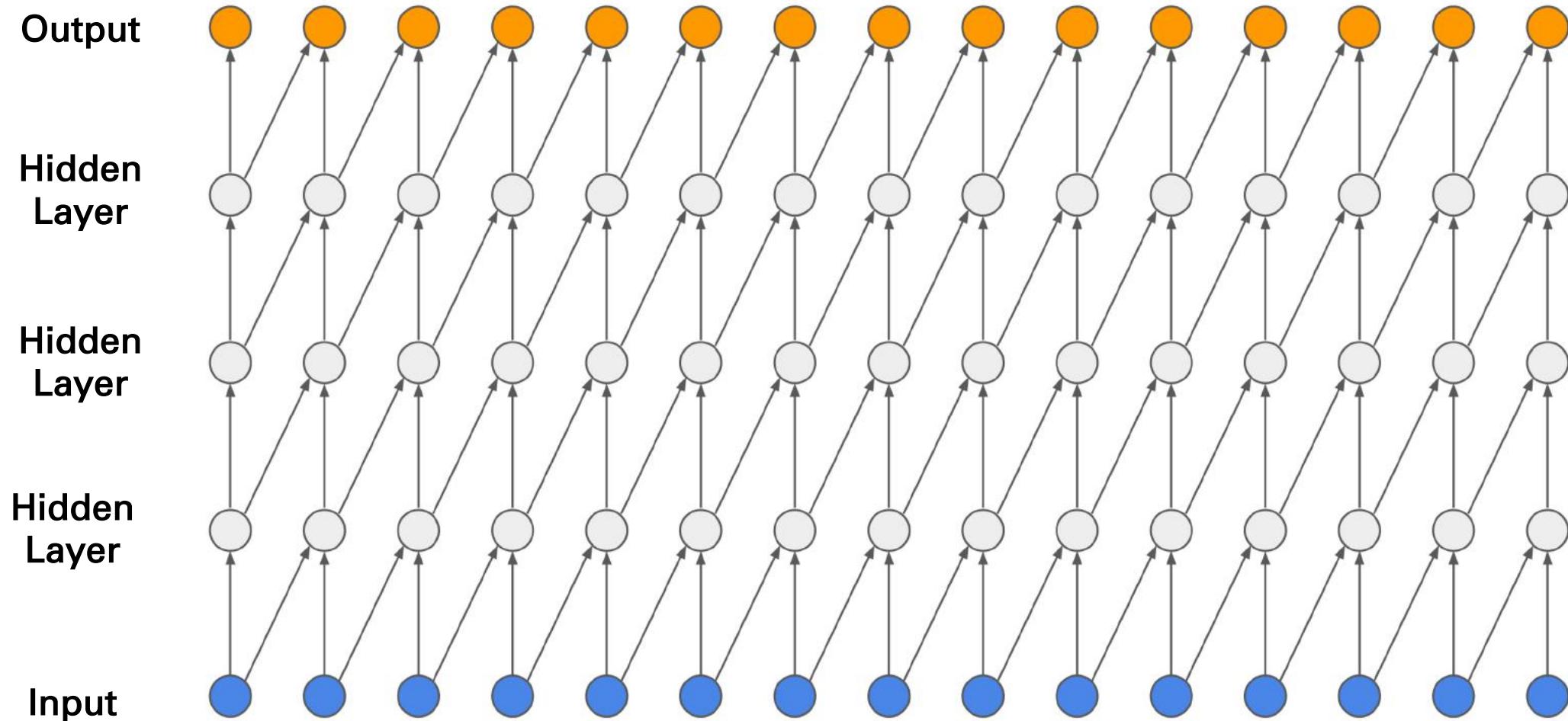
# Causal Convolution



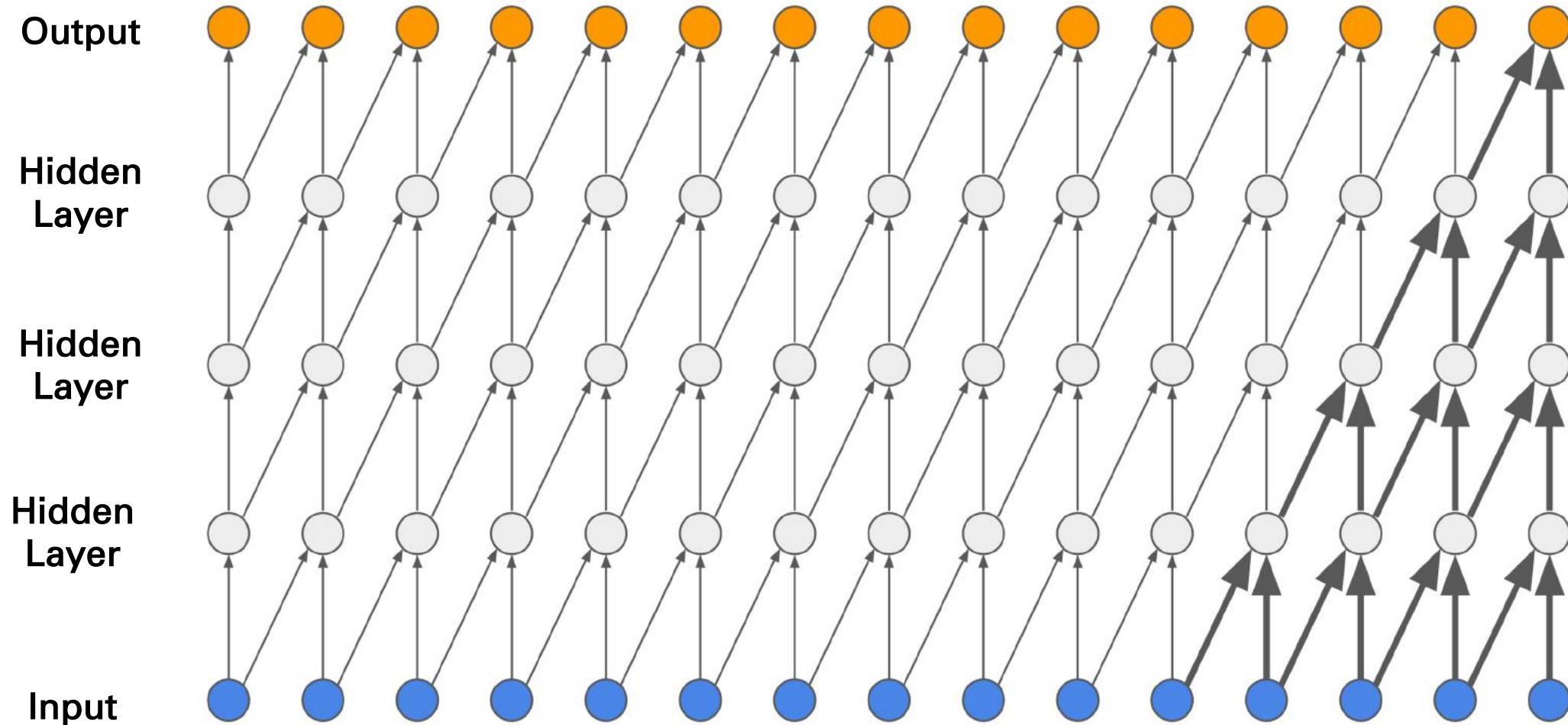
# Causal Convolution



# Causal Convolution



# Causal Convolution

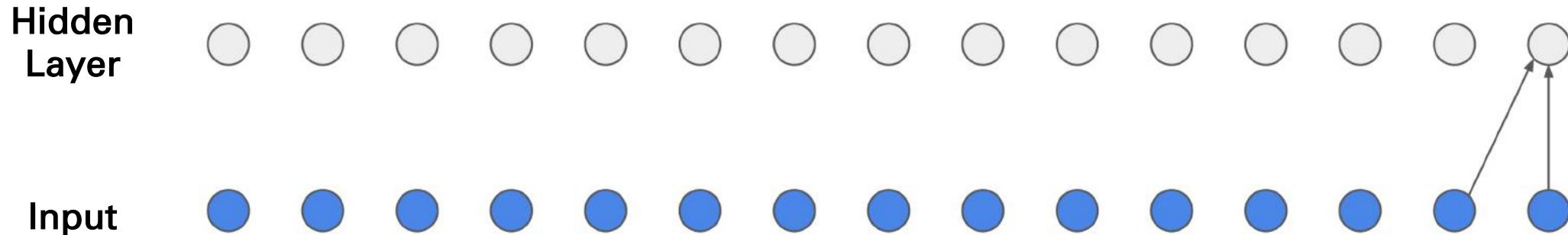


# Causal Dilated Convolution

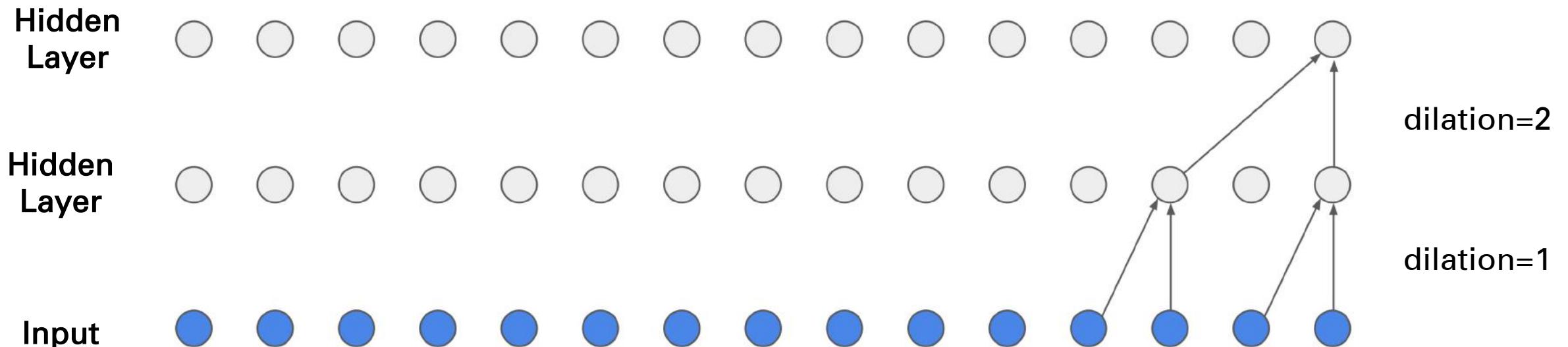
Input



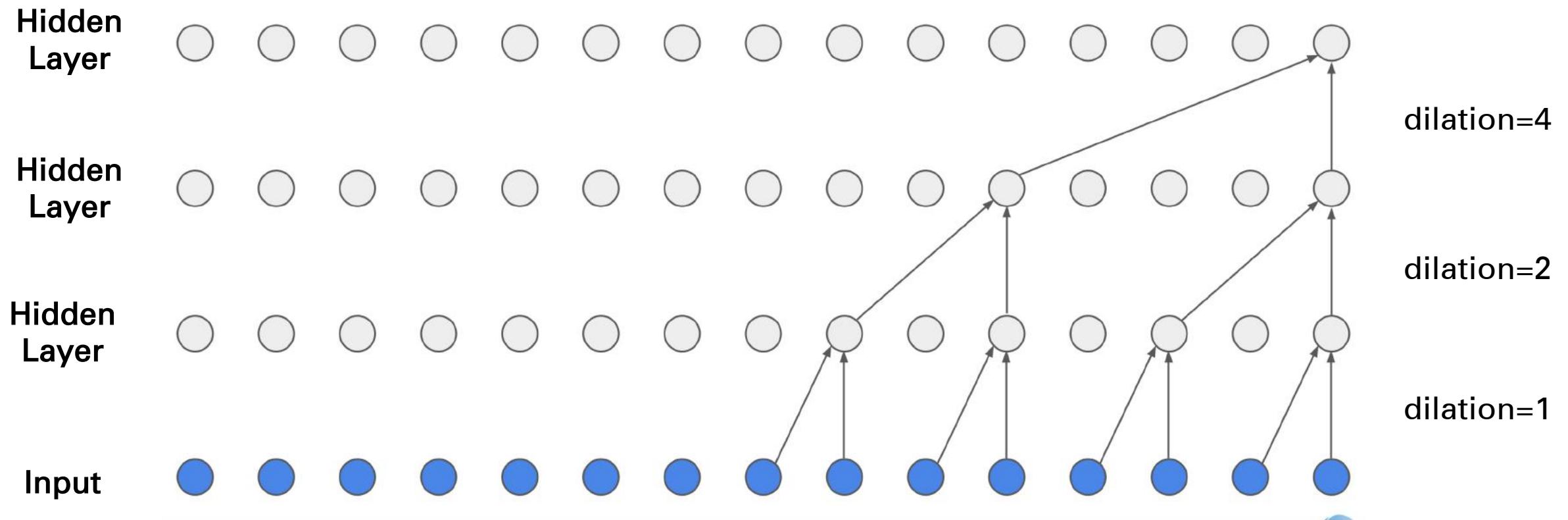
# Causal Dilated Convolution



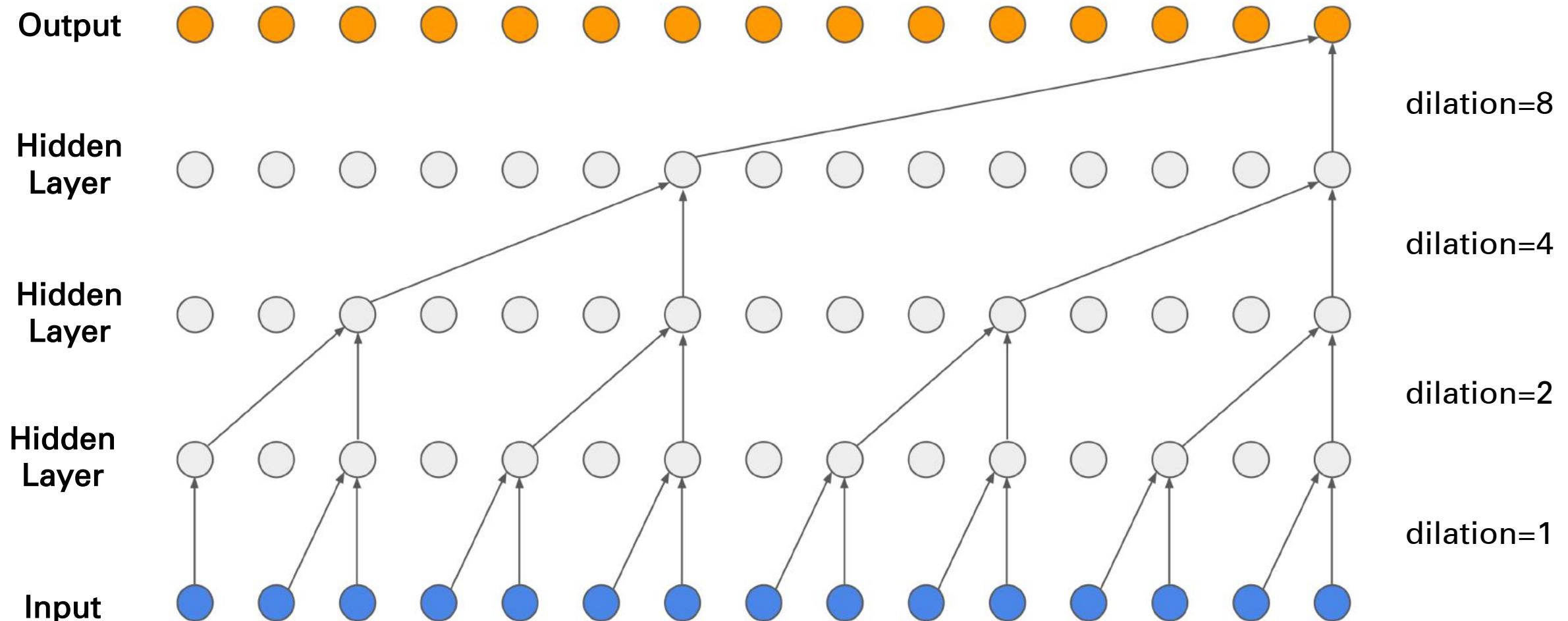
# Causal Dilated Convolution



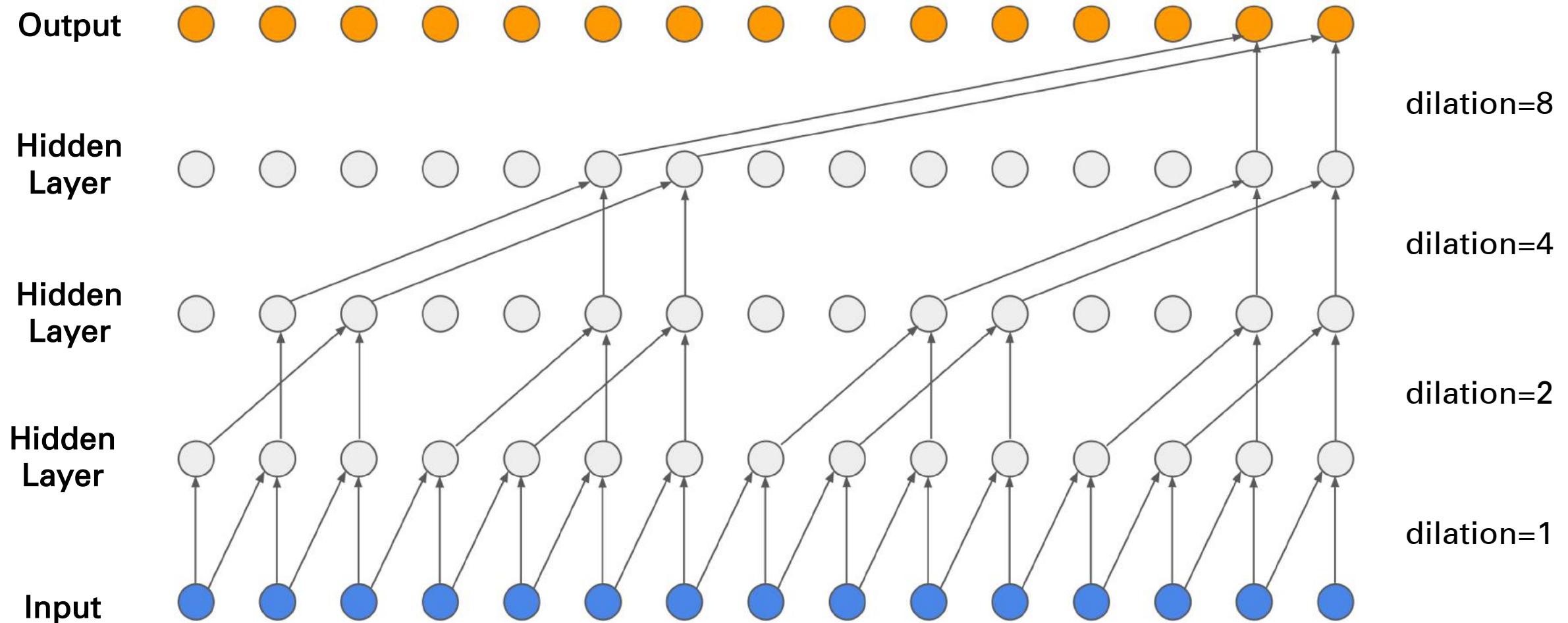
# Causal Dilated Convolution



# Causal Dilated Convolution

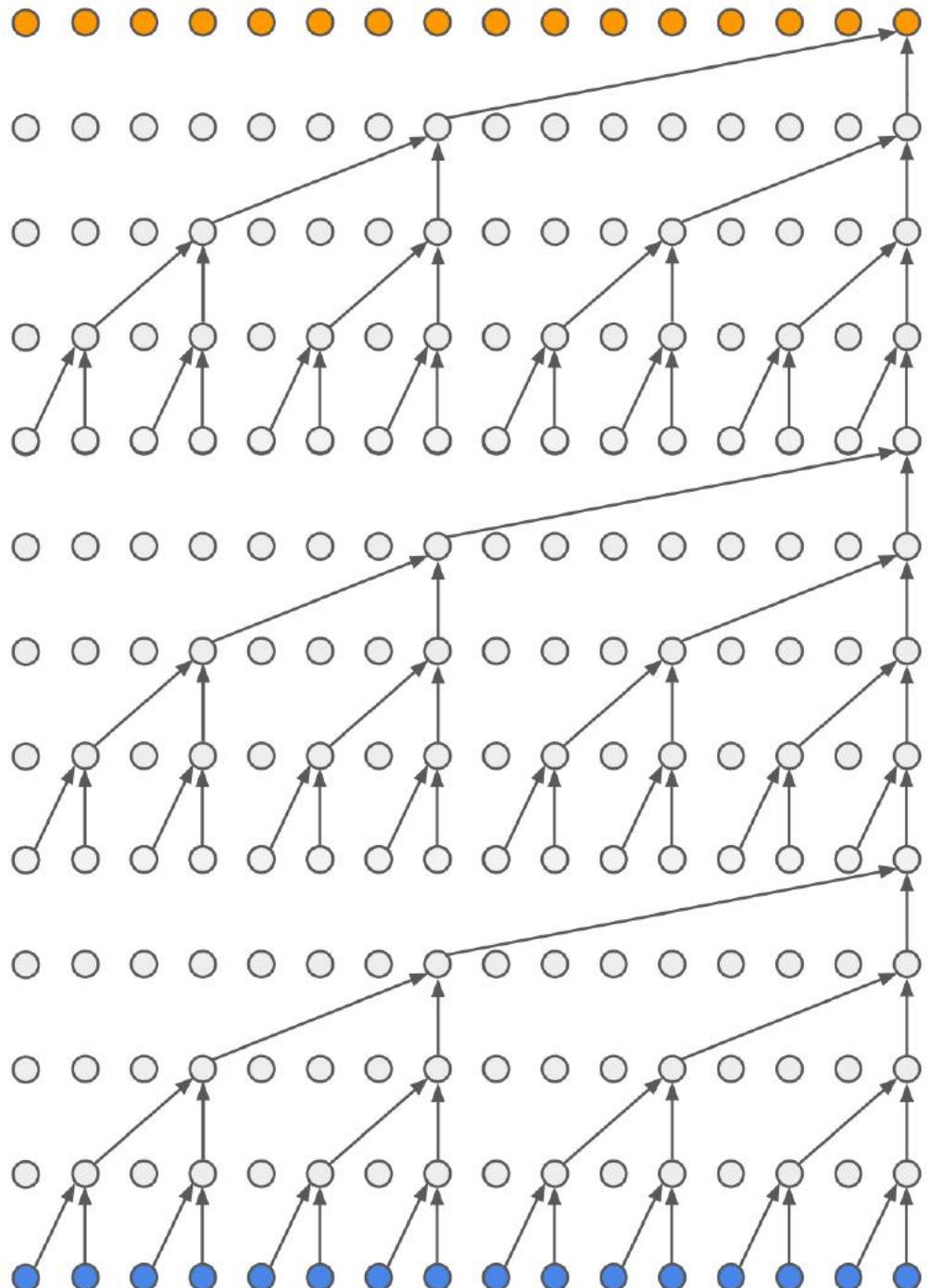
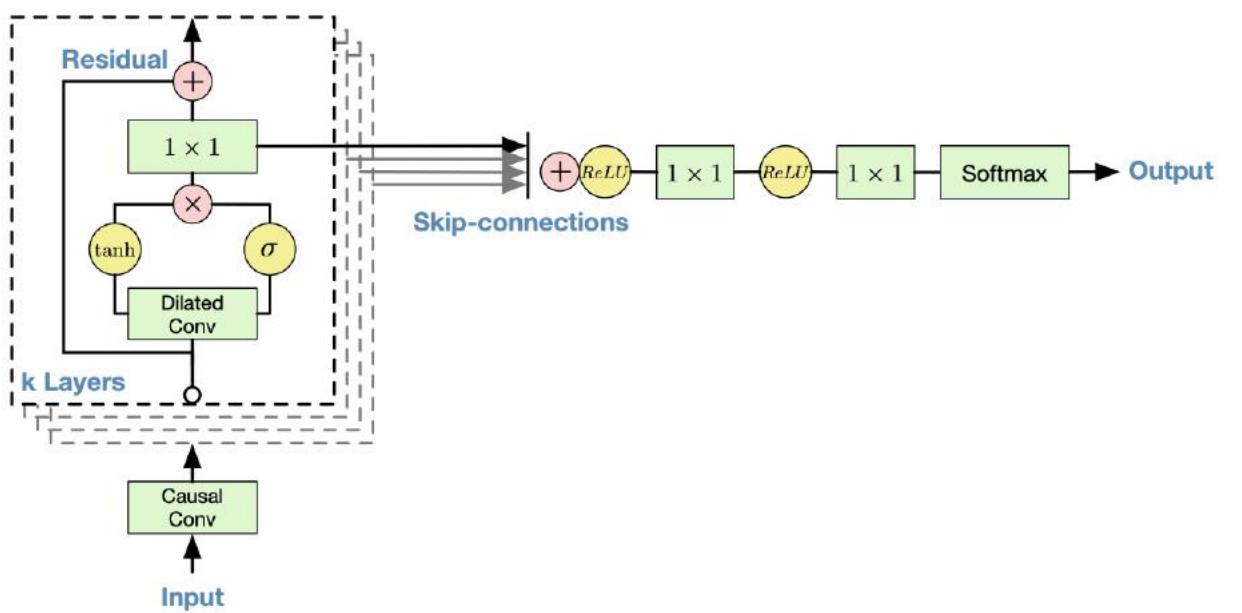


# Causal Dilated Convolution



# Multiple Stacks

- Improved receptive field with dilated convolutions
- Gated Residual block with skip connections



# Sampling

Output



Hidden  
Layer



Hidden  
Layer



Hidden  
Layer



Input

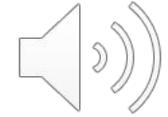


# Sampling

sample  
speech



sample  
music



Output



Hidden  
Layer



Hidden  
Layer



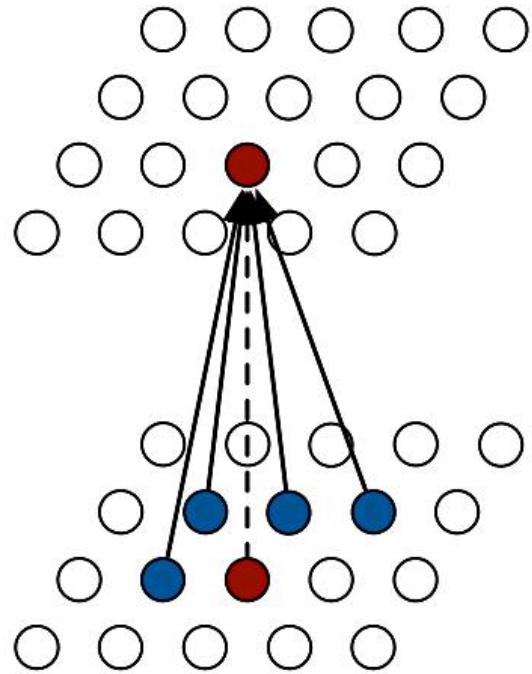
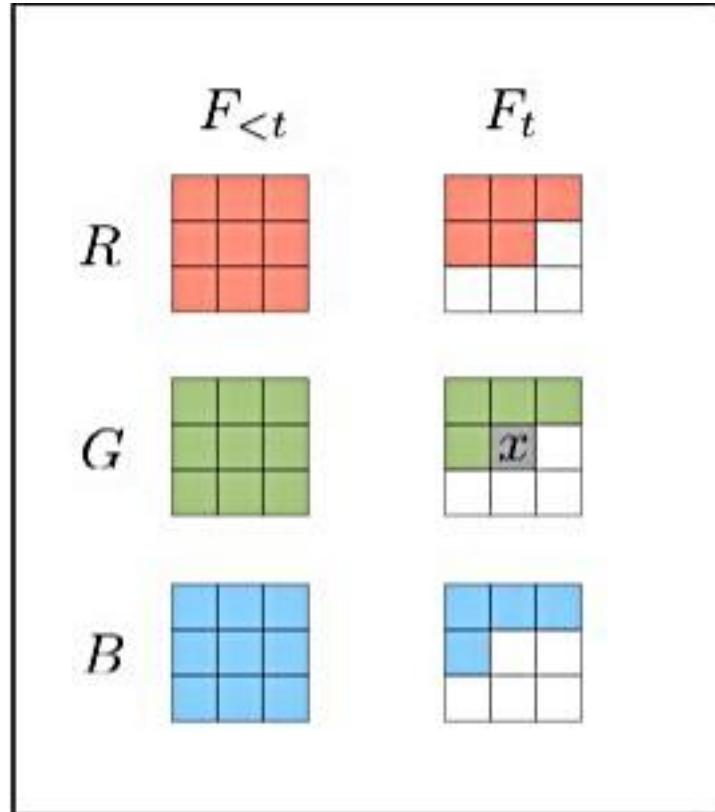
Hidden  
Layer



Input



# Video Pixel Net (VPN)

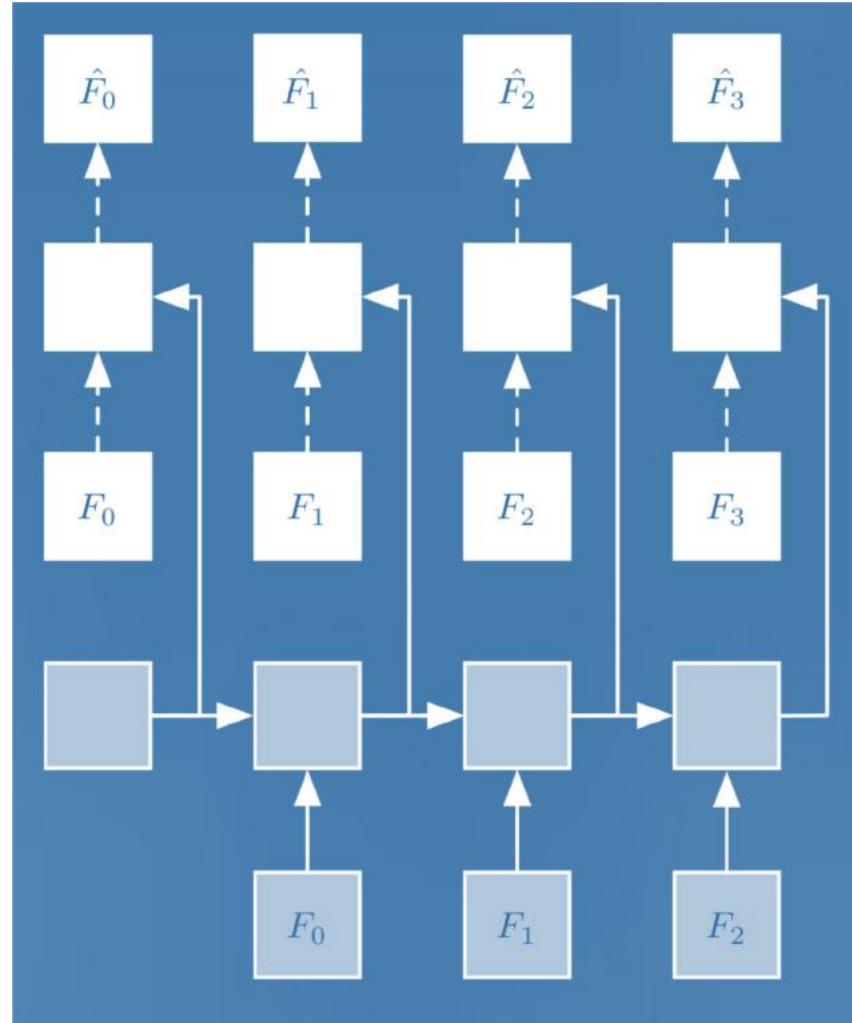


masked convolution



VPN Samples for Robotic Pushing

# Video Pixel Net (VPN)



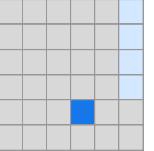
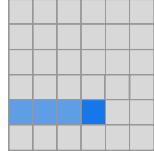
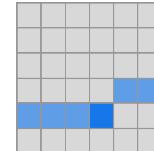
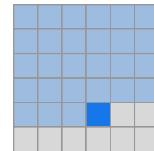
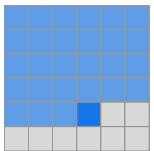
PixelCNN  
Decoders

Resolution Preserving  
CNN Encoders



VPN Samples for Robotic Pushing

# Sparse Transformers



Normal  
Transformer

Sparse  
Transformer  
(strided)

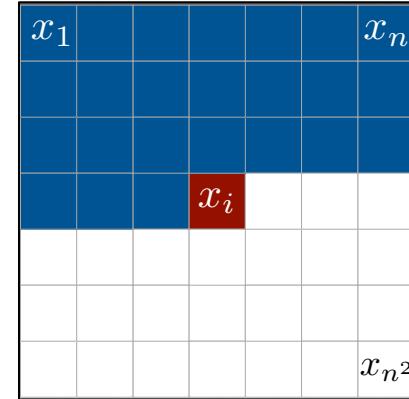
Sparse  
Transformer  
(fixed)

- Strided attention is roughly equivalent to each position attending to its row and its column
- Fixed attention attends to a fixed column and the elements after the latest column element (especially used for text).

# Autoregressive Models

- Explicitly model conditional probabilities:

$$p_{\text{model}}(\mathbf{x}) = p_{\text{model}}(x_1) \prod_{i=2}^n p_{\text{model}}(x_i \mid x_1, \dots, x_{i-1})$$



*Each conditional can be  
a complicated neural net*

## Advantages:

- $p_{\text{model}}(x)$  is tractable (easy to train and sample)

## Disadvantages:

- Generation can be too costly
- Generation can not be controlled by a latent code



PixelCNN elephants  
(van den Ord et al. 2016)

# **Next Lecture:**

# Generative Adversarial Networks and Flow Models