CS 4803 / 7643: Deep Learning

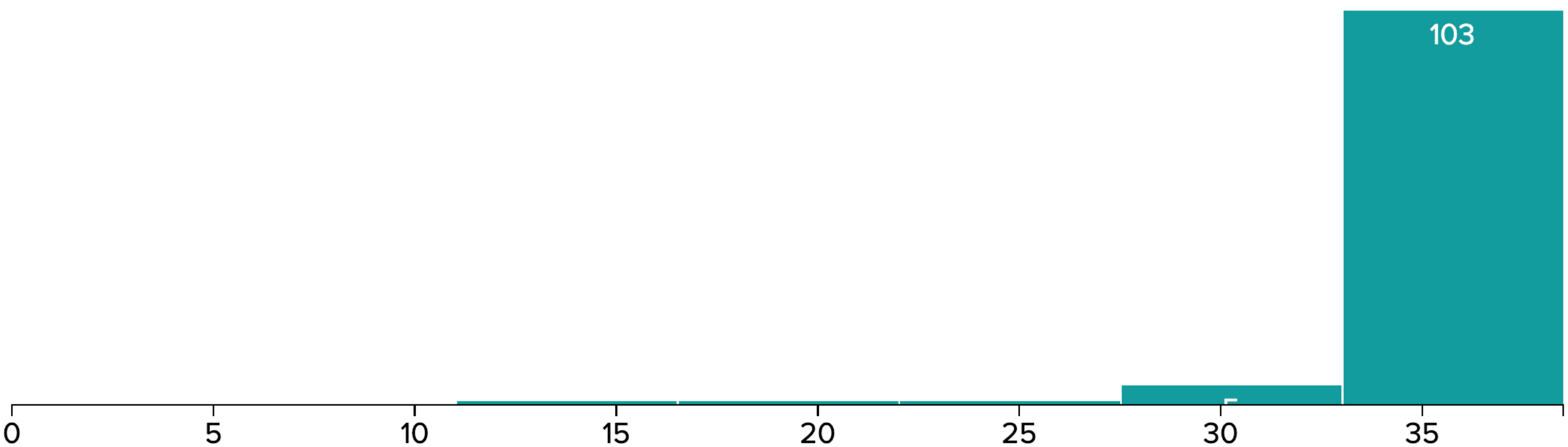
Topics:

- Unsupervised Learning
- Generative Models (PixelRNNs, VAEs)

Dhruv Batra
Georgia Tech

Administrativia

- HW4 Grades Released
  - Regrade requests close: 11/09, 11:59pm
- Grade histogram: 7643
  - Max possible: 38.5 (regular credit) + 6.5 (extra credit)

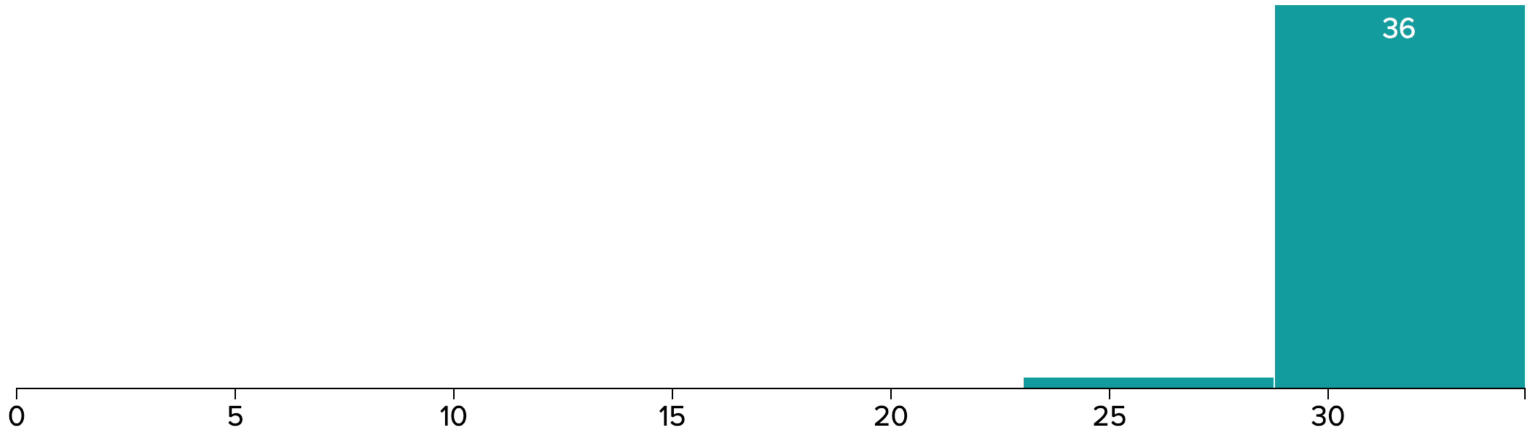


MINIMUM	MEDIAN	MAXIMUM	MEAN	STD DEV
11.0	39.5	45.0	39.26	5.12

Administrativia

- HW3 Grades Released
  - Regrade requests close: 11/09, 11:59pm

- Grade histogram: 4803
  - Max possible: 34.5 (regular) + 10.5 (extra credit)



MINIMUM	MEDIAN	MAXIMUM	MEAN	STD DEV
23.25	37.5	44.0	36.9	3.51

Administrativia

- Project submission instructions
  - Due: 11/24, 11:59pm
  - Last deliverable in the class
  - Can't use late days
  - https://www.cc.gatech.edu/classes/AY2021/cs7643_fall/

Administrativia

- Guest Lecture: Emily Denton (Google AI)
  - Next class (11/10)
  - Ethics in AI

https://cephaloponderer.com/

Overview

- Unsupervised Learning

- Generative Models
    - PixelRNN and PixelCNN
    - Variational Autoencoders (VAE)
    - Generative Adversarial Networks (GAN)

Supervised vs Reinforcement vs Unsupervised Learning

Supervised vs Reinforcement vs Unsupervised Learning

Supervised Learning

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

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

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

$x \rightarrow \quad \rightarrow y$

x → Cat
y

Classification

Supervised vs Unsupervised Learning

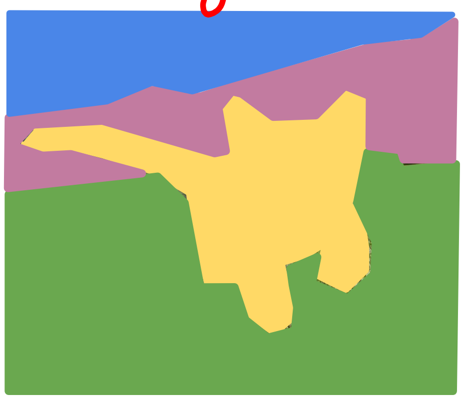
$x \rightarrow f_\theta \rightarrow y$

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

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

Caption generated using neuraltalk2
Image is CC0 Public domain.

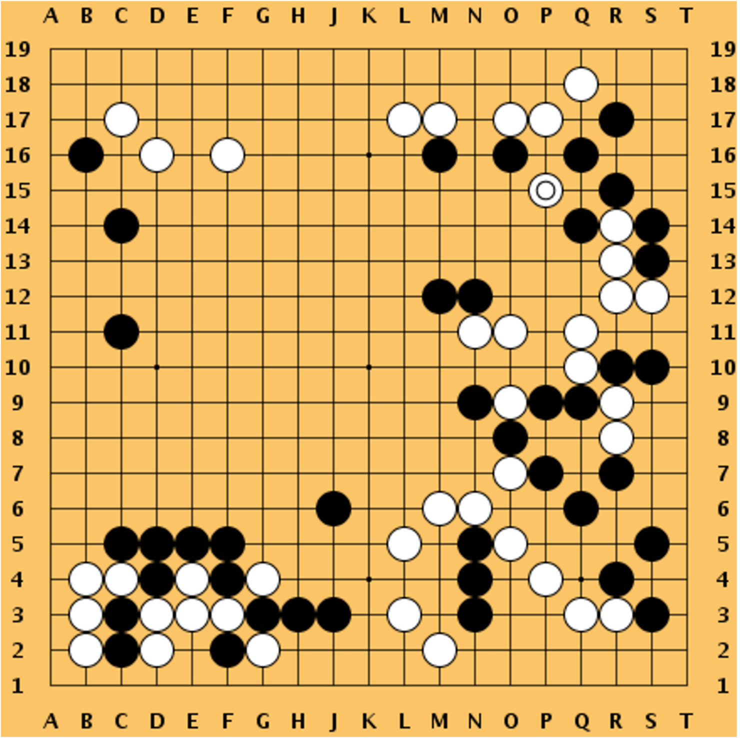
Supervised vs Reinforcement vs Unsupervised Learning

Reinforcement Learning

Given: (e, r)
Environment e, Reward function r
(evaluative feedback)

Goal: Maximize expected reward

Examples: Robotic control, video games, board games, etc.



$\pi: s_t \rightarrow a_t$

θ

$Q($

$s_t, a_t \rightarrow R$

$P(s_{t+1} | s_t, a_t)$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Supervised vs Reinforcement 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.

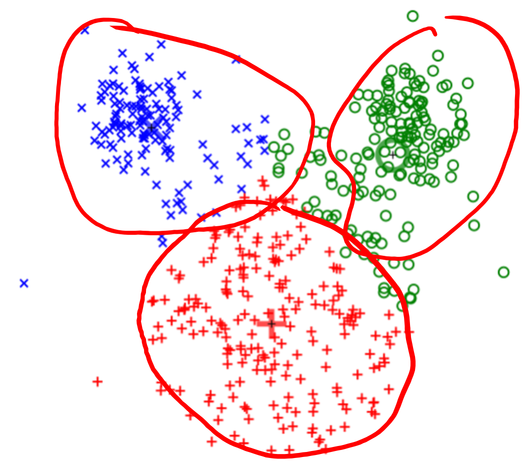
Supervised vs Reinforcement vs Unsupervised Learning

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

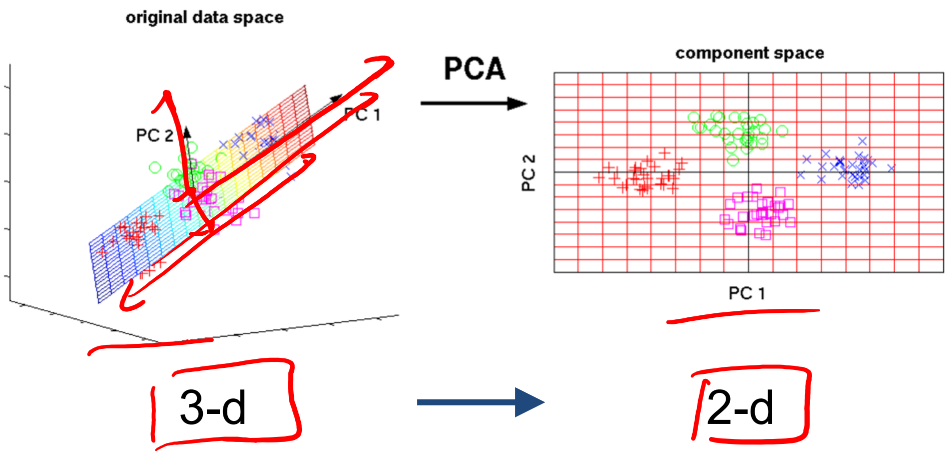
Supervised vs Reinforcement vs Unsupervised Learning

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.



Principal Component Analysis
(Dimensionality reduction)

Supervised vs Reinforcement vs Unsupervised Learning

$x \rightarrow p(x)$

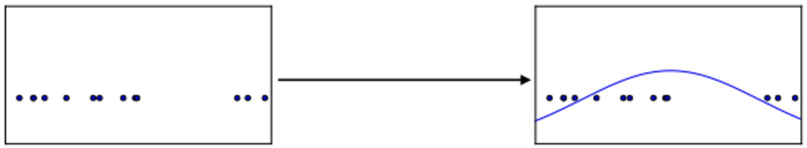
$p(x)$

Unsupervised Learning

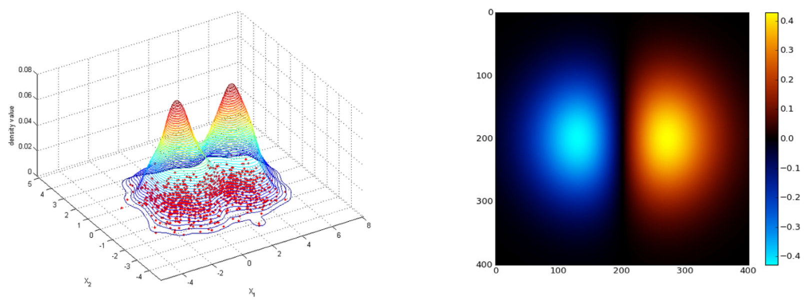
Data: x
Just data, no labels!

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

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



1-d density estimation



2-d density estimation

$\vec{x} = [x_1 \; \ldots \; x_d]$

$p(x_i \mid x_j, \ldots, x_{j+n})$

Tasks

Supervised Learning

Input	Task	Output	Type
x	Classification	y	Discrete
x	Regression	y	Continuous

Unsupervised Learning

Input	Task	Output	Type
x	Clustering	c	Discrete
x	Dimensionality Reduction	z	Continuous
x	Density Estimation	p(x)	On simplex if x discrete

Supervised vs Reinforcement 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.

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.

Supervised vs Reinforcement vs Unsupervised Learning

Unsupervised Learning

Training data is cheap

Data: x
Just data, no labels!

Holy grail: Solve unsupervised learning => understand structure of visual world

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

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

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.

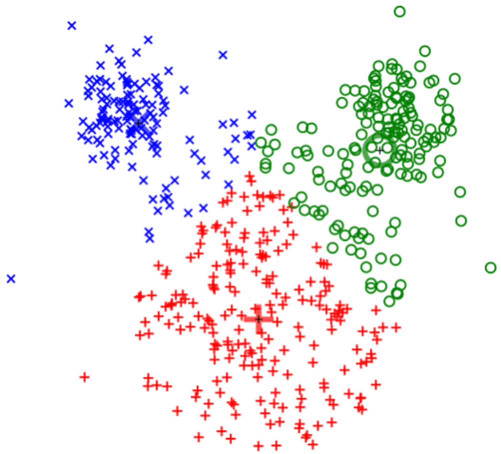
Supervised vs Reinforcement vs Unsupervised Learning

Unsupervised Learning

Data: x
Just data, no labels!

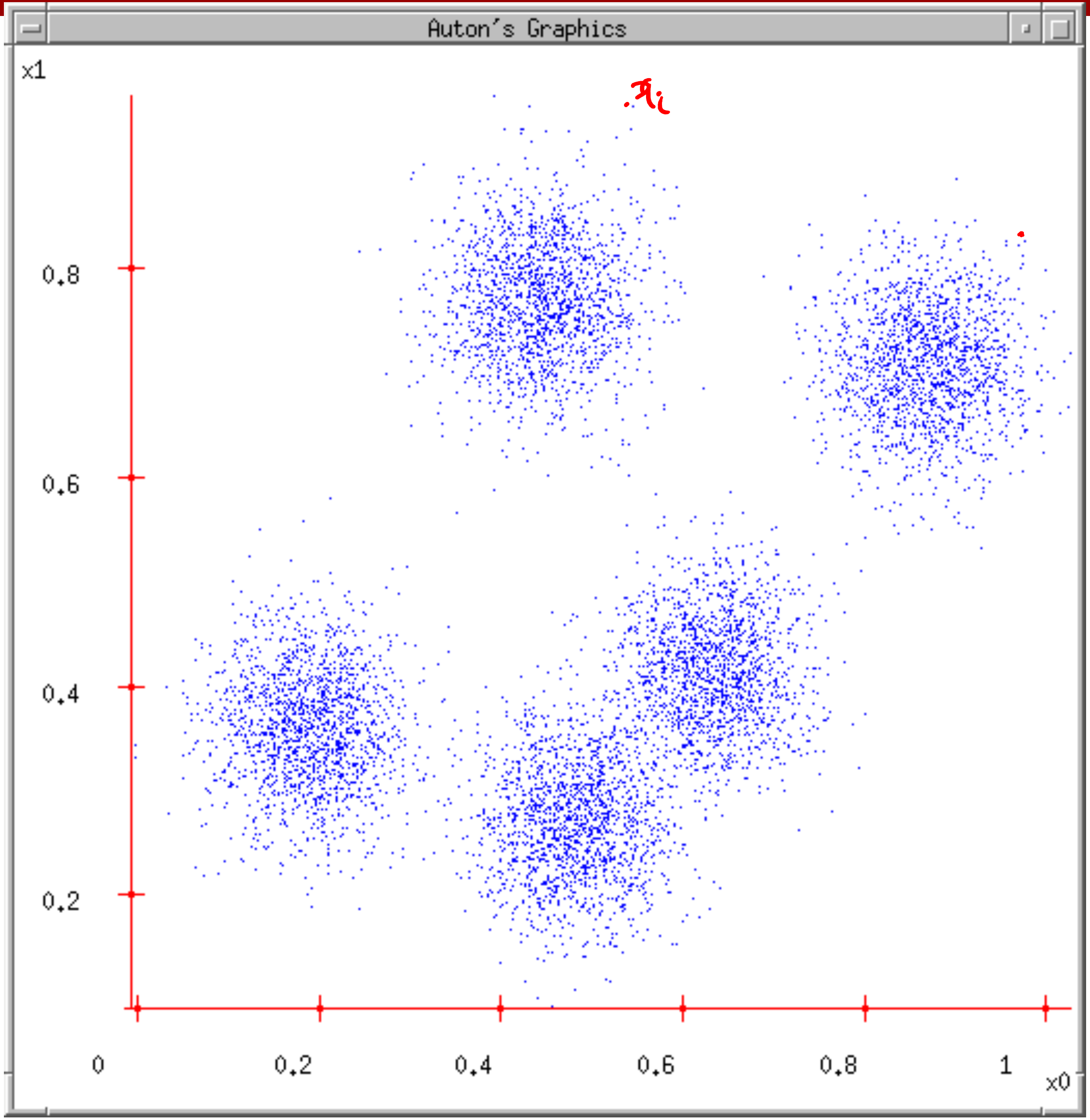
Goal: Learn some underlying hidden *structure* of the data

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



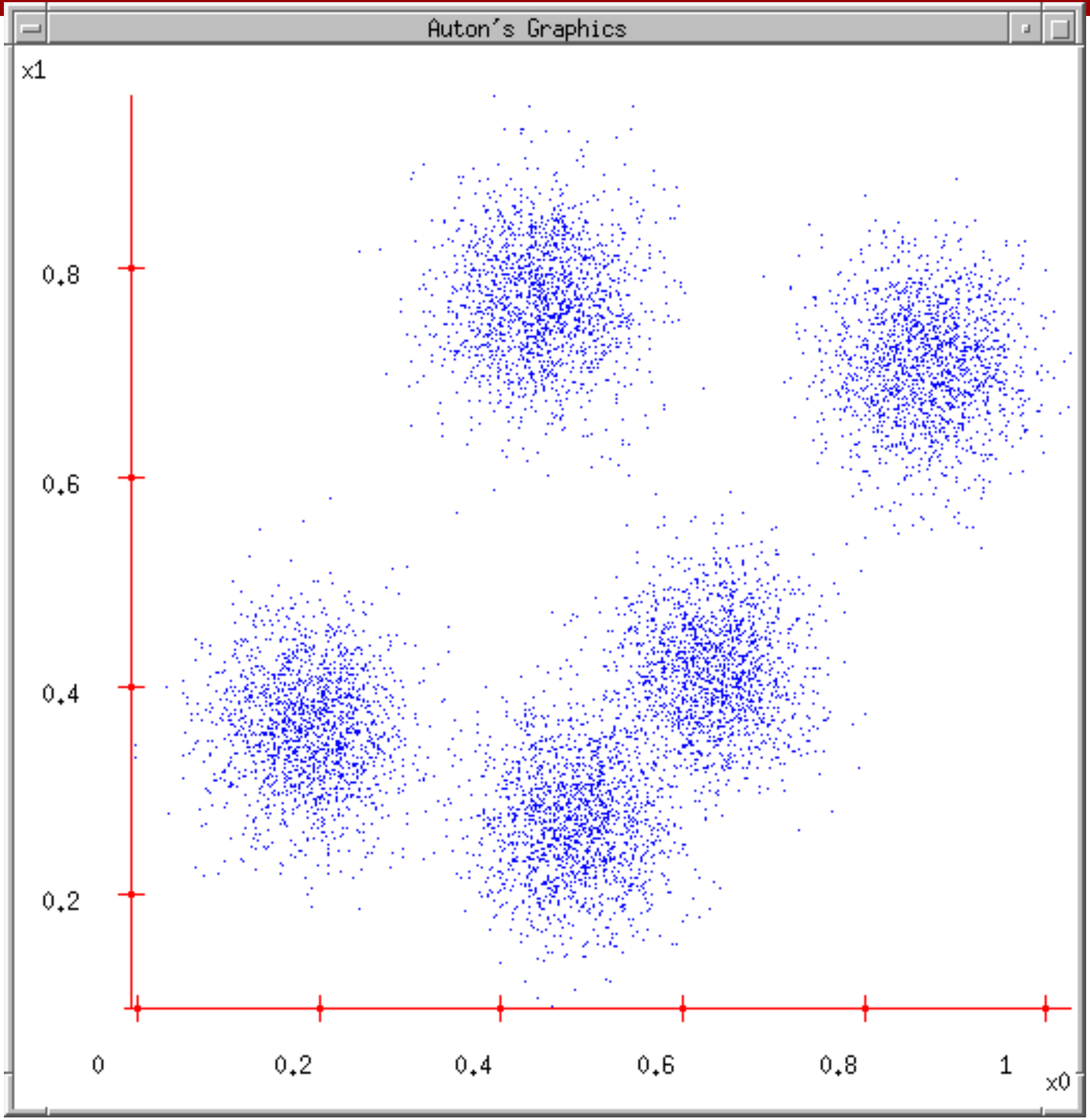
K-means clustering

Some Data



K-means

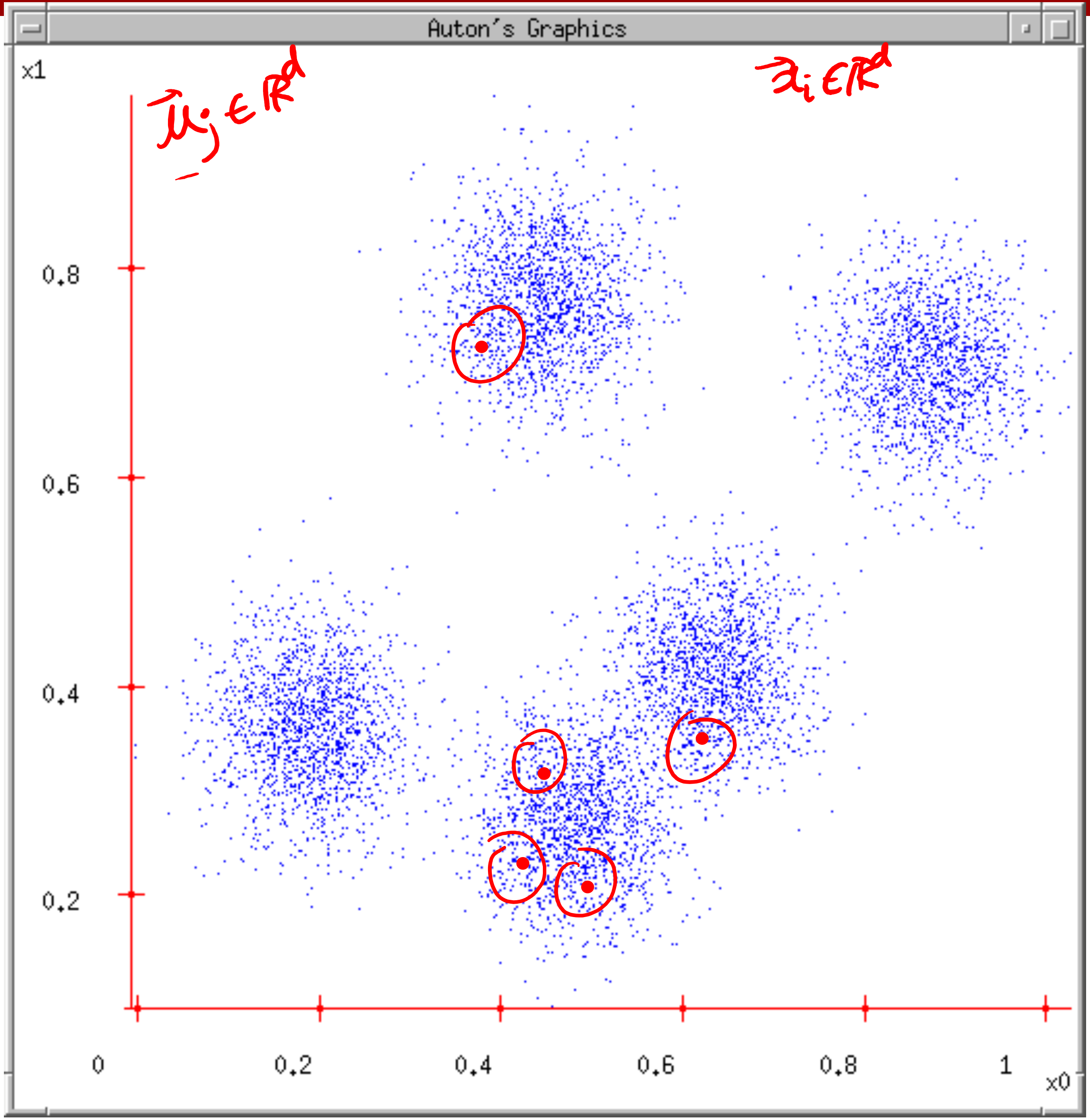
1. Ask user how many clusters they'd like.
   (e.g. k=5)



K-means

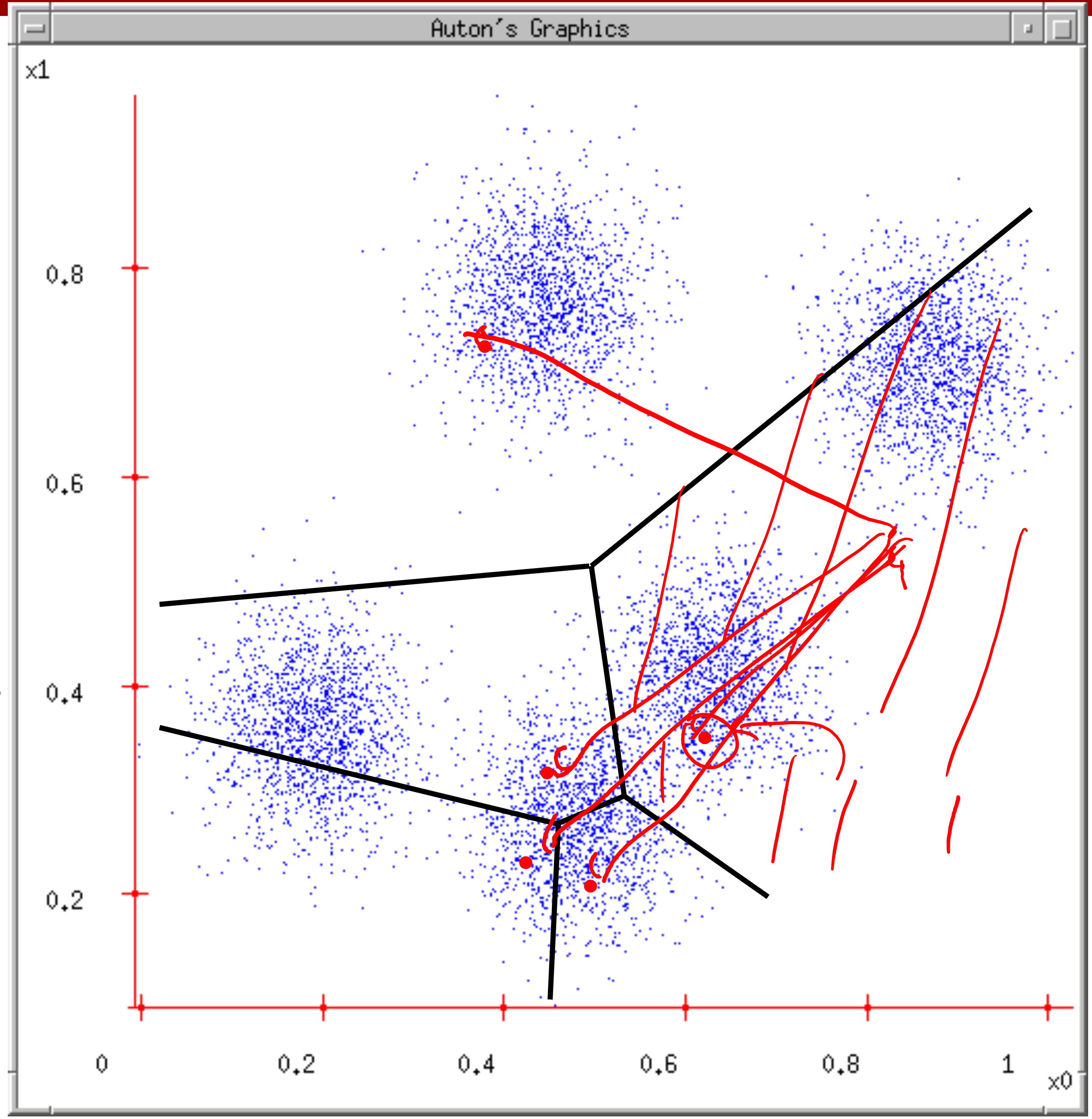
1. Ask user how many clusters they'd like. *(e.g. k=5)*

2. Randomly guess k cluster Center locations



K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



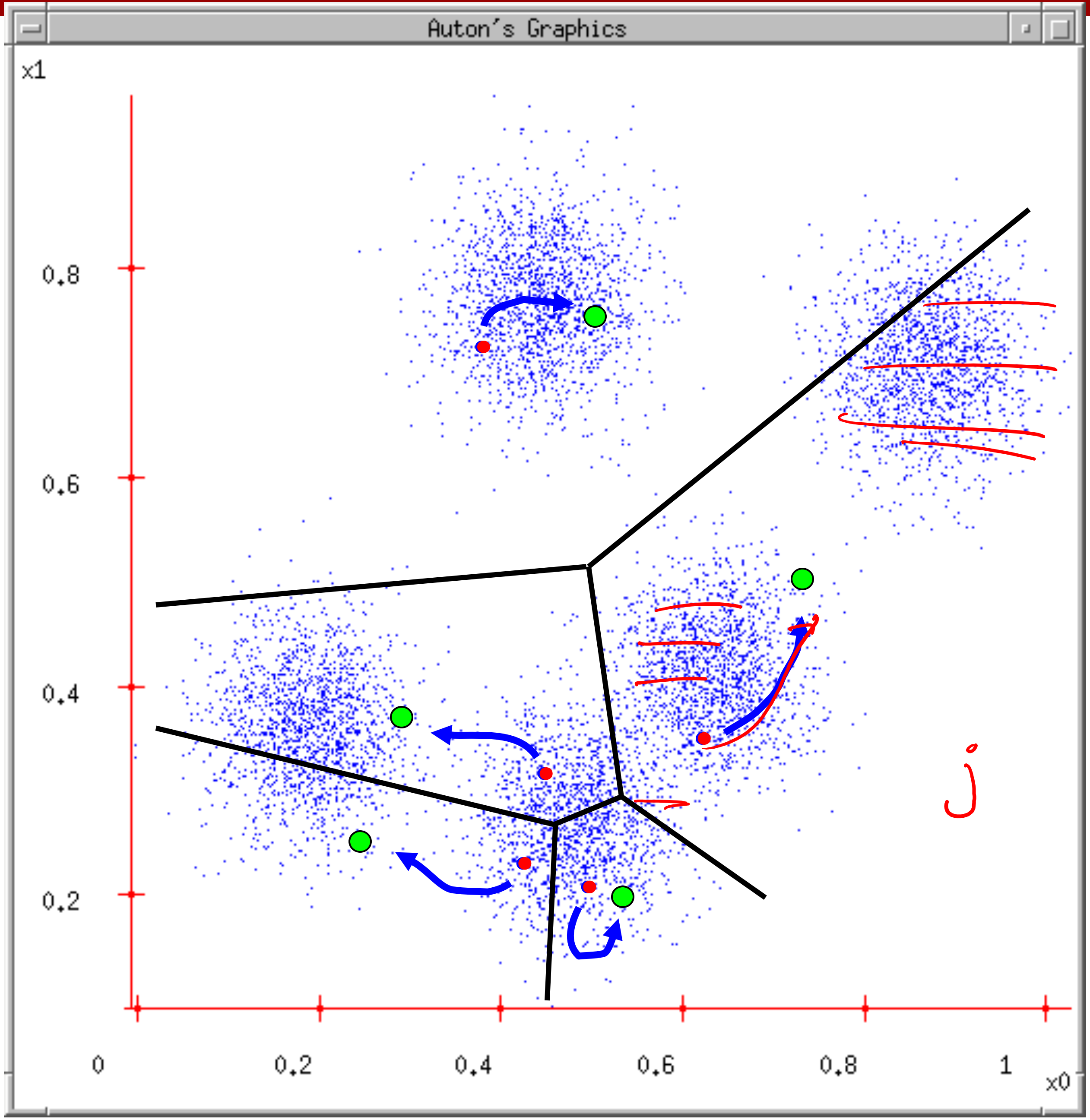
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*

2. Randomly guess k cluster Center locations

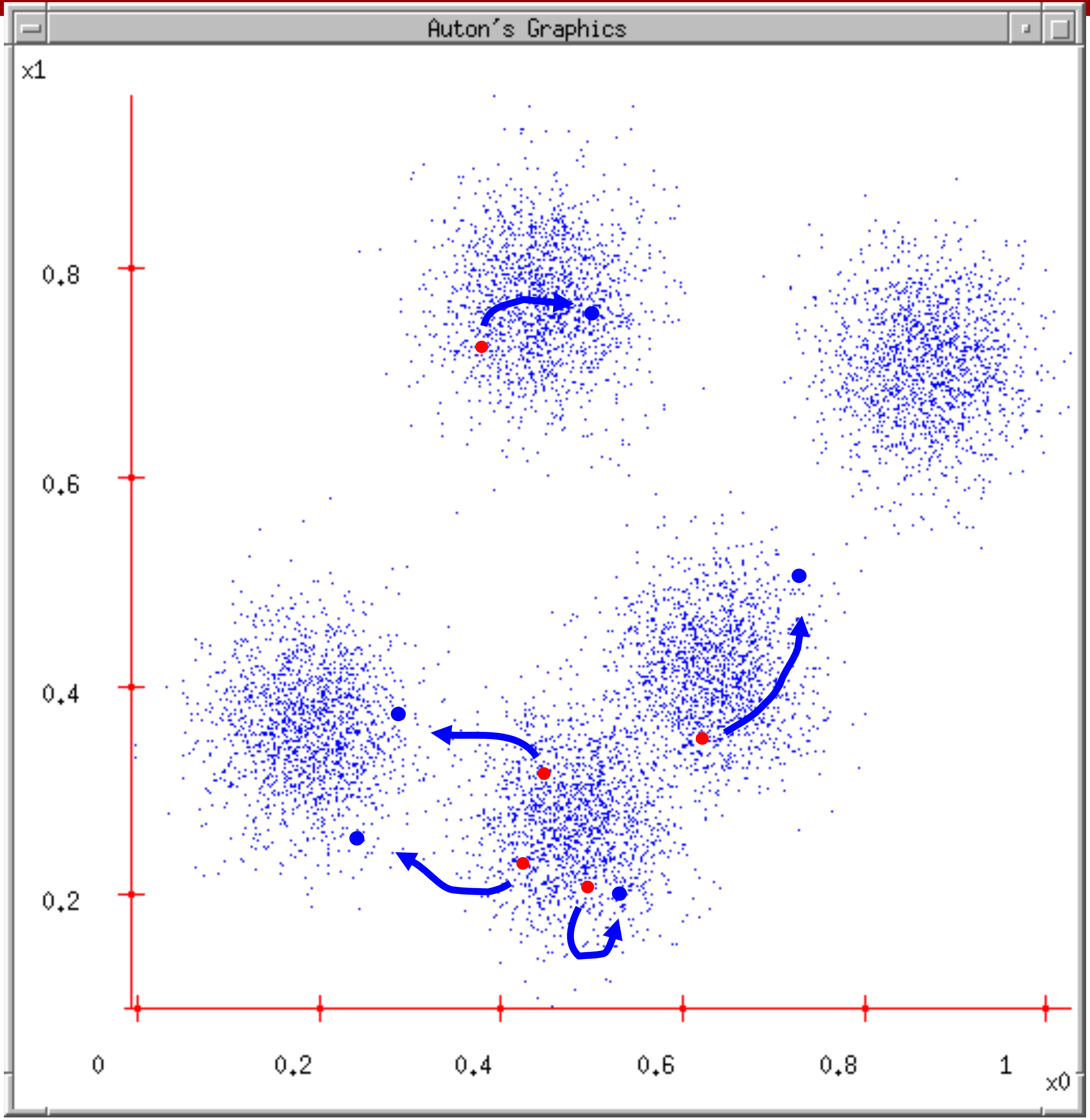
3. Each datapoint finds out which Center it's closest to.

4. Each Center finds the centroid of the points it owns



K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!



K-means

- Randomly initialize *k* centers
  - $\mu^{(0)} = \mu_1^{(0)}, \ldots, \mu_k^{(0)}$ $\mu_j \in \mathbb{R}^d$
- **Assign**:
  - Assign each point $i \in \{1, \ldots n\}$ to nearest center:
  - $\boxed{C(i)} \longleftarrow \underset{j}{\operatorname{argmin}} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2$ $d(x_i, \mu_j)$
- **Recenter**:
  - μ_j becomes $\boxed{\text{centroid}}$ of points assigned to cluster j

K-means

- Demo
  - http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html

What is K-means optimizing?

- Objective $\boxed{F(\mu, C)}$: function of centers μ and point allocations C:
  - $F(\boldsymbol{\mu}, C) = \boxed{\sum_{i=1}^{N}} \underbrace{||\mathbf{x}_i - \boldsymbol{\mu}_{C(i)}||^2}_{d(x_i, \mu_j) \text{ if } C_i = j}$
  - 1-of-k encoding $\boxed{F(\boldsymbol{\mu}, \boldsymbol{a})} = \underbrace{\sum_{i=1}^{N}\sum_{j=1}^{k} a_{ij}\boxed{||\mathbf{x}_i - \boldsymbol{\mu}_j||^2}}$

$C(i) \in \{1, \dots, k\}$

$\vec{a}_i = \begin{bmatrix} a_{i1} \\ a_{i2} \\ a_{ik} \end{bmatrix}$

$a_{ij} = 1$ iff $C(i) = j$

$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$

- Optimal K-means:
  - $\boxed{\min_{\mu} \min_{a}} F(\mu, a)$

"reconstruction"

① fix a, $\boxed{\min \vec{\mu}}$

② fix $\vec{\mu}$, min a

Supervised vs Reinforcement vs Unsupervised Learning

$\boxed{p(x)}$

Unsupervised Learning

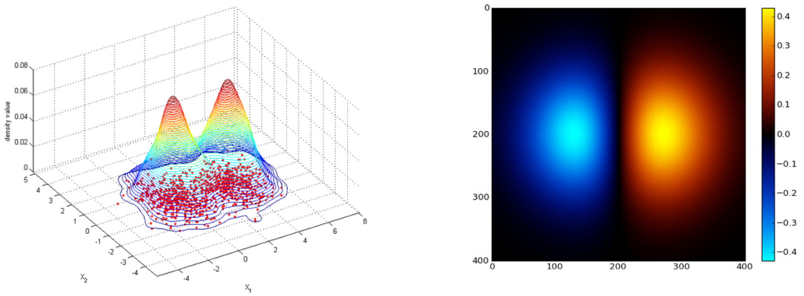
Data: x
Just data, no labels!

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-d density estimation

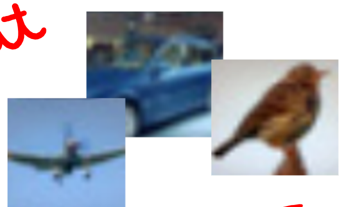


2-d density estimation

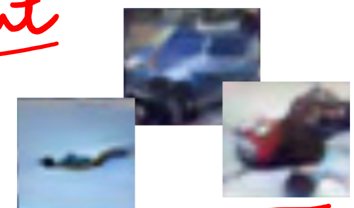
Generative Models

Given training data, generate new samples from same distribution

Input



Ouput



$x \sim P_{model}(x)$

Training data ~ $\boxed{p_{data}(x)}$

Generated samples ~ $p_{model}(x)$

$D = \{\vec{x}_1, \ldots, \vec{x}_n\}$ Want to learn $\boxed{p_{model}(x)}$ similar to $\boxed{p_{data}(x)}$ ← Density estimation

Generative Classification vs Discriminative Classification vs Density Estimation

- Generative Classification
  - Model p(x, y); estimate p(x|y) and p(y)
  - Use Bayes Rule to predict y
  - E.g Naïve Bayes

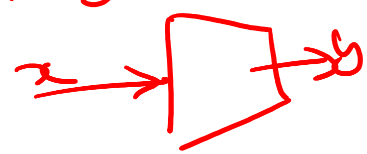
p(x

$\boxed{p(x|y)}$ $p(y)$

$$p(y|x) = \frac{p(x|y)\,p(y)}{p(x)}$$

- Discriminative Classification
  - Estimate p(y|x) directly
  - E.g. Logistic Regression

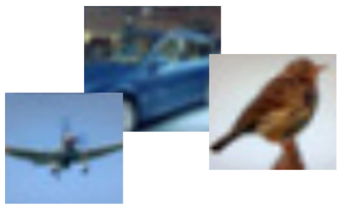
$p(y|x)$



- Density Estimation
  - Model $\boxed{p(x)}$
  - E.g. VAEs

Generative Models

Given training data, generate new samples from same distribution



Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

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

Addresses density estimation, a core problem in unsupervised learning

Several flavors:

- Explicit density estimation: explicitly define and solve for $\boxed{p_{model}(x)}$
- Implicit density estimation: learn model that can sample from $p_{model}(x)$ w/o explicitly defining it

$$\boxed{\underline{x} \sim P_{model}(x)}$$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Why Generative Models?

$\boxed{p(x)}$ $p(x|y)$ $p(x|x')$

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



- Generative models of time-series data can be used for simulation and planning (reinforcement learning applications!)

$p(z)$

- Training generative models can also enable inference of latent representations that can be useful as general features

Taxonomy of Generative Models

black box

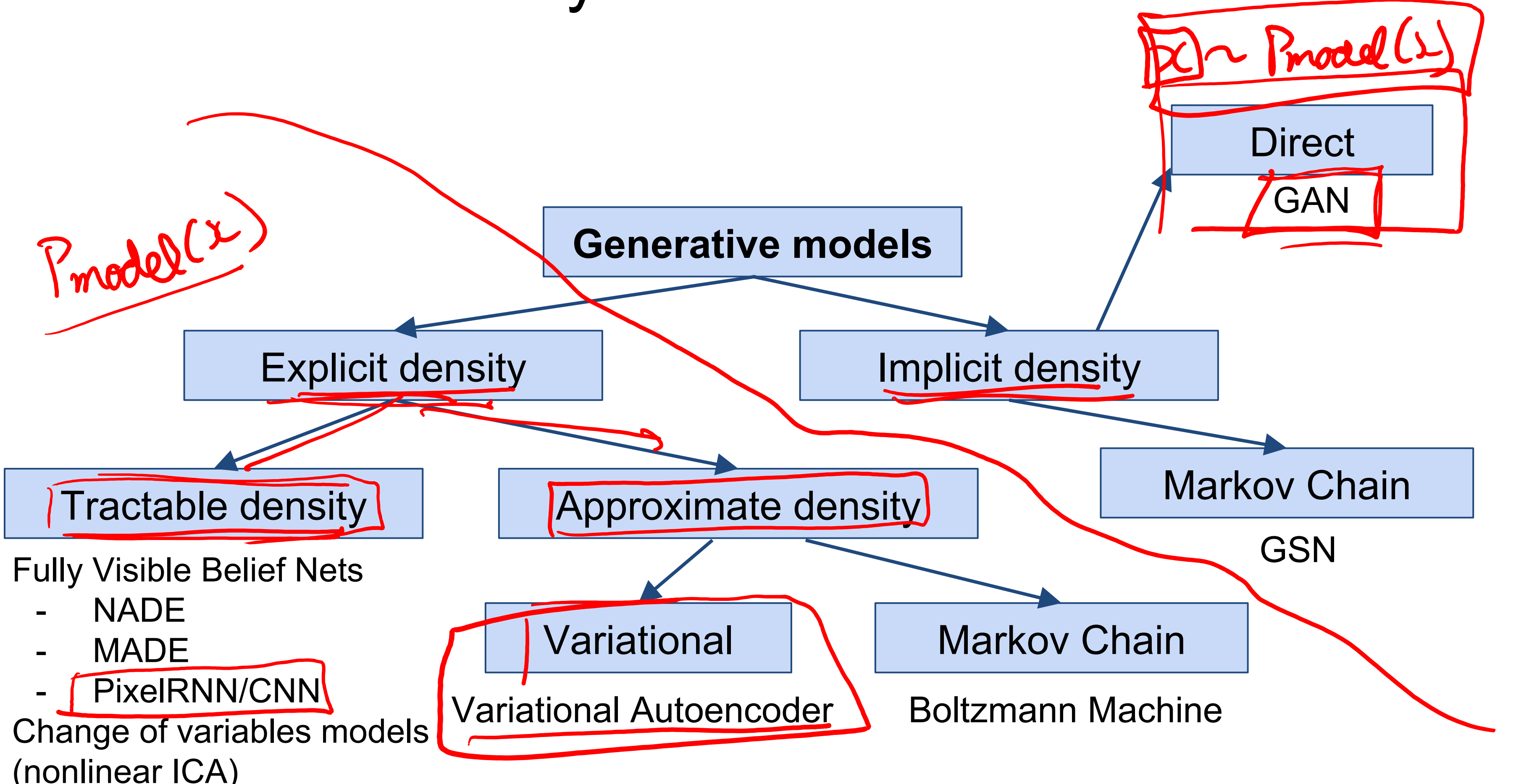


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Taxonomy of Generative Models

We will discuss 3 most popular types of generative models

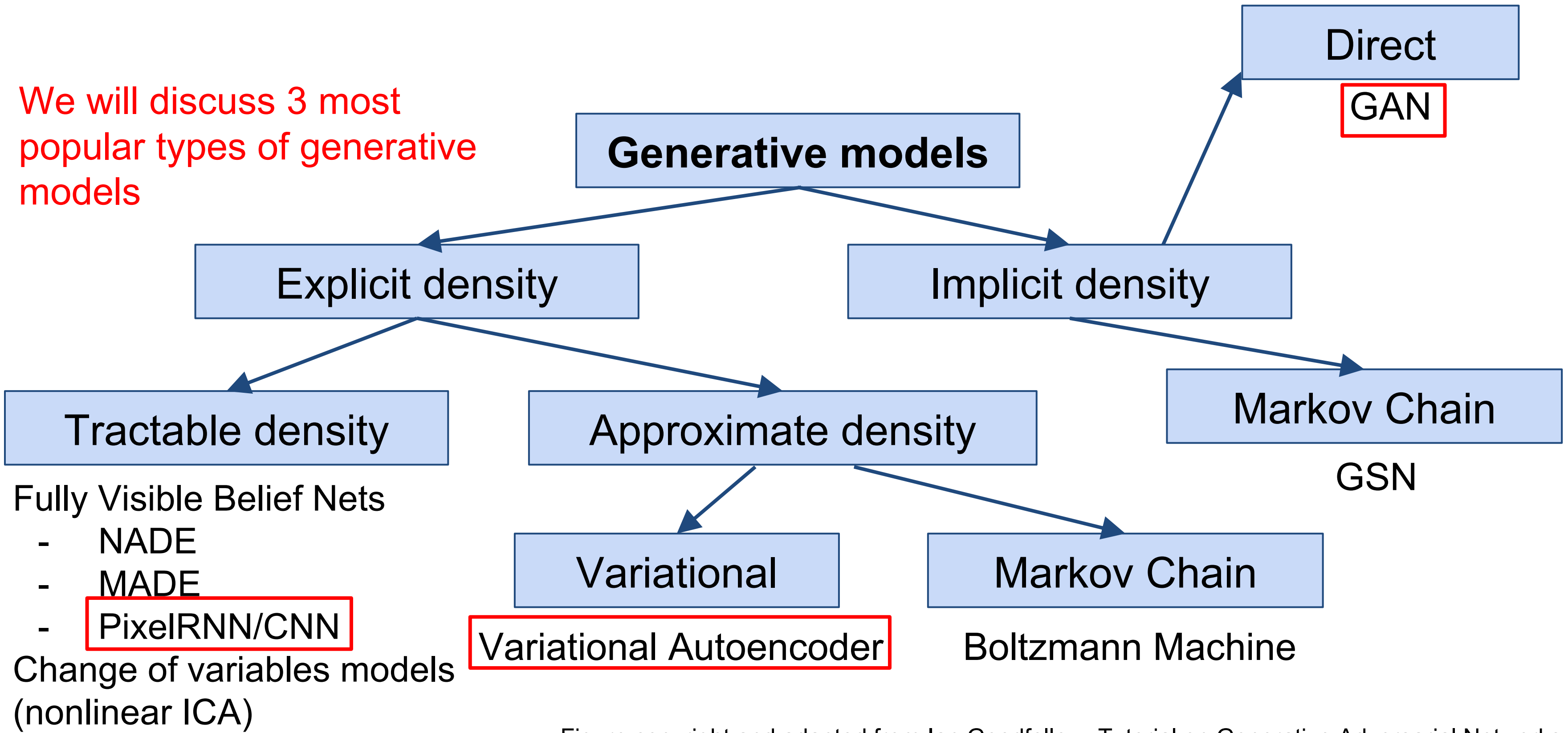


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

PixelRNN and PixelCNN

Fully Observable Model

Explicit density model

$\vec{x} \in \mathbb{R}^d$

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

$P_{model}(x)$

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

Likelihood of image x

Probability of i'th pixel value given all previous pixels

Loss

Then maximize likelihood of training data

$x_1 \dots x_{i-1} \rightarrow NN_\theta \rightarrow x_i$

$p(x_i | x_1, \dots, x_{i-1})$

$\min_\theta - \log p(x_i^{gt} | \dots)$

Fully Observable Model

Explicit density model

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

$D = \{ \cdot\ \cdot\ \cdot\ \cdot\ \cdot \}$

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

Likelihood of image x

Probability of i'th pixel value given all previous pixels

Then maximize likelihood of training data

Complex distribution over pixel values
=> Express using a neural network!

Fully Observable Model

Explicit density model

Use chain rule to decompose likelihood of an image x into product of 1-d distributions:

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

Likelihood of image x

Probability of i'th pixel value given all previous pixels

Will need to define ordering of "previous pixels"

Then maximize likelihood of training data

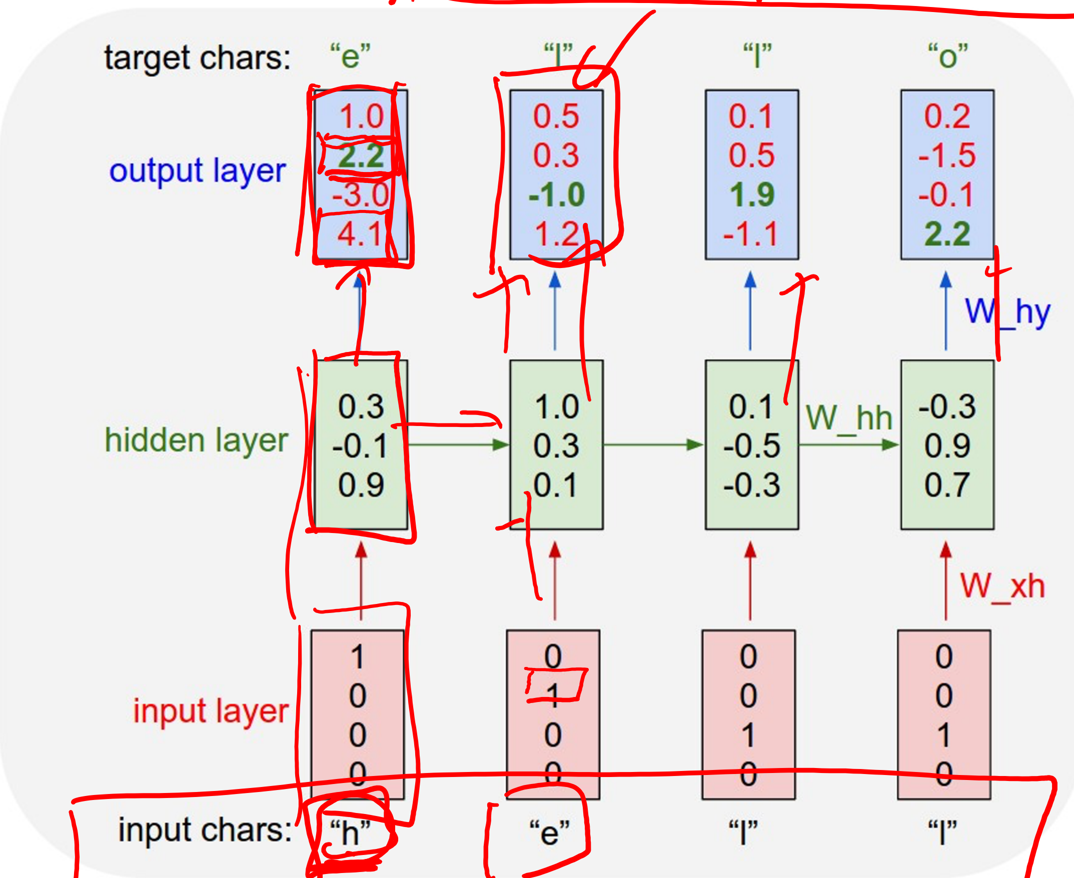
Complex distribution over pixel values => Express using a neural network!

$$\max_{w} \underbrace{\log P(x_1, \dots, x_s \mid w)}_{} = \sum_{t} \underbrace{\log P(x_t \mid x_1, \dots, x_{t-1}, w)}_{}$$

Example: Character-level Language Model

Vocabulary:
[h,e,l,o]

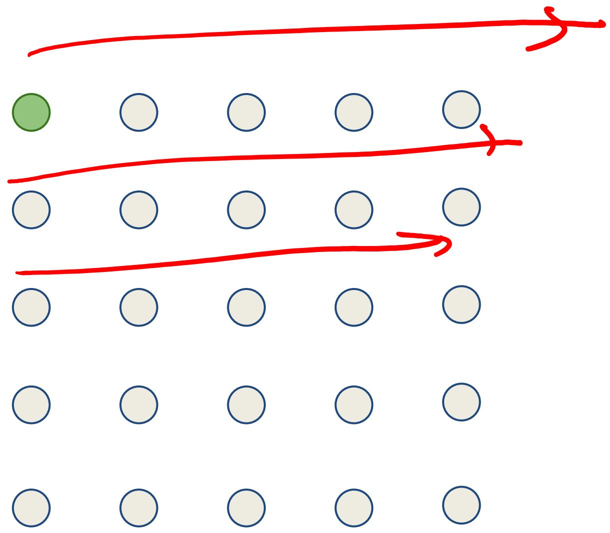
Example training sequence:
“hello”



PixelRNN *[van der Oord et al. 2016]*

Generate image pixels starting from corner

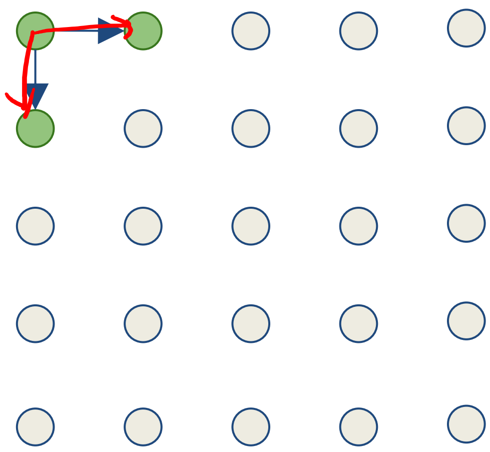
Dependency on previous pixels modeled using an RNN (LSTM)



PixelRNN *[van der Oord et al. 2016]*

Generate image pixels starting from corner

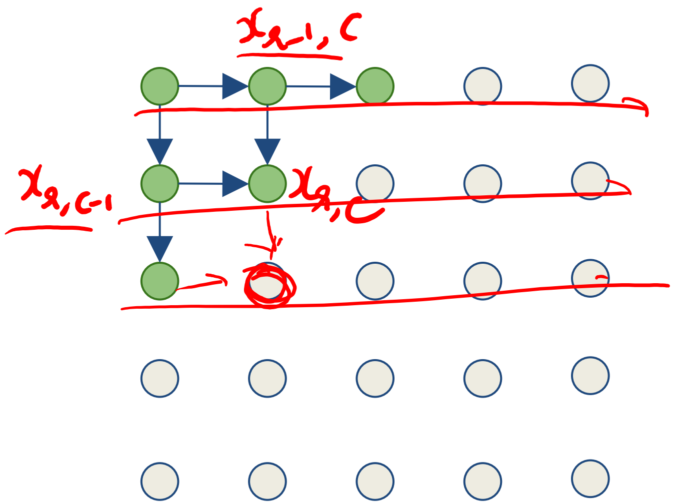
Dependency on previous pixels modeled using an RNN (LSTM)



PixelRNN *[van der Oord et al. 2016]*

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

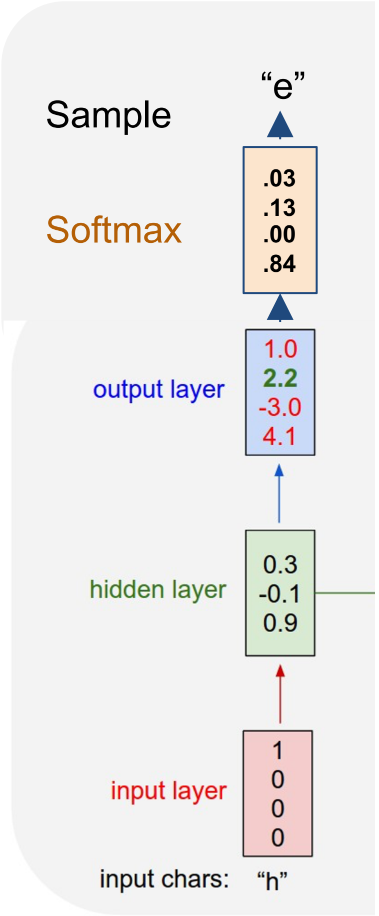


Test Time: Sample / Argmax / Beam Search

Example:
Character-level
Language Model
Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model

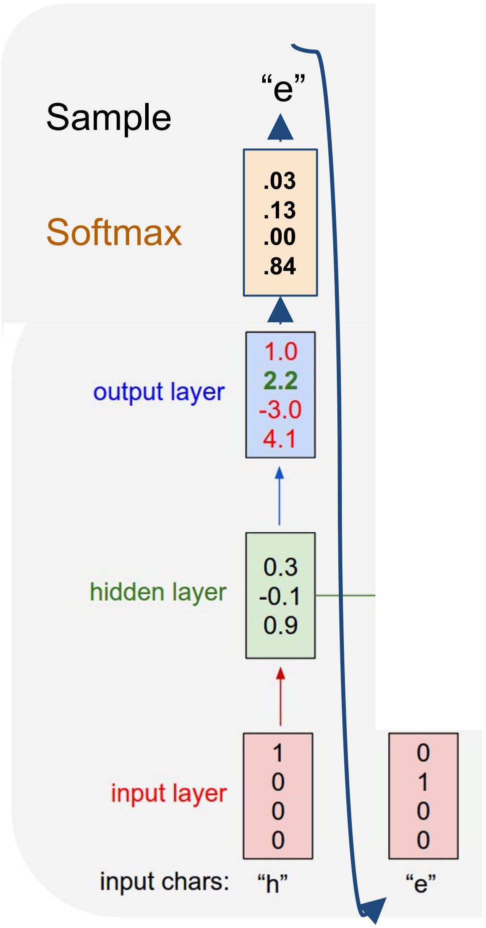


Test Time: Sample / Argmax / Beam Search

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model

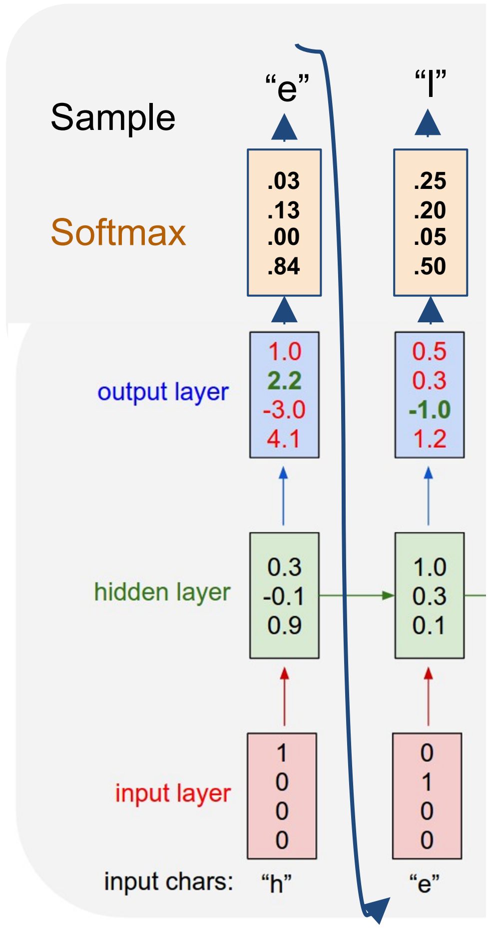


Test Time: Sample / Argmax / Beam Search

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model

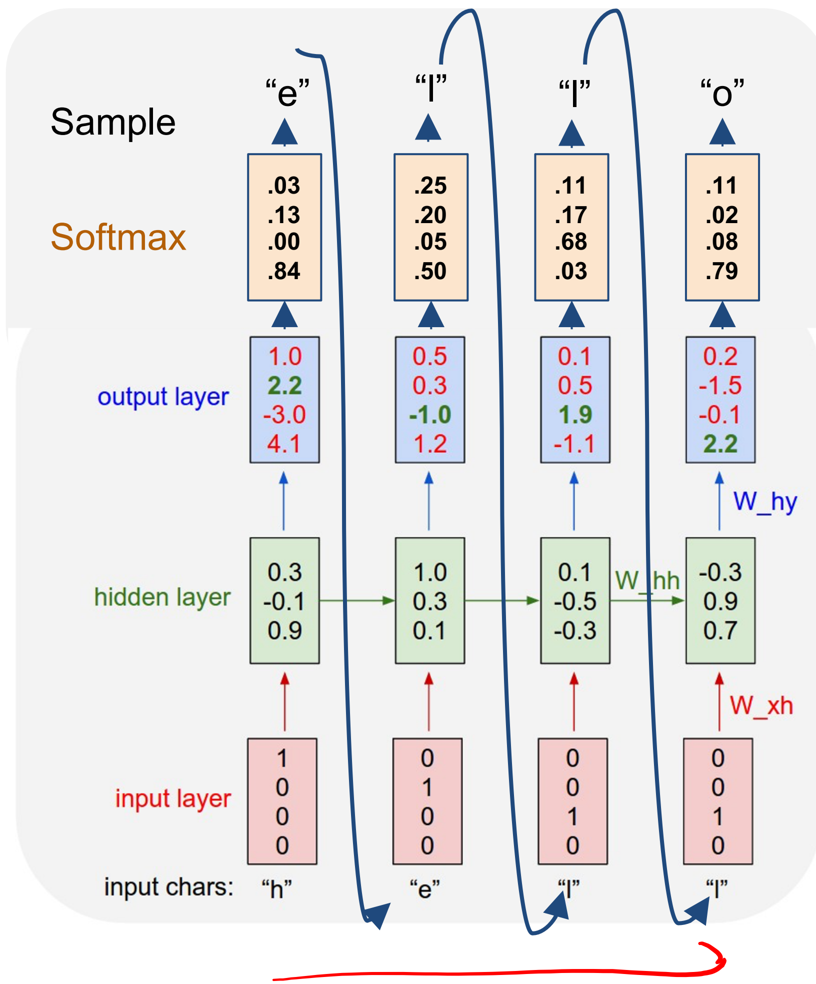


Test Time: Sample / Argmax / Beam Search

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

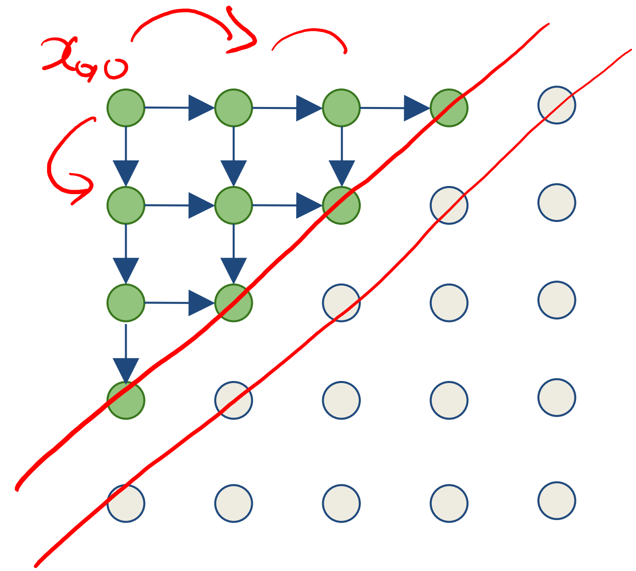
PixelRNN *[van der Oord et al. 2016]*

$\log p(x_{a,c} \mid \_\_\_\_)$

Generate image pixels starting from corner

Dependency on previous pixels modeled using an RNN (LSTM)

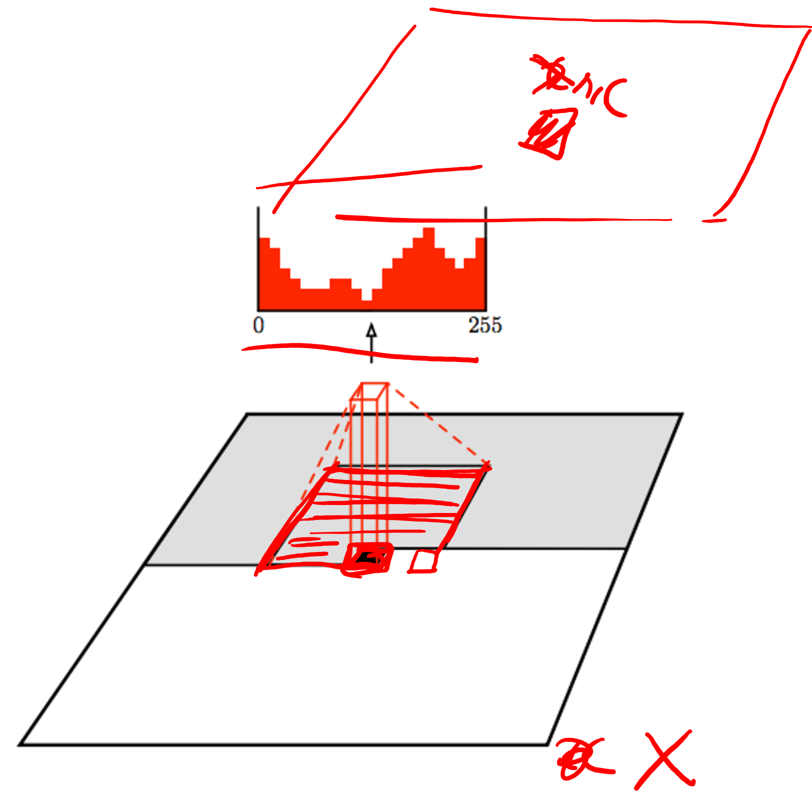
Drawback: sequential generation is slow!

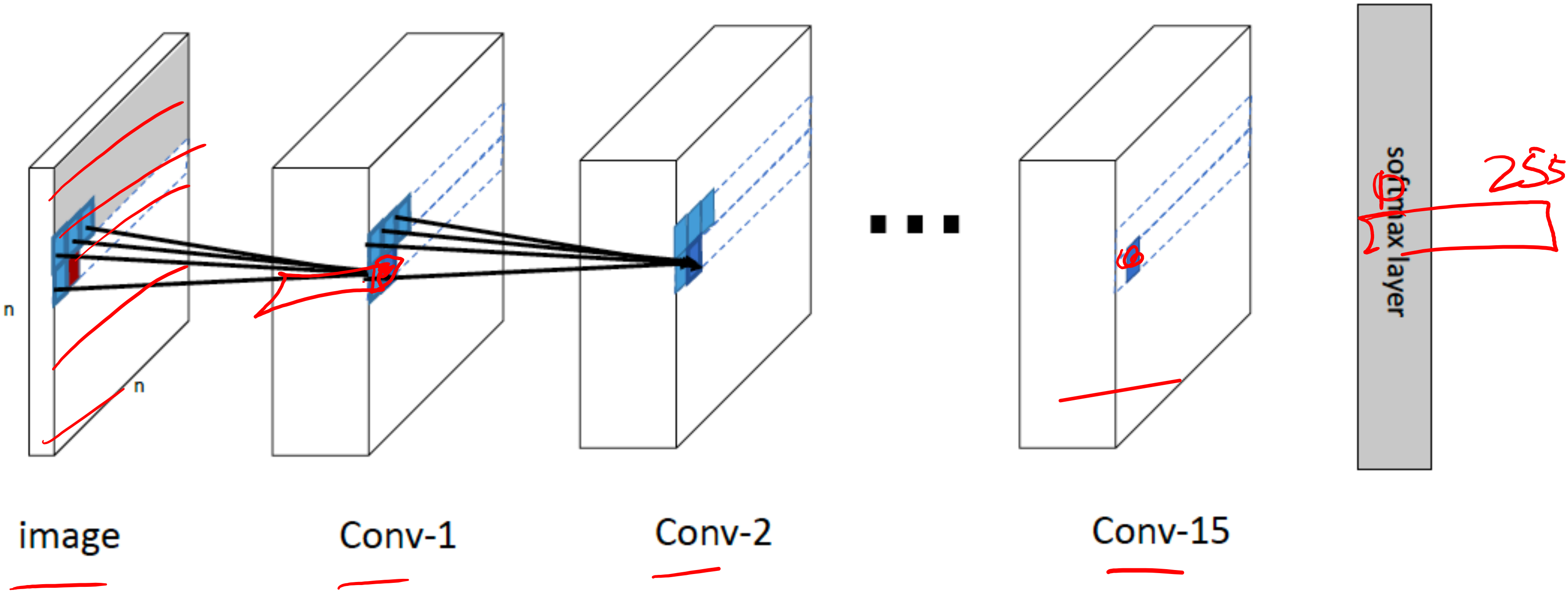


PixelCNN *[van der Oord et al. 2016]*

Still generate image pixels starting from corner

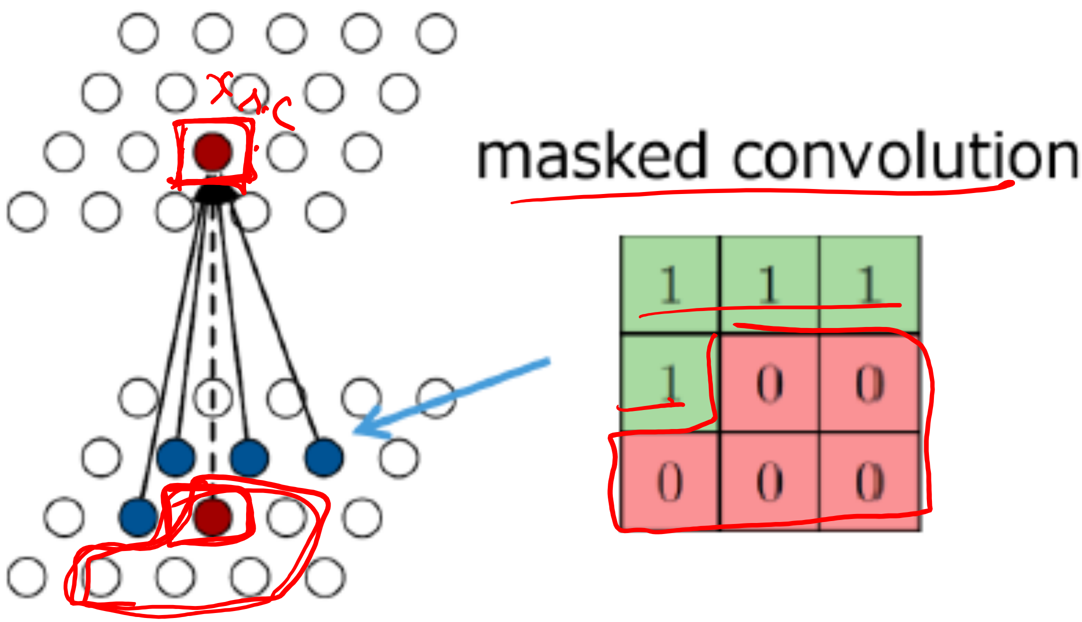
Dependency on previous pixels now modeled using a CNN over context region




n
n
softmax layer
255
0
0
image
Conv-1
Conv-2
Conv-15

Masked Convolutions

- Apply masks so that a pixel does not see "future" pixels



PixelCNN *[van der Oord et al. 2016]*

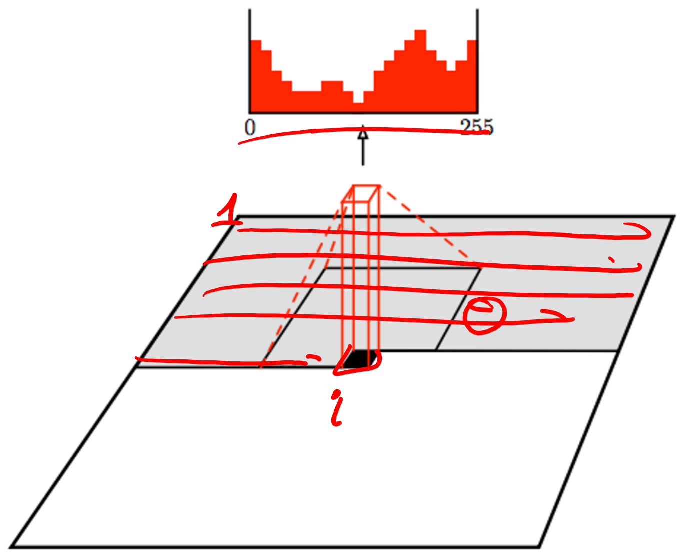
Still generate image pixels starting from corner

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

Training: maximize likelihood of training images

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

Softmax loss at each pixel



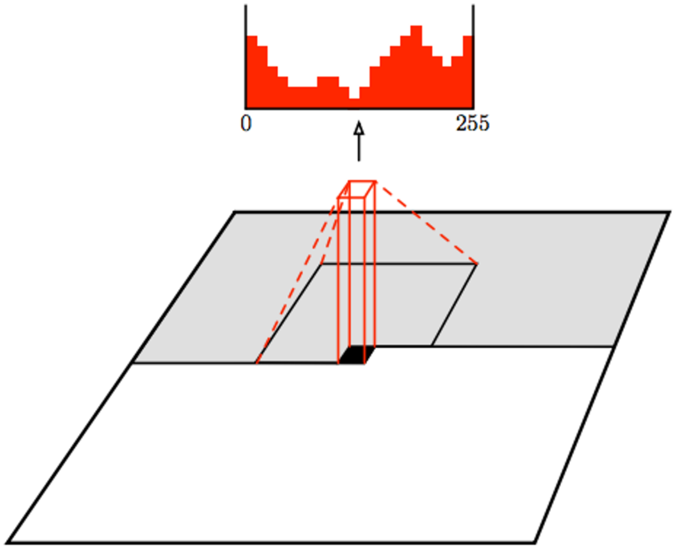
PixelCNN *[van der Oord et al. 2016]*

Still generate image pixels starting from corner

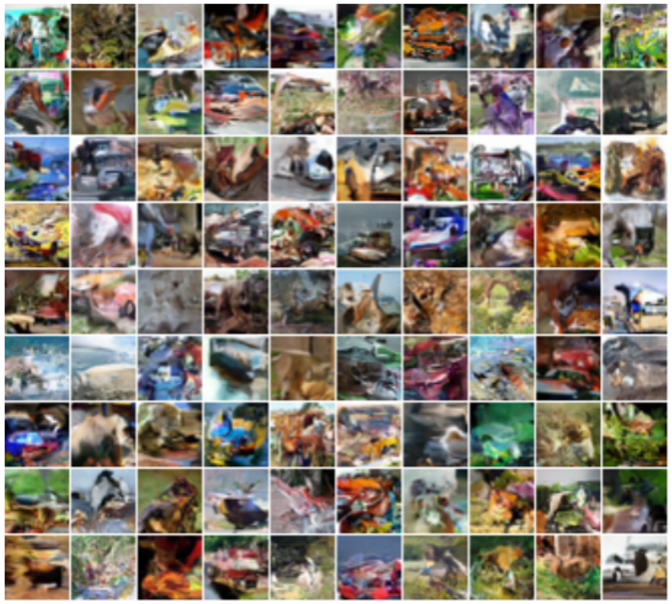
Dependency on previous pixels now modeled using a CNN over context region

Training is faster than PixelRNN
(can parallelize convolutions since context region values known from training images)

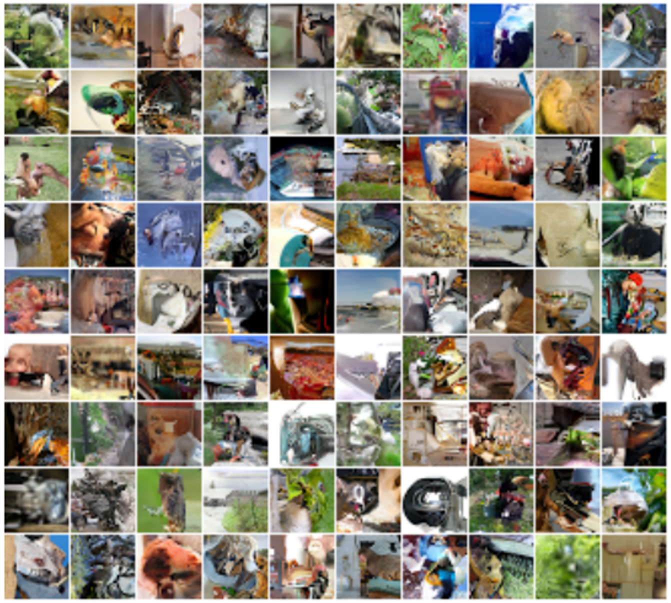
Generation must still proceed sequentially
=> still slow



Generation Samples



32x32 CIFAR-10



32x32 ImageNet

Image Completion

$p(\vec{x})$

$\sim \boxed{p(\vec{x}_{i+1:d} \mid x_{1:i})}$

Figure 1. Image completions sampled from a PixelRNN.

Results from generating sounds

- https://deepmind.com/blog/wavenet-generative-model-raw-audio/

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…

See

- Van der Oord et al. NIPS 2016
- Salimans et al. 2017 (PixelCNN++)