Deep Unsupervised Learning

Russ Salakhutdinov

Machine Learning Department
Carnegie Mellon University
Canadian Institute of Advanced Research

Carnegie Mellon University

CIFAR
CANADIAN INSTITUTE for ADVANCED RESEARCH

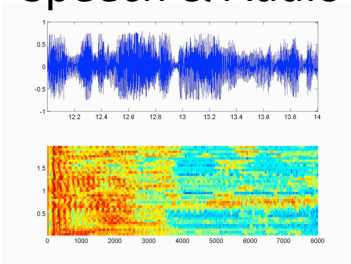
Mining for Structure

Massive increase in both computational power and the amount of data available from web, video cameras, laboratory measurements.

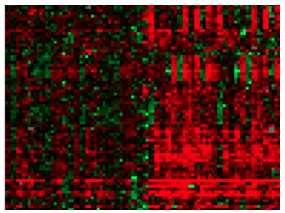
Images & Video

Text & Language

Speech & Audio

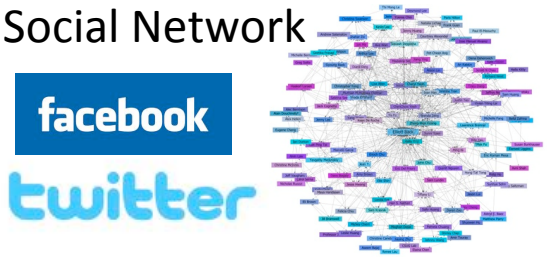


Gene Expression

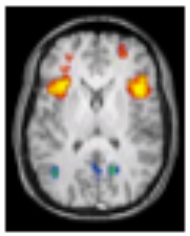


Product Recommendation

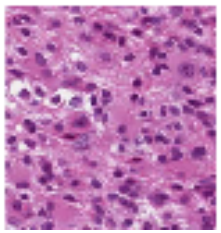
Relational Data/ Social Network



fMRI



Tumor region



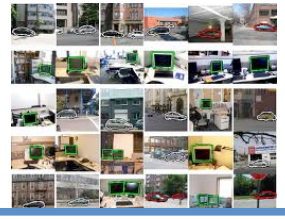
Mostly Unlabeled

- Develop statistical models that can discover underlying structure, cause, or statistical correlation from data in **unsupervised** or **semi-supervised** way.
- Multiple application domains.

Mining for Structure

Massive increase in both computational power and the amount of data available from web, video cameras, laboratory measurements.

Images & Video

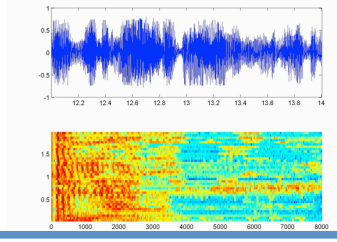


Text & Language

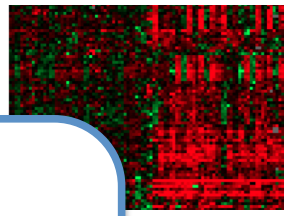
REUTERS

Associated Press

Speech & Audio



Gene Expression



Product

Recomm

ama

NETFLI

Deep Learning Models that support inferences and discover structure at multiple levels.

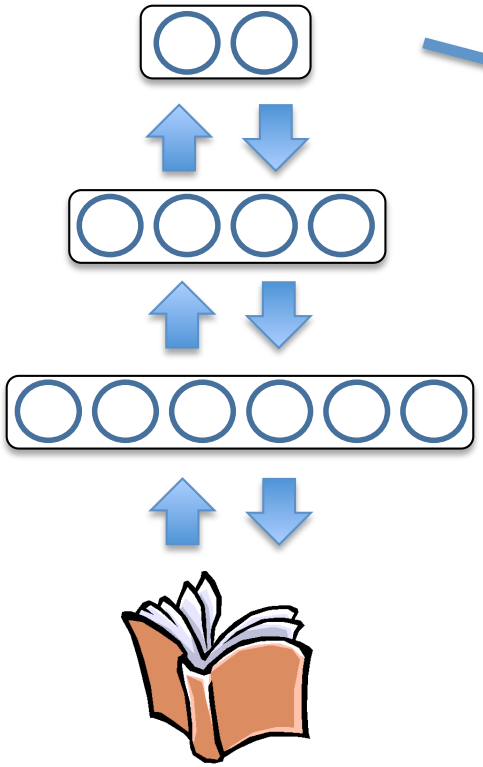
on

Mostly Unlabeled

- Develop statistical models that can discover underlying structure, cause, or statistical correlation from data in **unsupervised** or **semi-supervised** way.
- Multiple application domains.

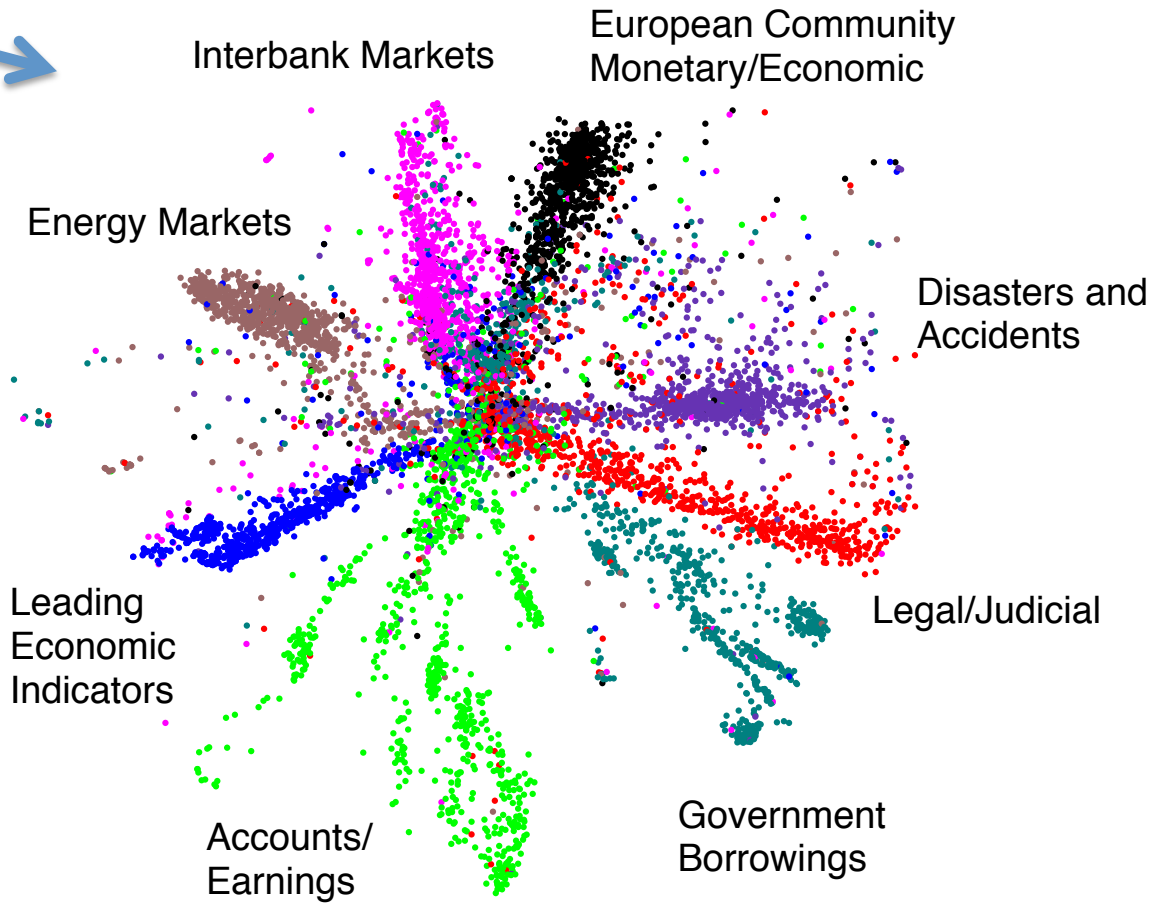
Deep Autoencoder Model

Learned latent code

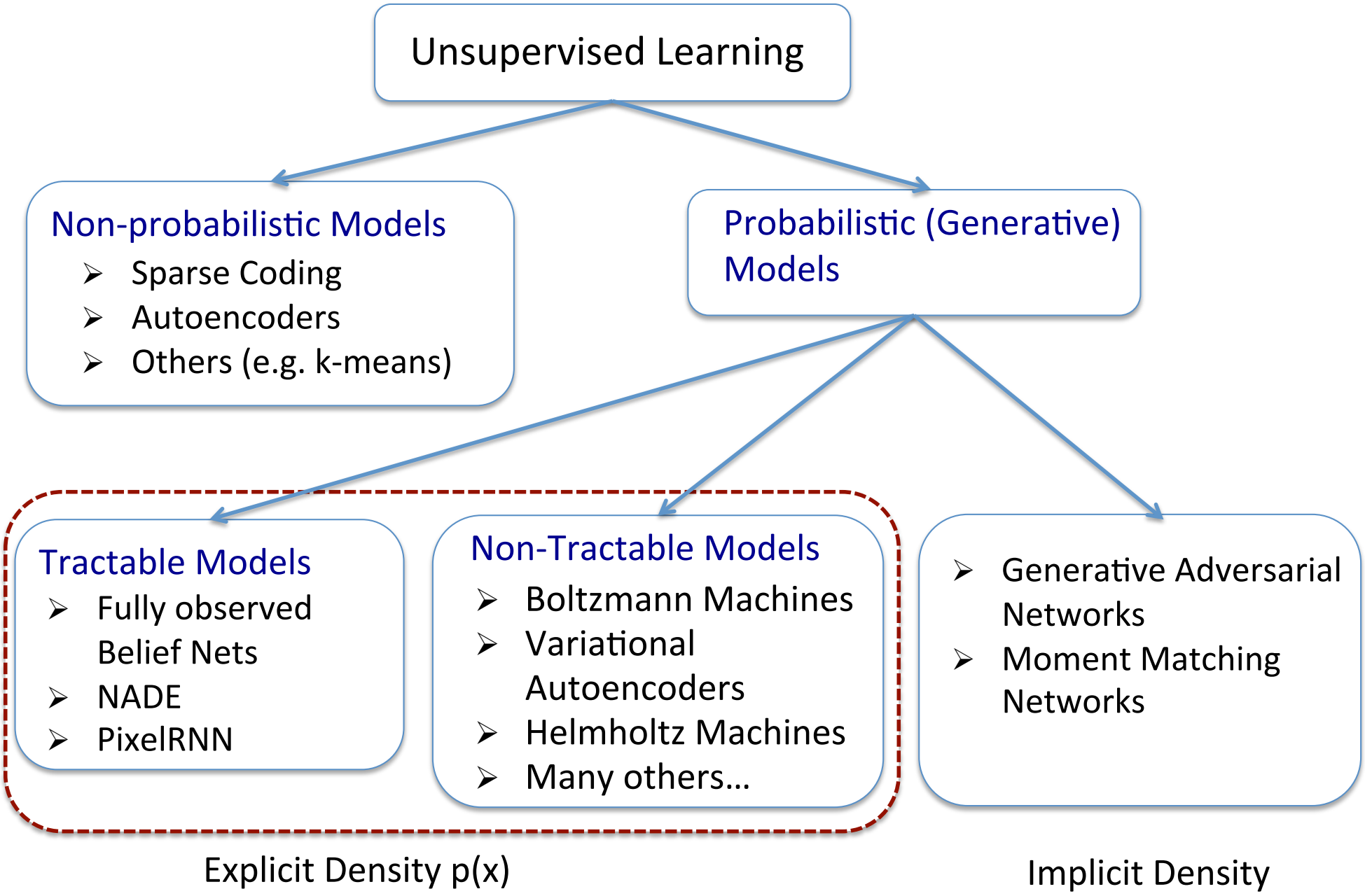


Bag of words

Reuters dataset: 804,414
newswire stories: **unsupervised**



(Hinton & Salakhutdinov, Science 2006)


Unsupervised Learning
Non-probabilistic Models
Sparse Coding
Autoencoders
Others (e.g. k-means)
Probabilistic (Generative) Models
Tractable Models
Fully observed Belief Nets
NADE
PixelRNN
Non-Tractable Models
Boltzmann Machines
Variational Autoencoders
Helmholtz Machines
Many others…
Generative Adversarial Networks
Moment Matching Networks
Explicit Density p(x)
Implicit Density

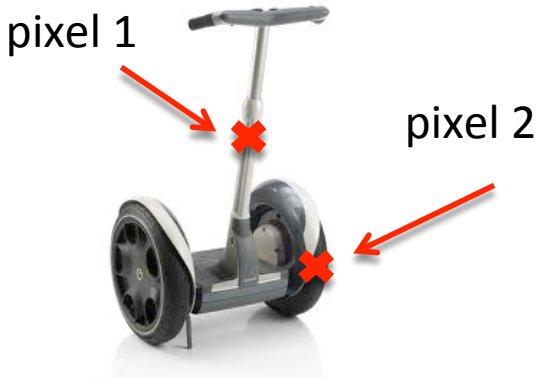
Talk Roadmap

- Basic Building Blocks:
  - Sparse Coding
  - Autoencoders
- Deep Generative Models
  - Restricted Boltzmann Machines
  - Deep Boltzmann Machines
  - Helmholtz Machines / Variational Autoencoders
- Generative Adversarial Networks

Talk Roadmap

- Basic Building Blocks:
  - Sparse Coding
  - Autoencoders
- Deep Generative Models
  - Restricted Boltzmann Machines
  - Deep Boltzmann Machines
  - Helmholtz Machines / Variational Autoencoders
- Generative Adversarial Networks

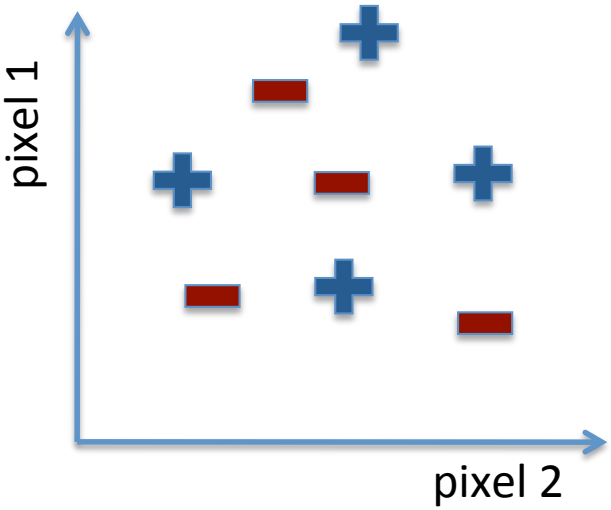
Learning Feature Representations



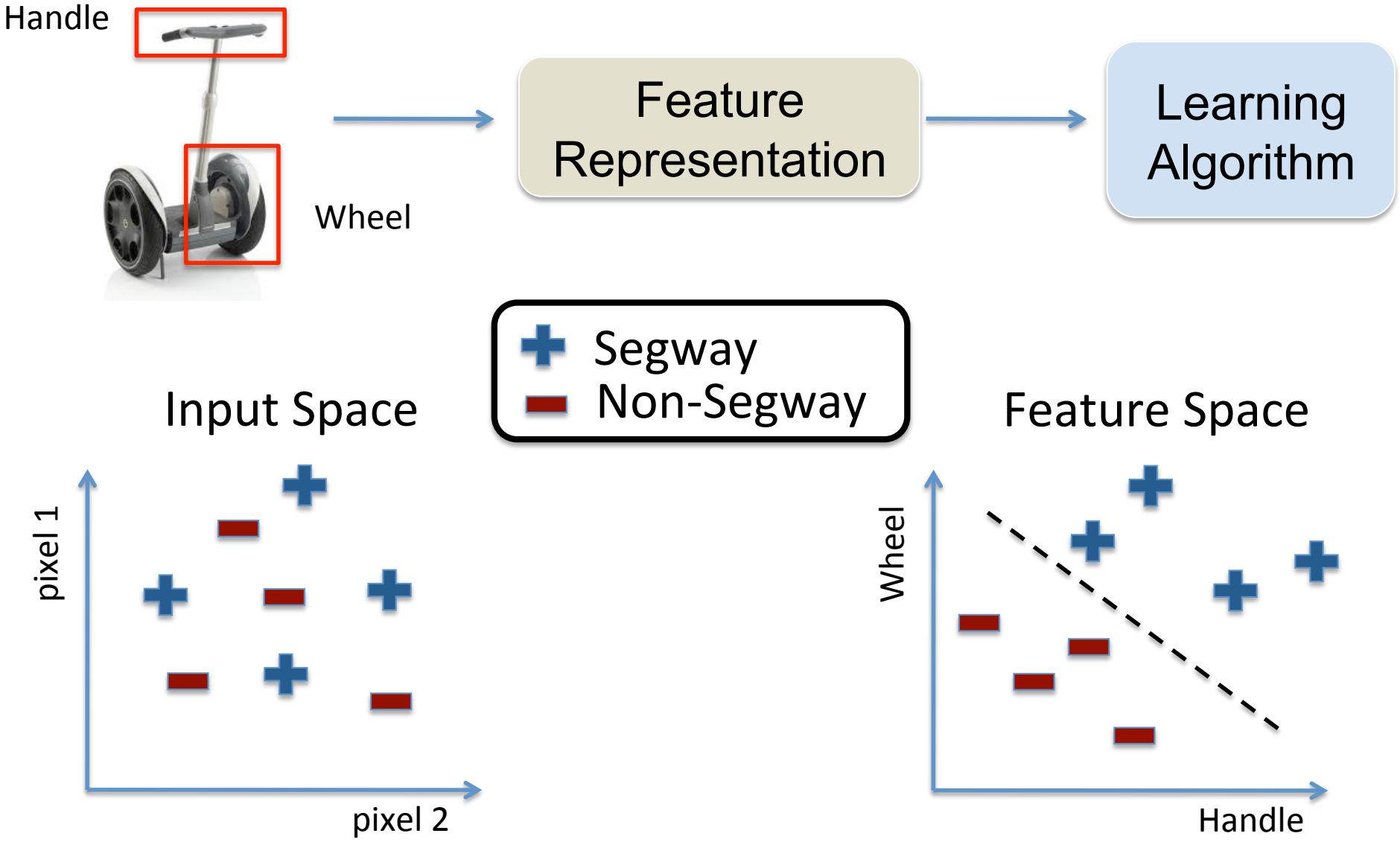
Learning Algorithm

Segway
Non-Segway

Input Space



Learning Feature Representations



Traditional Approaches

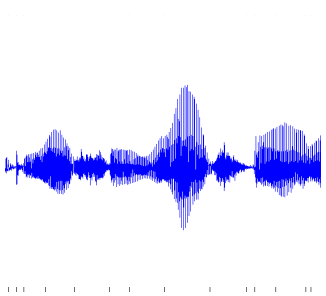
Object detection

Image

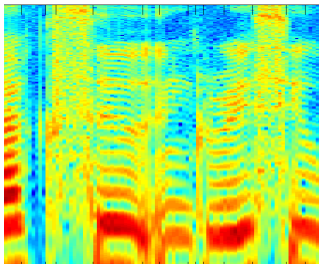
vision features

Recognition

Audio classification



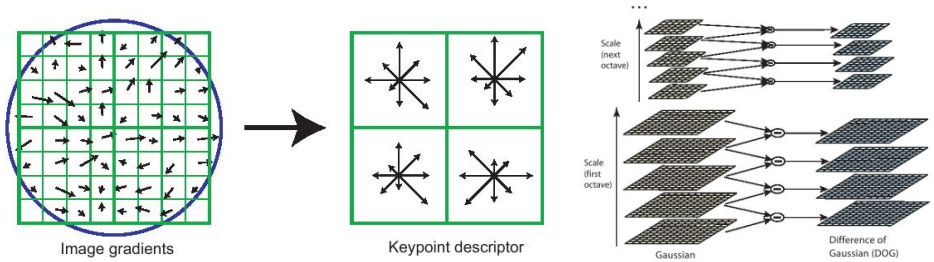
Audio



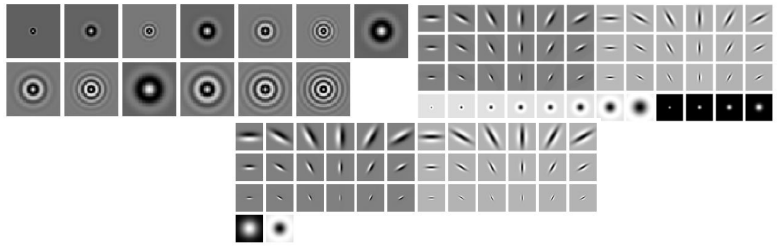
audio features

Speaker identification

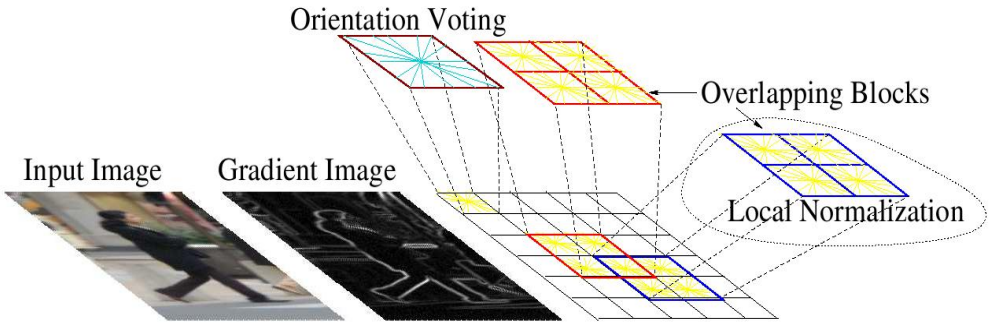
Computer Vision Features



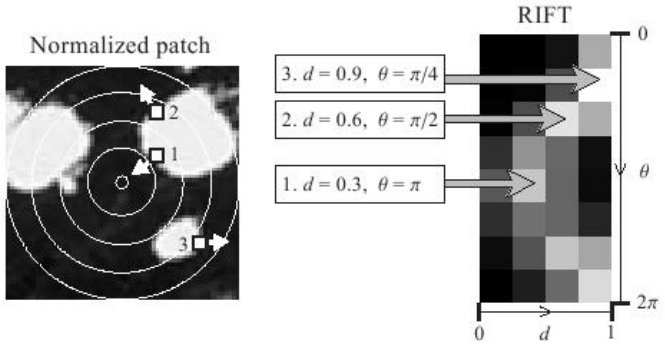
SIFT



Textons

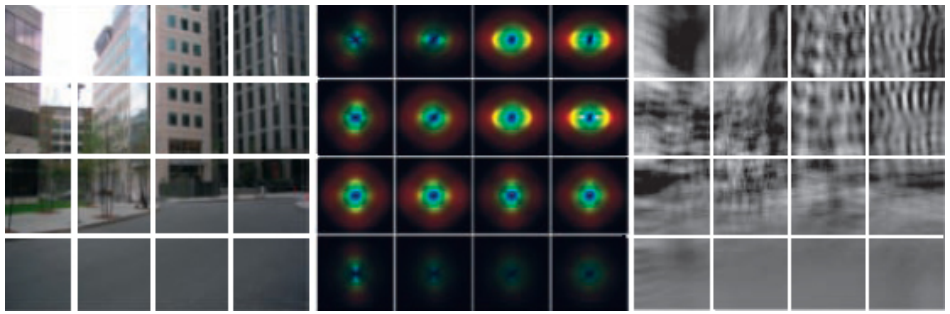


HoG

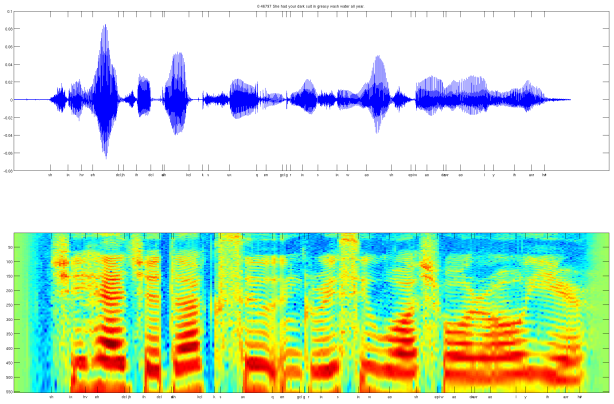


RIFT

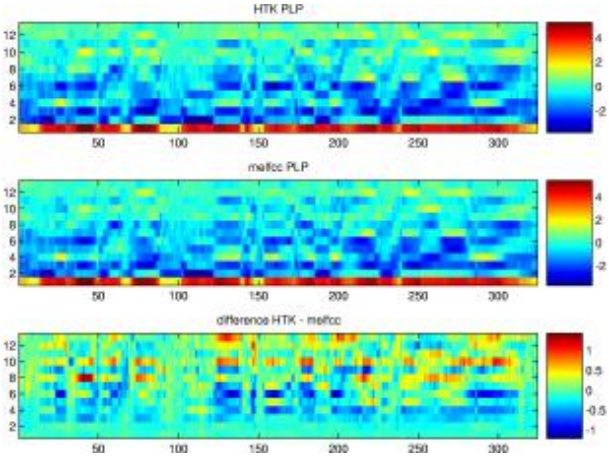
GIST



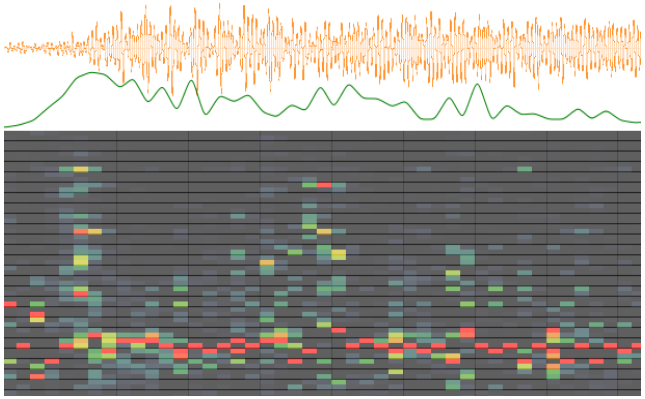
Audio Features



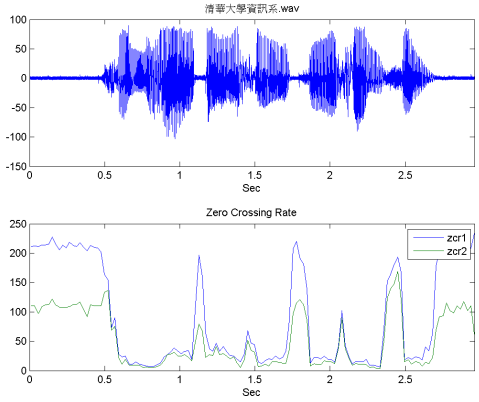
Spectrogram



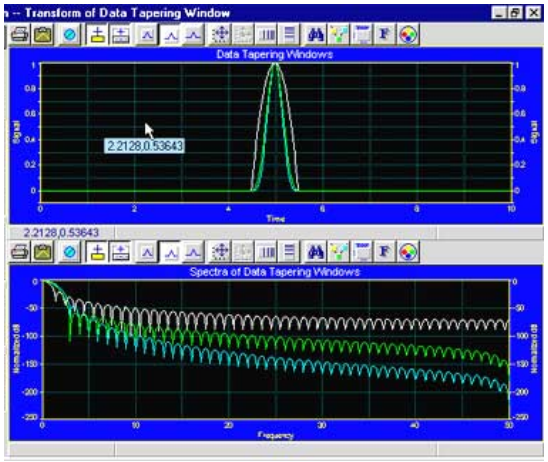
MFCC



Flux

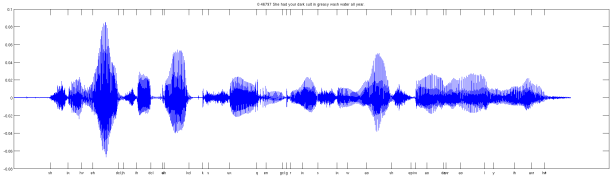


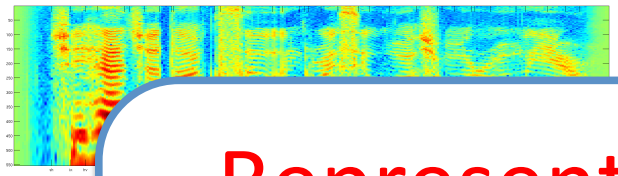
ZCR

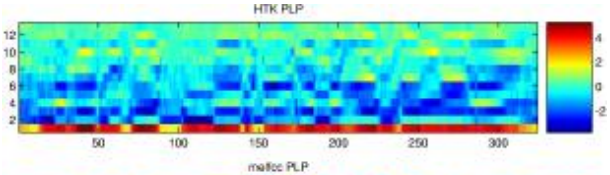


Rolloff

Audio Features

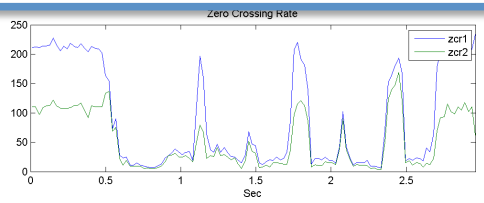






Representation Learning:
Can we automatically learn these representations?

Flux



ZCR

Rolloff

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 $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, learn a dictionary of bases $\{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_K\}$, such that:

$$\mathbf{x}_n = \sum_{k=1}^{K} a_{nk}\boldsymbol{\phi}_k,$$

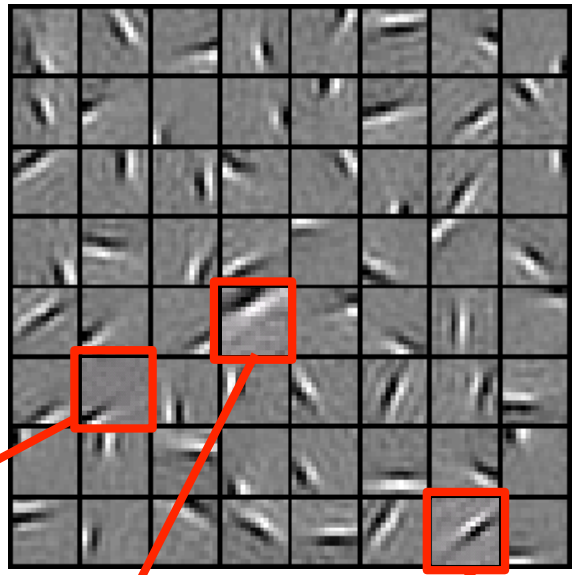
Sparse: mostly zeros

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

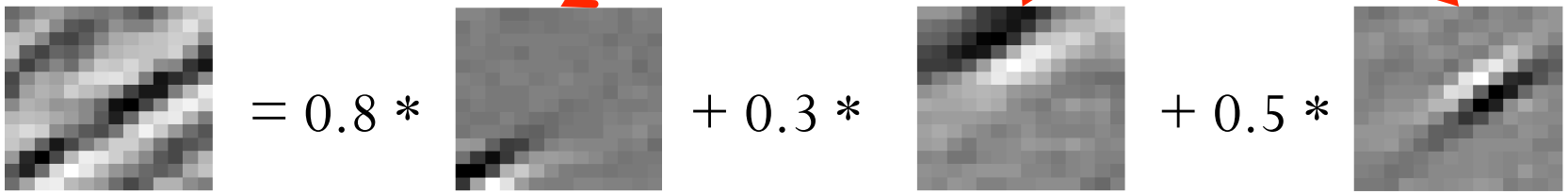
Sparse Coding

Natural Images

Learned bases: "Edges"



New example



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

[0, 0, … **0.8**, …, **0.3**, …, **0.5**, …] = coefficients (feature representation)

Slide Credit: Honglak Lee

Sparse Coding: Training

- Input image patches: $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N \in \mathbb{R}^D$
- Learn dictionary of bases: $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_K \in \mathbb{R}^D$

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

Reconstruction error Sparsity penalty

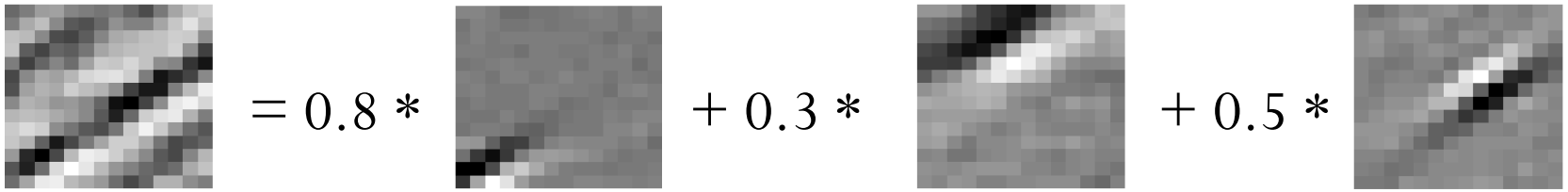
- Alternating Optimization:

1. Fix dictionary of bases $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_K$ and solve for activations **a** (a standard Lasso problem).
2. Fix activations **a**, optimize the dictionary of bases (convex QP problem).

Sparse Coding: Testing Time

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

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

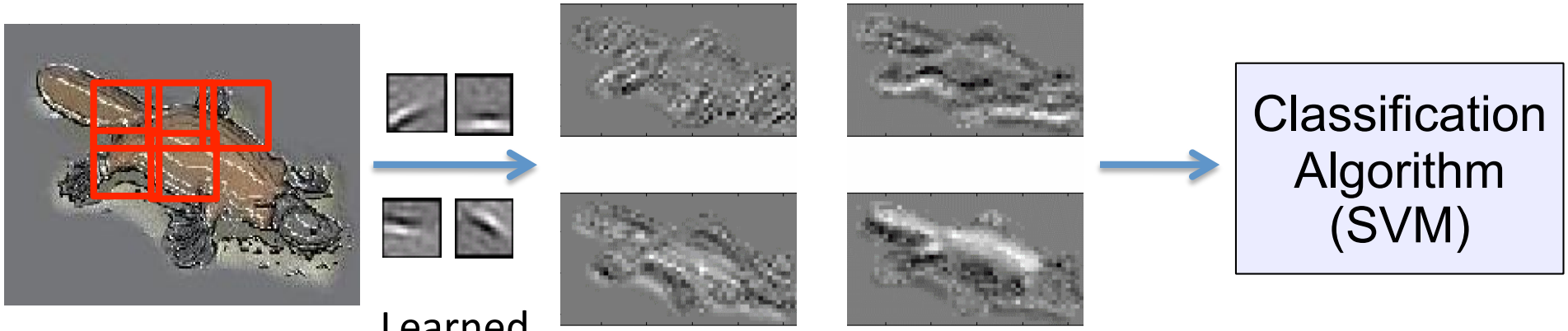


x^* $= 0.8 *$ ϕ_{36} $+ 0.3 *$ ϕ_{42} $+ 0.5 *$ ϕ_{65}

[0, 0, … **0.8**, …, **0.3**, …, **0.5**, …] = coefficients (feature representation)

Image Classification

Evaluated on Caltech101 object category dataset.

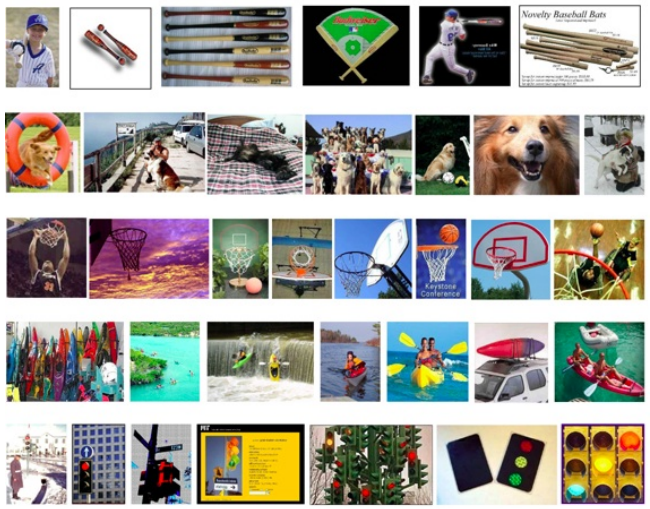


Input Image

Learned bases

Features (coefficients)

9K images, 101 classes



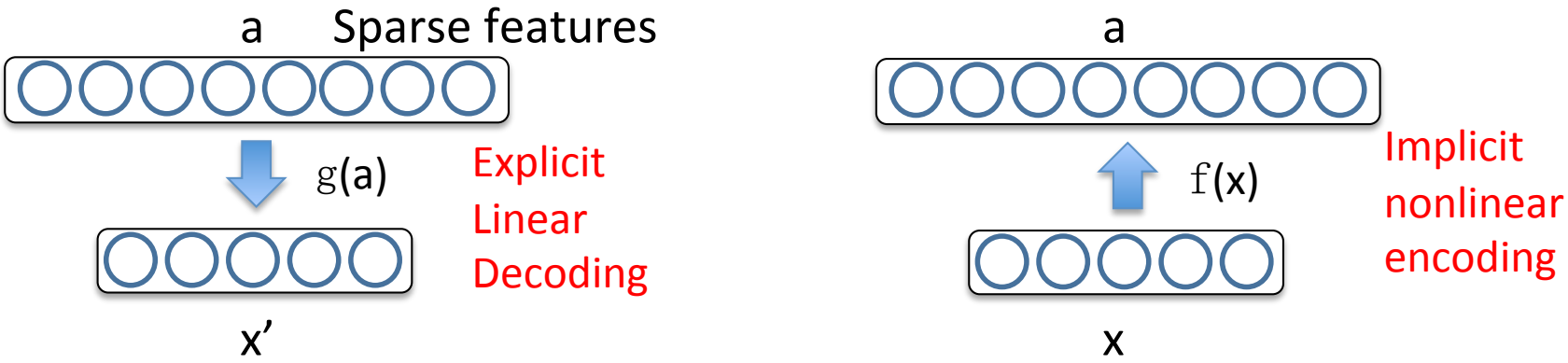
Algorithm	Accuracy
Baseline (Fei-Fei et al., 2004)	16%
PCA	37%
Sparse Coding	**47%**

Slide Credit: Honglak Lee

Lee, Battle, Raina, Ng, 2006

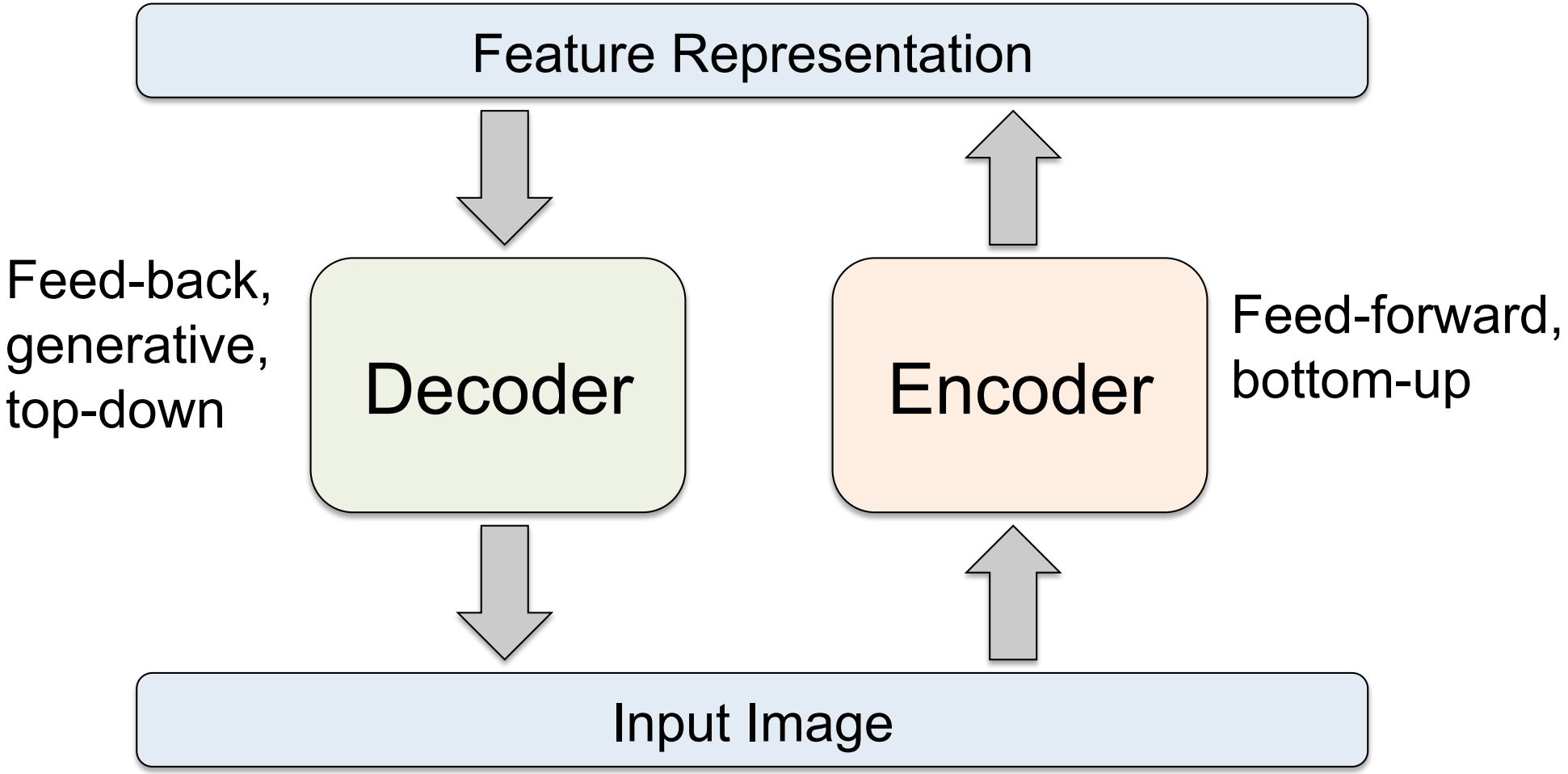
Interpreting Sparse Coding

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



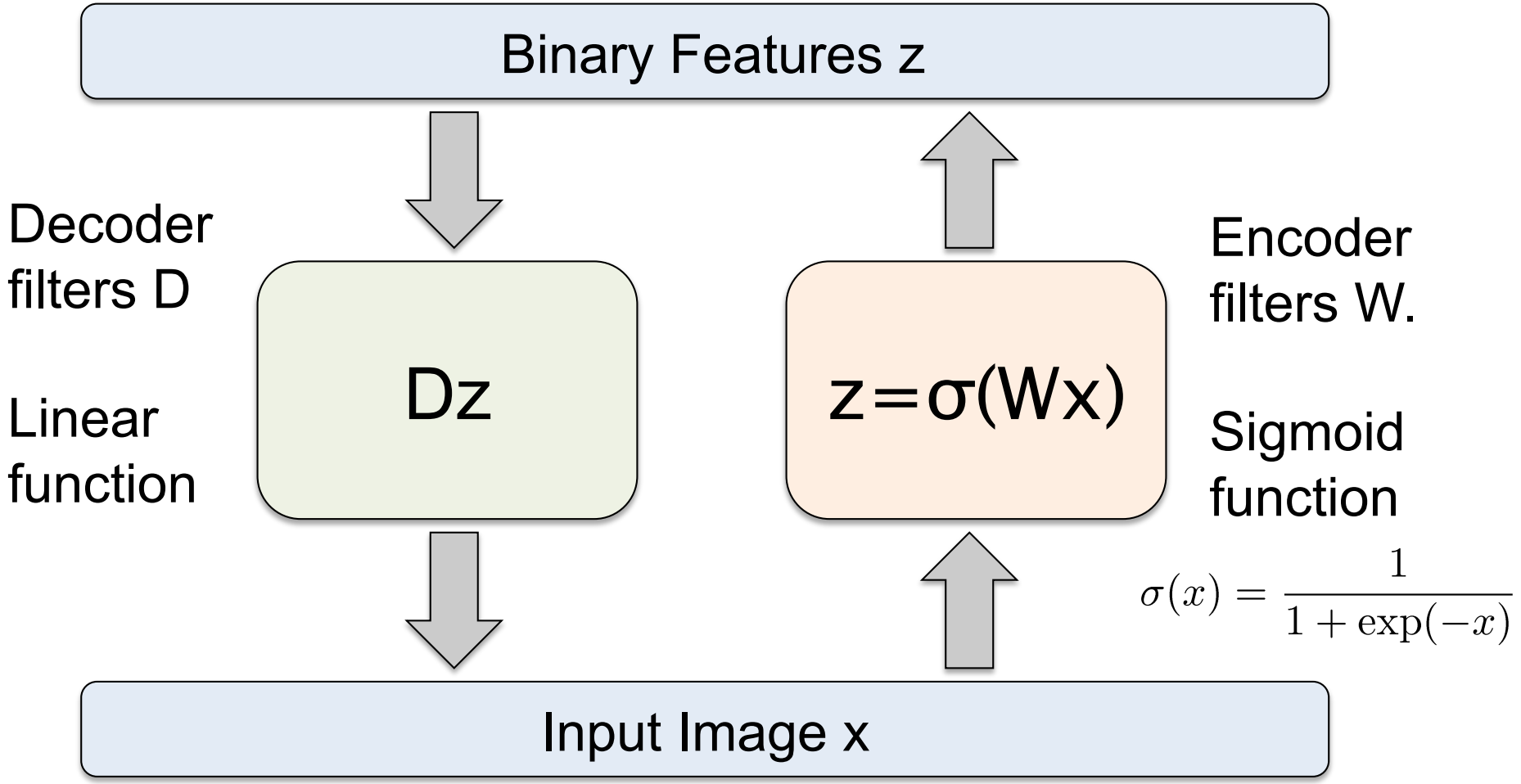
- Sparse, over-complete representation **a.**
- Encoding **a** = f(**x**) is implicit and nonlinear function of **x**.
- Reconstruction (or decoding) **x'** = g(**a**) is linear and explicit.

Autoencoder

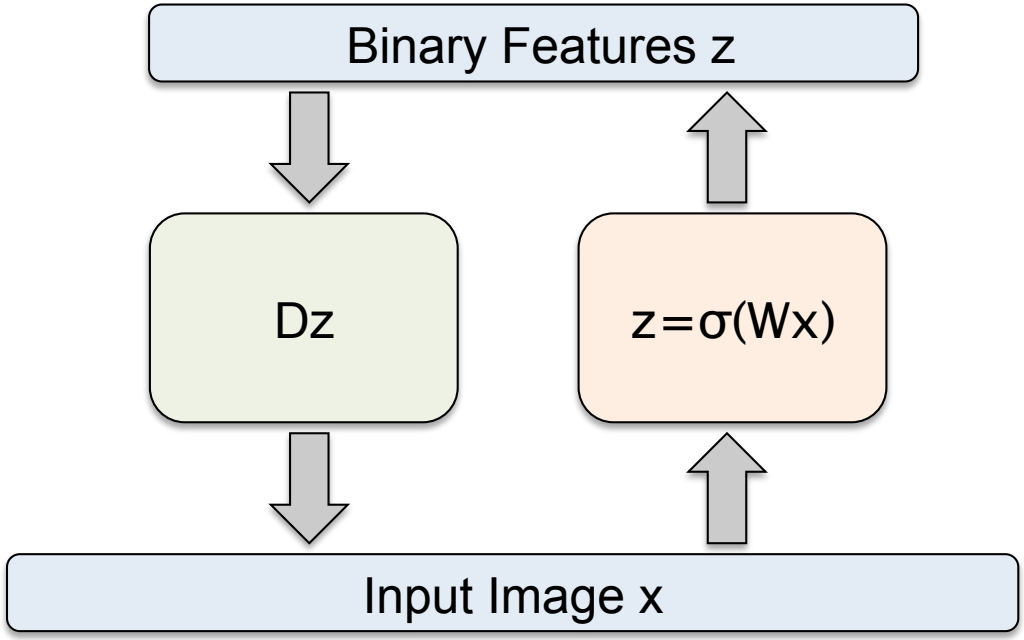


- Details of what goes insider the encoder and decoder matter!
- Need constraints to avoid learning an identity.

Autoencoder



Autoencoder



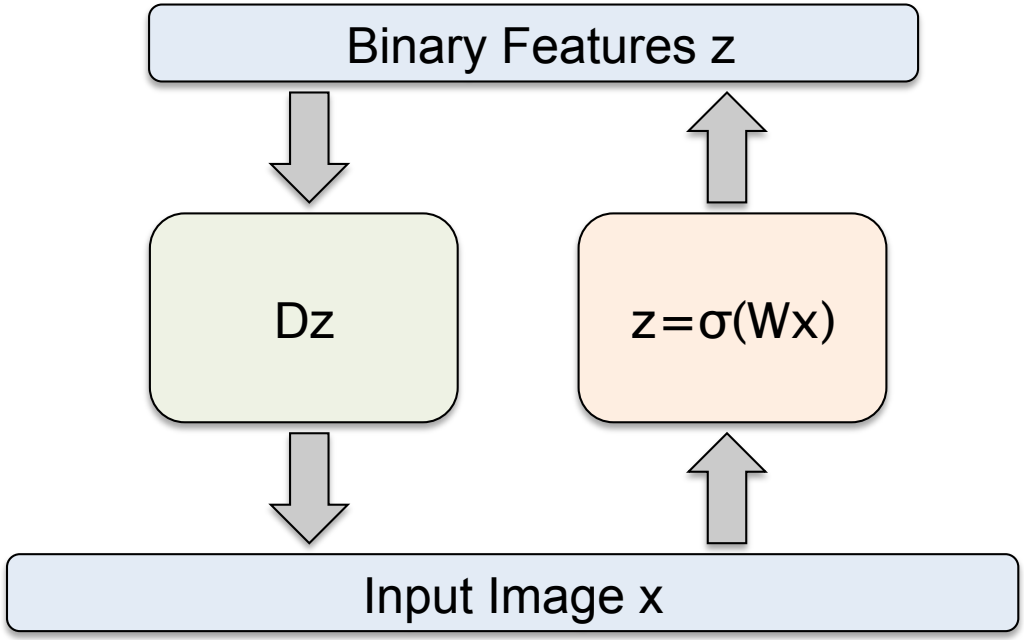
- An autoencoder with D inputs, D outputs, and K hidden units, with K<D.

- Given an input x, its reconstruction is given by:

$$y_j(\mathbf{x}, W, D) = \underbrace{\sum_{k=1}^{K} D_{jk}}_{\text{Decoder}} \underbrace{\sigma\left(\sum_{i=1}^{D} W_{ki} x_i\right)}_{\text{Encoder}}, \quad j = 1, .., D.$$

$$y_j = \sum_{k=1}^{K} D_{jk} z_k \qquad z_k = \sigma\left(\sum_{i=1}^{D} W_{ki} x_i\right)$$

Autoencoder

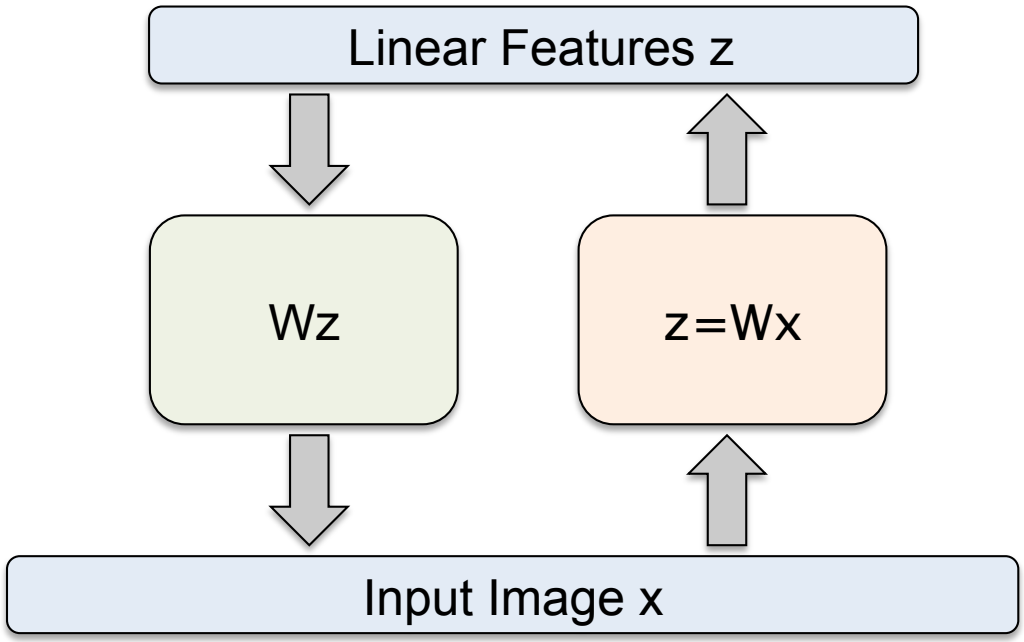


• An autoencoder with D inputs, D outputs, and K hidden units, with K<D.

• We can determine the network parameters W and D by minimizing the reconstruction error:

$$E(W, D) = \frac{1}{2} \sum_{n=1}^{N} ||y(\mathbf{x}_n, W, D) - \mathbf{x}_n||^2.$$

Autoencoder

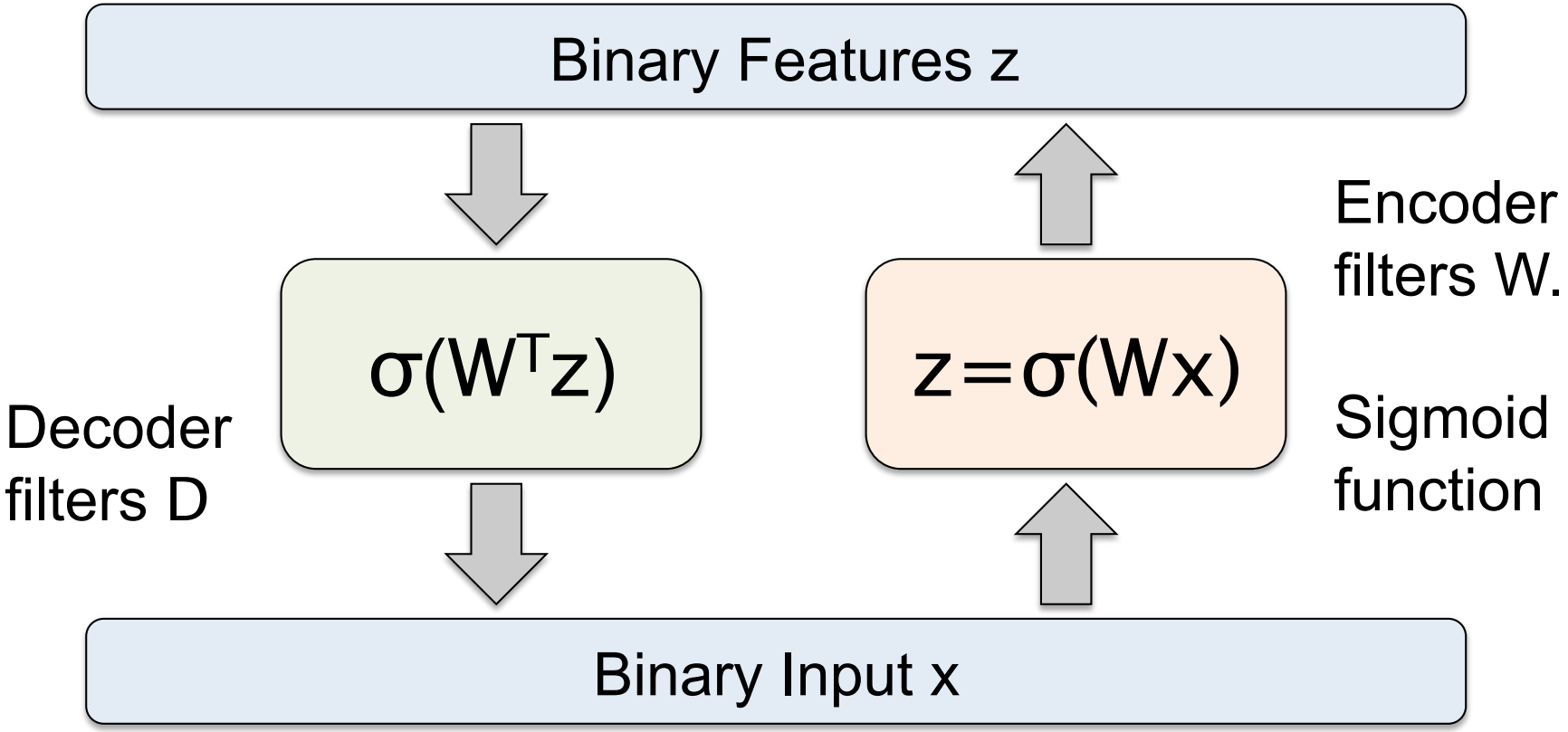


- If the hidden and output layers are linear, it will learn hidden units that are a linear function of the data and minimize the squared error.

- The K hidden units will span the same space as the first k principal components. The weight vectors may not be orthogonal.

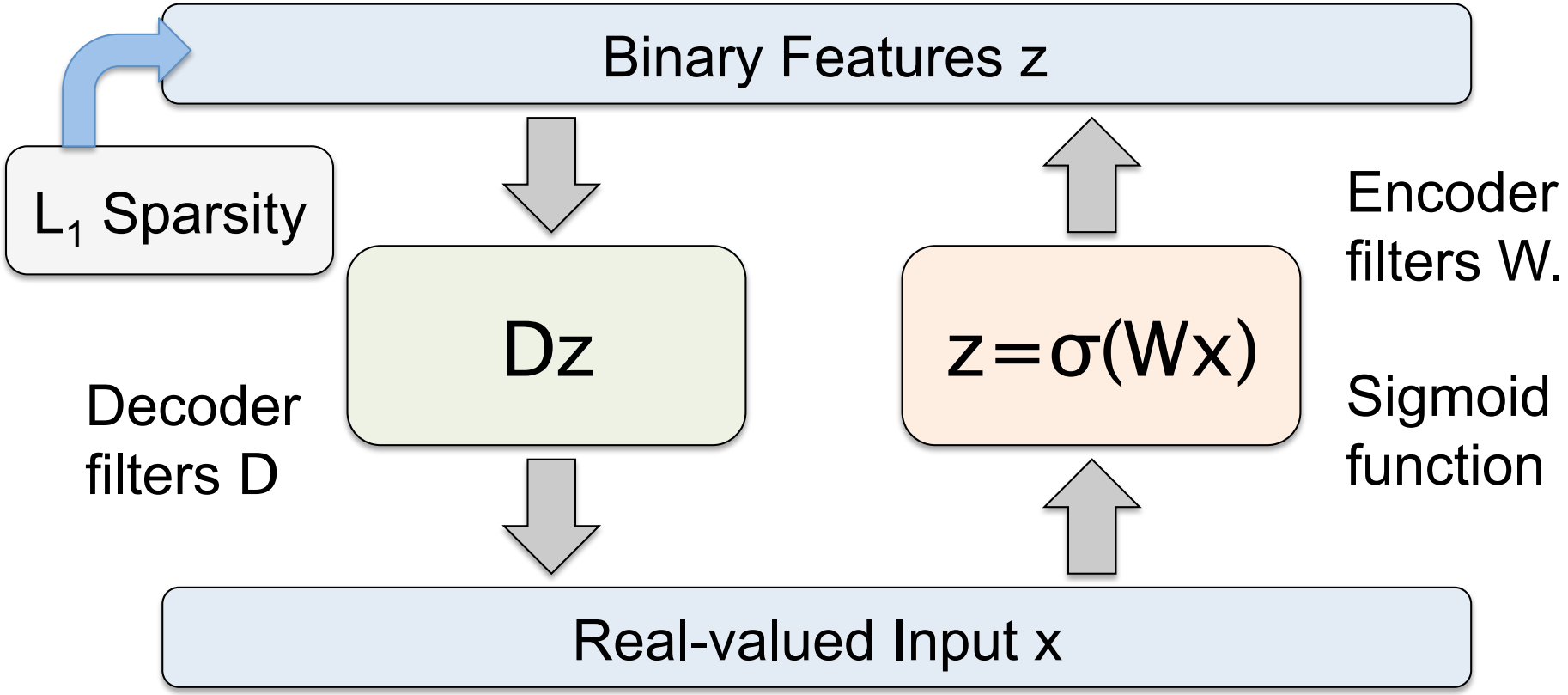
- With nonlinear hidden units, we have a nonlinear generalization of PCA.

Another Autoencoder Model



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

Predictive Sparse Decomposition

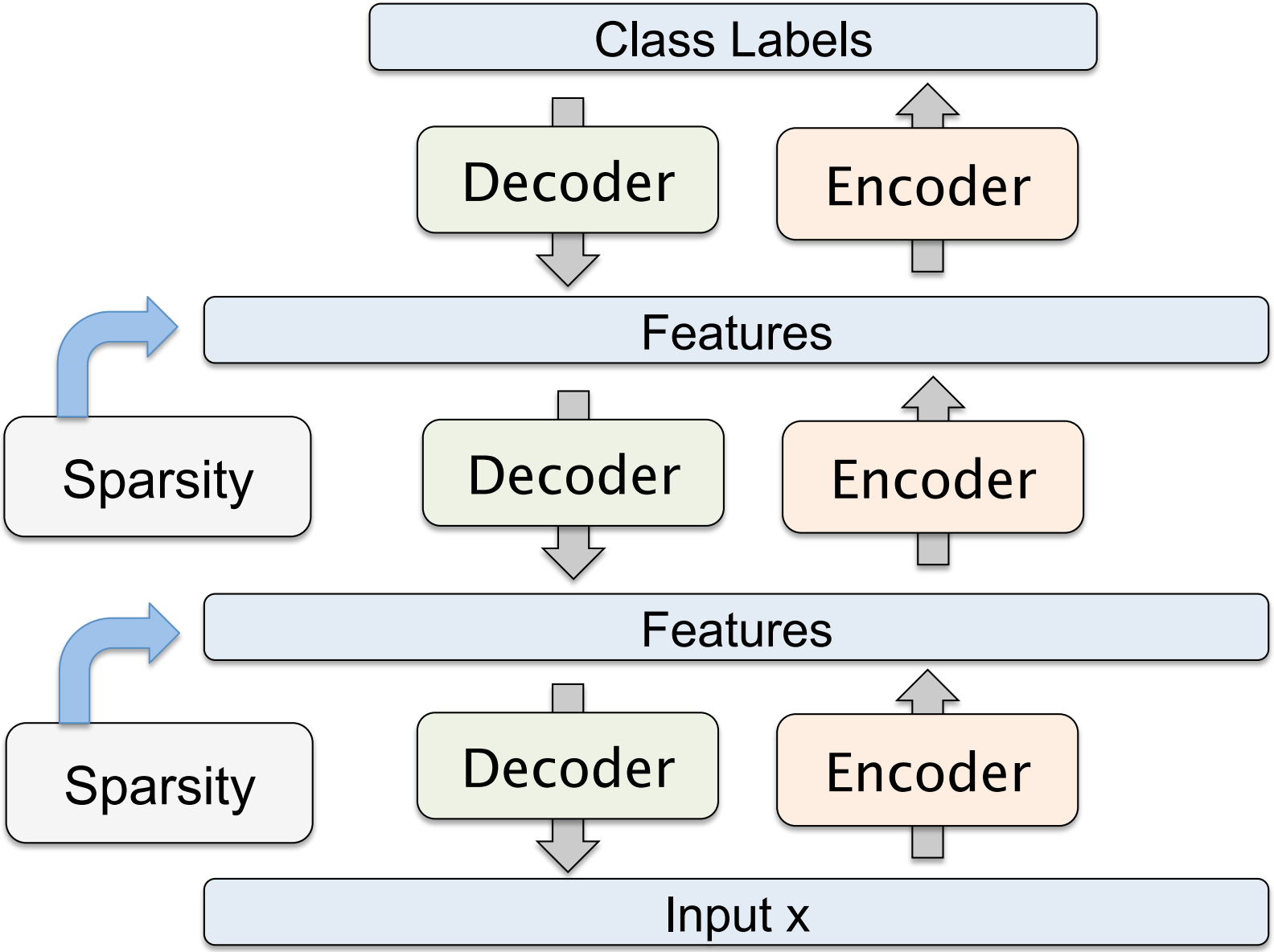


At training time

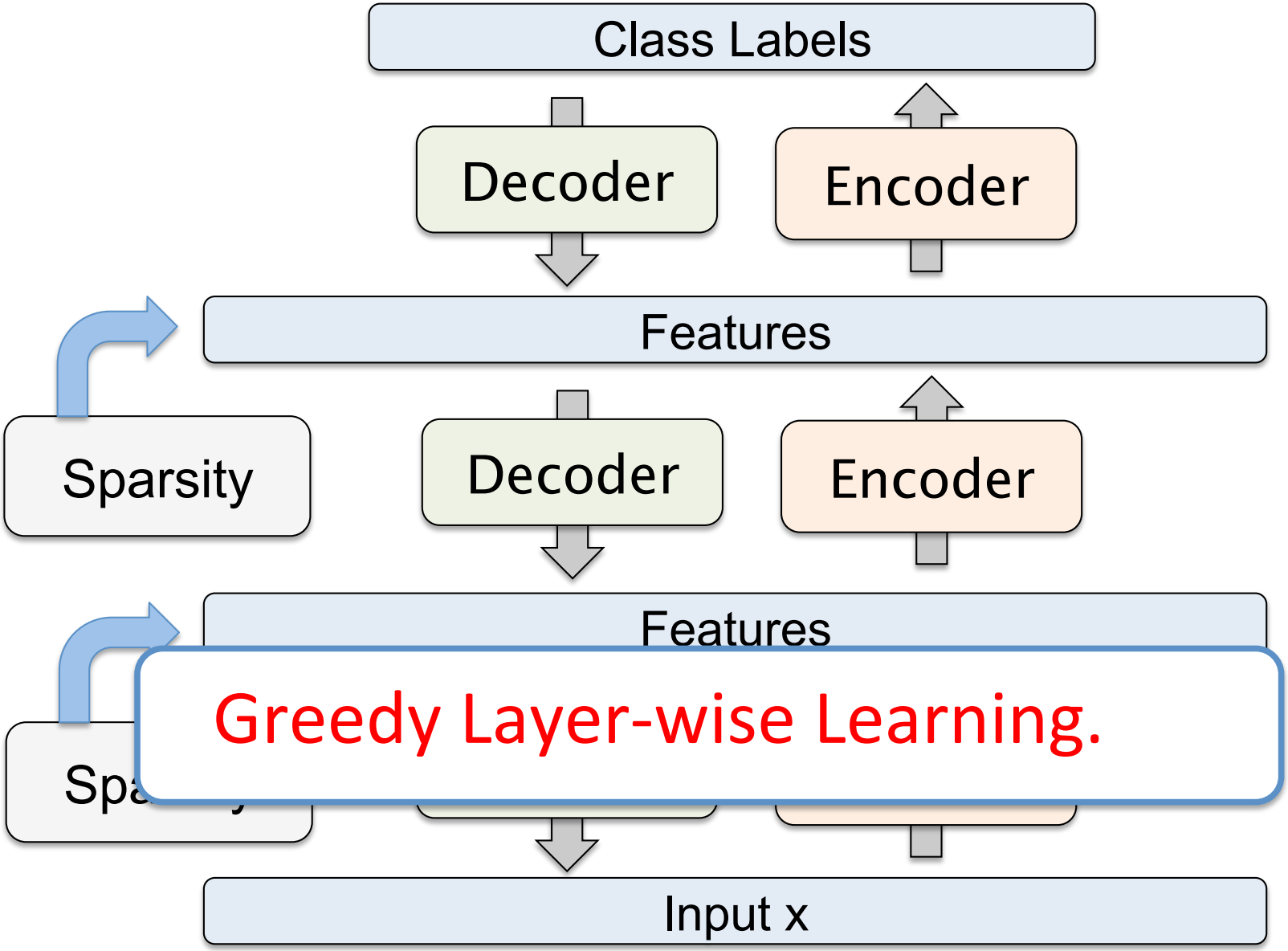
$$\min_{D,W,\mathbf{z}} \underbrace{||D\mathbf{z}-\mathbf{x}||_2^2+\lambda|\mathbf{z}|_1}_{\text{Decoder}} + \underbrace{||\sigma(W\mathbf{x})-\mathbf{z}||_2^2}_{\text{Encoder}}$$

Kavukcuoglu, Ranzato, Fergus, LeCun, 2009

Stacked Autoencoders

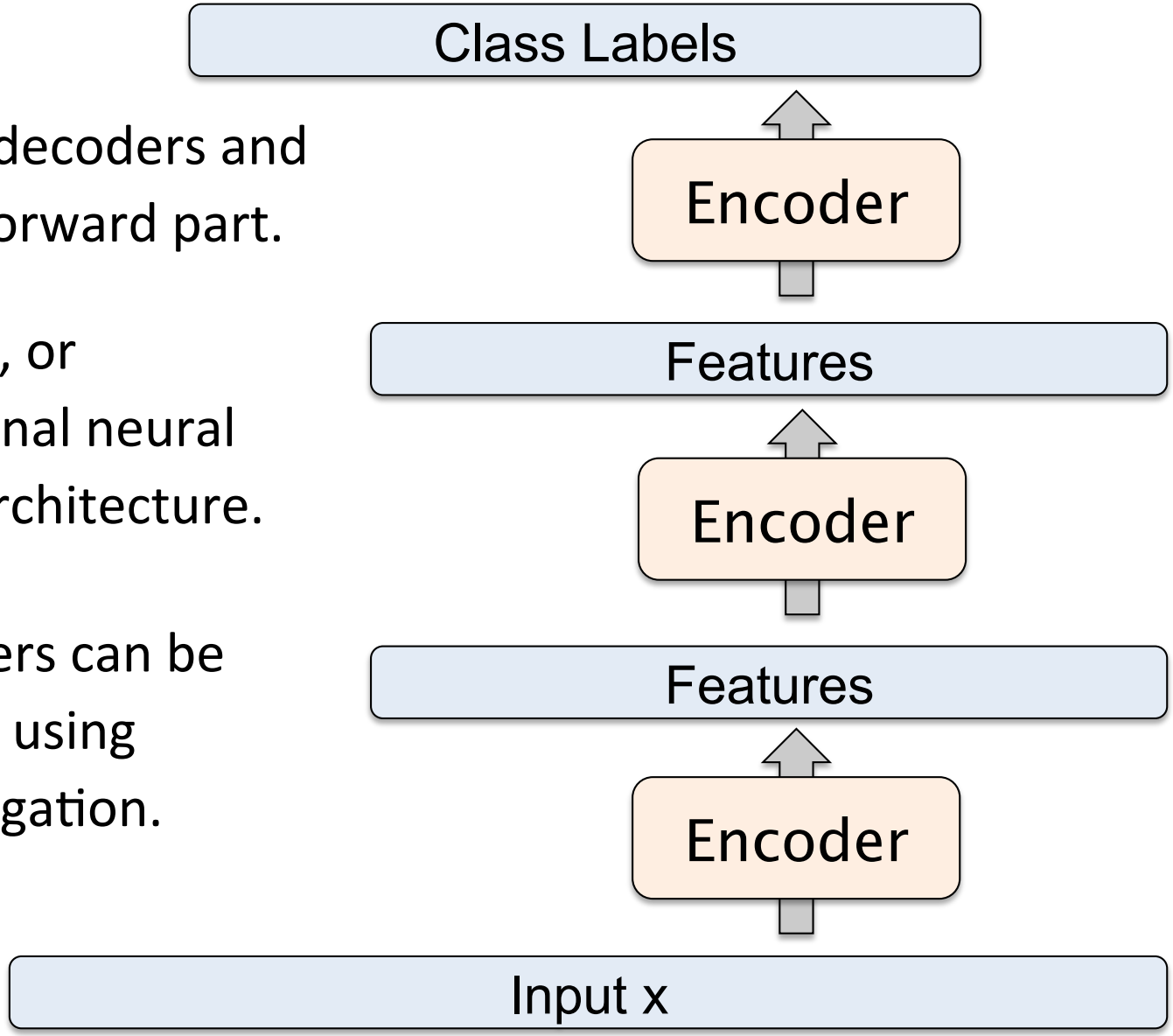


Stacked Autoencoders

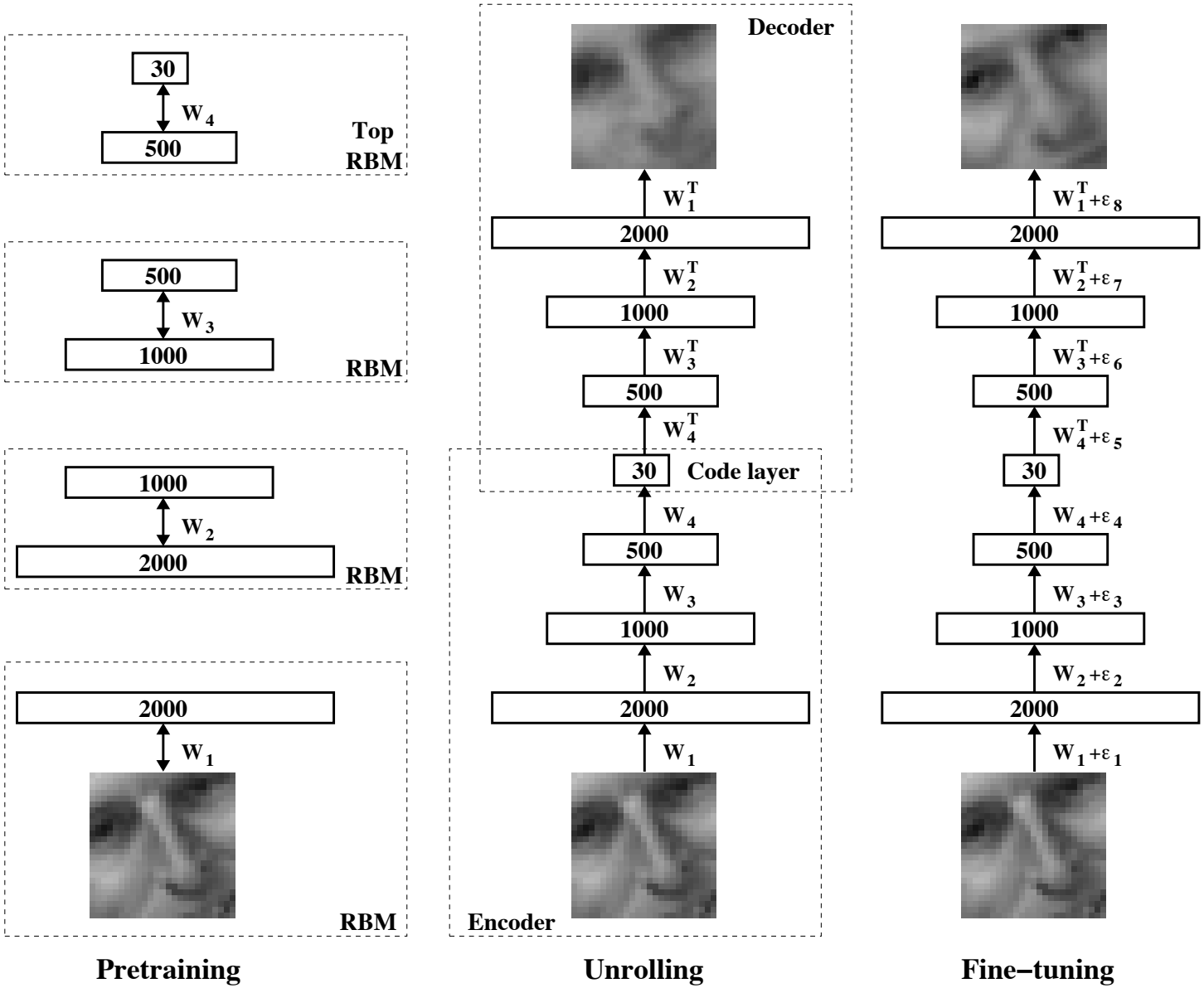


Stacked Autoencoders

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

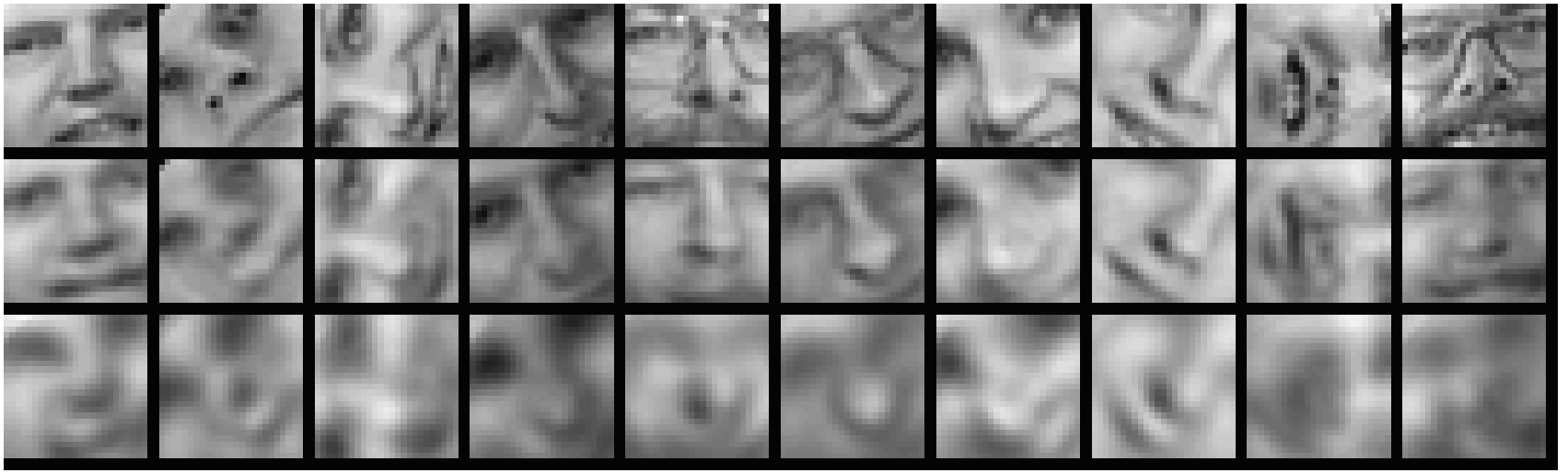


Deep Autoencoders



Deep Autoencoders

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

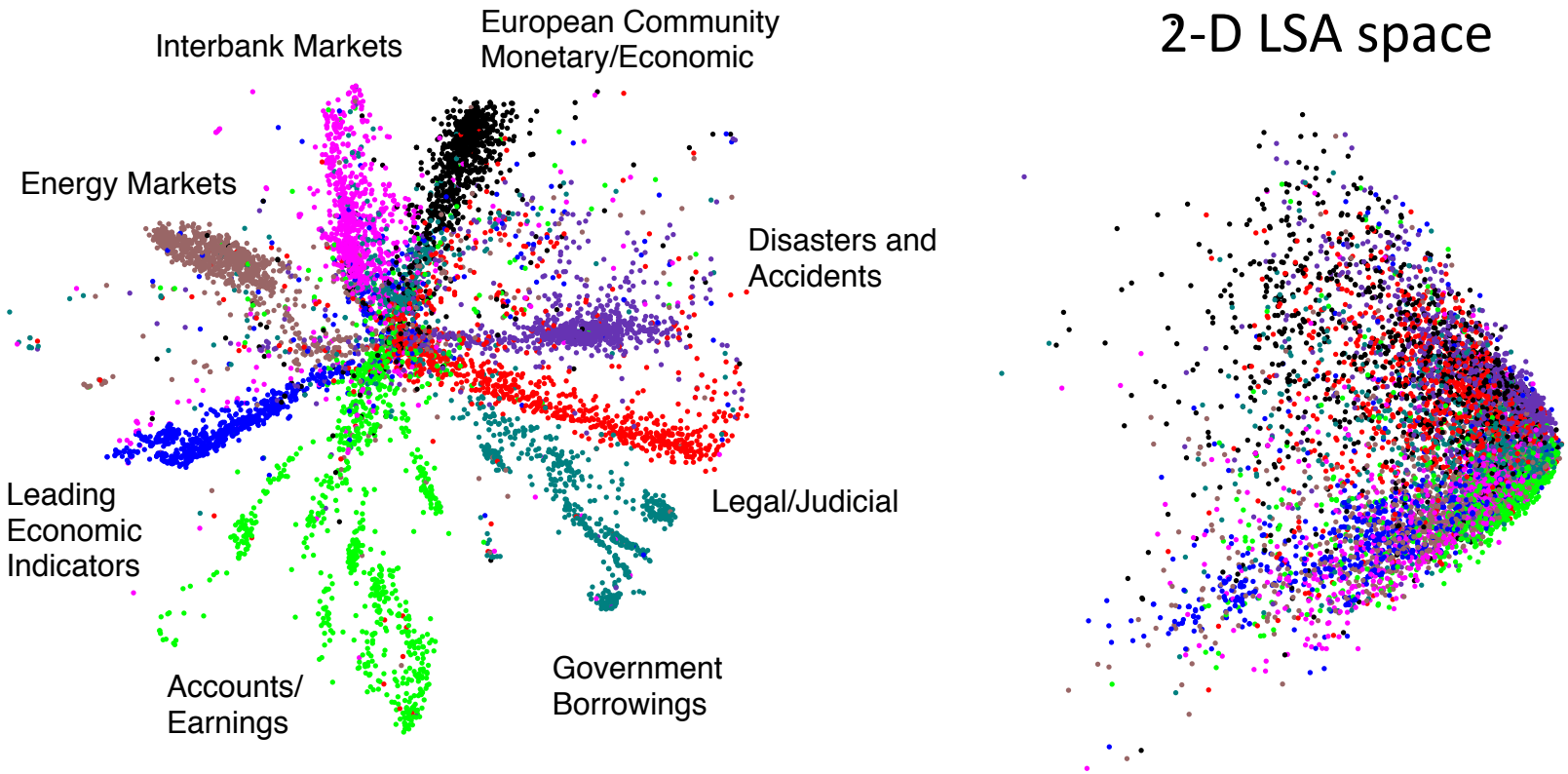


• **Top**: Random samples from the test dataset.

• **Middle**: Reconstructions by the 30-dimensional deep autoencoder.

• **Bottom**: Reconstructions by the 30-dimentinoal PCA.

Information Retrieval

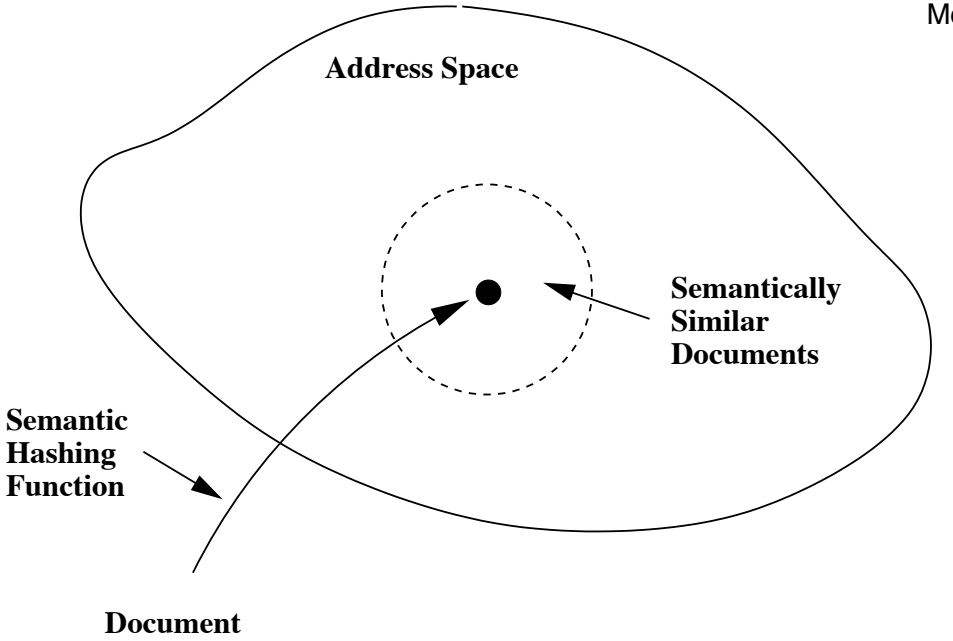


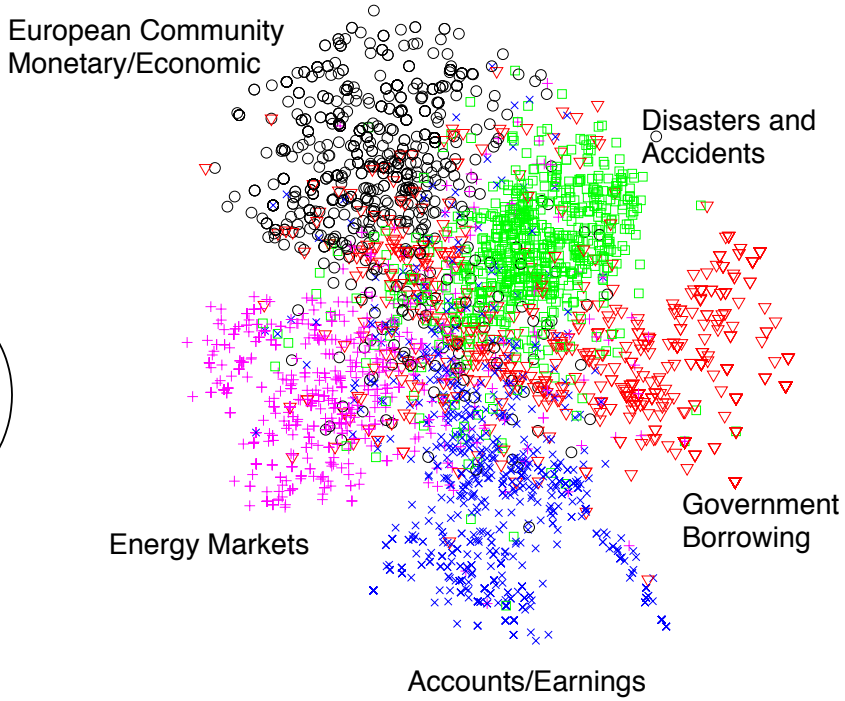
- The Reuters Corpus Volume II contains 804,414 newswire stories (randomly split into **402,207 training** and **402,207 test).**

- "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.

(Hinton and Salakhutdinov, Science 2006)

Semantic Hashing



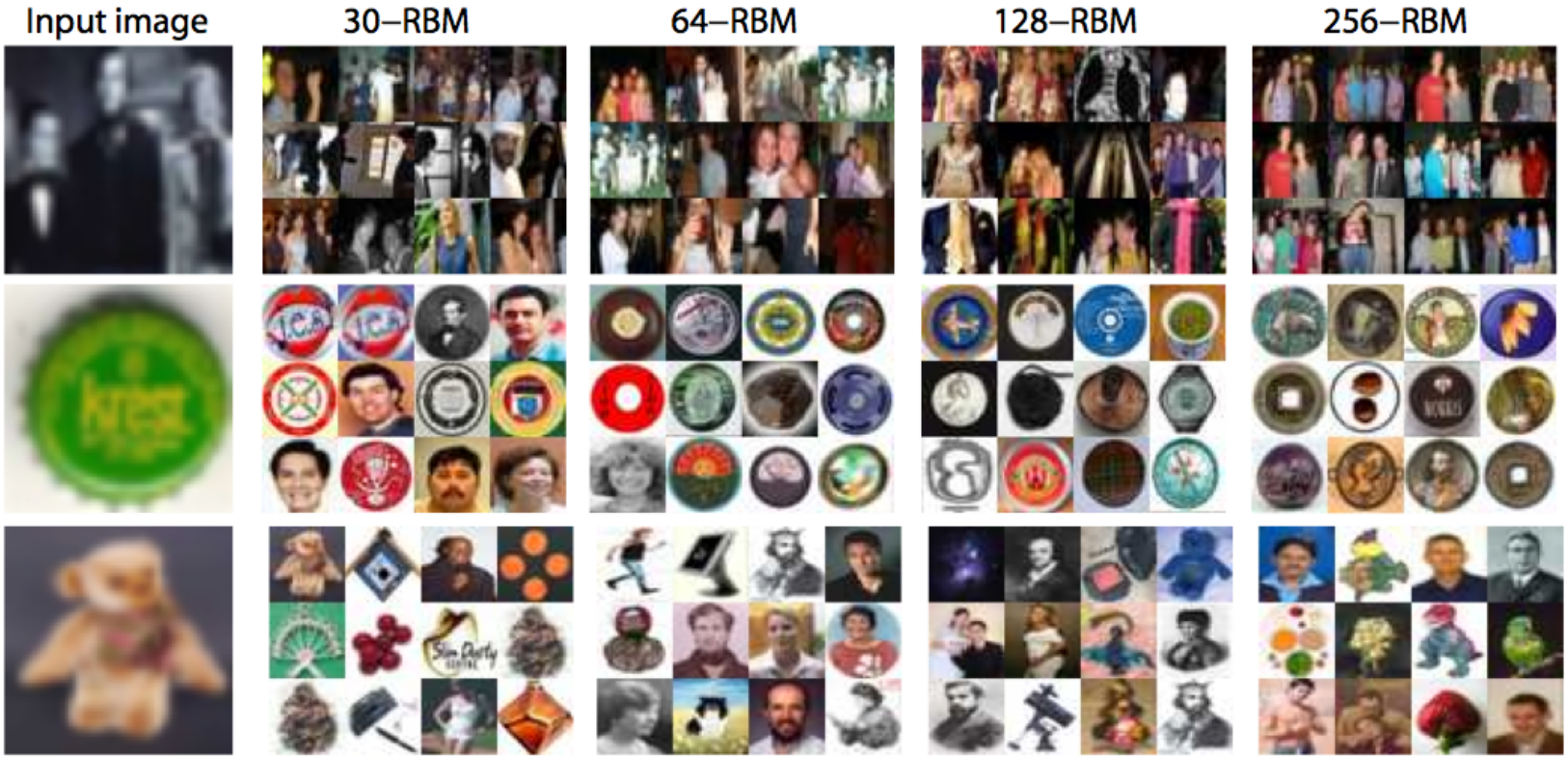


- Learn to map documents into **semantic 20-D binary codes.**
- Retrieve similar documents stored at the nearby addresses **with no search at all.**

(Salakhutdinov and Hinton, SIGIR 2007)

Searching Large Image Database using Binary Codes

- Map images into binary codes for fast retrieval.



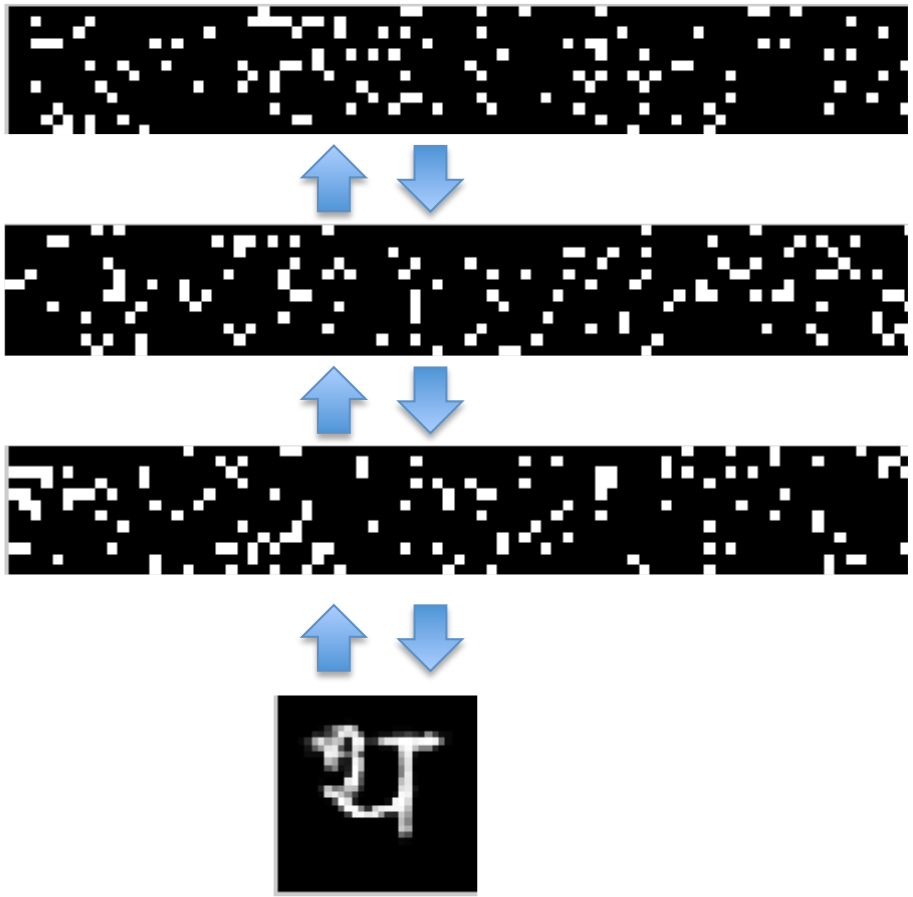
- Small Codes, Torralba, Fergus, Weiss, CVPR 2008
- Spectral Hashing, Y. Weiss, A. Torralba, R. Fergus, NIPS 2008
- Kulis and Darrell, NIPS 2009, Gong and Lazebnik, CVPR 20111
- Norouzi and Fleet, ICML 2011,

Talk Roadmap

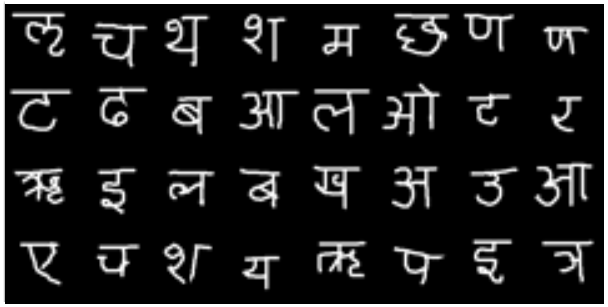
- Basic Building Blocks:
  - Sparse Coding
  - Autoencoders
- Deep Generative Models
  - Restricted Boltzmann Machines
  - Deep Boltzmann Machines
  - Helmholtz Machines / Variational Autoencoders
- Generative Adversarial Networks

Deep Generative Model

Sanskrit



Model P(image)

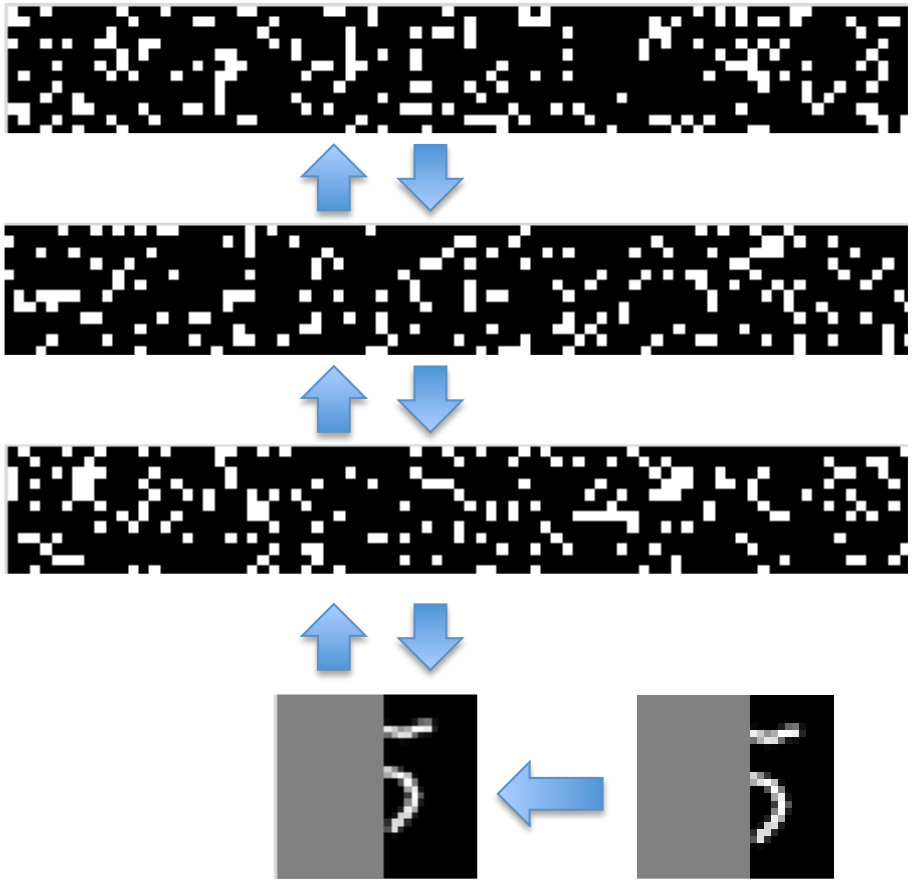


25,000 characters from 50 alphabets around the world.

- 3,000 hidden variables
- 784 observed variables (28 by 28 images)
- About 2 million parameters

Bernoulli Markov Random Field

Deep Generative Model

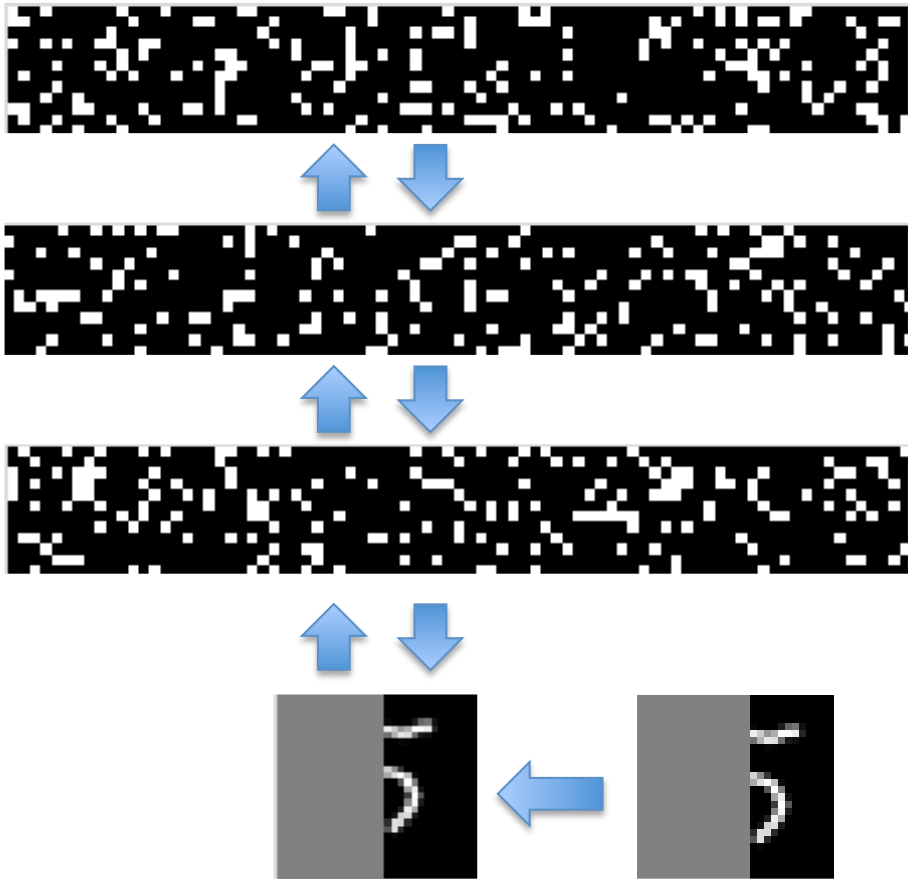


Conditional Simulation

P(image|partial image)

Bernoulli Markov Random Field

Deep Generative Model



P(image|partial image)

Conditional Simulation

Why so difficult?

28

28

$2^{28\times 28}$ possible images!

Bernoulli Markov Random Field

Fully Observed Models

- Explicitly model conditional probabilities:

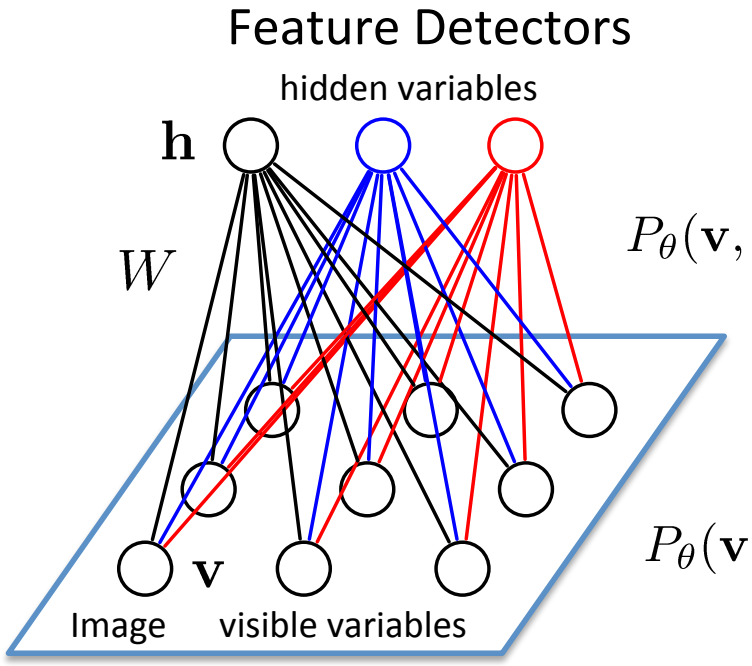
$$p_{\text{model}}(\boldsymbol{x}) = p_{\text{model}}(x_1) \prod_{i=2}^{n} p_{\text{model}}(x_i \mid x_1, \ldots, x_{i-1})$$

Each conditional can be a complicated neural network

- A number of successful models, including
  - NADE, RNADE (Larochelle, et.al. 20011)
  - Pixel CNN (van den Ord et. al. 2016)
  - Pixel RNN (van den Ord et. al. 2016)

Pixel CNN

Restricted Boltzmann Machines



$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\underbrace{\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij} v_i h_j}_{\text{Pair-wise}} + \underbrace{\sum_{i=1}^{D} v_i b_i}_{\text{Unary}} + \underbrace{\sum_{j=1}^{F} h_j a_j}_{\text{Unary}}\right)$$

$$\theta = \{W, a, b\}$$

$$P_\theta(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{D} P_\theta(v_i|\mathbf{h}) = \prod_{i=1}^{D} \frac{1}{1+\exp(-\sum_{j=1}^{F} W_{ij} v_i h_j - b_i)}$$

RBM is a Markov Random Field with:

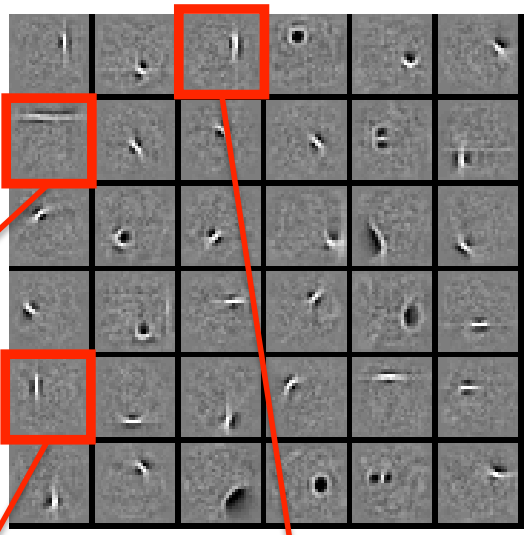
- Stochastic binary visible variables $\mathbf{v} \in \{0,1\}^D$.
- Stochastic binary hidden variables $\mathbf{h} \in \{0,1\}^F$.
- Bipartite connections.

Markov random fields, Boltzmann machines, log-linear models.

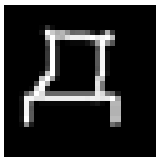
Learning Features

Observed Data
Subset of 25,000 characters

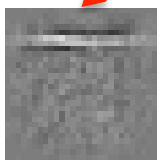
Learned W: "edges"
Subset of 1000 features



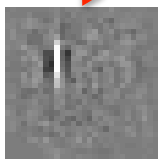
New Image:



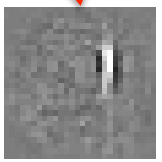
$$= \sigma\left(0.99 \times \right.$$



$$+ \quad 0.97 \times$$



$$+ \quad 0.82 \times$$



$$\left. \cdots \right)$$

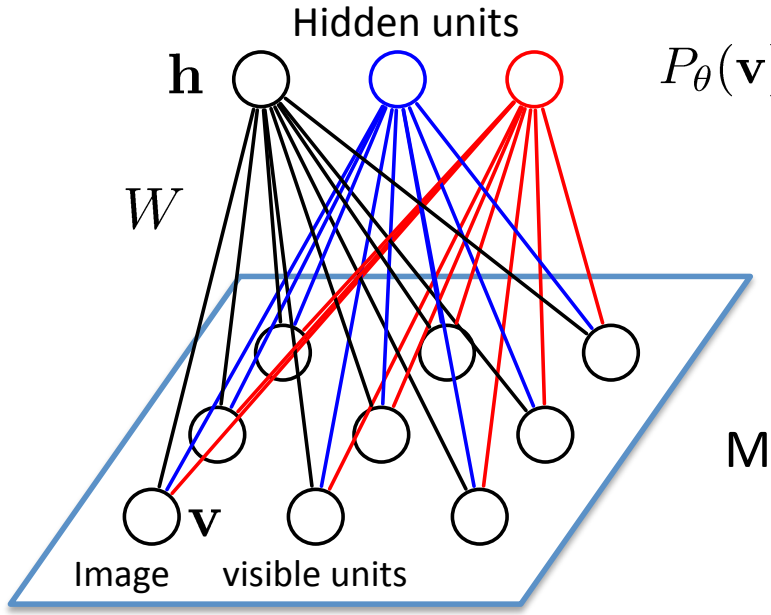
$p(h_7 = 1|v)$

$p(h_{29} = 1|v)$

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

Logistic Function: Suitable for modeling binary images

Model Learning



$$P_\theta(\mathbf{v}) = \frac{P^*(\mathbf{v})}{\mathcal{Z}(\theta)} = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} \exp\left[\mathbf{v}^\top W \mathbf{h} + \mathbf{a}^\top \mathbf{h} + \mathbf{b}^\top \mathbf{v}\right]$$

Given a set of *i.i.d.* training examples $\mathcal{D} = \{\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, ..., \mathbf{v}^{(N)}\}$, we want to learn model parameters $\theta = \{W, a, b\}$.

Maximize log-likelihood objective:

$$L(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log P_\theta(\mathbf{v}^{(n)})$$

Derivative of the log-likelihood:

$$\frac{\partial L(\theta)}{\partial W_{ij}} = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial}{\partial W_{ij}} \log\left(\sum_{\mathbf{h}} \exp\left[\mathbf{v}^{(n)\top} W \mathbf{h} + \mathbf{a}^\top \mathbf{h} + \mathbf{b}^\top \mathbf{v}^{(n)}\right]\right) - \frac{\partial}{\partial W_{ij}} \log \mathcal{Z}(\theta)$$

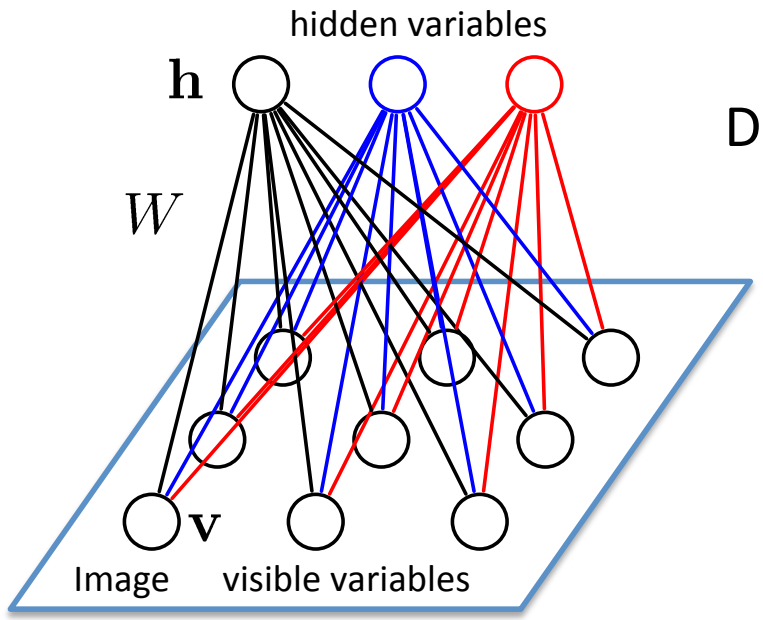
$$= \mathrm{E}_{P_{data}}[v_i h_j] - \underbrace{\mathrm{E}_{P_\theta}[v_i h_j]}$$

Difficult to compute: exponentially many configurations

$$P_{data}(\mathbf{v}, \mathbf{h}; \theta) = P(\mathbf{h}|\mathbf{v}; \theta) P_{data}(\mathbf{v})$$

$$P_{data}(\mathbf{v}) = \frac{1}{N} \sum_{n} \delta(\mathbf{v} - \mathbf{v}^{(n)})$$

Model Learning



Derivative of the log-likelihood:

$$\frac{\partial L(\theta)}{\partial W_{ij}} = \mathbb{E}_{P_{data}}[v_i h_j] - \mathbb{E}_{P_\theta}[v_i h_j]$$

$$\sum_{\mathbf{v},\mathbf{h}} v_i h_j P_\theta(\mathbf{v},\mathbf{h})$$

Easy to compute exactly

Difficult to compute: exponentially many configurations.

Use MCMC

$$P_{data}(\mathbf{v},\mathbf{h};\theta) = P(\mathbf{h}|\mathbf{v};\theta)P_{data}(\mathbf{v})$$

$$P_{data}(\mathbf{v}) = \frac{1}{N}\sum_n \delta(\mathbf{v}-\mathbf{v}^{(n)})$$

Approximate maximum likelihood learning

Approximate Learning

- An approximation to the gradient of the log-likelihood objective:

$$\frac{\partial L(\theta)}{\partial W_{ij}} = \mathbb{E}_{P_{data}}[v_i h_j] - \mathbb{E}_{P_\theta}[v_i h_j]$$

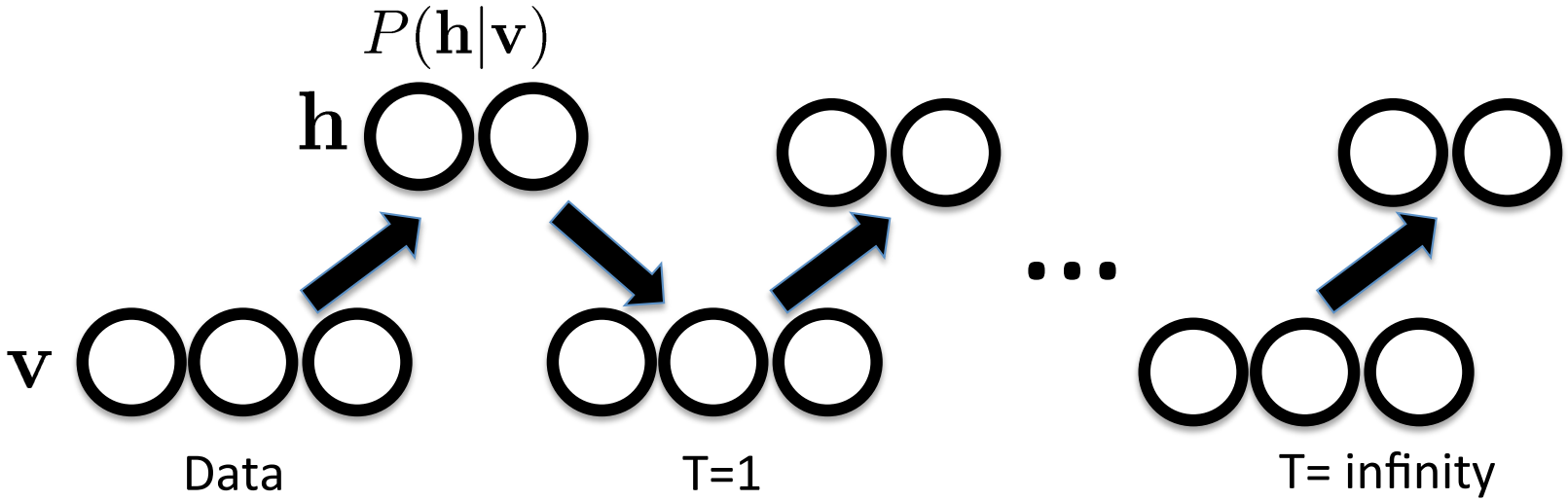
$$\sum_{\mathbf{v},\mathbf{h}} v_i h_j P_\theta(\mathbf{v},\mathbf{h})$$

- Replace the average over all possible input configurations by samples.
- Run MCMC chain (Gibbs sampling) starting from the observed examples.

- Initialize $v^0 = v$
- Sample h^0 from $P(h \mid v^0)$
- For t=1:T
  - Sample v^t from $P(v \mid h^{t-1})$
  - Sample h^t from $P(h \mid v^t)$

Approximate ML Learning for RBMs

Run Markov chain (alternating Gibbs Sampling):



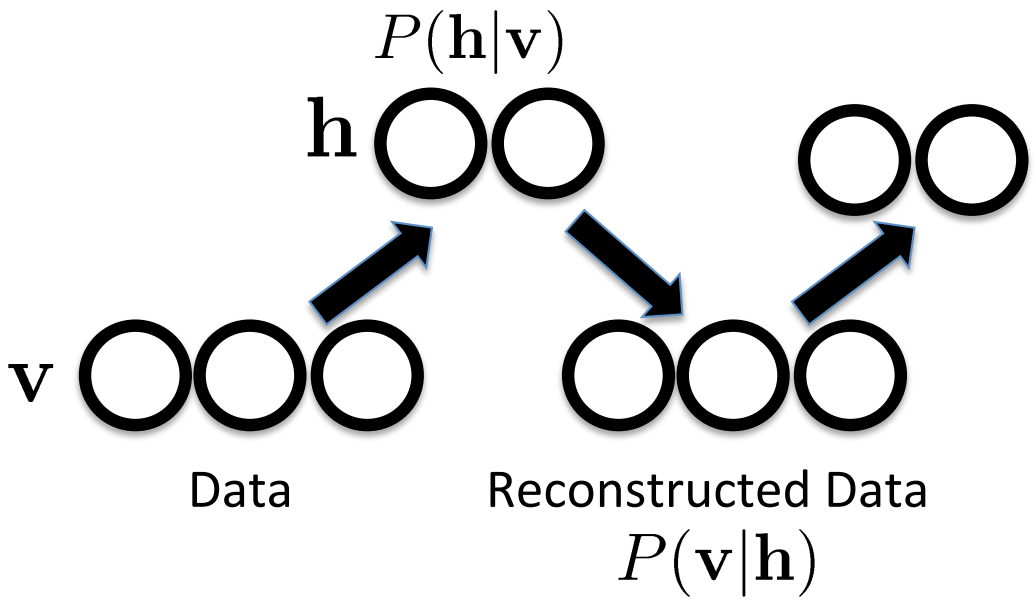
Equilibrium Distribution

$$P(\mathbf{h}|\mathbf{v}) = \prod_j P(h_j|\mathbf{v}) \quad P(h_j = 1|\mathbf{v}) = \frac{1}{1 + \exp(-\sum_i W_{ij} v_i - a_j)}$$

$$P(\mathbf{v}|\mathbf{h}) = \prod_i P(v_i|\mathbf{h}) \quad P(v_i = 1|\mathbf{h}) = \frac{1}{1 + \exp(-\sum_j W_{ij} h_j - b_i)}$$

Contrastive Divergence

A quick way to learn RBM:



- Start with a training vector on the visible units.
- Update all the hidden units in parallel.
- Update the all the visible units in parallel to get a "reconstruction".
- Update the hidden units again.

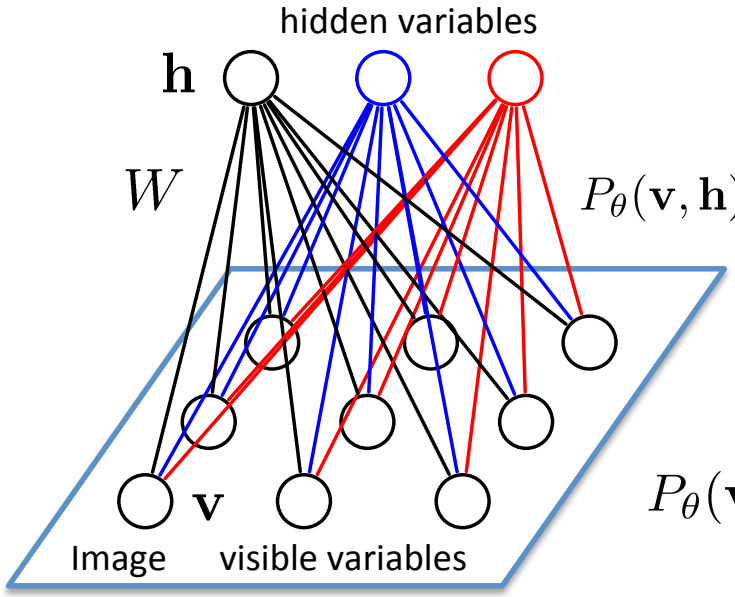
Update model parameters:

$$\Delta W_{ij} = \mathrm{E}_{P_{data}}[v_i h_j] - \mathrm{E}_{P_1}[v_i h_j]$$

Implementation: ~10 lines of Matlab code.

Hinton, Neural Computation 2002

RBMs for Real-valued Data



Pair-wise Unary

$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij} h_j \frac{v_i}{\sigma_i} + \sum_{i=1}^{D} \frac{(v_i - b_i)^2}{2\sigma_i^2} + \sum_{j=1}^{F} a_j h_j\right)$$

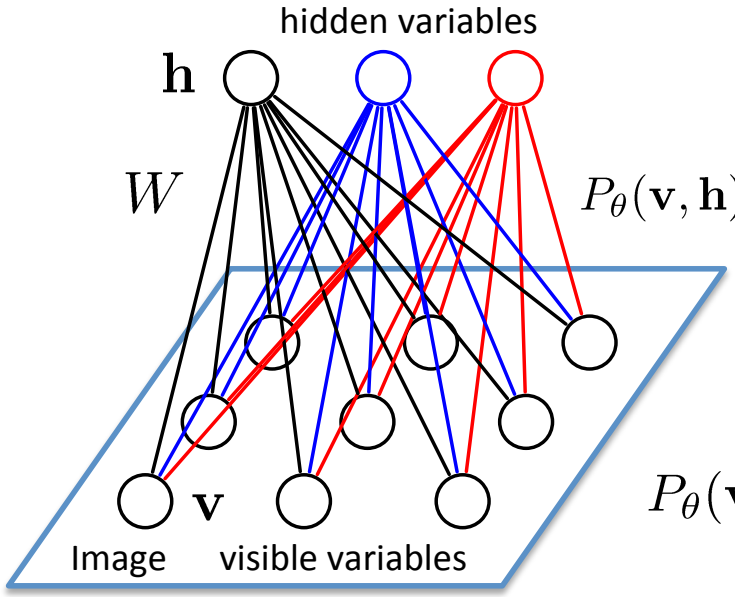
$$\theta = \{W, a, b\}$$

$$P_\theta(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{D} P_\theta(v_i|\mathbf{h}) = \prod_{i=1}^{D} \mathcal{N}\left(b_i + \sum_{j=1}^{F} W_{ij} h_j, \sigma_i^2\right)$$

Gaussian-Bernoulli RBM:

- Stochastic real-valued visible variables $\mathbf{v} \in \mathbb{R}^D$.
- Stochastic binary hidden variables $\mathbf{h} \in \{0,1\}^F$.
- Bipartite connections.

RBMs for Real-valued Data



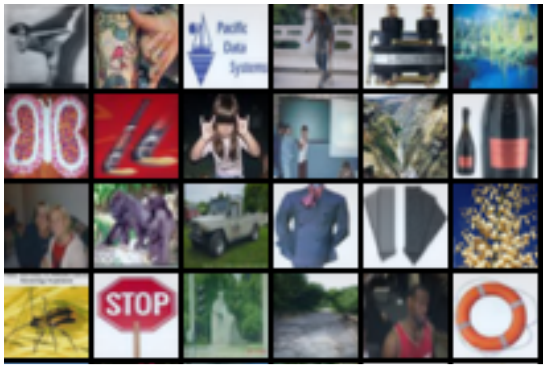
Pair-wise **Unary**

$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij} h_j \frac{v_i}{\sigma_i} + \sum_{i=1}^{D} \frac{(v_i - b_i)^2}{2\sigma_i^2} + \sum_{j=1}^{F} a_j h_j\right)$$

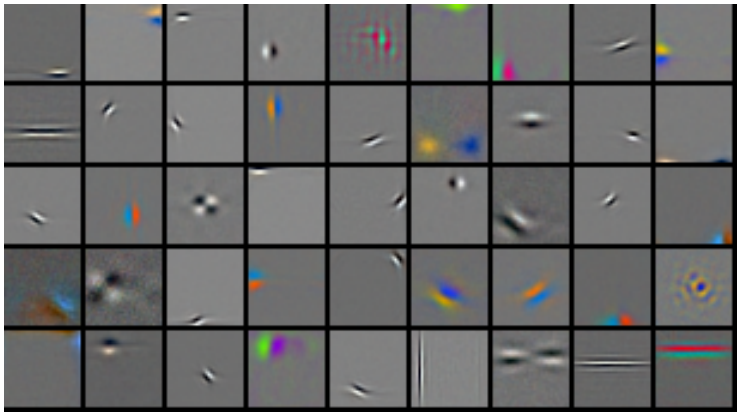
$$\theta = \{W, a, b\}$$

$$P_\theta(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{D} P_\theta(v_i|\mathbf{h}) = \prod_{i=1}^{D} \mathcal{N}\left(b_i + \sum_{j=1}^{F} W_{ij} h_j, \sigma_i^2\right)$$

4 million **unlabelled** images

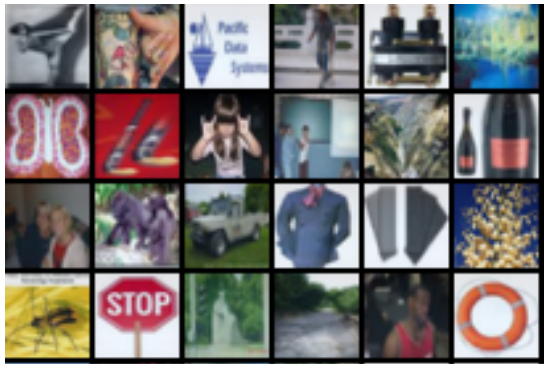


Learned features (out of 10,000)

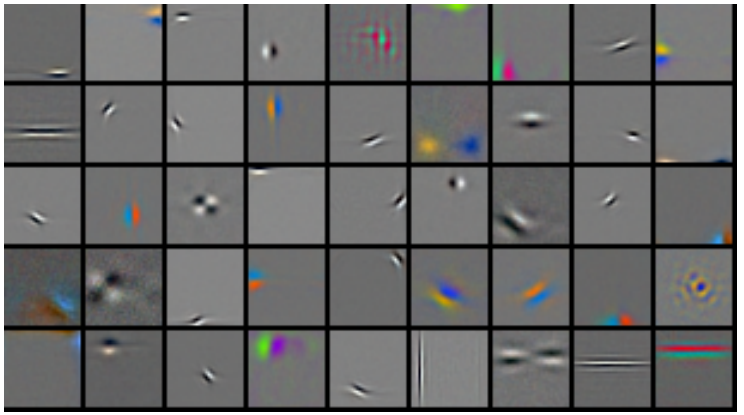


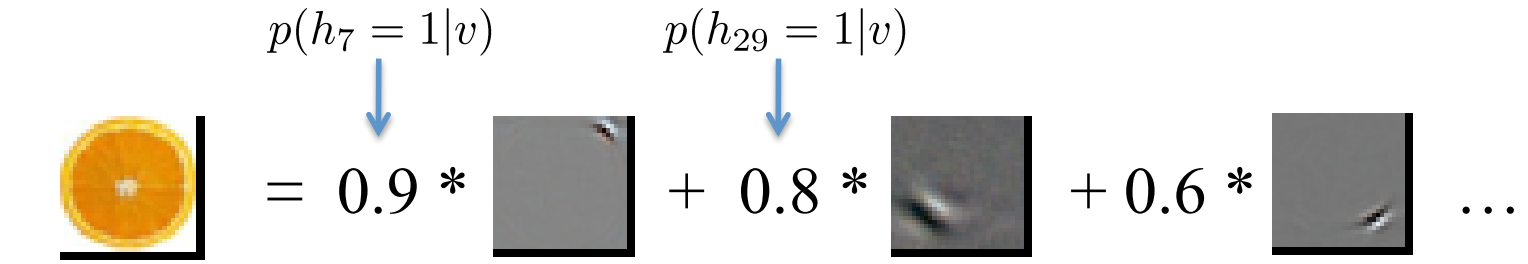
RBMs for Real-valued Data

4 million **unlabelled** images

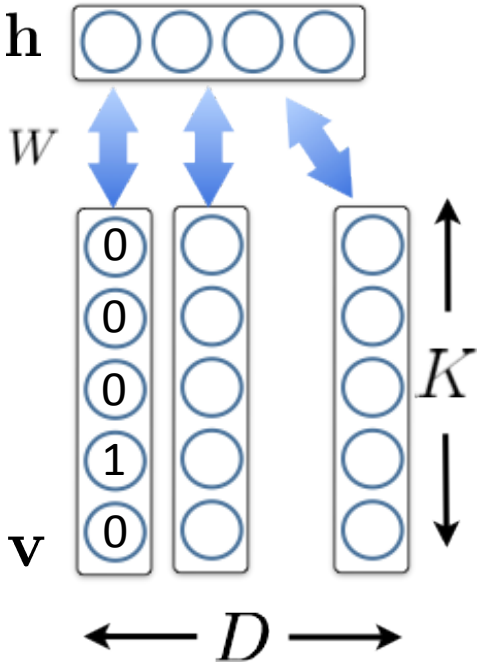


Learned features (out of 10,000)





RBMs for Word Counts



Pair-wise Unary

$$P_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\sum_{i=1}^{D} \sum_{k=1}^{K} \sum_{j=1}^{F} W_{ij}^k v_i^k h_j + \sum_{i=1}^{D} \sum_{k=1}^{K} v_i^k b_i^k + \sum_{j=1}^{F} h_j a_j \right)$$

$$\theta = \{W, a, b\}$$

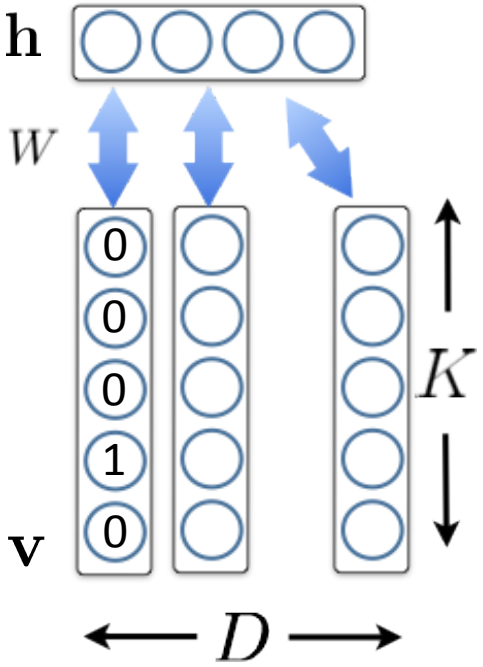
$$P_\theta(v_i^k = 1 | \mathbf{h}) = \frac{\exp\left(b_i^k + \sum_{j=1}^{F} h_j W_{ij}^k\right)}{\sum_{q=1}^{K} \exp\left(b_i^q + \sum_{j=1}^{F} h_j W_{ij}^q\right)}$$

Replicated Softmax Model: undirected topic model:

- Stochastic 1-of-K visible variables.
- Stochastic binary hidden variables $\mathbf{h} \in \{0, 1\}^F$.
- Bipartite connections.

(Salakhutdinov & Hinton, NIPS 2010, Srivastava & Salakhutdinov, NIPS 2012)

RBMs for Word Counts



Pair-wise **Unary**

$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\sum_{i=1}^{D}\sum_{k=1}^{K}\sum_{j=1}^{F} W_{ij}^k v_i^k h_j + \sum_{i=1}^{D}\sum_{k=1}^{K} v_i^k b_i^k + \sum_{j=1}^{F} h_j a_j\right)$$

$$\theta = \{W, a, b\}$$

$$P_\theta(v_i^k = 1|\mathbf{h}) = \frac{\exp\left(b_i^k + \sum_{j=1}^{F} h_j W_{ij}^k\right)}{\sum_{q=1}^{K}\exp\left(b_i^q + \sum_{j=1}^{F} h_j W_{ij}^q\right)}$$

Reuters dataset:
804,414 **unlabeled**
newswire stories
Bag-of-Words

Learned features: ``topics''

russian	clinton	computer	trade	stock
russia	house	system	country	wall
moscow	president	product	import	street
yeltsin	bill	software	world	point
soviet	congress	develop	economy	dow

RBMs for Word Counts

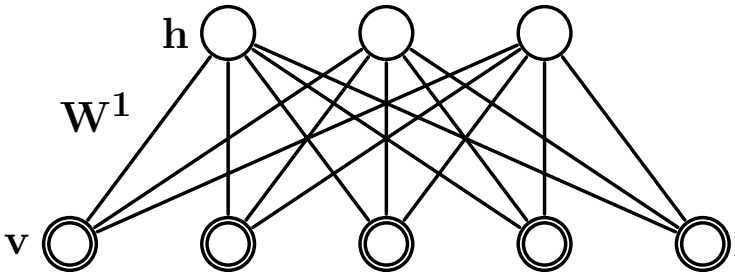
One-step reconstruction from the Replicated Softmax model.

Input	**Reconstruction**
chocolate, cake	cake, chocolate, sweets, dessert, cupcake, food, sugar, cream, birthday
nyc	nyc, newyork, brooklyn, queens, gothamist, manhattan, subway, streetart
dog	dog, puppy, perro, dogs, pet, filmshots, tongue, pets, nose, animal
flower, high, 花	flower, 花, high, japan, sakura, 日本, blossom, tokyo, lily, cherry
girl, rain, station, norway	norway, station, rain, girl, oslo, train, umbrella, wet, railway, weather
fun, life, children	children, fun, life, kids, child, playing, boys, kid, play, love
forest, blur	forest, blur, woods, motion, trees, movement, path, trail, green, focus
españa, agua, granada	españa, agua, spain, granada, water, andalucía, naturaleza, galicia, nieve

Collaborative Filtering

$$P_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\sum_{ijk} W_{ij}^k v_i^k h_j + \sum_{ik} b_i^k v_i^k + \sum_j a_j h_j \right)$$

Binary hidden: user preferences



Multinomial visible: user ratings

Netflix dataset:
480,189 users
17,770 movies
Over 100 million ratings

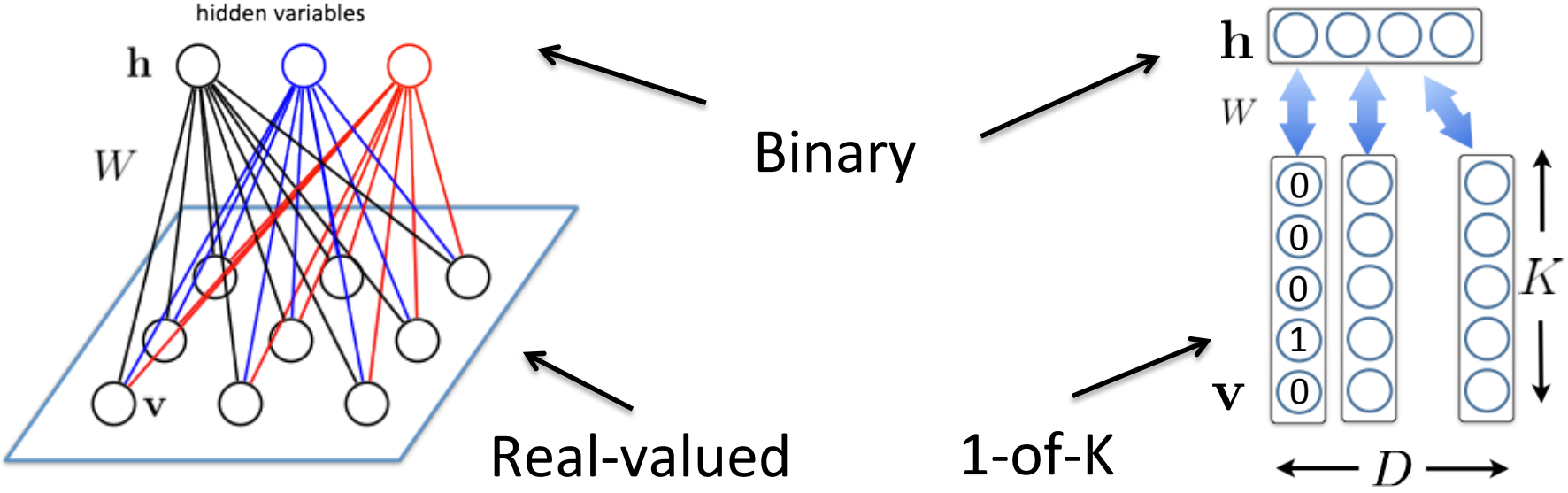
Learned features: ``genre''

Fahrenheit 9/11	Independence Day
Bowling for Columbine	The Day After Tomorrow
The People vs. Larry Flynt	Con Air
Canadian Bacon	Men in Black II
La Dolce Vita	Men in Black
Friday the 13th	Scary Movie
The Texas Chainsaw Massacre	Naked Gun
Children of the Corn	Hot Shots!
Child's Play	American Pie
The Return of Michael Myers	Police Academy

(Salakhutdinov, Mnih, Hinton, ICML 2007)

Different Data Modalities

- Binary/Gaussian/Softmax RBMs: All have binary hidden variables but use them to model different kinds of data.



- It is easy to infer the states of the hidden variables:

$$P_\theta(\mathbf{h}|\mathbf{v}) = \prod_{j=1}^{F} P_\theta(h_j|\mathbf{v}) = \prod_{j=1}^{F} \frac{1}{1+\exp(-a_j - \sum_{i=1}^{D} W_{ij} v_i)}$$

Product of Experts

The joint distribution is given by:

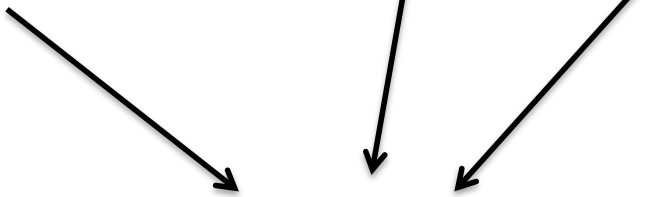
$$P_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\Big(\sum_{ij} W_{ij} v_i h_j + \sum_i b_i v_i + \sum_j a_j h_j\Big)$$

Marginalizing over hidden variables:

Product of Experts

$$P_\theta(\mathbf{v}) = \sum_{\mathbf{h}} P_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \prod_i \exp(b_i v_i) \prod_j \Big(1 + \exp(a_j + \sum_i W_{ij} v_i)\Big)$$

government authority power empire federation	clinton house president bill congress	bribery corruption dishonesty corrupt fraud	mafia business gang mob insider	stock wall street point dow	…



Silvio Berlusconi

Topics “government”, ”corruption” and ”mafia” can combine to give very high probability to a word “Silvio Berlusconi”.

Product of Experts

The joint distribution is given by:

$$P_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\Big(\sum_{ij} W_{ij} v_i h_j + \sum_i b_i v_i + \sum_j a_j h_j\Big)$$

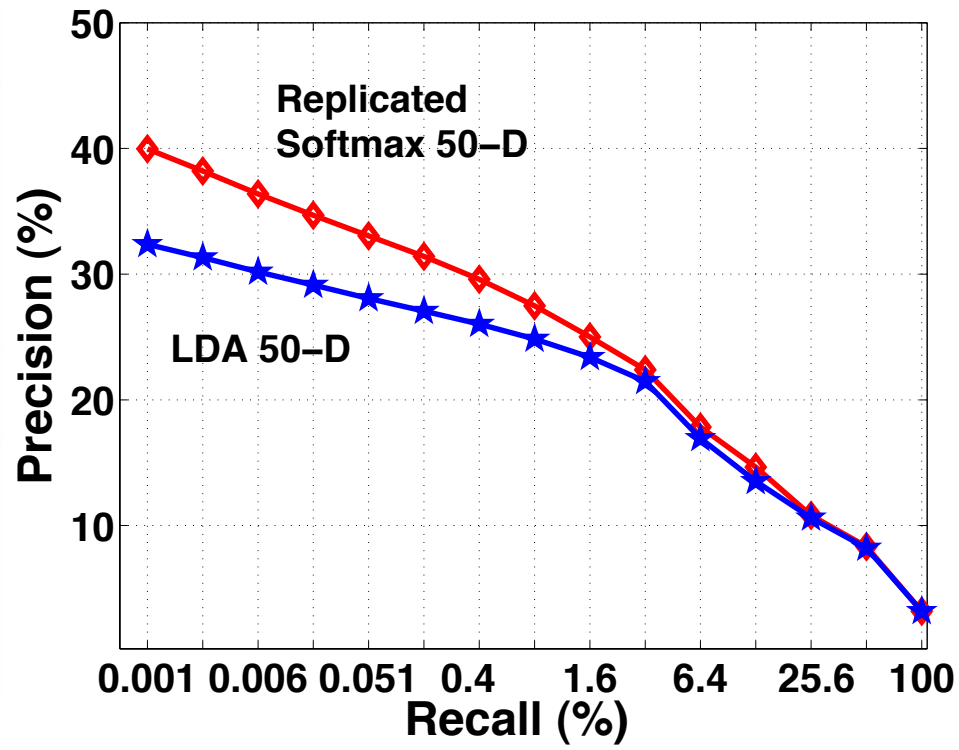
Marginalizing o

$$P_\theta(\mathbf{v}) = \sum_{\mathbf{h}}$$

duct of Experts

$$W_{ij} v_i)\Big)$$

government	clint
authority	hou
power	pres
empire	bill
federation	cong



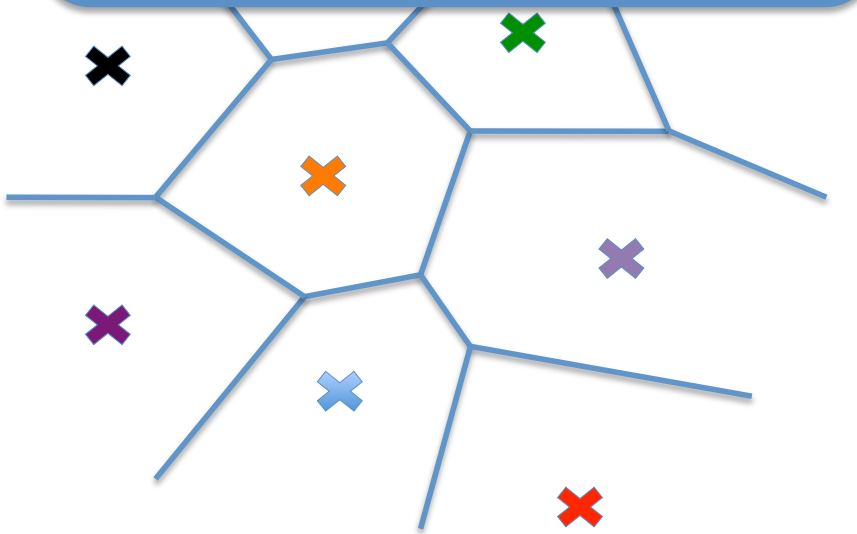
, "corruption"
bine to give very
word "Silvio

Berlusconi".

Local vs. Distributed Representations

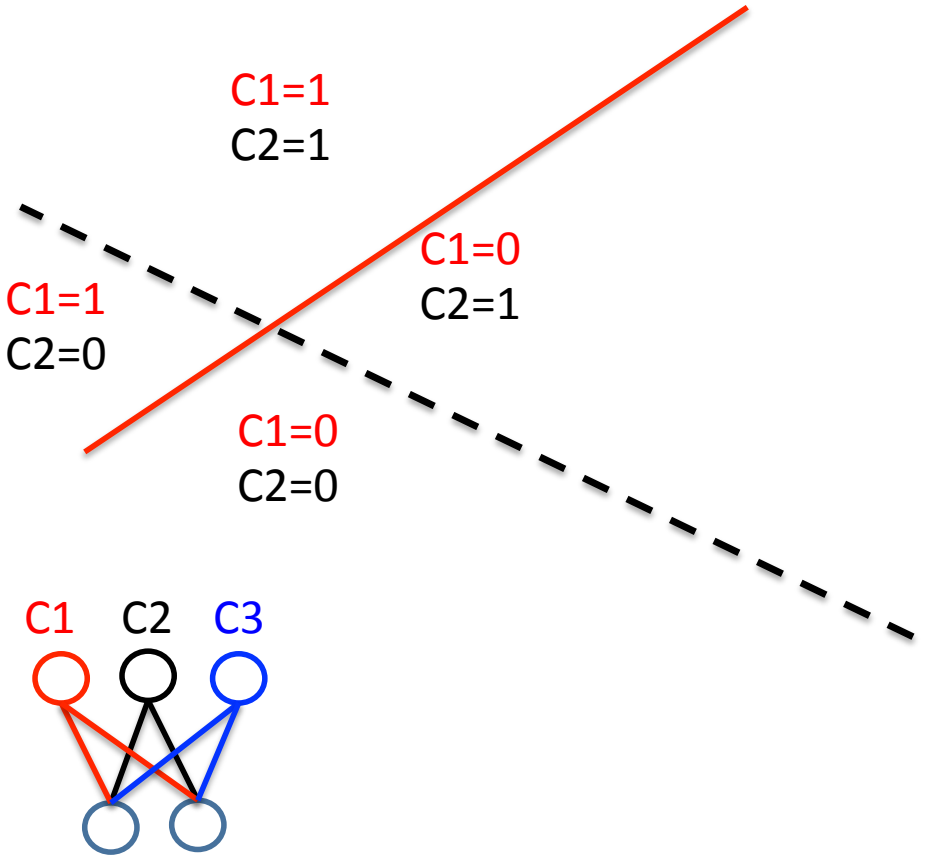
• Clustering, Nearest Neighbors, RBF SVM, local density estimators

• Parameters for each region.
• # of regions is linear with # of parameters.



Learned prototypes

• RBMs, Factor models, PCA, Sparse Coding, Deep models

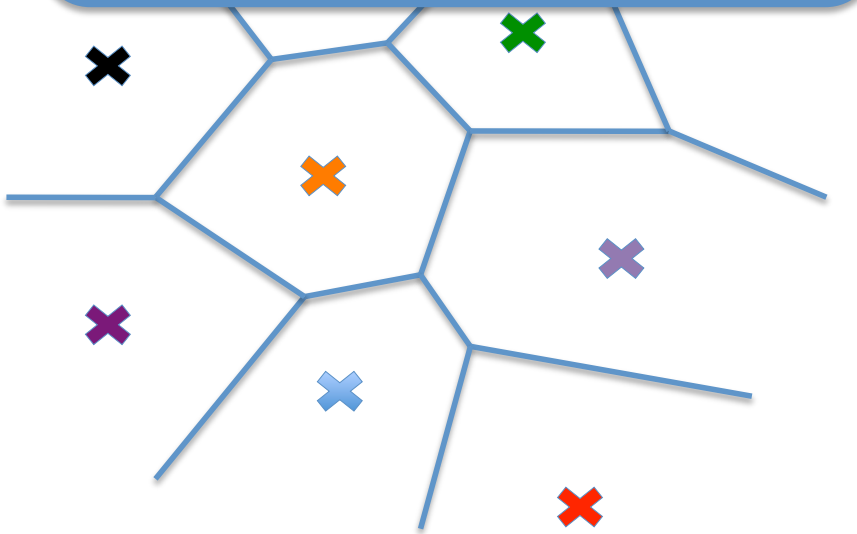


Bengio, 2009, Foundations and Trends in Machine Learning

Local vs. Distributed Representations

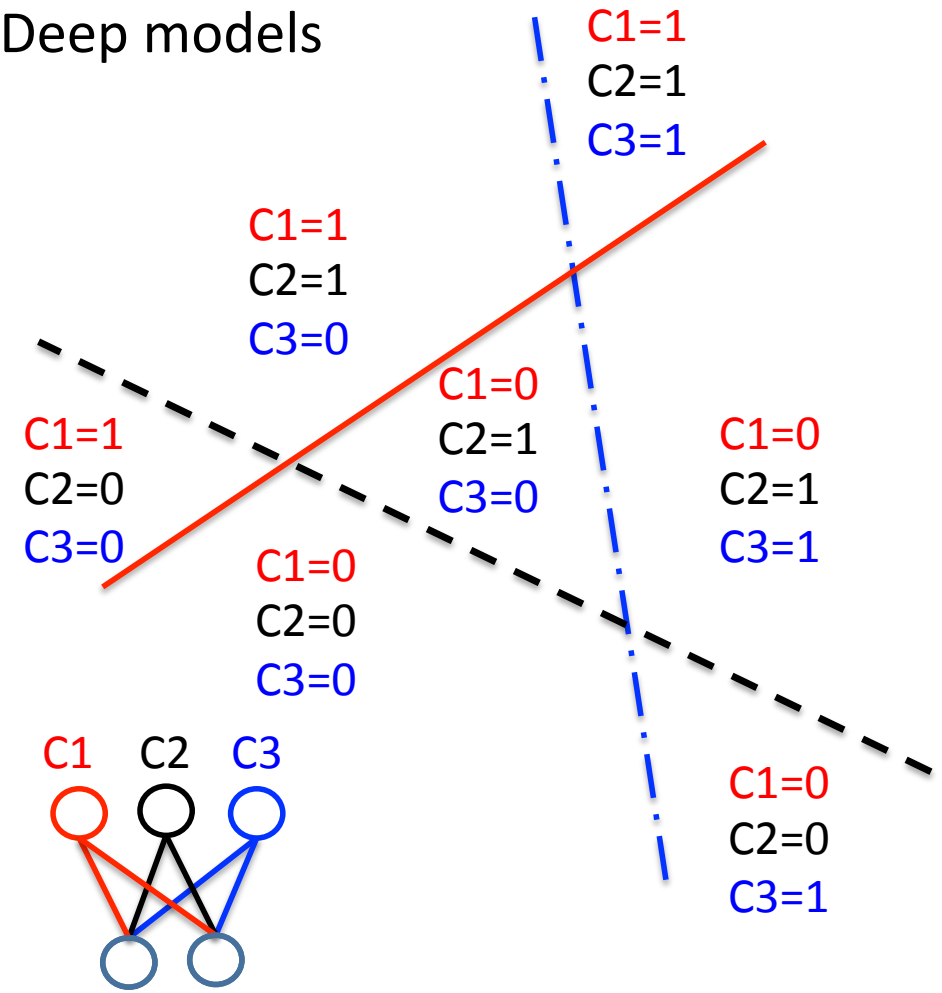
• Clustering, Nearest Neighbors, RBF SVM, local density estimators

• Parameters for each region.
• # of regions is linear with # of parameters.



Learned prototypes

• RBMs, Factor models, PCA, Sparse Coding, Deep models

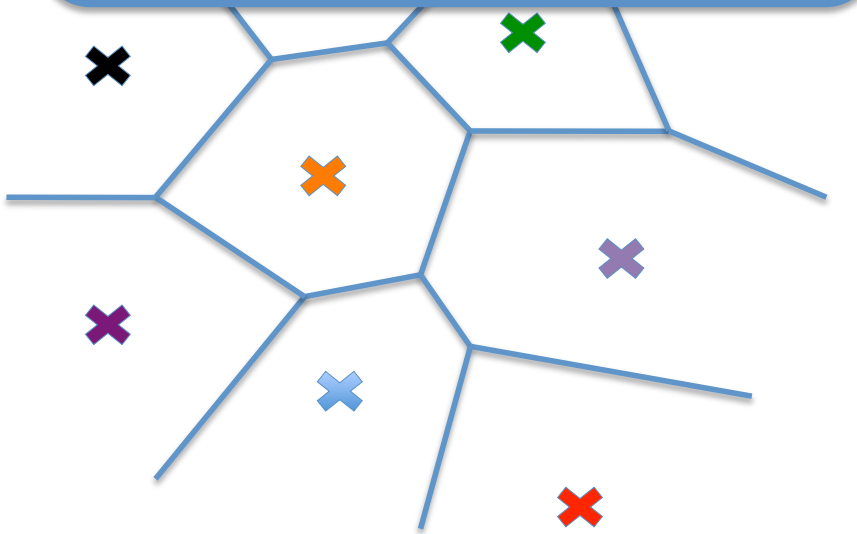


Bengio, 2009, Foundations and Trends in Machine Learning

Local vs. Distributed Representations

- Clustering, Nearest Neighbors, RBF SVM, local density estimators

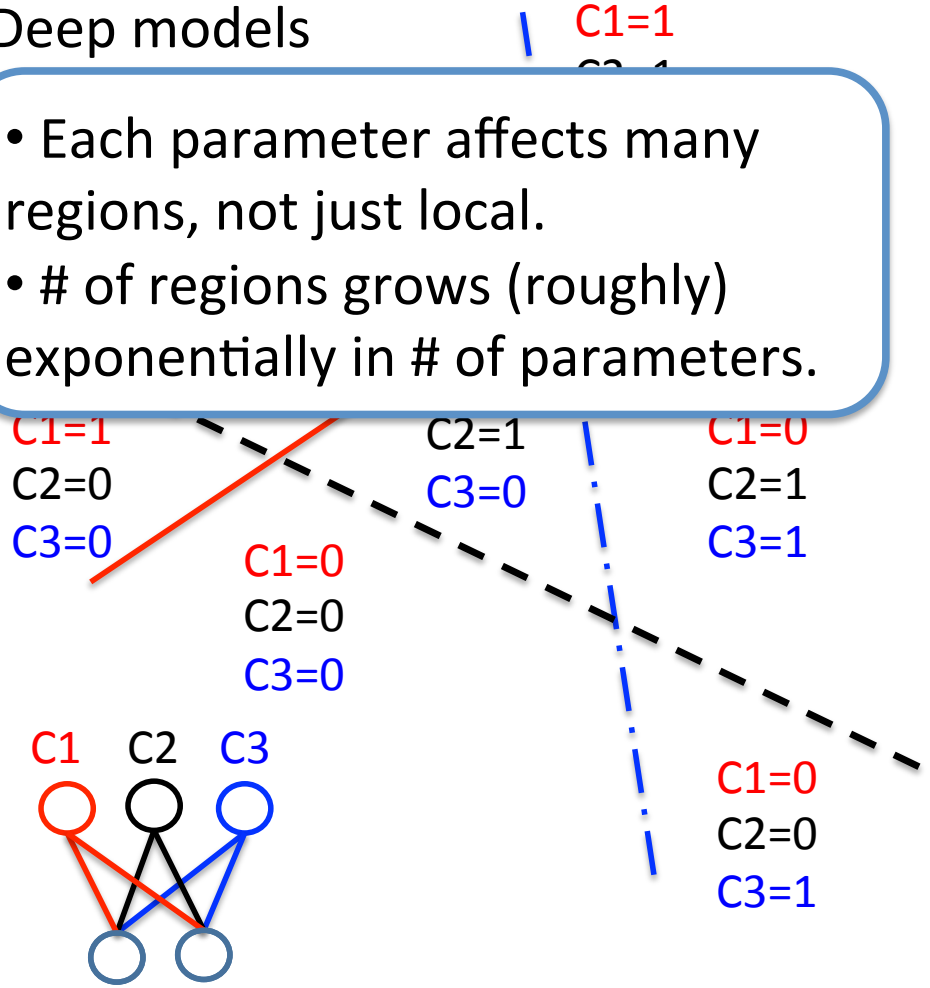
- Parameters for each region.
- # of regions is linear with # of parameters.



Learned prototypes

- RBMs, Factor models, PCA, Sparse Coding, Deep models

- Each parameter affects many regions, not just local.
- # of regions grows (roughly) exponentially in # of parameters.

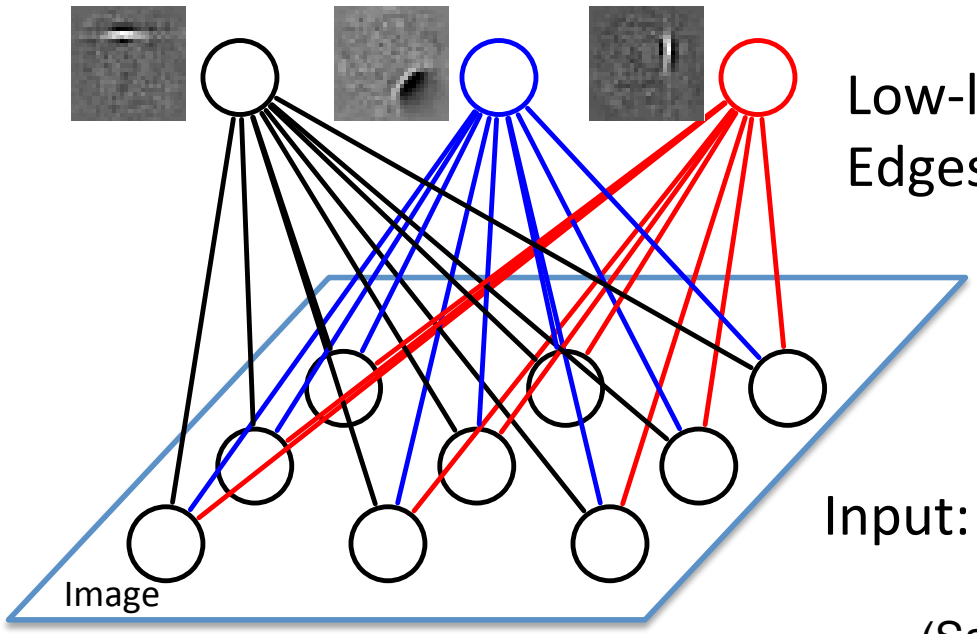


Bengio, 2009, Foundations and Trends in Machine Learning

Talk Roadmap

- Basic Building Blocks (non-probabilistic models):
  - Sparse Coding
  - Autoencoders
- Deep Generative Models
  - Restricted Boltzmann Machines
  - Deep Boltzmann Machines
  - Helmholtz Machines / Variational Autoencoders
- Generative Adversarial Networks

Deep Boltzmann Machines



Low-level features:
Edges

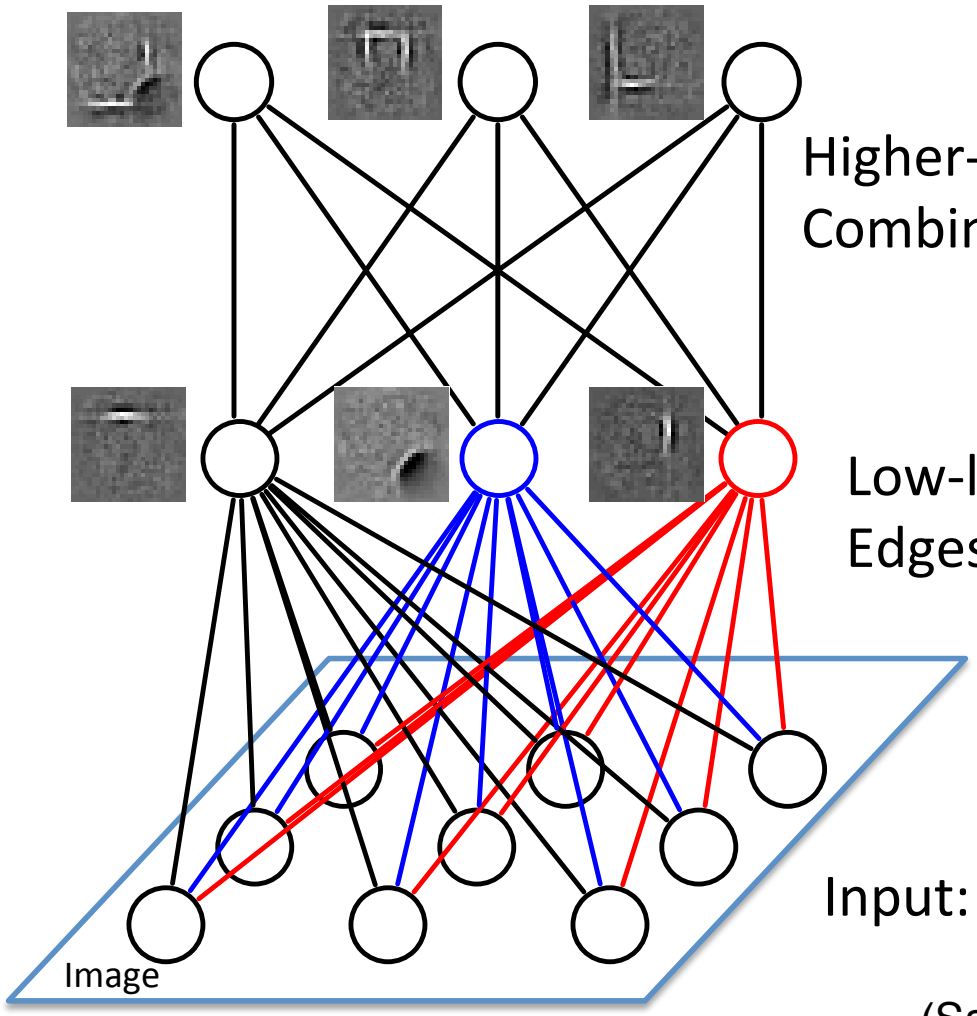
Built from **unlabeled** inputs.

Input: Pixels

(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)

Deep Boltzmann Machines

Learn simpler representations,
then compose more complex ones



Higher-level features:
Combination of edges

Low-level features:
Edges

Input: Pixels

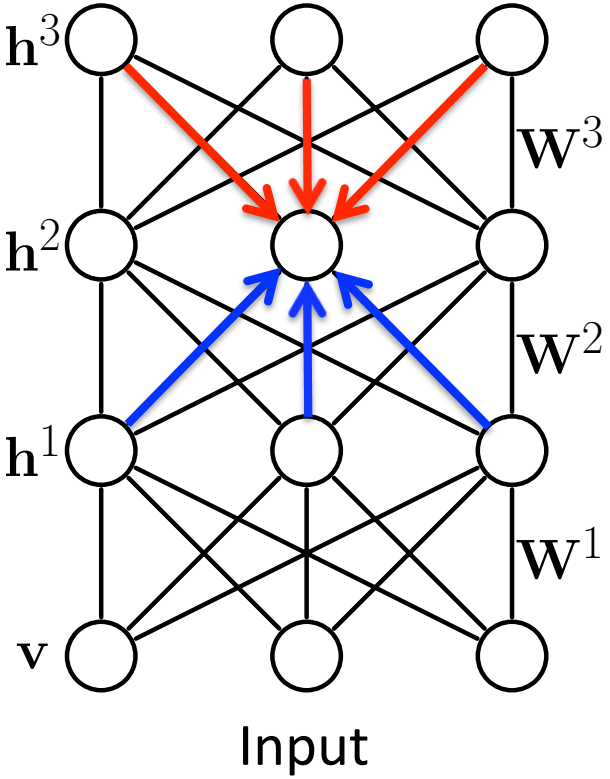
Built from **unlabeled** inputs.

(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)

Model Formulation

$$P_\theta(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left[\mathbf{v}^\top W^{(1)} \mathbf{h}^{(1)} + \mathbf{h}^{(1)\top} W^{(2)} \mathbf{h}^{(2)} + \mathbf{h}^{(2)\top} W^{(3)} \mathbf{h}^{(3)}\right]$$

Same as RBMs



$\theta = \{W^1, W^2, W^3\}$ model parameters

- Dependencies between hidden variables.
- All connections are undirected.
- Bottom-up and Top-down:

$$P(h_j^2 = 1 | \mathbf{h}^1, \mathbf{h}^3) = \sigma\left(\sum_k W_{kj}^3 h_k^3 + \sum_m W_{mj}^2 h_m^1\right)$$

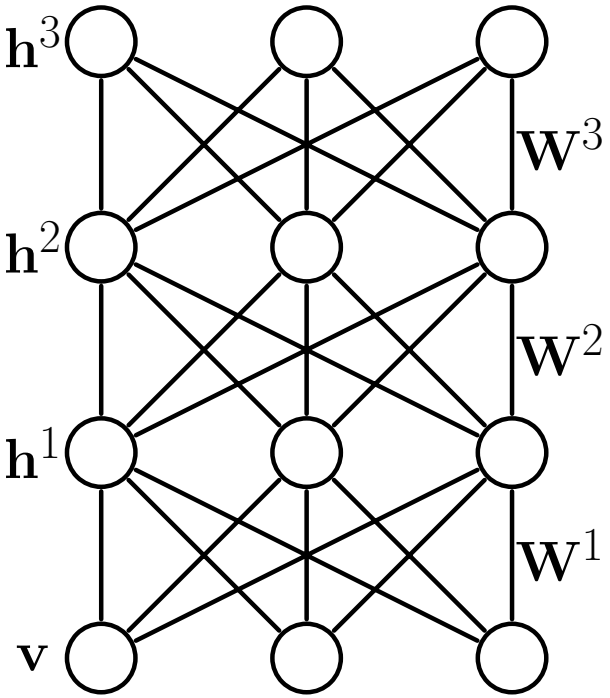
Top-down

Bottom-up

- Hidden variables are dependent even when **conditioned on the input**.

Approximate Learning

$$P_\theta(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left[\mathbf{v}^\top W^{(1)} \mathbf{h}^{(1)} + \mathbf{h}^{(1)\top} W^{(2)} \mathbf{h}^{(2)} + \mathbf{h}^{(2)\top} W^{(3)} \mathbf{h}^{(3)}\right]$$



(Approximate) Maximum Likelihood:

$$\frac{\partial \log P_\theta(\mathbf{v})}{\partial W^1} = \mathbb{E}_{P_{data}}[\mathbf{v}\mathbf{h^1}^\top] - \mathbb{E}_{P_\theta}[\mathbf{v}\mathbf{h^1}^\top]$$

- Both expectations are intractable!

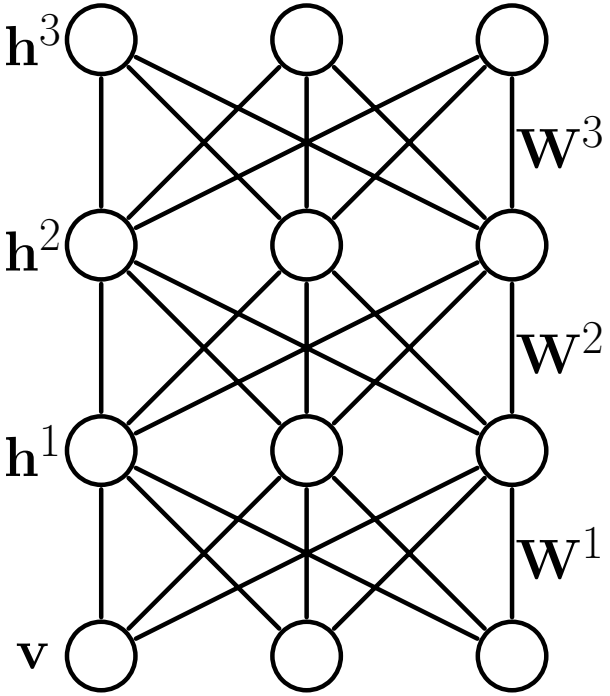
$$P_{data}(\mathbf{v}, \mathbf{h^1}) = P_\theta(\mathbf{h^1}|\mathbf{v}) P_{data}(\mathbf{v})$$

$$P_{data}(\mathbf{v}) = \frac{1}{N} \sum_{n=1}^{N} \delta(\mathbf{v} - \mathbf{v_n})$$

Not factorial any more!

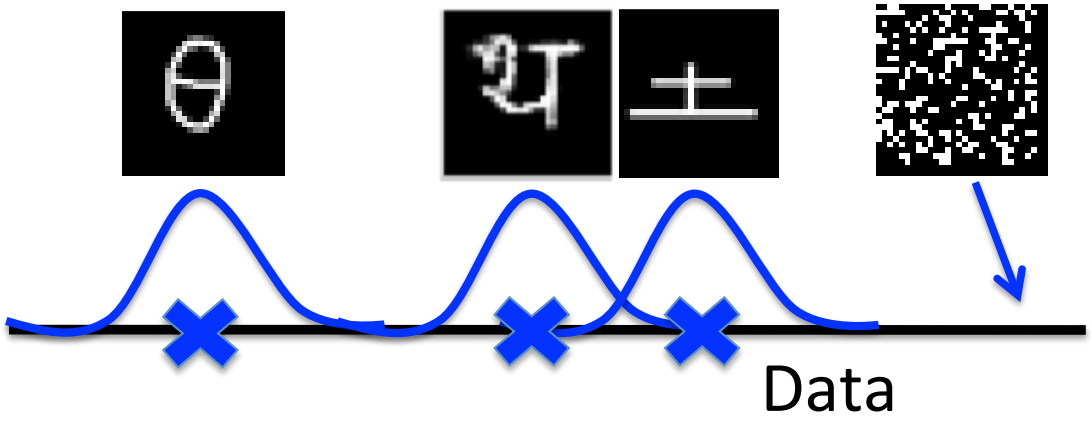
Approximate Learning

$$P_\theta(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left[\mathbf{v}^\top W^{(1)} \mathbf{h}^{(1)} + \mathbf{h}^{(1)\top} W^{(2)} \mathbf{h}^{(2)} + \mathbf{h}^{(2)\top} W^{(3)} \mathbf{h}^{(3)}\right]$$



(Approximate) Maximum Likelihood:

$$\frac{\partial \log P_\theta(\mathbf{v})}{\partial W^1} = \mathbb{E}_{P_{data}}[\mathbf{v}\mathbf{h^1}^\top] - \mathbb{E}_{P_\theta}[\mathbf{v}\mathbf{h^1}^\top]$$



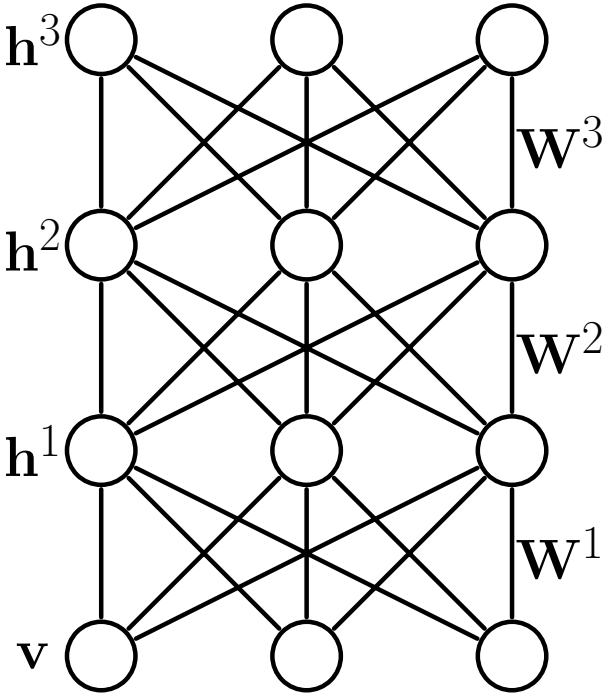
$$P_{data}(\mathbf{v}, \mathbf{h^1}) = P_\theta(\mathbf{h^1}|\mathbf{v}) P_{data}(\mathbf{v})$$

$$P_{data}(\mathbf{v}) = \frac{1}{N} \sum_{n=1}^{N} \delta(\mathbf{v} - \mathbf{v_n})$$

Not factorial any more!

Approximate Learning

$$P_\theta(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left[\mathbf{v}^\top W^{(1)} \mathbf{h}^{(1)} + \mathbf{h}^{(1)\top} W^{(2)} \mathbf{h}^{(2)} + \mathbf{h}^{(2)\top} W^{(3)} \mathbf{h}^{(3)}\right]$$



(Approximate) Maximum Likelihood:

$$\frac{\partial \log P_\theta(\mathbf{v})}{\partial W^1} = \mathbb{E}_{P_{data}}[\mathbf{v}\mathbf{h}^{\mathbf{1}\top}] - \mathbb{E}_{P_\theta}[\mathbf{v}\mathbf{h}^{\mathbf{1}\top}]$$

Variational Inference

Stochastic Approximation (MCMC-based)

$$P_{data}(\mathbf{v}, \mathbf{h}^{\mathbf{1}}) = P_\theta(\mathbf{h}^{\mathbf{1}}|\mathbf{v}) P_{data}(\mathbf{v})$$

$$P_{data}(\mathbf{v}) = \frac{1}{N} \sum_{n=1}^{N} \delta(\mathbf{v} - \mathbf{v_n})$$

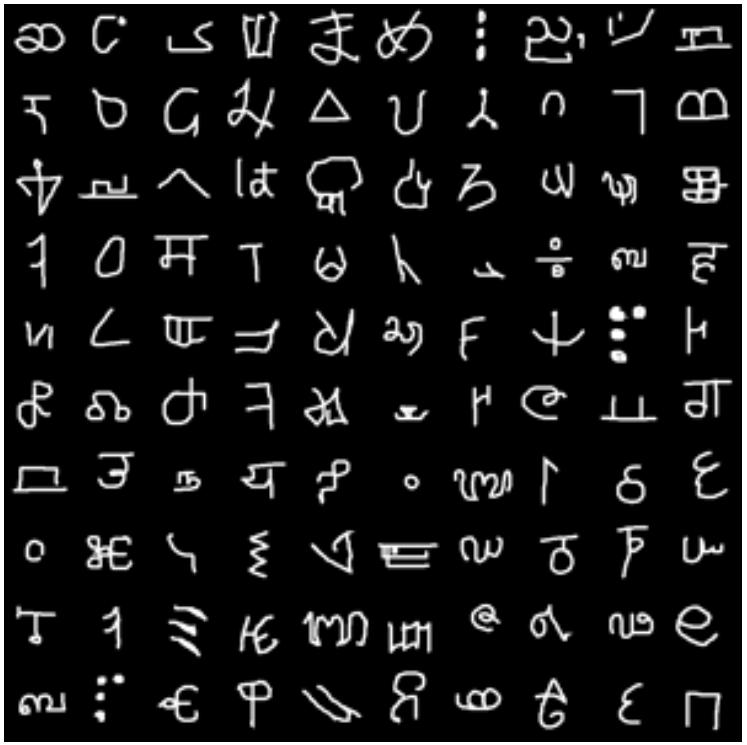
Not factorial any more!

Good Generative Model?

Handwritten Characters

Good Generative Model?

Handwritten Characters



Good Generative Model?

Handwritten Characters

Simulated

Real Data

Good Generative Model?

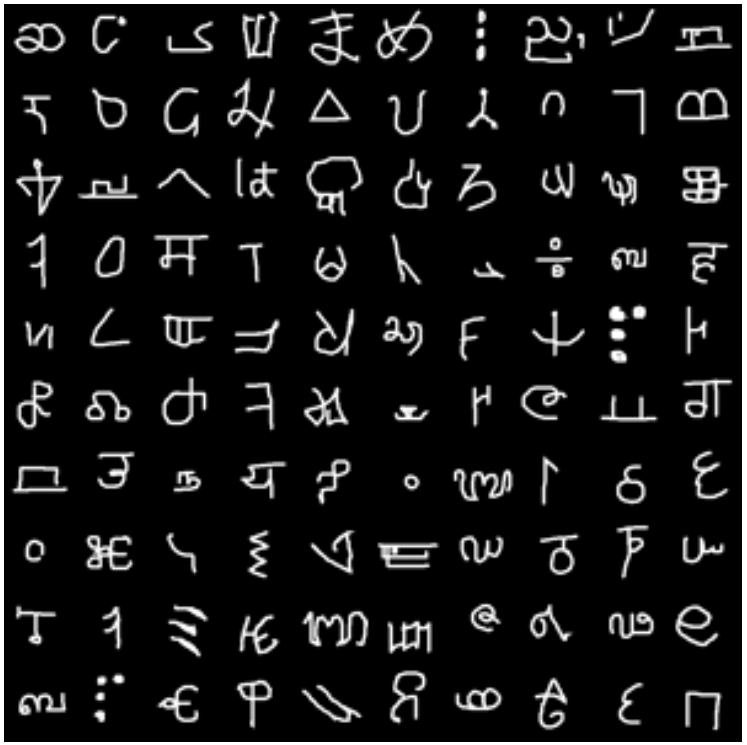
Handwritten Characters

Real Data

Simulated

Good Generative Model?

Handwritten Characters



Handwriting Recognition

MNIST Dataset
60,000 examples of 10 digits

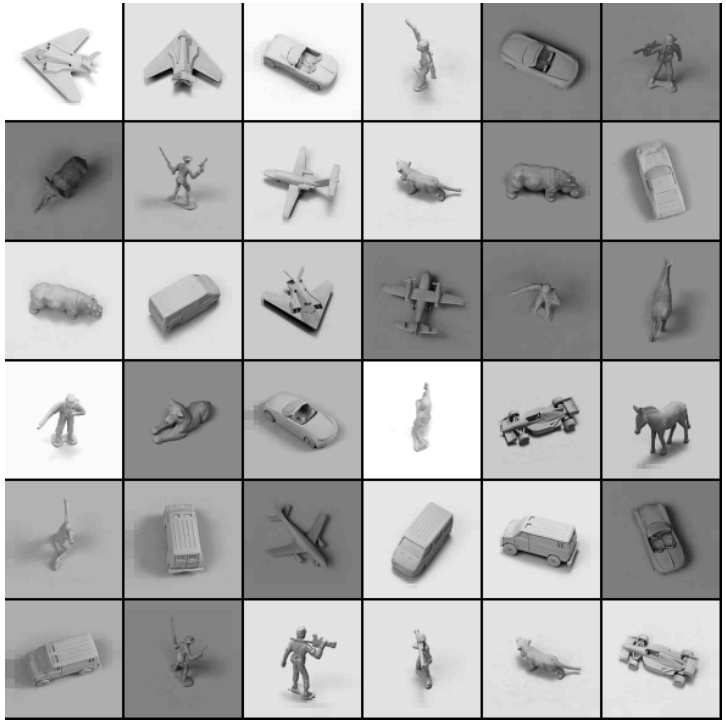
Learning Algorithm	Error
Logistic regression	12.0%
K-NN	3.09%
Neural Net (Platt 2005)	1.53%
SVM (Decoste et.al. 2002)	1.40%
Deep Autoencoder (Bengio et. al. 2007)	1.40%
Deep Belief Net (Hinton et. al. 2006)	1.20%
DBM	**0.95%**

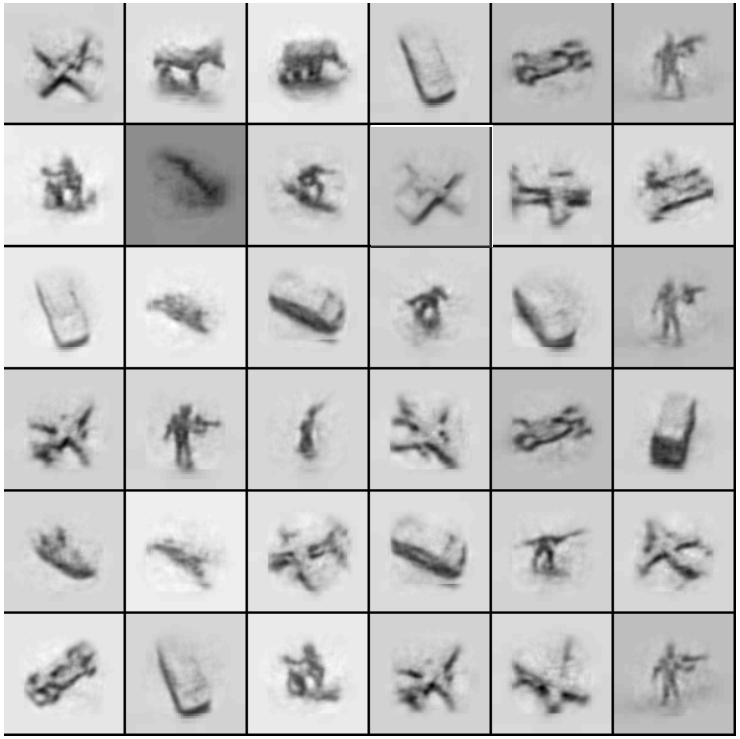
Optical Character Recognition
42,152 examples of 26 English letters

Learning Algorithm	Error
Logistic regression	22.14%
K-NN	18.92%
Neural Net	14.62%
SVM (Larochelle et.al. 2009)	9.70%
Deep Autoencoder (Bengio et. al. 2007)	10.05%
Deep Belief Net (Larochelle et. al. 2009)	9.68%
DBM	**8.40%**

Permutation-invariant version.

Generative Model of 3-D Objects



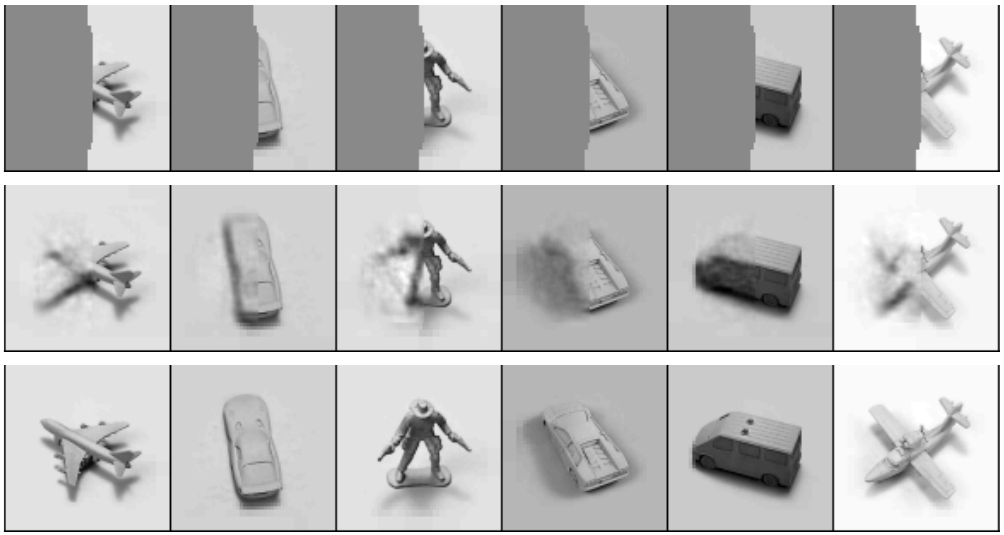


24,000 examples, 5 object categories, 5 different objects within each category, 6 lightning conditions, 9 elevations, 18 azimuths.

3-D Object Recognition

Learning Algorithm	Error
Logistic regression	22.5%
K-NN (LeCun 2004)	18.92%
SVM (Bengio & LeCun 2007)	11.6%
Deep Belief Net (Nair & Hinton 2009)	9.0%
DBM	**7.2%**

Pattern Completion

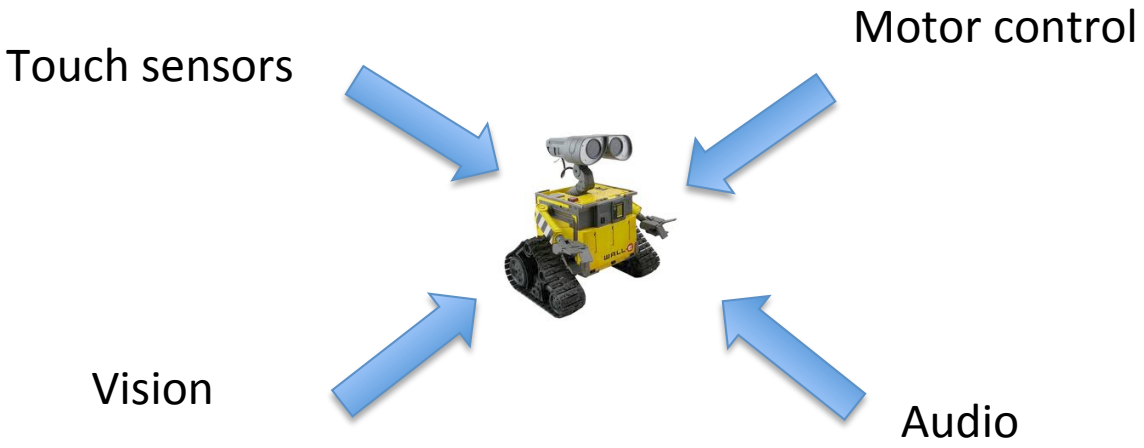


Where else can we use generative models?

Permutation-invariant version.

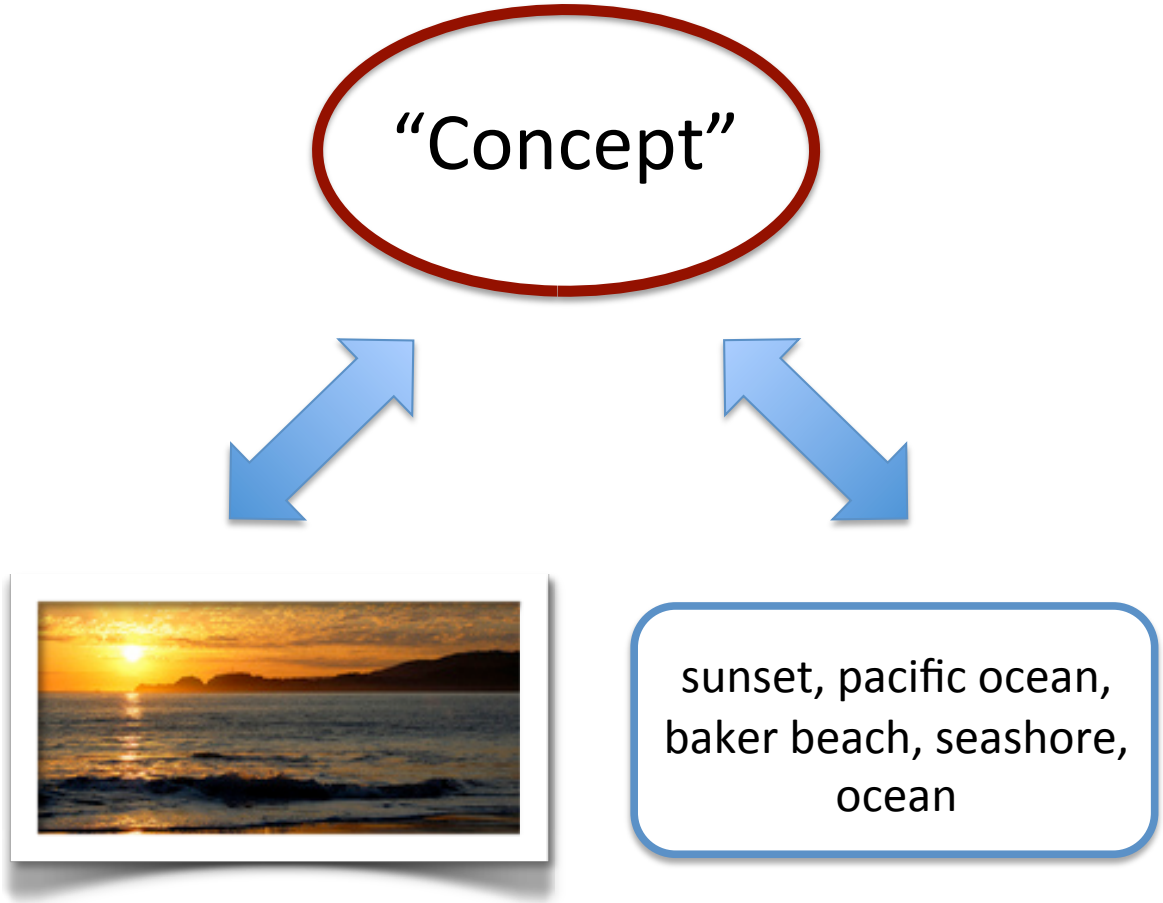
Data – Collection of Modalities

- Multimedia content on the web - image + text + audio.
- Product recommendation systems.
- Robotics applications.



Shared Concept

"Modality-free" representation



"Modality-full" representation

Challenges - I

Image

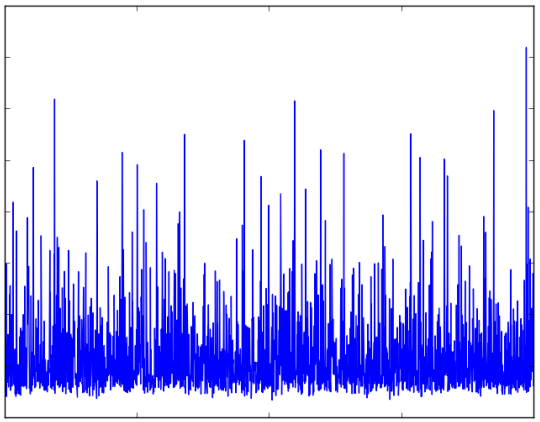
Text

sunset, pacific ocean, baker beach, seashore, ocean

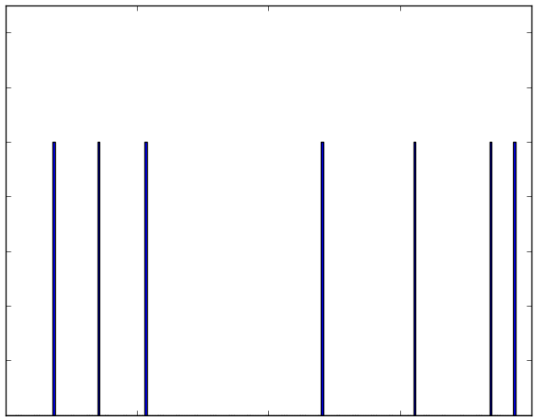
Very different input representations

- Images – real-valued, dense
- Text – discrete, sparse

Dense



Sparse



Difficult to learn cross-modal features from low-level representations.

Challenges - II

Image	Text
	pentax, k10d, pentaxda50200, kangarooisland, sa, australiansealion
	mickikrimmel, mickipedia, headshot
	< no text>
	unseulpixel, naturey

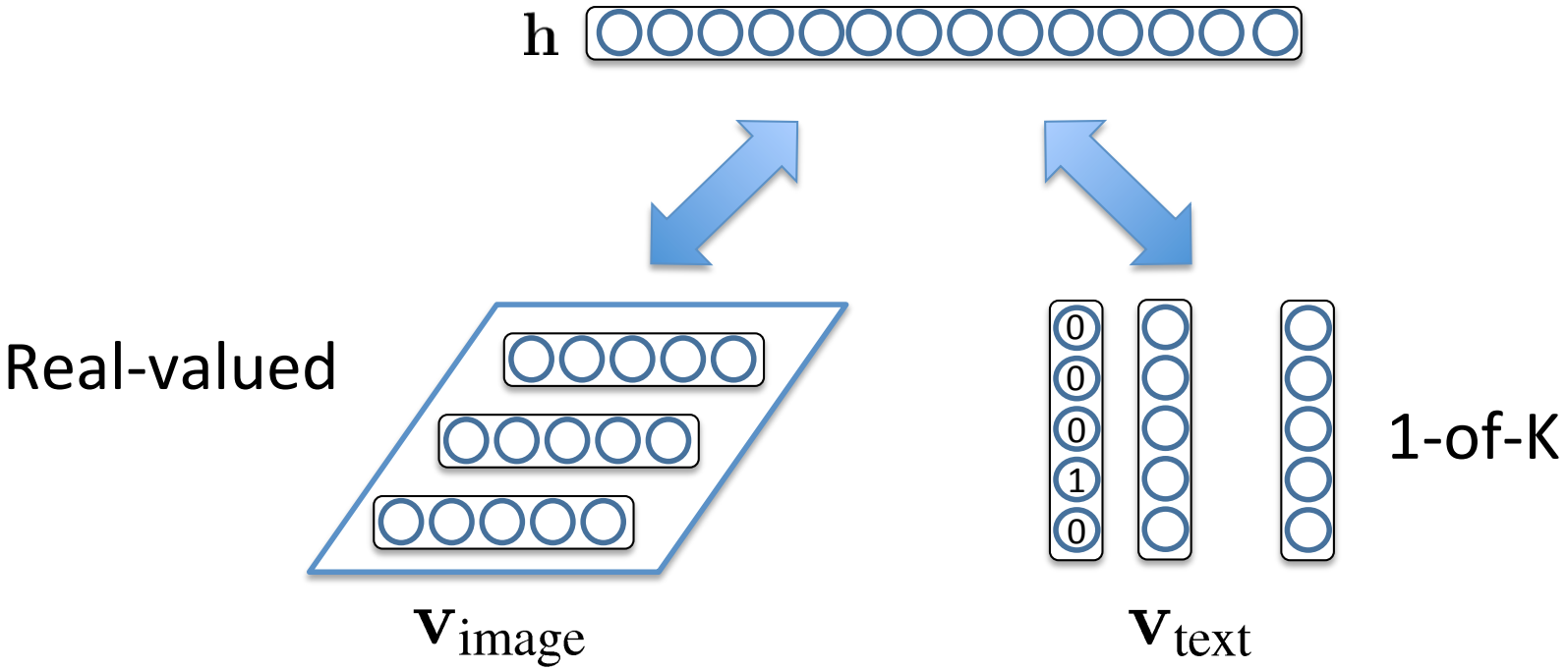
Noisy and missing data

Challenges - II

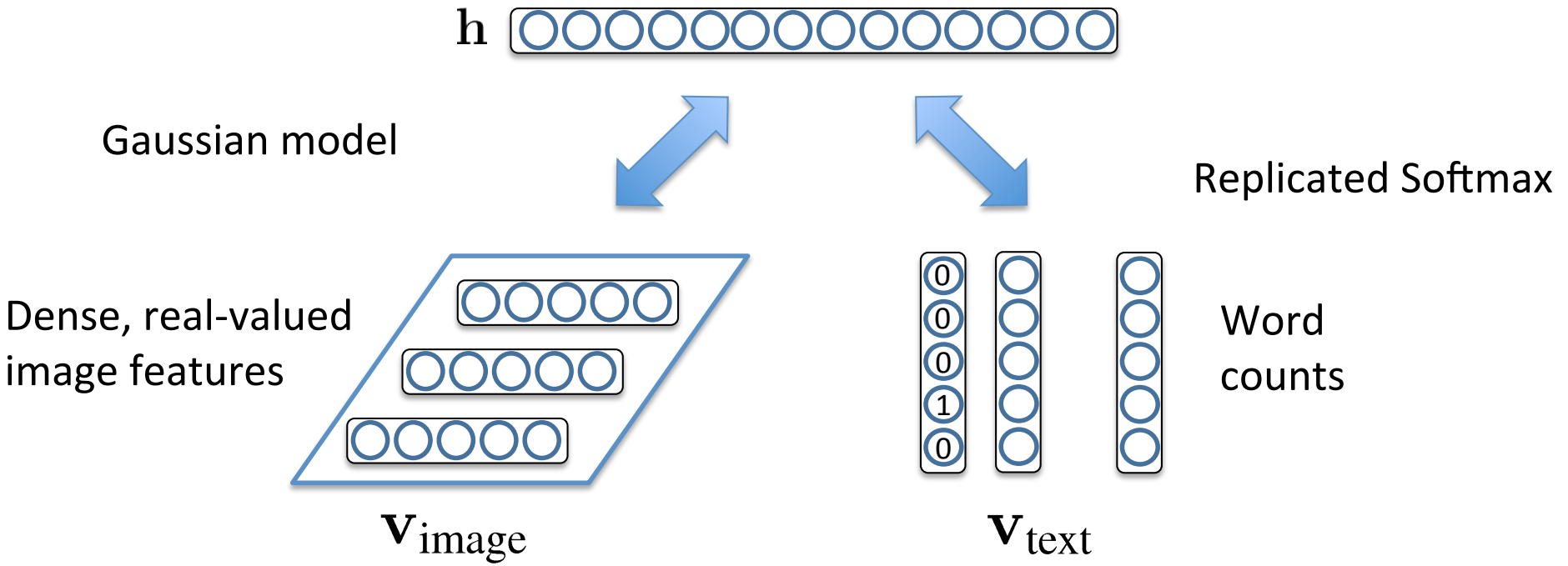
Image	Text	Text generated by the model
	pentax, k10d, pentaxda50200, kangarooisland, sa, australiansealion	beach, sea, surf, strand, shore, wave, seascape, sand, ocean, waves
	mickikrimmel, mickipedia, headshot	portrait, girl, woman, lady, blonde, pretty, gorgeous, expression, model
	< no text>	night, notte, traffic, light, lights, parking, darkness, lowlight, nacht, glow
	unseulpixel, naturey	fall, autumn, trees, leaves, foliage, forest, woods, branches, path

A Simple Multimodal Model

- Use a joint binary hidden layer.
- **Problem**: Inputs have very different statistical properties.
- Difficult to learn cross-modal features.

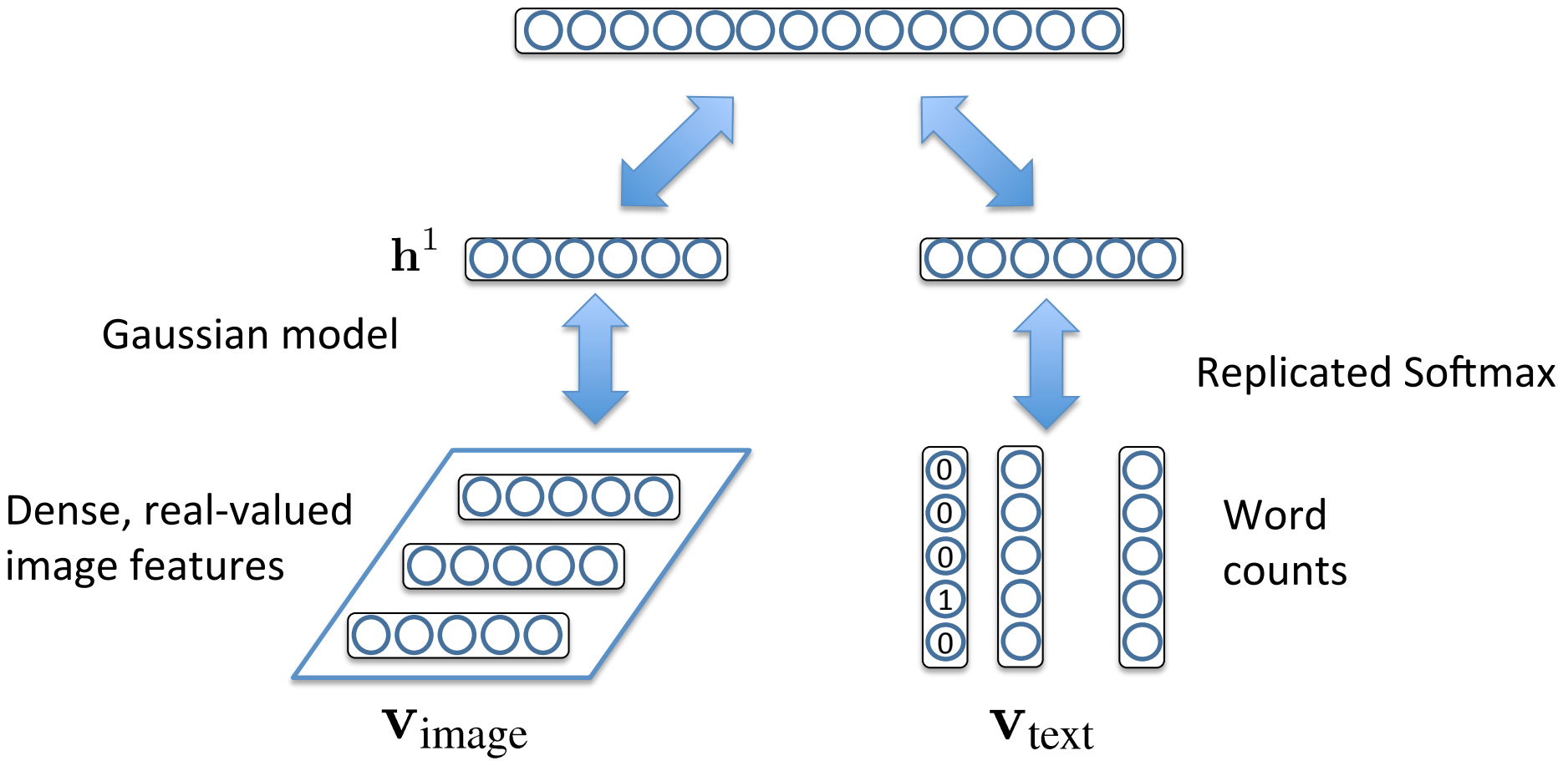


Multimodal DBM



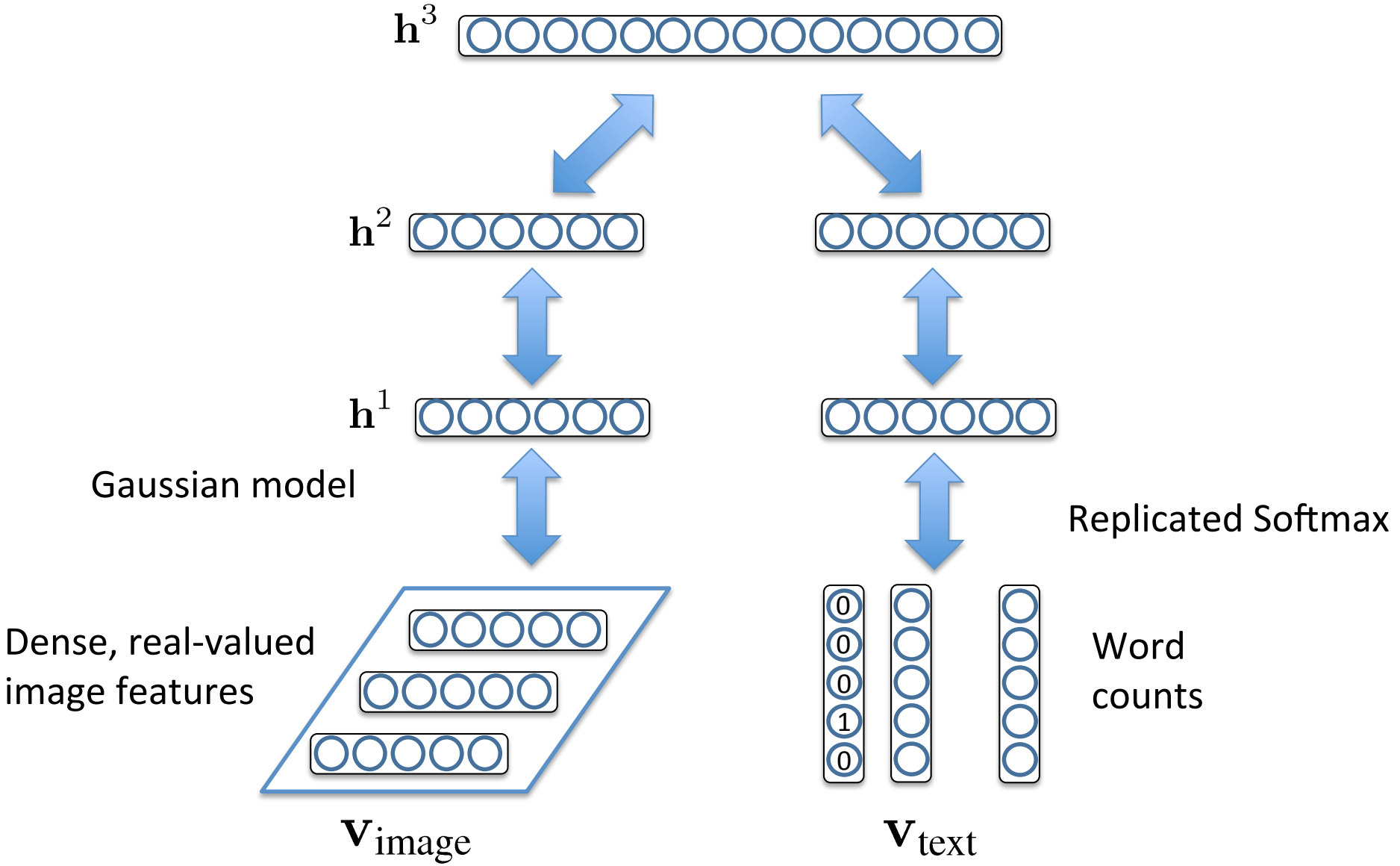
(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)

Multimodal DBM



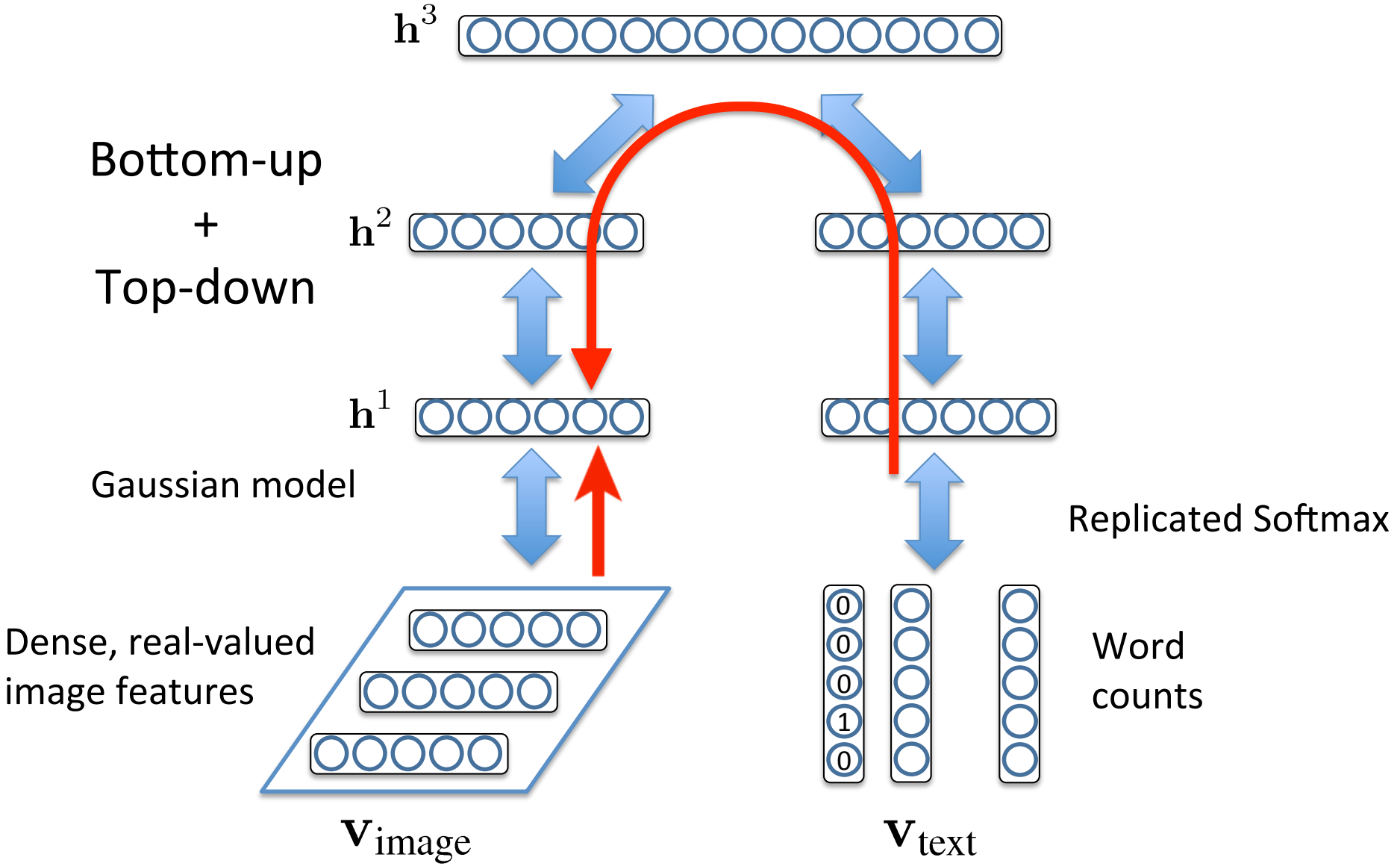
(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)

Multimodal DBM



(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)

Multimodal DBM



(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)

Text Generated from Images

Given	Generated
	dog, cat, pet, kitten, puppy, ginger, tongue, kitty, dogs, furry
	sea, france, boat, mer, beach, river, bretagne, plage, brittany
	portrait, child, kid, ritratto, kids, children, boy, cute, boys, italy

Given	Generated
	insect, butterfly, insects, bug, butterflies, lepidoptera
	graffiti, streetart, stencil, sticker, urbanart, graff, sanfrancisco
	canada, nature, sunrise, ontario, fog, mist, bc, morning

Text Generated from Images

Given

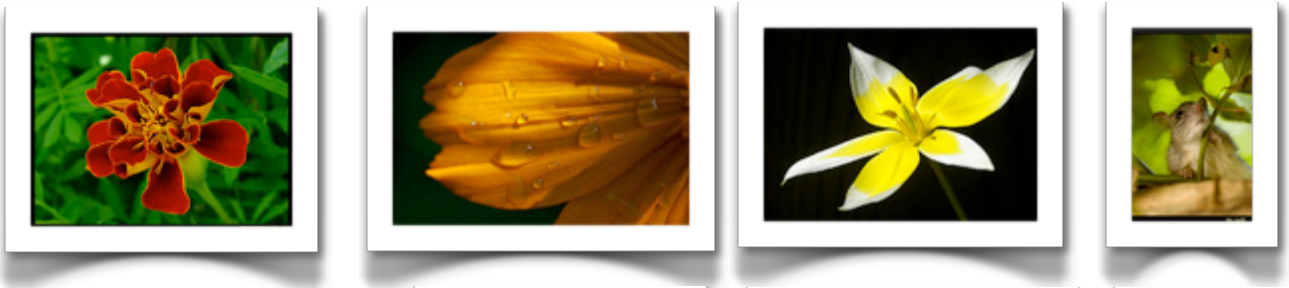
Generated

portrait, women, army, soldier, mother, postcard, soldiers

obama, barackobama, election, politics, president, hope, change, sanfrancisco, convention, rally

water, glass, beer, bottle, drink, wine, bubbles, splash, drops, drop

Images from Text

Given	Retrieved
water, red, sunset	
nature, flower, red, green	
blue, green, yellow, colors	
chocolate, cake	

MIR-Flickr Dataset

- 1 million images along with user-assigned tags.

sculpture, beauty, stone

d80

nikon, abigfave, goldstaraward, d80, nikond80

food, cupcake, vegan

anawesomeshot, theperfectphotographer, flash, damniwishidtakenthat, spiritofphotography

nikon, green, light, photoshop, apple, d70

white, yellow, abstract, lines, bus, graphic

sky, geotagged, reflection, cielo, bilbao, reflejo

Huiskes et. al.

Results

- Logistic regression on top-level representation.
- Multimodal Inputs

Mean Average Precision

Learning Algorithm	MAP	Precision@50
Random	0.124	0.124
LDA [Huiskes et. al.]	0.492	0.754
SVM [Huiskes et. al.]	0.475	0.758
DBM-Labelled	0.526	0.791

Labeled 25K examples

Results

- Logistic regression on top-level representation.
- Multimodal Inputs

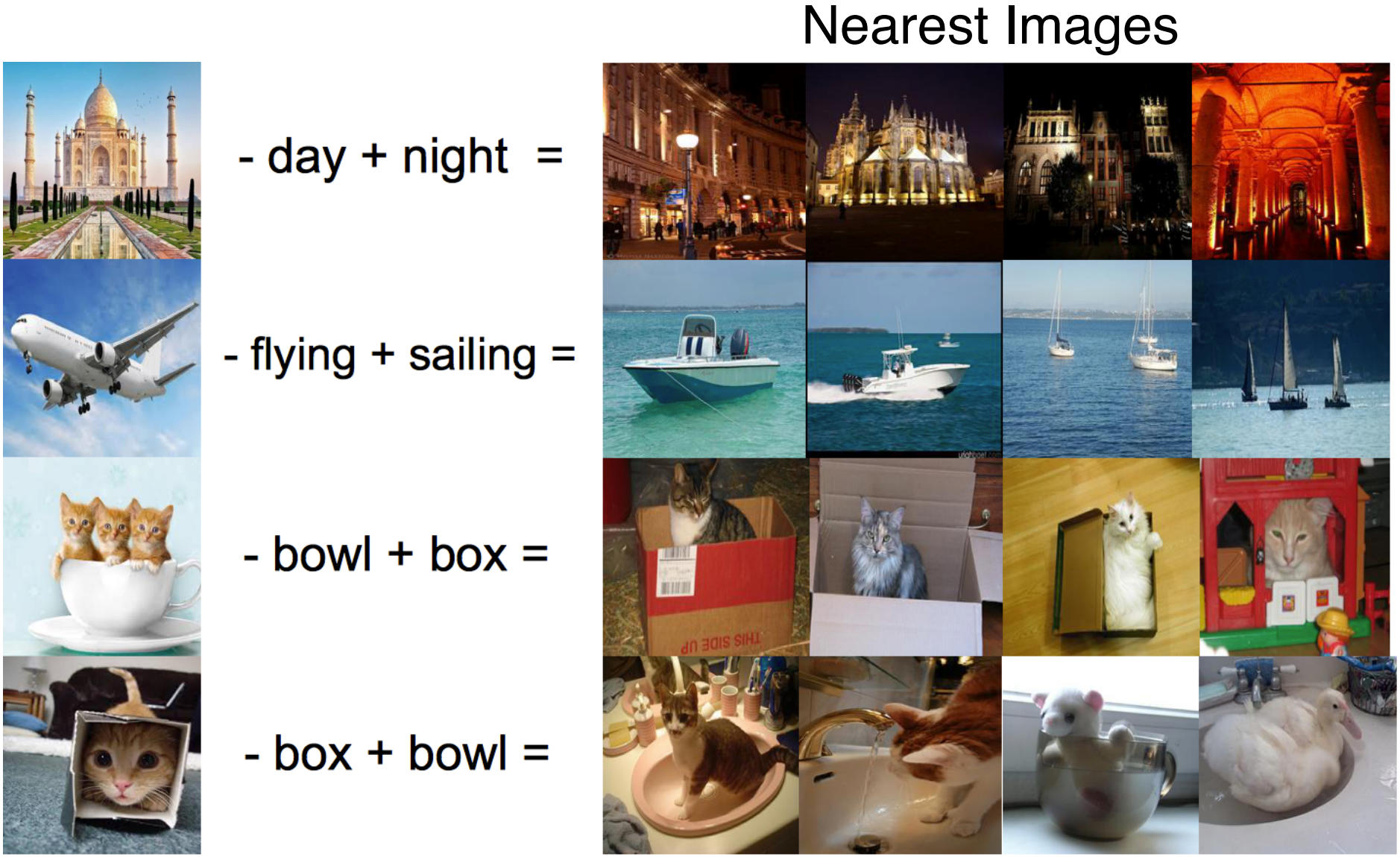
Mean Average Precision

Learning Algorithm	MAP	Precision@50
Random	0.124	0.124
LDA [Huiskes et. al.]	0.492	0.754
SVM [Huiskes et. al.]	0.475	0.758
DBM-Labelled	0.526	0.791
Deep Belief Net	0.638	0.867
Autoencoder	0.638	0.875
DBM	0.641	0.873

Labeled 25K examples

+ 1 Million unlabelled

Multimodal Linguistic Regularities



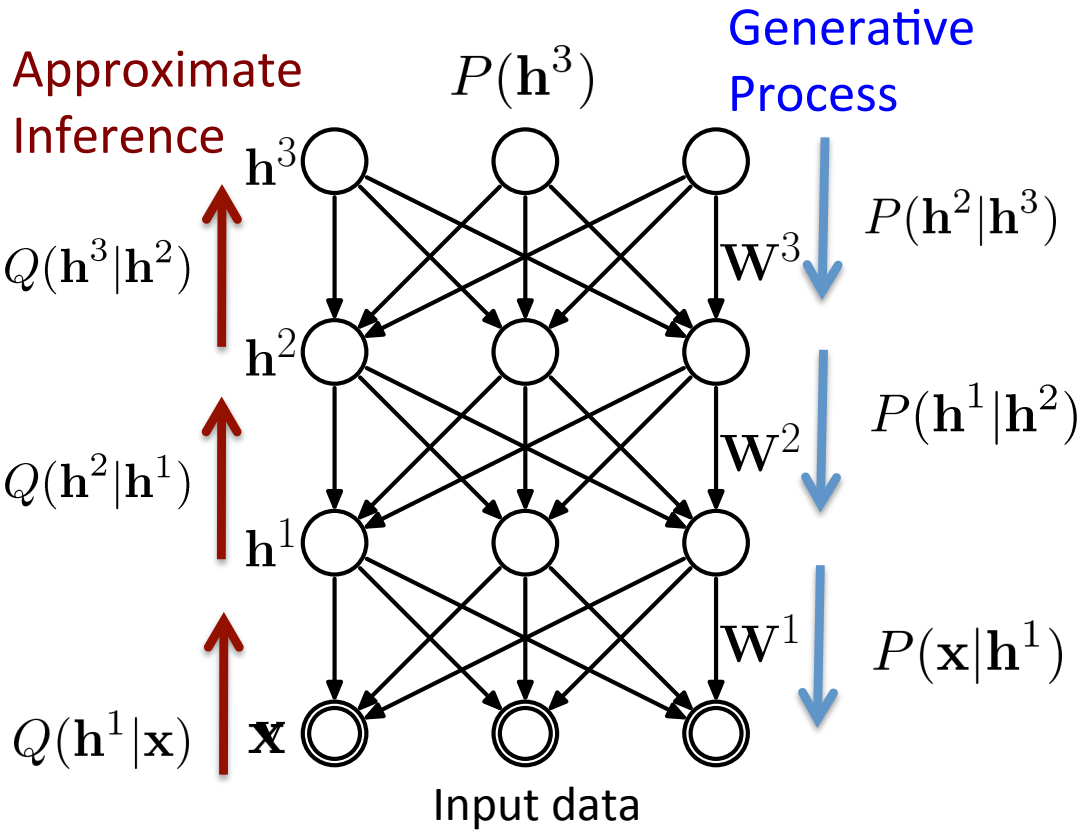
(Kiros, Salakhutdinov, Zemel, TACL 2015)

Talk Roadmap

- Basic Building Blocks:
  - Sparse Coding
  - Autoencoders
- Deep Generative Models
  - Restricted Boltzmann Machines
  - Deep Boltzmann Machines
  - Helmholtz Machines / Variational Autoencoders
- Generative Adversarial Networks

Helmholtz Machines

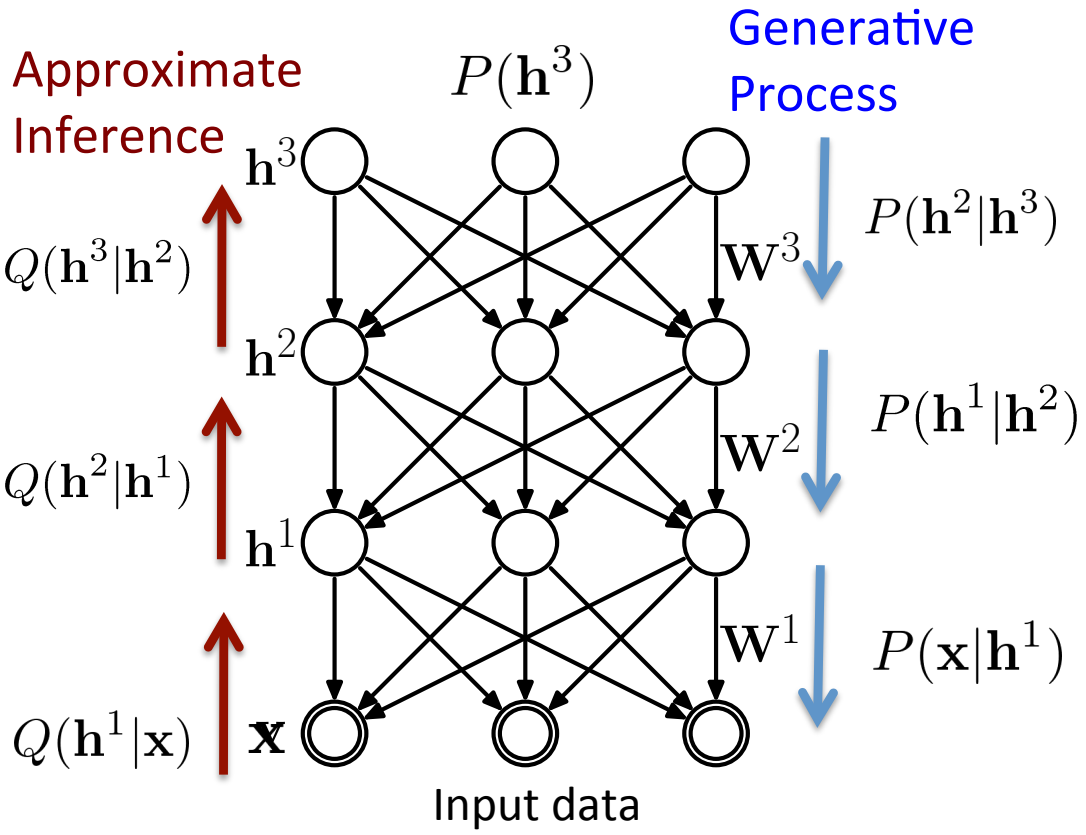
- Hinton, G. E., Dayan, P., Frey, B. J. and Neal, R., Science 1995



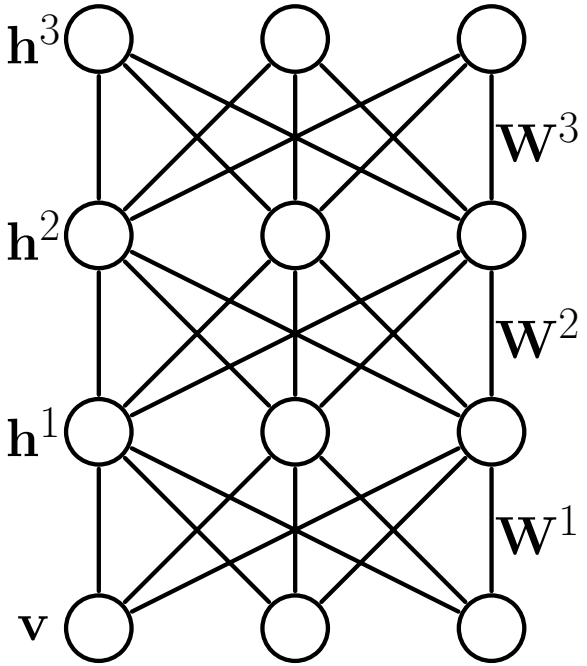
- Kingma & Welling, 2014
- Rezende, Mohamed, Daan, 2014
- Mnih & Gregor, 2014
- Bornschein & Bengio, 2015
- Tang & Salakhutdinov, 2013

Helmholtz Machines vs. DBMs

Helmholtz Machine



Deep Boltzmann Machine

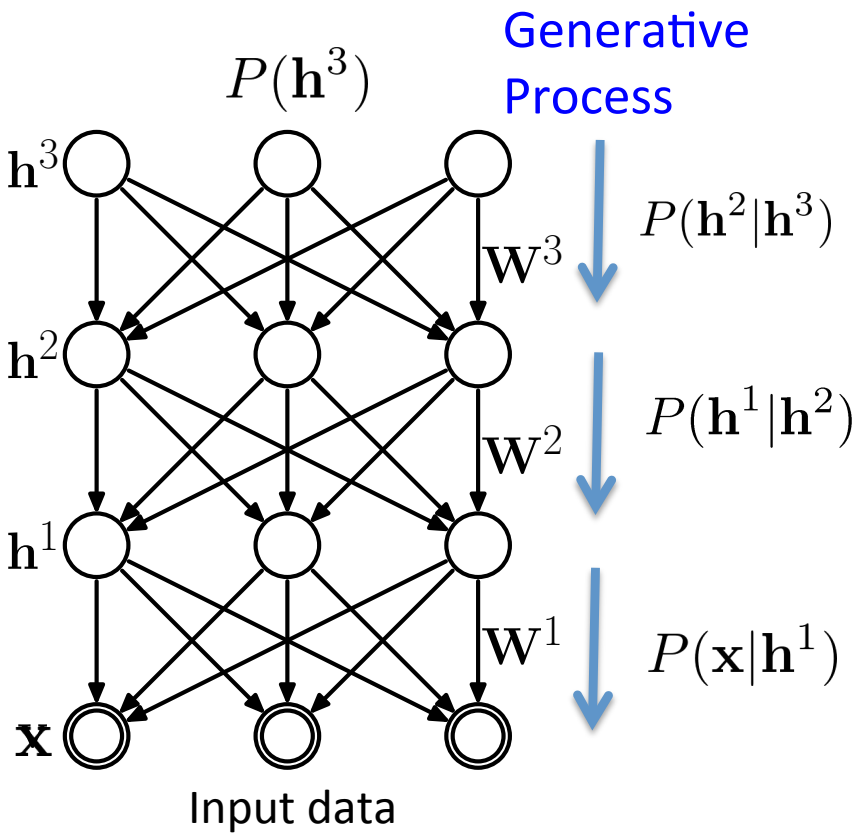


Variational Autoencoders (VAEs)

• The VAE defines a generative process in terms of ancestral sampling through a cascade of hidden stochastic layers:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{\mathbf{h}^1,\ldots,\mathbf{h}^L} p(\mathbf{h}^L|\boldsymbol{\theta})p(\mathbf{h}^{L-1}|\mathbf{h}^L,\boldsymbol{\theta})\cdots p(\mathbf{x}|\mathbf{h}^1,\boldsymbol{\theta})$$

Each term may denote a complicated nonlinear relationship



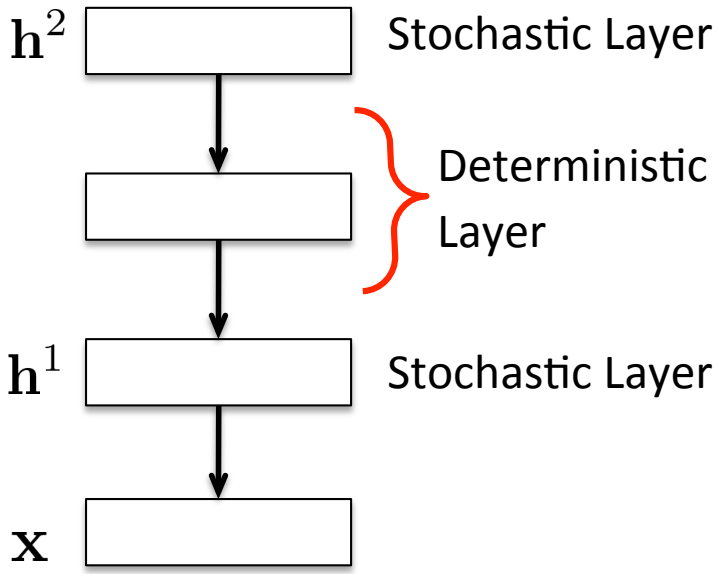
- $\boldsymbol{\theta}$ denotes parameters of VAE.
- L is the number of **stochastic** layers.
- Sampling and probability evaluation is tractable for each $p(\mathbf{h}^\ell|\mathbf{h}^{\ell+1})$.

VAE: Example

• The VAE defines a generative process in terms of ancestral sampling through a cascade of hidden stochastic layers:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{\mathbf{h}^1,\mathbf{h}^2} p(\mathbf{h}^2|\boldsymbol{\theta})p(\mathbf{h}^1|\mathbf{h}^2,\boldsymbol{\theta})p(\mathbf{x}|\mathbf{h}^1,\boldsymbol{\theta})$$

This term denotes a one-layer neural net.



- $\boldsymbol{\theta}$ denotes parameters of VAE.
- L is the number of **stochastic** layers.
- Sampling and probability evaluation is tractable for each $p(\mathbf{h}^\ell|\mathbf{h}^{\ell+1})$.

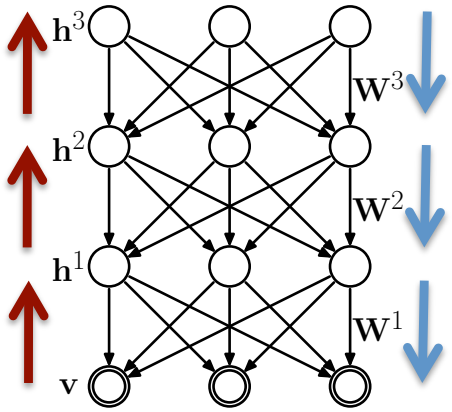
Variational Bound

- The VAE is trained to maximize the variational lower bound:

$$\log p(\mathbf{x}) = \log \mathbb{E}_{q(\mathbf{h}|\mathbf{x})} \left[\frac{p(\mathbf{x}, \mathbf{h})}{q(\mathbf{h}|\mathbf{x})} \right] \geq \mathbb{E}_{q(\mathbf{h}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{h})}{q(\mathbf{h}|\mathbf{x})} \right] = \mathcal{L}(\mathbf{x})$$

$$\mathcal{L}(\mathbf{x}) = \log p(\mathbf{x}) - \mathrm{D}_{\mathrm{KL}} \left(q(\mathbf{h}|\mathbf{x}) || p(\mathbf{h}|\mathbf{x}) \right)$$

- Trading off the data log-likelihood and the KL divergence from the true posterior.



Input data

- Hard to optimize the variational bound with respect to the recognition network (high-variance).
- Key idea of Kingma and Welling is to use reparameterization trick.

Reparameterization Trick

- Assume that the recognition distribution is Gaussian:

$$q(\mathbf{h}^{\ell}|\mathbf{h}^{\ell-1},\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{h}^{\ell-1},\boldsymbol{\theta}),\boldsymbol{\Sigma}(\mathbf{h}^{\ell-1},\boldsymbol{\theta}))$$

  with mean and covariance computed from the state of the hidden units at the previous layer.

- Alternatively, we can express this in term of auxiliary variable:

$$\boldsymbol{\epsilon}^{\ell} \sim \mathcal{N}(\mathbf{0},\boldsymbol{I})$$

$$\mathbf{h}^{\ell}\left(\boldsymbol{\epsilon}^{\ell},\mathbf{h}^{\ell-1},\boldsymbol{\theta}\right) = \boldsymbol{\Sigma}(\mathbf{h}^{\ell-1},\boldsymbol{\theta})^{1/2}\boldsymbol{\epsilon}^{\ell} + \boldsymbol{\mu}(\mathbf{h}^{\ell-1},\boldsymbol{\theta})$$

Reparameterization Trick

- Assume that the recognition distribution is Gaussian:

$$q(\mathbf{h}^{\ell}|\mathbf{h}^{\ell-1},\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{h}^{\ell-1},\boldsymbol{\theta}),\boldsymbol{\Sigma}(\mathbf{h}^{\ell-1},\boldsymbol{\theta}))$$

- Or

$$\boldsymbol{\epsilon}^{\ell} \sim \mathcal{N}(\mathbf{0},\boldsymbol{I})$$

$$\mathbf{h}^{\ell}\left(\boldsymbol{\epsilon}^{\ell},\mathbf{h}^{\ell-1},\boldsymbol{\theta}\right) = \boldsymbol{\Sigma}(\mathbf{h}^{\ell-1},\boldsymbol{\theta})^{1/2}\boldsymbol{\epsilon}^{\ell} + \boldsymbol{\mu}(\mathbf{h}^{\ell-1},\boldsymbol{\theta})$$

- The recognition distribution $q(\mathbf{h}^{\ell}|\mathbf{h}^{\ell-1},\boldsymbol{\theta})$ can be expressed in terms of a deterministic mapping:

$$\underbrace{\mathbf{h}\left(\boldsymbol{\epsilon},\mathbf{x},\boldsymbol{\theta}\right)}, \quad \text{with} \quad \boldsymbol{\epsilon} = \underbrace{(\boldsymbol{\epsilon}^{1},\ldots,\boldsymbol{\epsilon}^{L})}$$

Deterministic Encoder

Distribution of $\boldsymbol{\epsilon}$ does not depend on $\boldsymbol{\theta}$

Computing the Gradients

- The gradient w.r.t the parameters: both recognition and generative:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{h} \sim q(\mathbf{h}|\mathbf{x},\boldsymbol{\theta})} \left[\log \frac{p(\mathbf{x}, \mathbf{h}|\boldsymbol{\theta})}{q(\mathbf{h}|\mathbf{x}, \boldsymbol{\theta})} \right]$$

Autoencoder

$$= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\epsilon}^1, \ldots, \boldsymbol{\epsilon}^L \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[\log \frac{p(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \boldsymbol{\theta})|\boldsymbol{\theta})}{q(\mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \boldsymbol{\theta})|\mathbf{x}, \boldsymbol{\theta})} \right]$$

$$= \mathbb{E}_{\boldsymbol{\epsilon}^1, \ldots, \boldsymbol{\epsilon}^L \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[\nabla_{\boldsymbol{\theta}} \log \frac{p(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \boldsymbol{\theta})|\boldsymbol{\theta})}{q(\mathbf{h}(\boldsymbol{\epsilon}, \mathbf{x}, \boldsymbol{\theta})|\mathbf{x}, \boldsymbol{\theta})} \right]$$

Gradients can be computed by backprop

The mapping **h** is a deterministic neural net for fixed $\boldsymbol{\epsilon}$.

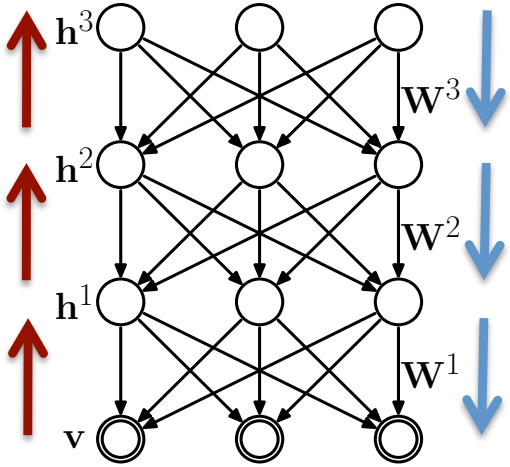
Importance Weighted Autoencoders

- Can improve VAE by using following k-sample importance weighting of the log-likelihood:

$$\mathcal{L}_k(\mathbf{x}) = \mathbb{E}_{\mathbf{h}_1,\ldots,\mathbf{h}_k \sim q(\mathbf{h}|\mathbf{x})}\left[\log \frac{1}{k}\sum_{i=1}^{k} \frac{p(\mathbf{x},\mathbf{h}_i)}{q(\mathbf{h}_i|\mathbf{x})}\right]$$

$$= \mathbb{E}_{\mathbf{h}_1,\ldots,\mathbf{h}_k \sim q(\mathbf{h}|\mathbf{x})}\left[\log \frac{1}{k}\sum_{i=1}^{k} w_i\right]$$

unnormalized importance weights

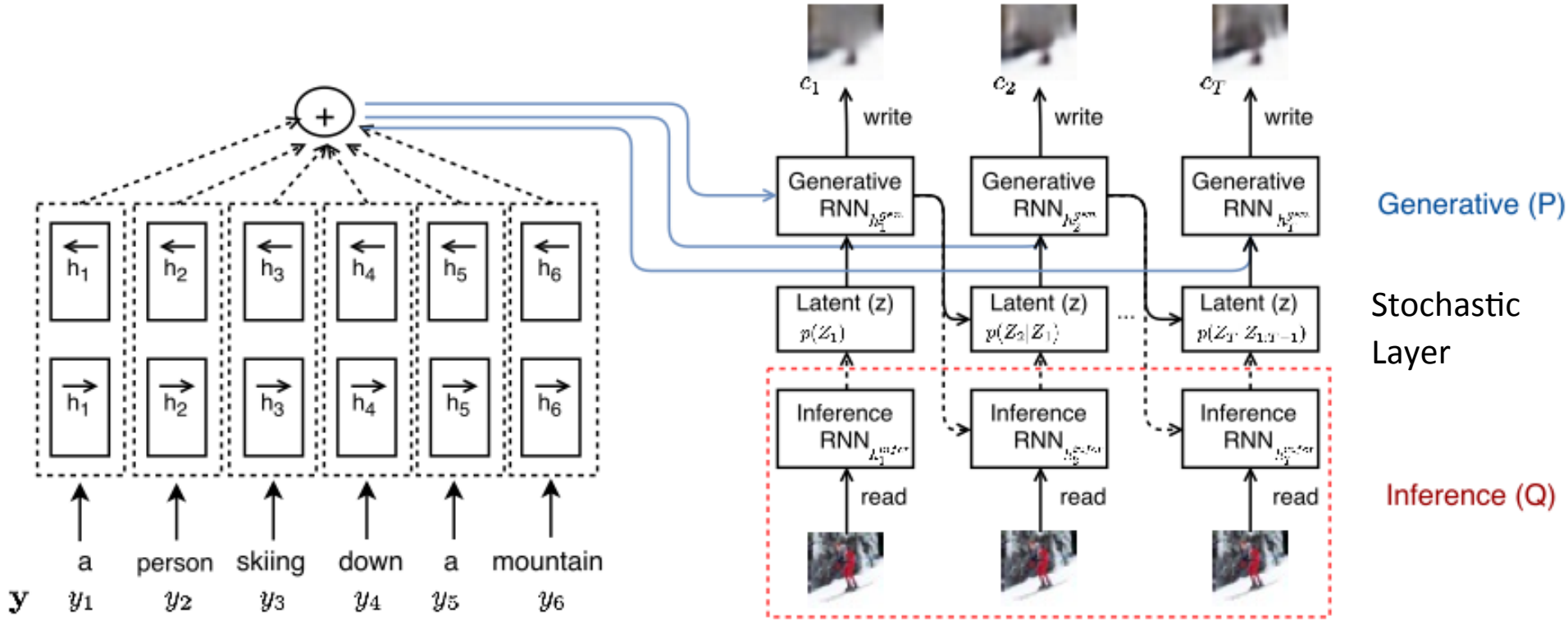


Input data

where $\mathbf{h}_1, \ldots, \mathbf{h}_k$ are sampled from the recognition network.

Burda, Grosse, Salakhutdinov, 2015

Generating Images from Captions



- Generative Model: Stochastic Recurrent Network, chained sequence of Variational Autoencoders, with a single stochastic layer.
- Recognition Model: Deterministic Recurrent Network.

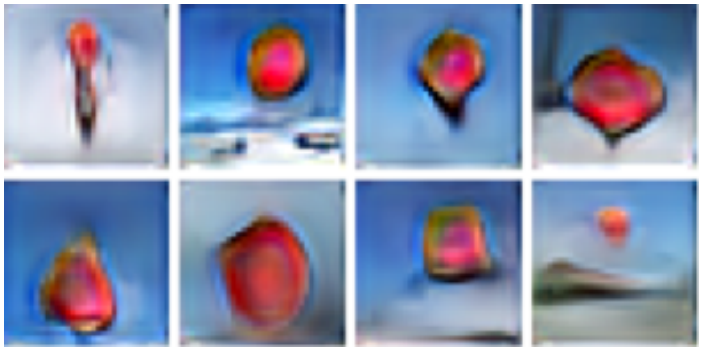
Gregor et. al. 2015

(Mansimov, Parisotto, Ba, Salakhutdinov, 2015)

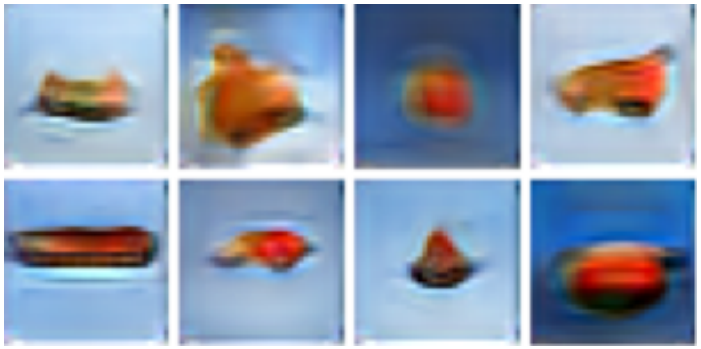
Motivating Example

- Can we generate images from natural language descriptions?

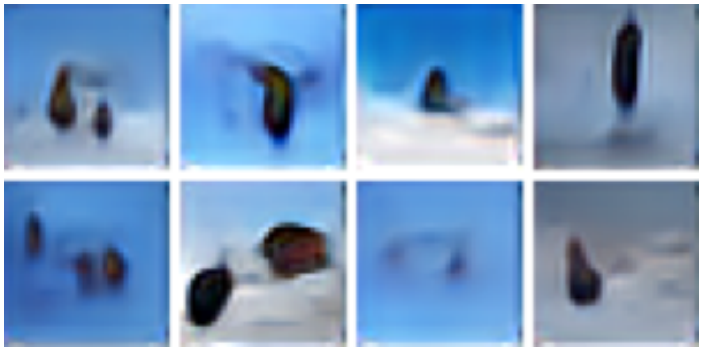
A **stop sign** is flying in blue skies



A **pale yellow school bus** is flying in blue skies



A **herd of elephants** is flying in blue skies



A **large commercial airplane** is flying in blue skies

(Mansimov, Parisotto, Ba, Salakhutdinov, 2015)

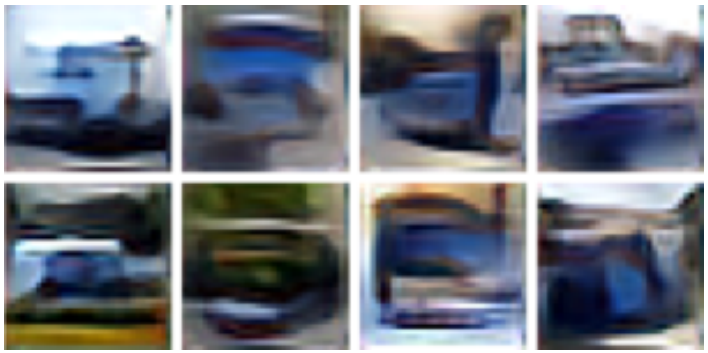
Flipping Colors

A **yellow school bus** parked in the parking lot

A **red school bus** parked in the parking lot

A **green school bus** parked in the parking lot

A **blue school bus** parked in the parking lot



(Mansimov, Parisotto, Ba, Salakhutdinov, 2015)

Qualitative Comparison

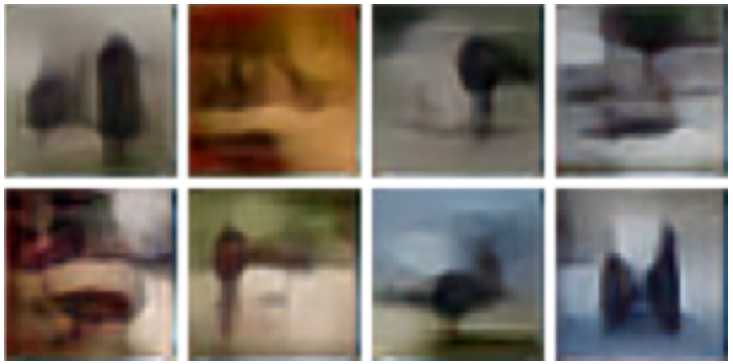
A group of people walk on a beach with surf boards

Our Model

LAPGAN (Denton et. al. 2015)

Conv-Deconv VAE

Fully Connected VAE



Novel Scene Compositions

A toilet seat sits open in the bathroom

A toilet seat sits open in the grass field

Ask Google?

Neural Story Telling

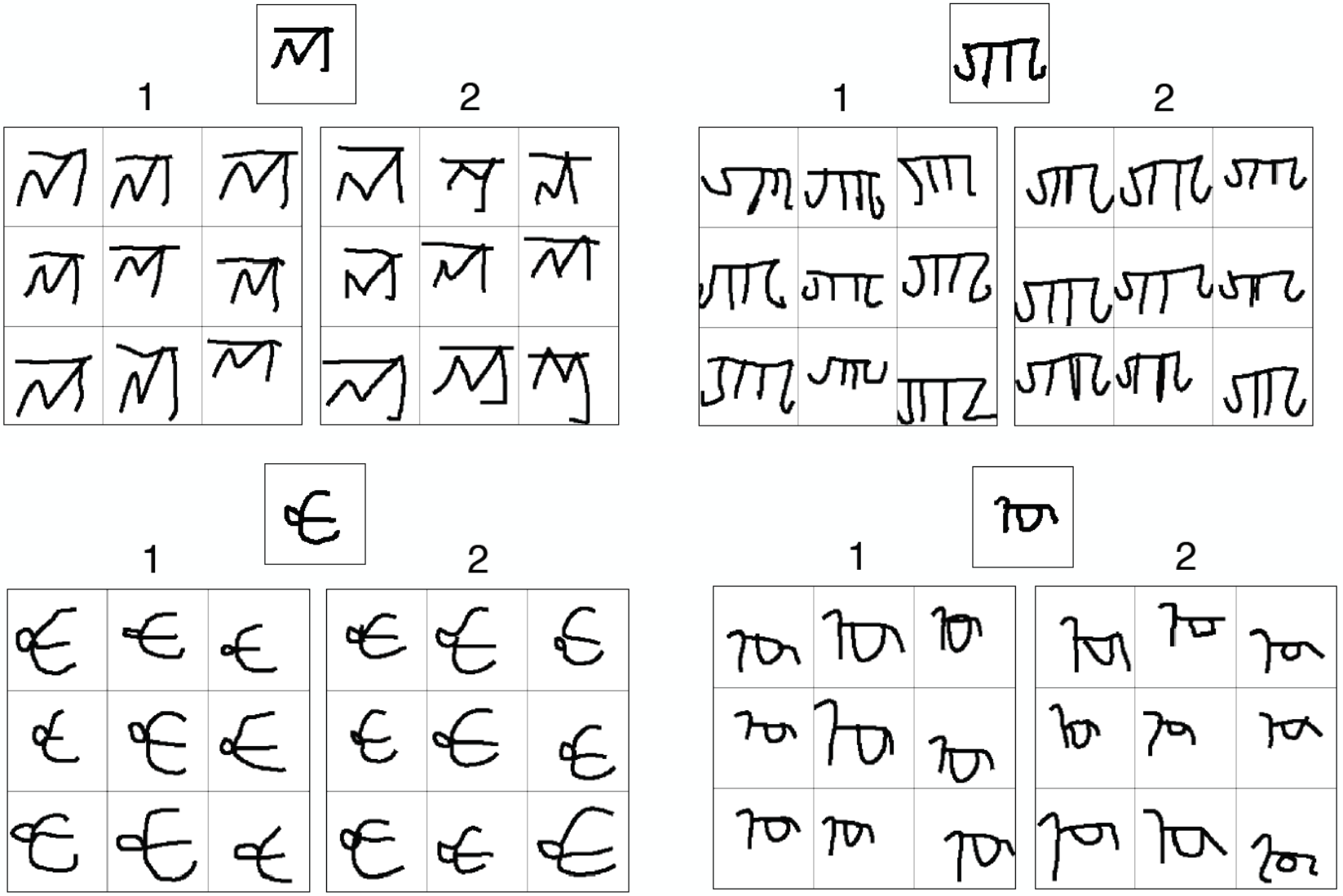
Sample from the Generative Model (recurrent neural network):

We were barely able to catch the breeze at the beach, and it felt as if someone stepped out of my mind.

She was in love with him for the first time in months, so she had no intention of escaping. The sun had risen from the ocean, making her feel more alive than normal . She is beautiful, but the truth is that I do not know what to do. The sun was just starting to fade away, leaving people scattered around the Atlantic Ocean.

Kiros et.al., NIPS 2015

One-Shot Learning: Humans vs. Machines



(Lake, Salakhutdinov, Tenenbaum, Science, 2015)

Talk Roadmap

- Basic Building Blocks:
  - Sparse Coding
  - Autoencoders
- Deep Generative Models
  - Restricted Boltzmann Machines
  - Deep Boltzmann Machines
  - Helmholtz Machines / Variational Autoencoders
- Generative Adversarial Networks

Generative Adversarial Networks

• There is no explicit definition of the density for p(x) – Only need to be able to sample from it.

• No variational learning, no maximum-likelihood estimation, no MCMC. How?

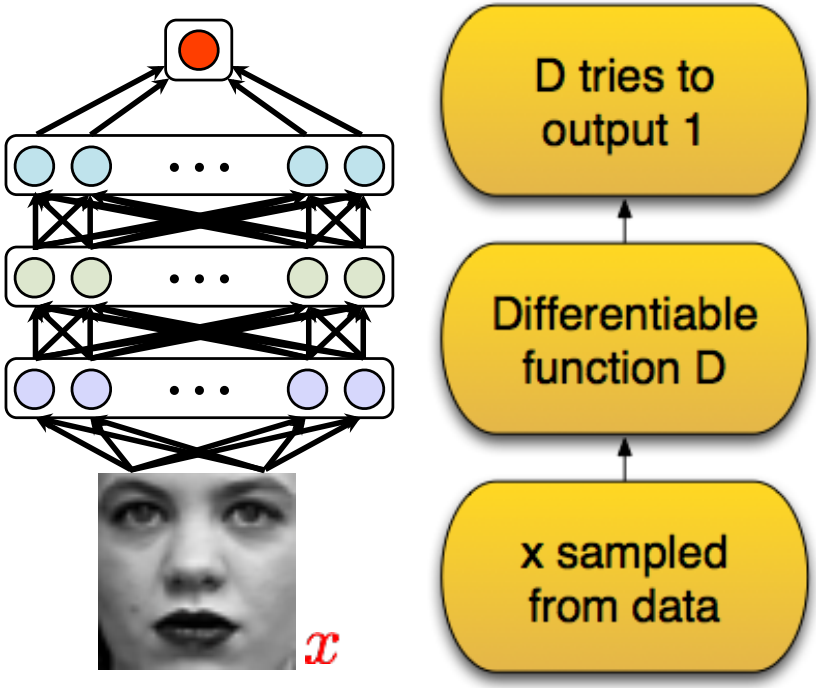
• By playing a game!

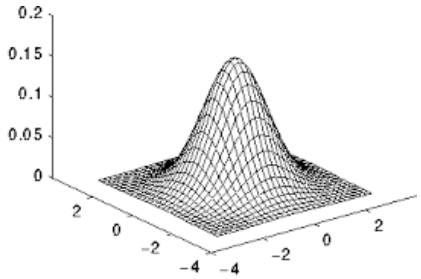
Generative Adversarial Networks

- Set up a game between two players:
  - Discriminator D
  - Generator G
- Discriminator D tries to discriminate between:
  - A sample from the data distribution.
  - And a sample from the generator G.
- The Generator G attempts to "fool" D by generating samples that are hard for D to distinguish from the real data.

Generative Adversarial Networks" Goodfellow et al., NIPS 2014

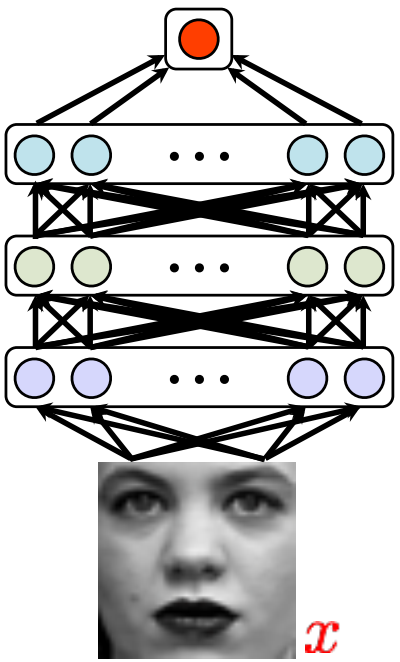
Generative Adversarial Networks





Slide Credit: Ian Goodfellow

Generative Adversarial Networks

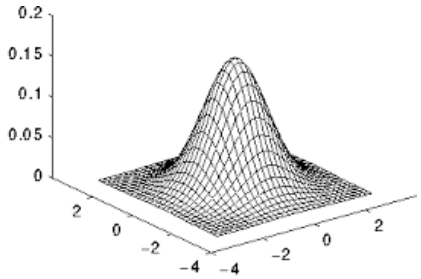


x

D tries to output 1

Differentiable function D

x sampled from data

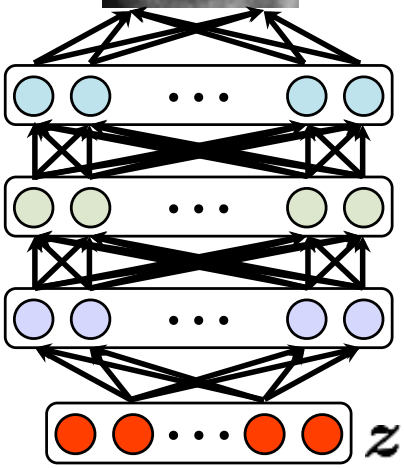


x sampled from model

Differentiable function G

Input noise Z

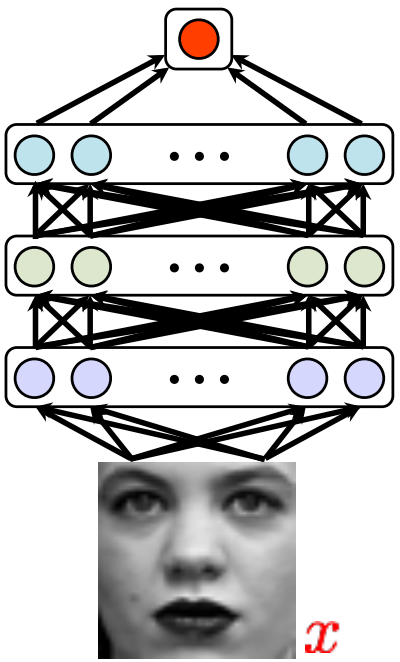
x



z

Slide Credit: Ian Goodfellow

Generative Adversarial Networks

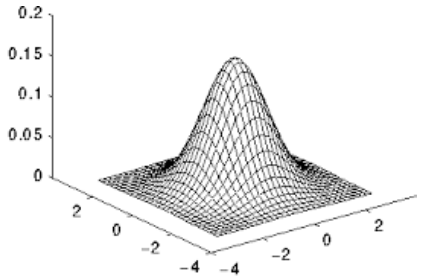


x

D tries to output 1

Differentiable function D

x sampled from data



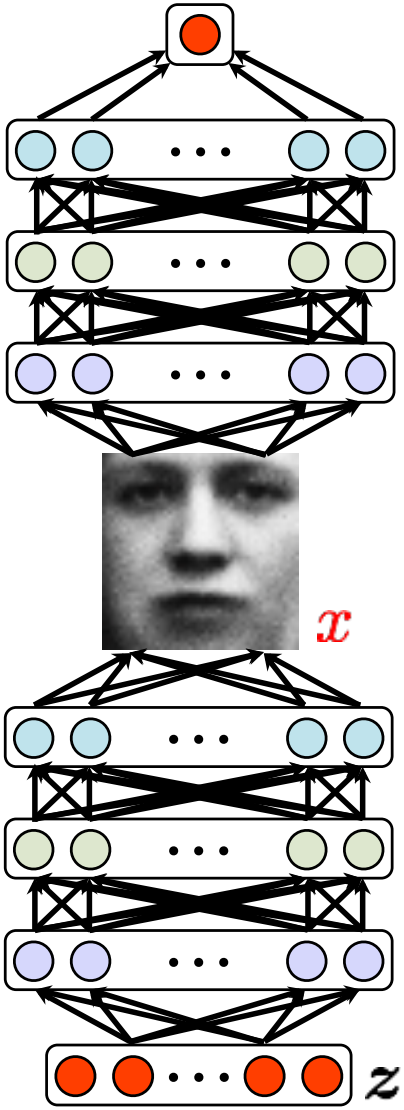
D tries to output 0

Differentiable function D

x sampled from model

Differentiable function G

Input noise Z



x

z

Slide Credit: Ian Goodfellow

Generative Adversarial Networks

- Minimax value function

Generator: generate samples that D would classify as real

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Discriminator: Pushes up

Discriminator: Classify data as being real

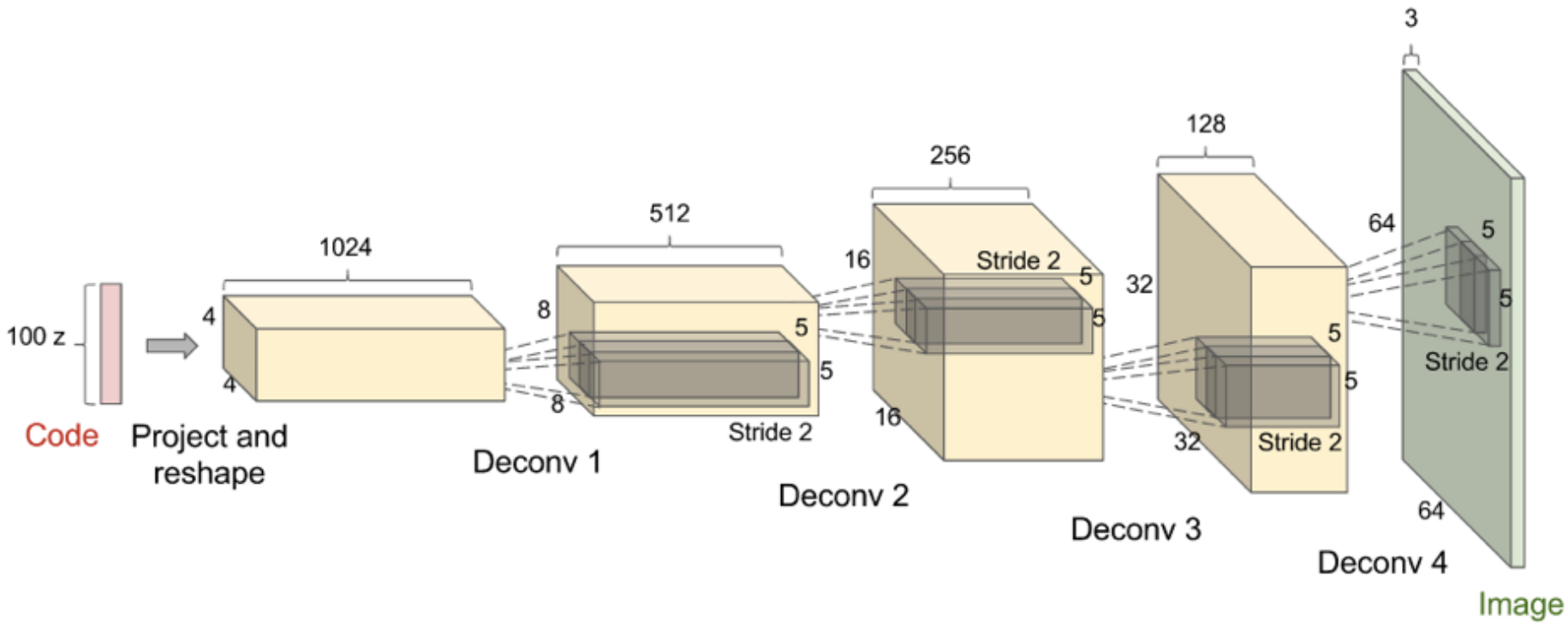
Discriminator: Classify generator samples as being fake

Generator: Pushes down

- Optimal strategy for Discriminator is:

$$D(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{model}}(x)}$$

DCGAN Architecture

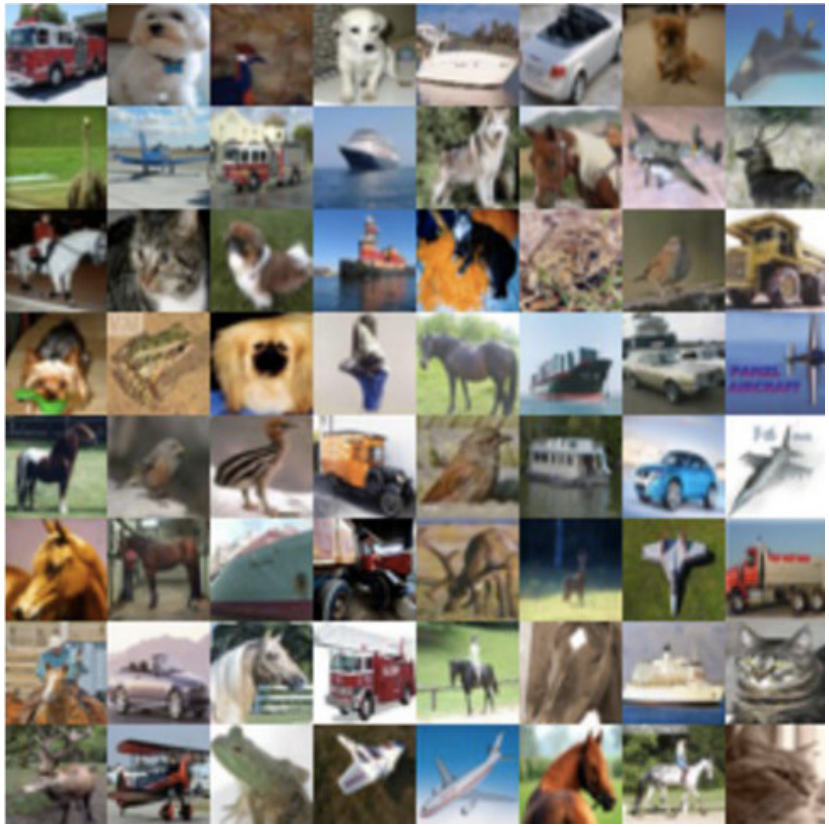


(Radford et al 2015)

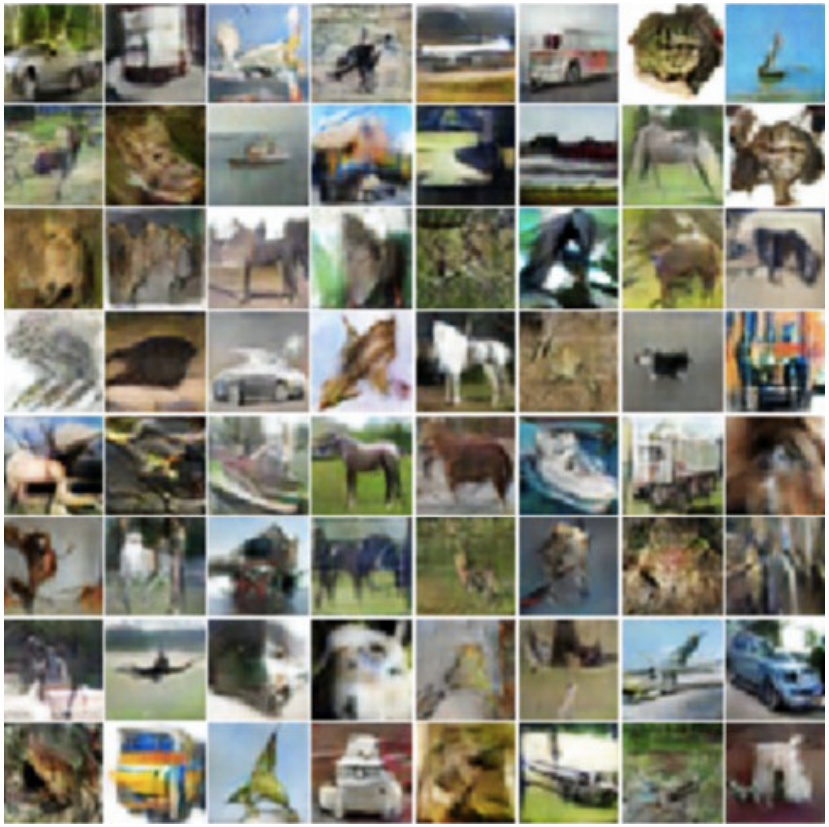
LSUN Bedrooms: Samples

(Radford et al 2015)

CIFAR



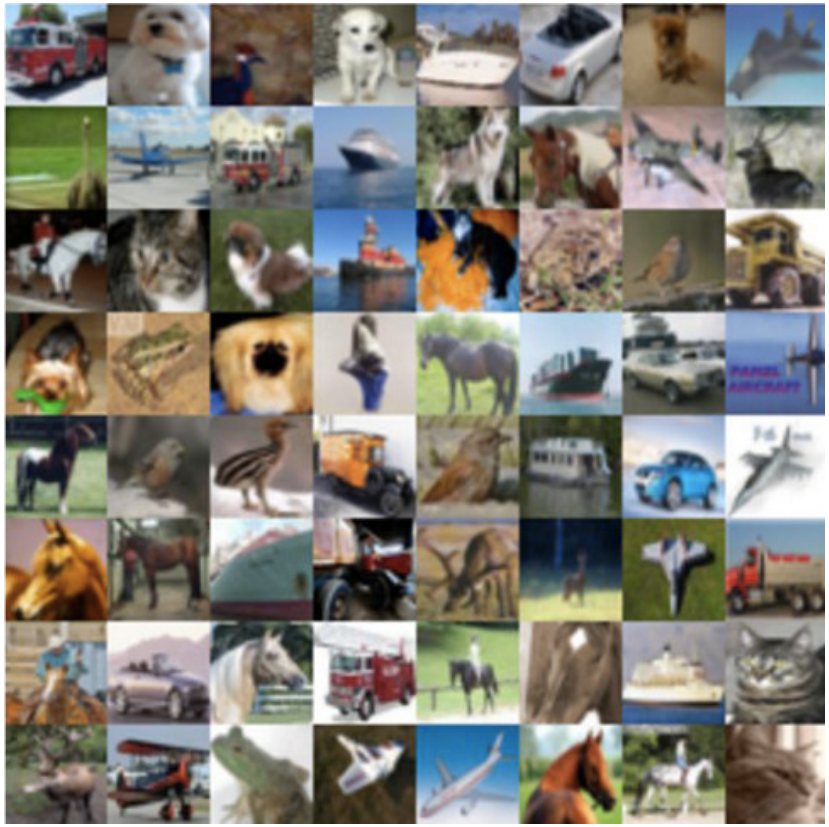
Training



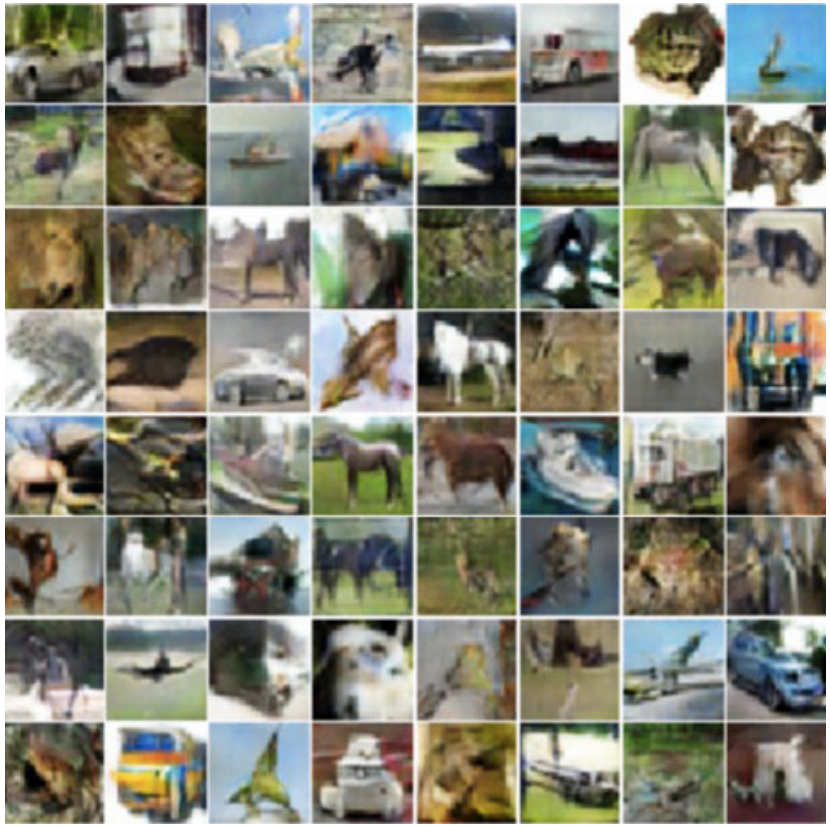
Samples

(Salimans et. al., 2016)

IMAGENET



Training

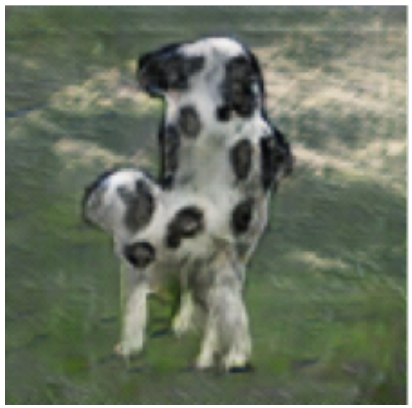
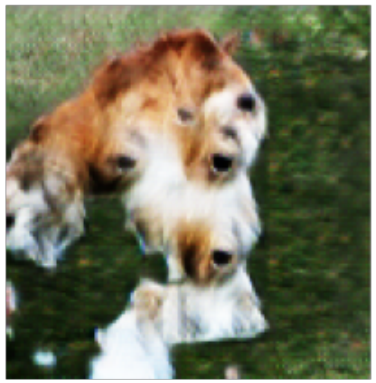


Samples

(Salimans et. al., 2016)

ImageNet: Cherry-Picked Results



- Open Question: How can we quantitatively evaluate these models!

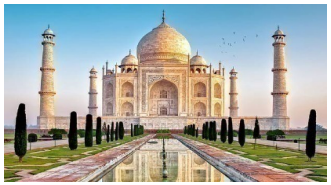
Slide Credit: Ian Goodfellow

Summary

- Efficient learning algorithms for Deep Unsupervised Models

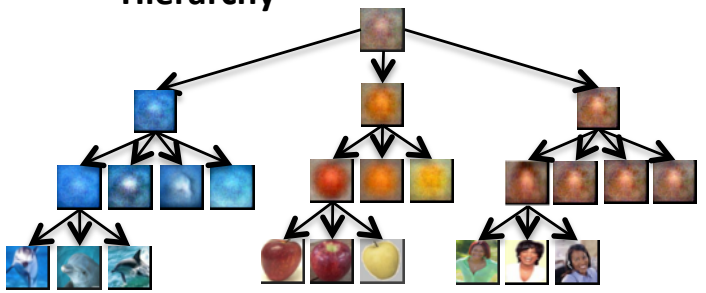
Text & image retrieval / Object recognition

Image Tagging

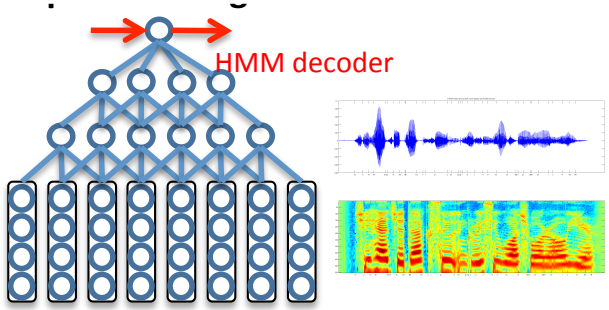


mosque, tower, building, cathedral, dome, castle

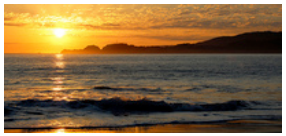
Learning a Category Hierarchy



Object Detection



Multimodal Data



sunset, pacific ocean, beach, seashore

- Deep models improve the current state-of-the art in many application domains:
  - Object recognition and detection, text and image retrieval, handwritten character and speech recognition, and others.

Thank you