Deep Learning

Russ Salakhutdinov

Machine Learning Department
Carnegie Mellon University
Canadian Institute for Advanced Research
MLSS 2017: Lecture 1

Carnegie Mellon University

MACHINE LEARNING DEPARTMENT

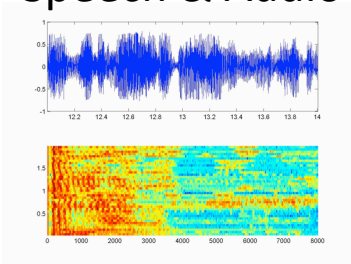
Mining for Structure

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

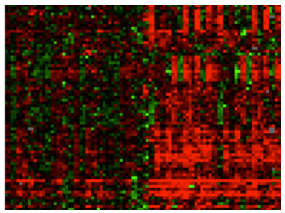
Images & Video

Text & Language

Speech & Audio

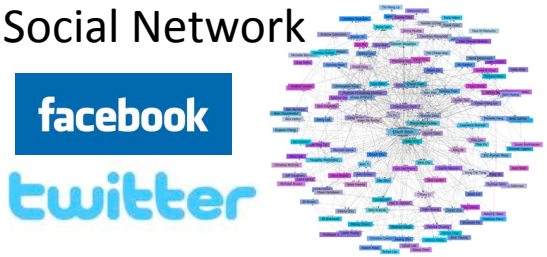


Gene Expression

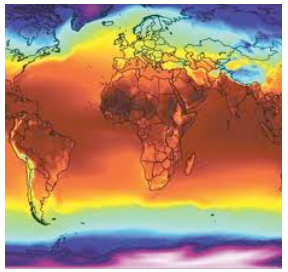


Product Recommendation

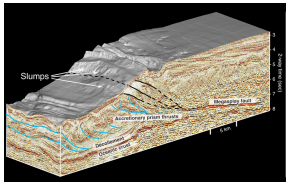
Relational Data/ Social Network



Climate Change



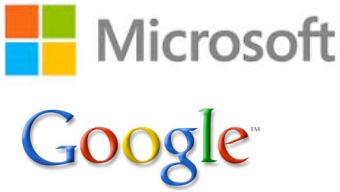
Geological Data



- Develop statistical models that can discover underlying structure, semantic relations, constraints, or invariances from data.
- Robust, adaptive models models that can deal with missing measurements, nonstationary distributions, multimodal data.

Impact of Deep Learning

- Speech Recognition



- Computer Vision

- Recommender Systems

- Language Understanding

- Drug Discovery and Medical Image Analysis

Example: Understanding Images

TAGS:

strangers, coworkers, conventioneers, attendants, patrons

Nearest Neighbor Sentence:

people taking pictures of a crazy person

Model Samples

- a group of people in a crowded area .
- a group of people are walking and talking .
- a group of people, standing around and talking .

Tutorial Roadmap

Part 1: Supervised (Discriminative) Learning: Deep Networks

Part 2: Unsupervised Learning: Deep Generative Models

Part 3: Open Research Questions

Supervised Learning

• Given a set of labeled training examples: $\{\mathbf{x}^{(t)}, y^{(t)}\}$, we perform Empirical Risk Minimization:

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{T} \sum_{t} \underbrace{l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})}_{\text{Loss function}} + \lambda \Omega(\boldsymbol{\theta})$$

where

- $f(\mathbf{x}^{(t)}; \theta)$ (non-linear) function mapping inputs to outputs, parameterized by θ -> Non-convex optimization
- $l(\mathbf{f}(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$ is the loss function.

Supervised Learning

• Given a set of labeled training examples: $\{\mathbf{x}^{(t)}, y^{(t)}\}$, we perform Empirical Risk Minimization:

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{T} \sum_{t} \underbrace{l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})}_{\text{Loss function}} + \underbrace{\lambda \Omega(\boldsymbol{\theta})}_{\text{Regularizer}}$$

where

- $f(\mathbf{x}^{(t)}; \theta)$ (non-linear) function mapping inputs to outputs, parameterized by θ -> Non-convex optimization
- $l(\mathbf{f}(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$ is the loss function.
- $\Omega(\boldsymbol{\theta})$ is a regularization term.

Supervised Learning

• Given a set of labeled training examples: $\{\mathbf{x}^{(t)}, y^{(t)}\}$, we perform Empirical Risk Minimization:

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{T} \sum_{t} \underbrace{l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})}_{\text{Loss function}} + \underbrace{\lambda \Omega(\boldsymbol{\theta})}_{\text{Regularizer}}$$

• Learning is cast as optimization.

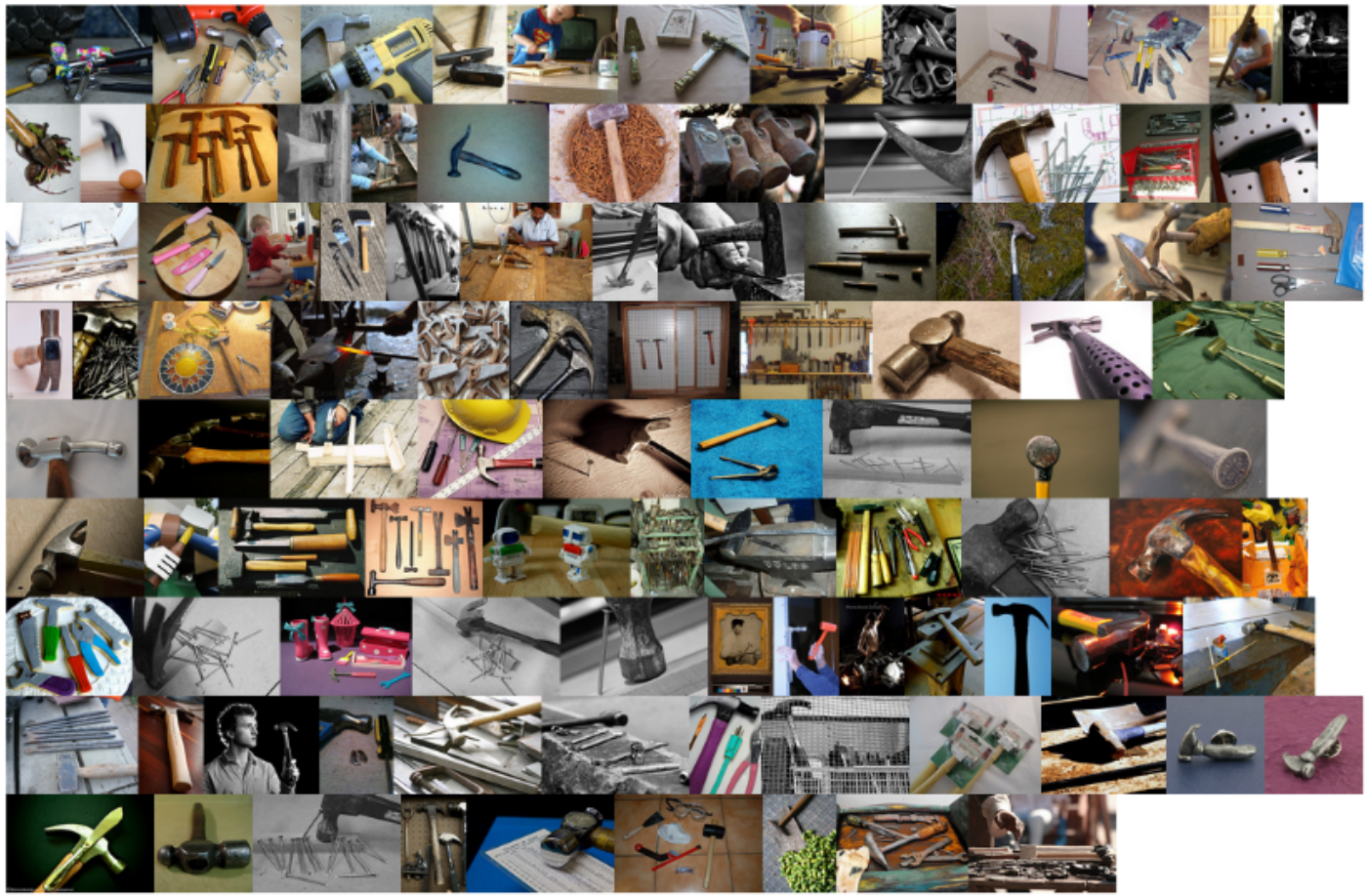
➢ For classification problems, we would like to minimize classification error.

➢ Loss function can sometimes be viewed as a surrogate for what we want to optimize (e.g. upper bound)

Example: ImageNet Dataset

- 1.2 million (225 x 225) images, 1000 classes

Examples of Hammer



(Deng et al., Imagenet: a large scale hierarchical image database, CVPR 2009)

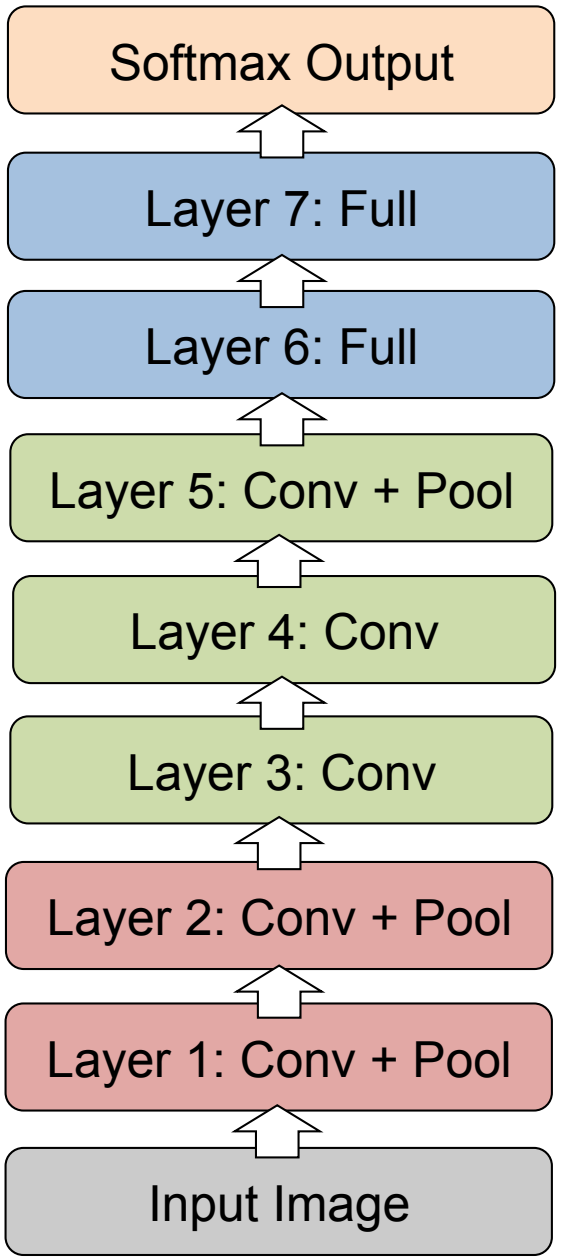
AlexNet

- Input: 225 x 225 image
- Output: Softmax over 1000 classes

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{T} \sum_{t} l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)}) + \lambda \Omega(\boldsymbol{\theta})$$

- $f(\mathbf{x}^{(t)}; \theta)$ differentiable, non-nonlinear function parameterized by θ: 8 layers, 60M parameters -> Non-convex optimization
- $l(\mathbf{f}(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$ is the cross entropy loss.
- $\Omega(\boldsymbol{\theta})$: L_2, early stopping, drop-out

- Achieves: 18.2% top-5 error

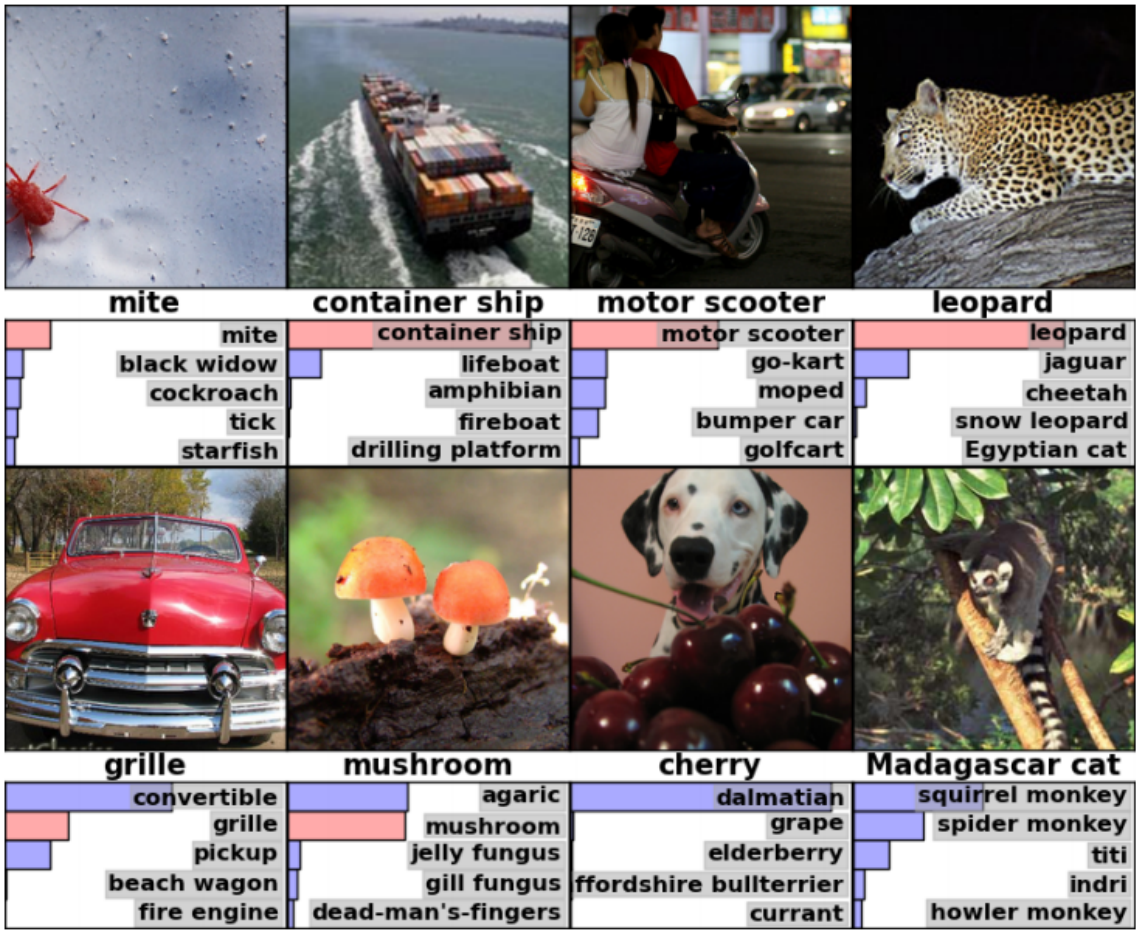


(Krizhevsky, Sutskever, Hinton, NIPS, 2012)

Important Breakthrough

- Deep Convolutional Nets for Vision (Supervised)

  Krizhevsky, A., Sutskever, I. and Hinton, G. E., ImageNet Classification with Deep Convolutional Neural Networks, NIPS, 2012.



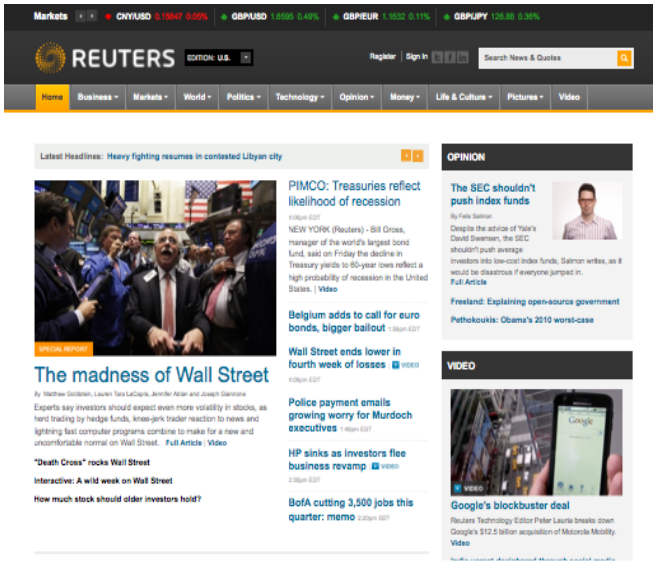
Unsupervised Learning

- Given a set of unlabeled training examples $\{\mathbf{x}^{(t)}\}$:

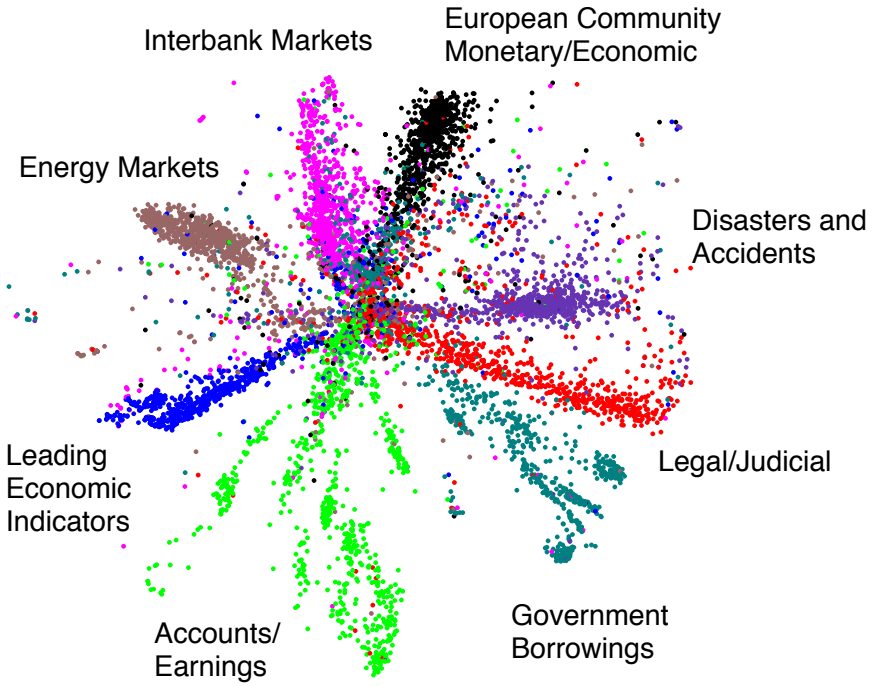
$$P(\mathbf{x}) = \frac{1}{\mathcal{Z}} \sum_{\mathbf{h}} \exp\left[\mathbf{x}^\top \mathbf{W} \mathbf{h}\right]$$

Vector of word counts on a webpage

Latent variables: hidden topics

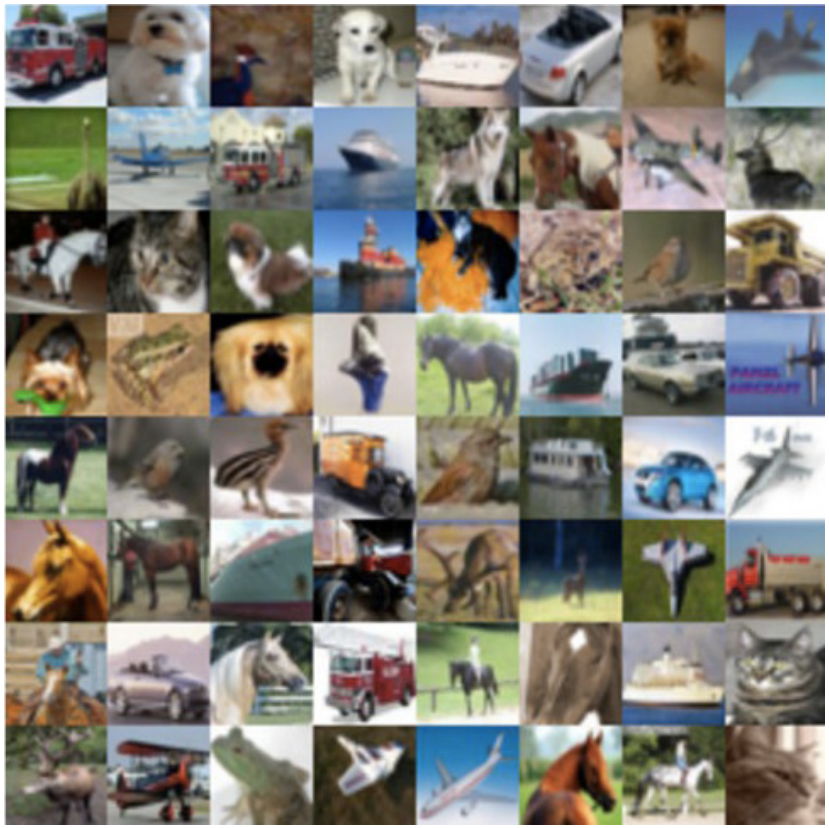


804,414 unlabelled newswire stories

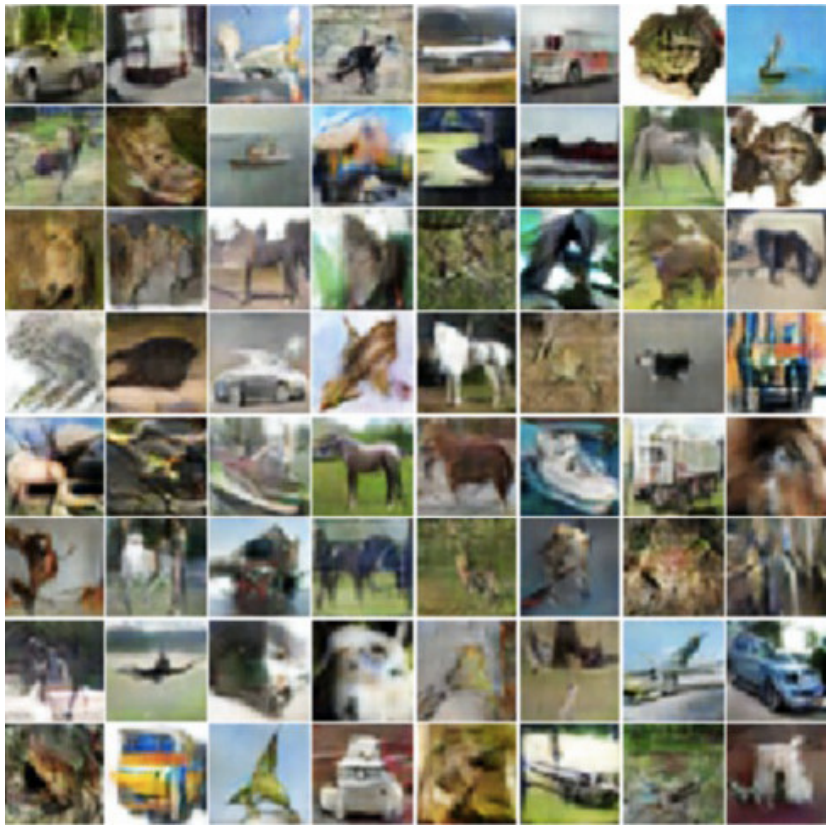


(Hinton & Salakhutdinov, Science, 2006)

Generative Adversarial Net Trained on ImageNet



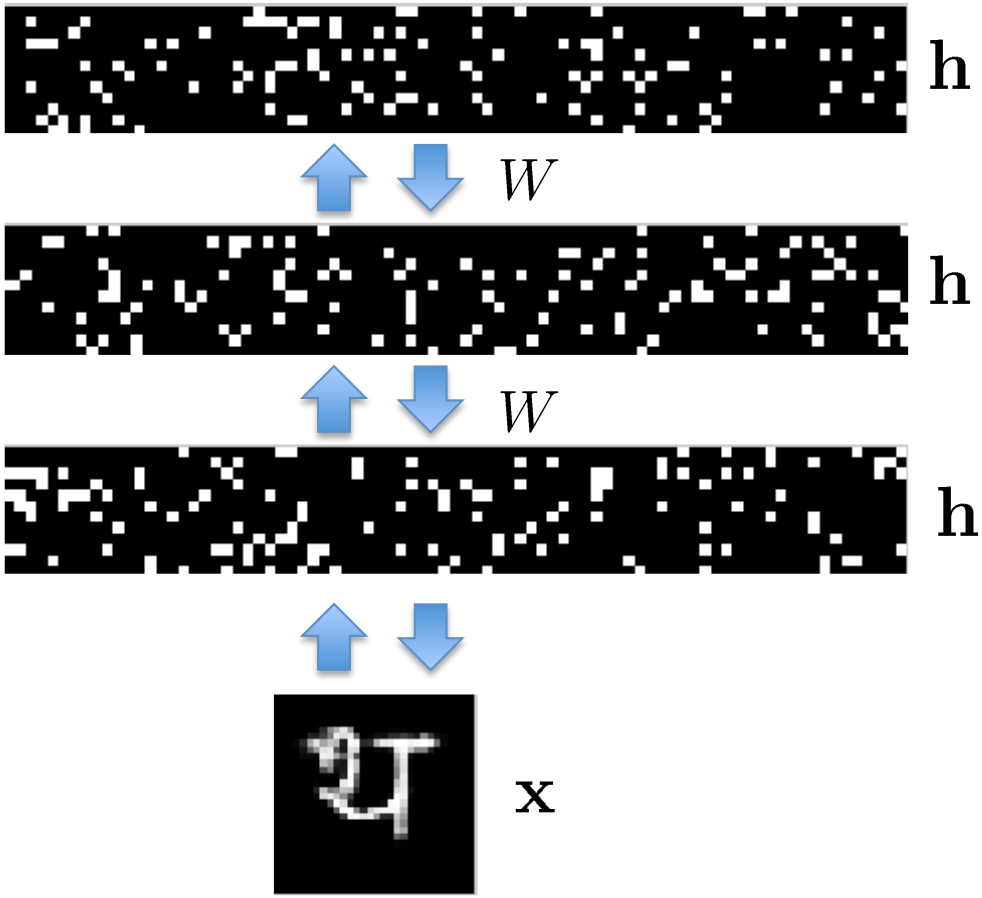
Training



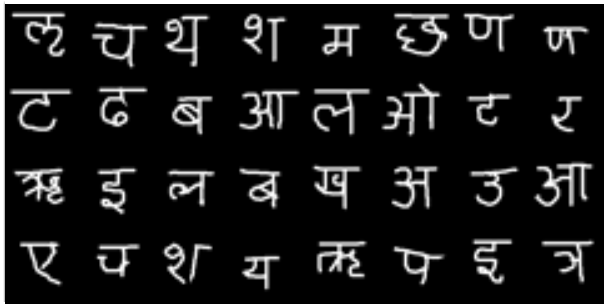
Samples

(Salimans et. al., 2016)

Boltzmann Machine



Observed Data



25,000 characters from 50 alphabets around the world.

Simulate a Markov chain whose stationary distribution is $P(\mathbf{x}|\mathbf{y} = \text{Sanskrit})$.

Talk Roadmap

Part 1: Supervised Learning: Deep Networks

- Definition and Training Neural Networks
- Recent Optimization / Regularization Techniques

Part 2: Unsupervised Learning: Learning Deep Generative Models

Part 3: Open Research Questions

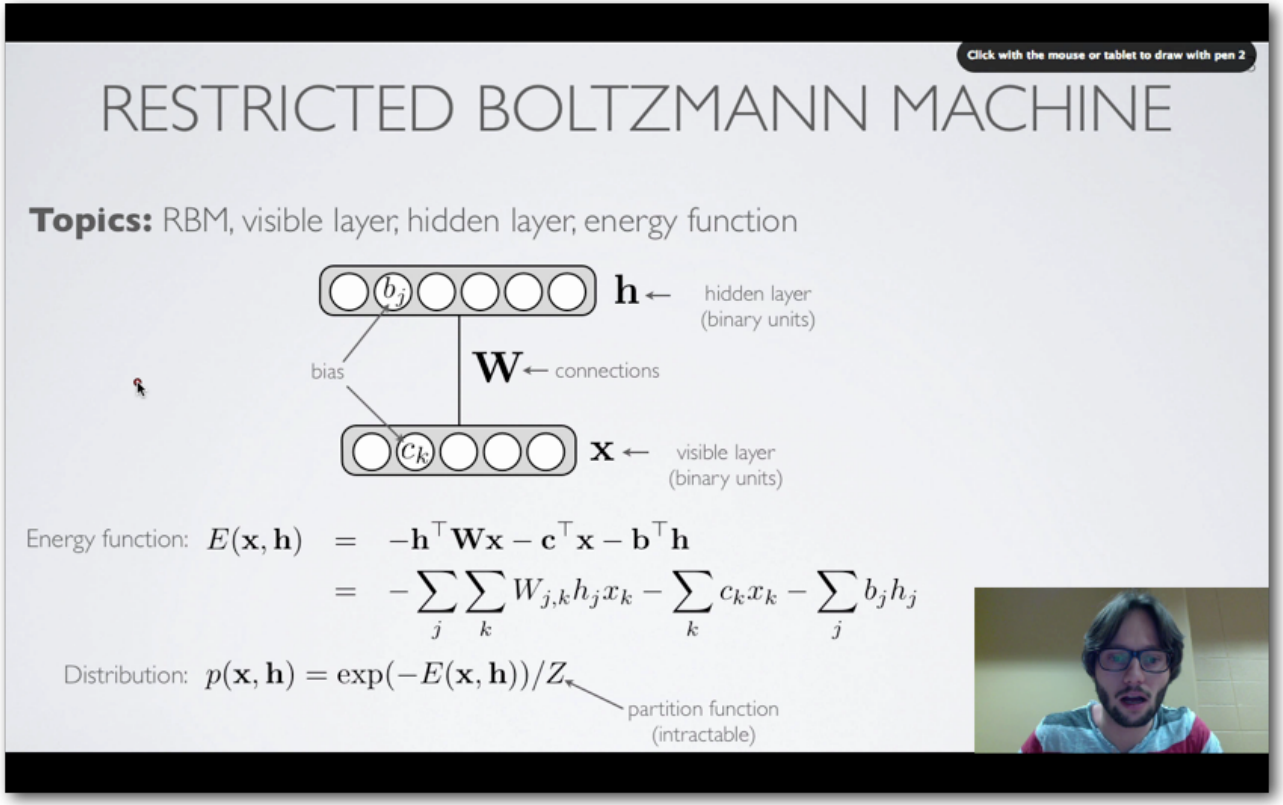
Neural Networks Online Course

• **Disclaimer**: Some of the material and slides for this lecture were borrowed from Hugo Larochelle's class on Neural Networks: https://sites.google.com/site/deeplearningsummerschool2016/

• Hugo's class covers many other topics: convolutional networks, neural language model, Boltzmann machines, autoencoders, sparse coding, etc.

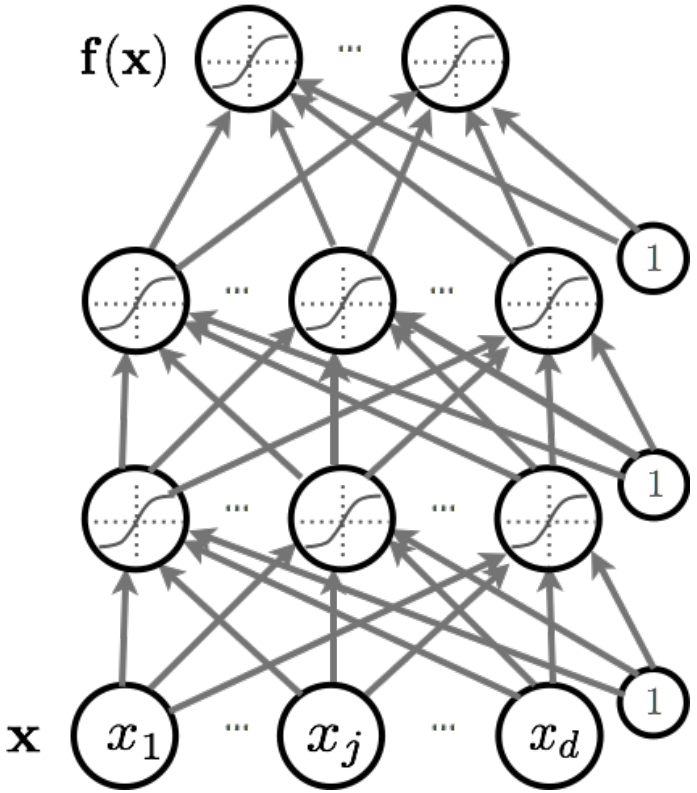
• We will use his material for some of the other lectures.

http://info.usherbrooke.ca/hlarochelle/neural_networks



Feedforward Neural Networks

- Definition of Neural Networks
  - Forward propagation
  - Types of units
  - Capacity of neural networks
- How to train neural nets:
  - Loss function
  - Backpropagation with gradient descent
- More recent techniques:
  - Dropout
  - Batch normalization
  - Unsupervised Pre-training



Artificial Neuron

- Neuron pre-activation (or input activation):

$$a(\mathbf{x}) = b + \sum_i w_i x_i = b + \mathbf{w}^\top \mathbf{x}$$

- Neuron output activation:

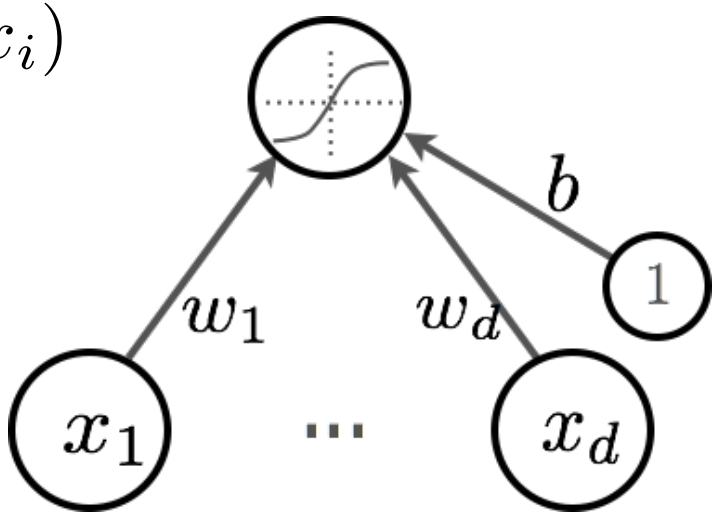
$$h(\mathbf{x}) = g(a(\mathbf{x})) = g(b + \sum_i w_i x_i)$$

where

$\mathbf{w}$ are the weights (parameters)

b is the bias term

$g(\cdot)$ is called the activation function

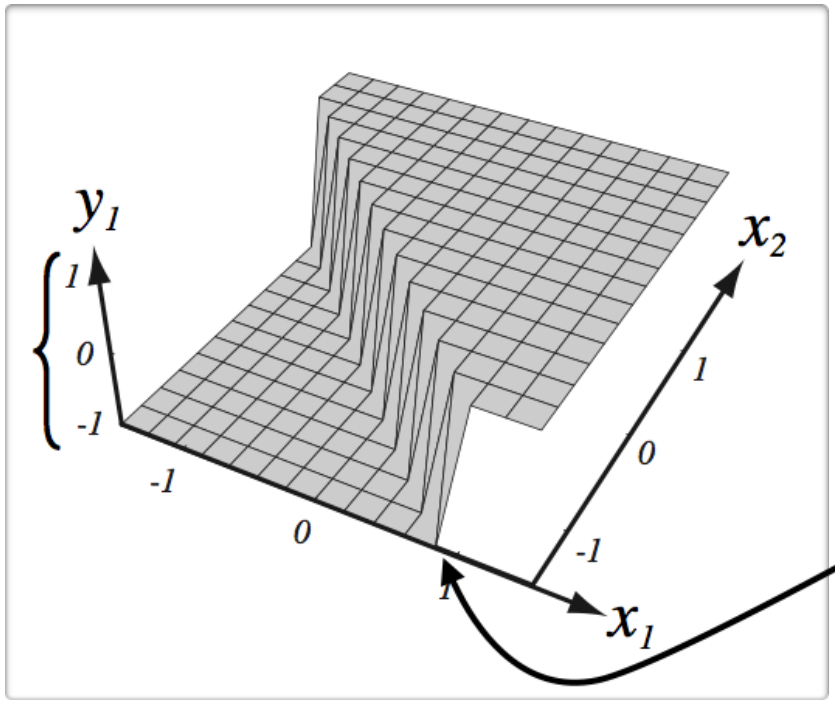


Artificial Neuron

- Output activation of the neuron:

$$h(\mathbf{x}) = g(a(\mathbf{x})) = g(b + \sum_i w_i x_i)$$

Range is determined by $g(\cdot)$



(from Pascal Vincent's slides)

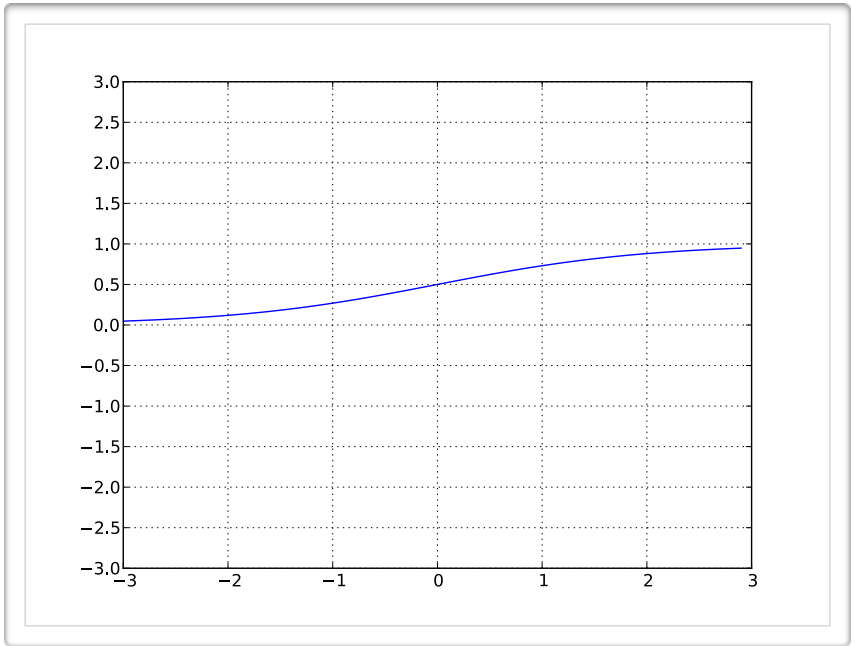
Bias only changes the position of the riff

Activation Function

- Sigmoid activation function:

$$g(a) = \text{sigm}(a) = \frac{1}{1+\exp(-a)}$$

  - Squashes the neuron's output between 0 and 1
  - Always positive
  - Bounded
  - Strictly Increasing

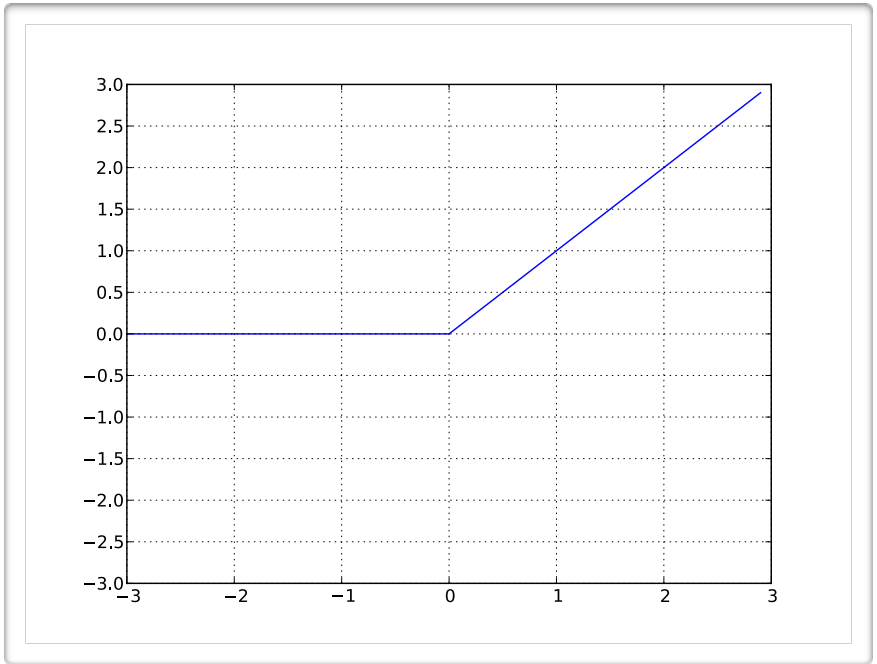


Activation Function

- Rectified linear (ReLU) activation function:

  - Bounded below by 0 (always non-negative)
  - Tends to produce units with sparse activities
  - Not upper bounded
  - Strictly increasing

$$g(a) = \text{reclin}(a) = \max(0, a)$$



Single Hidden Layer Neural Net

- Hidden layer pre-activation:

$$\mathbf{a}(\mathbf{x}) = \mathbf{b}^{(1)} + \mathbf{W}^{(1)}\mathbf{x}$$

$$\left(a(\mathbf{x})_i = b_i^{(1)} + \sum_j W_{i,j}^{(1)} x_j\right)$$

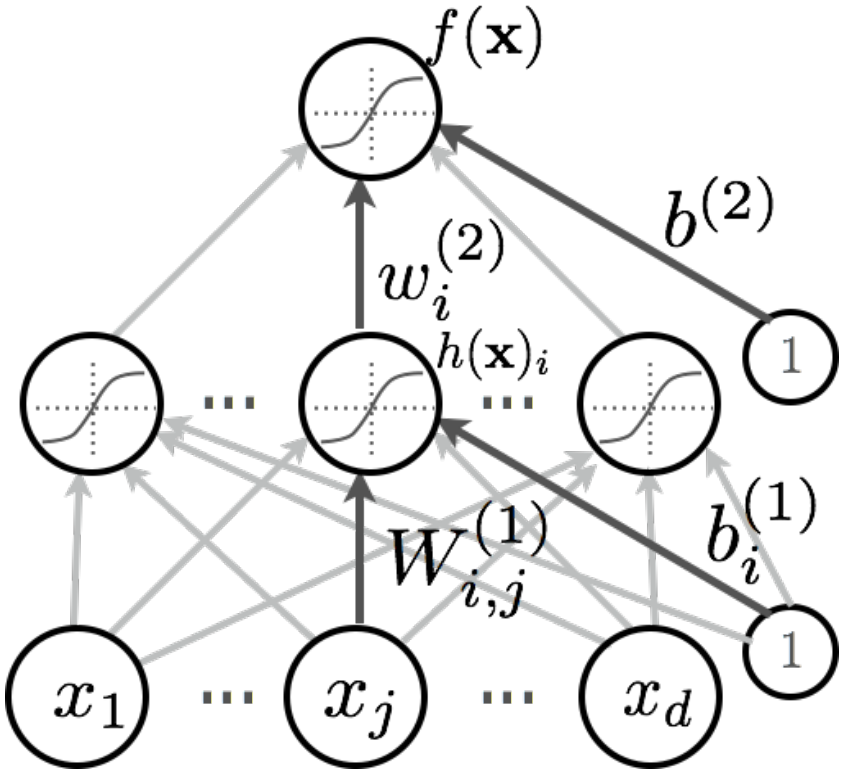
- Hidden layer activation:

$$\mathbf{h}(\mathbf{x}) = \mathbf{g}(\mathbf{a}(\mathbf{x}))$$

- Output layer activation:

$$f(\mathbf{x}) = o\left(b^{(2)} + \mathbf{w}^{(2)\top}\mathbf{h}^{(1)}\mathbf{x}\right)$$

Output activation function



Multilayer Neural Net

- Consider a network with L hidden layers.
  - layer pre-activation for k>0

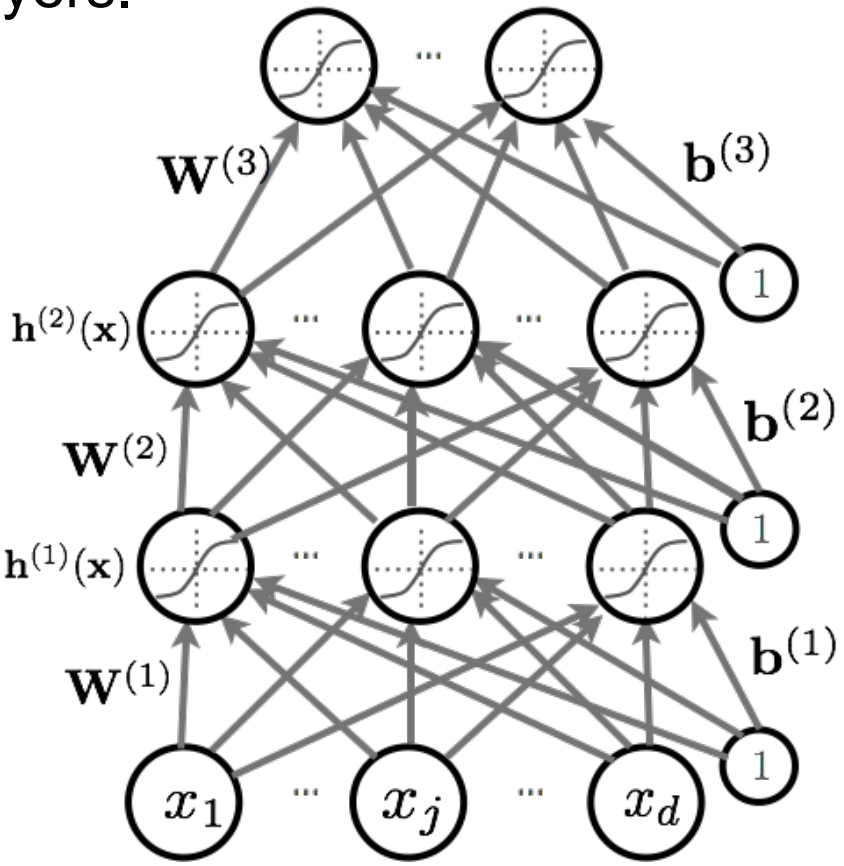
$$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{b}^{(k)} + \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}(\mathbf{x})$$

  - hidden layer activation from 1 to L:

$$\mathbf{h}^{(k)}(\mathbf{x}) = \mathbf{g}(\mathbf{a}^{(k)}(\mathbf{x}))$$

  - output layer activation (k=L+1):

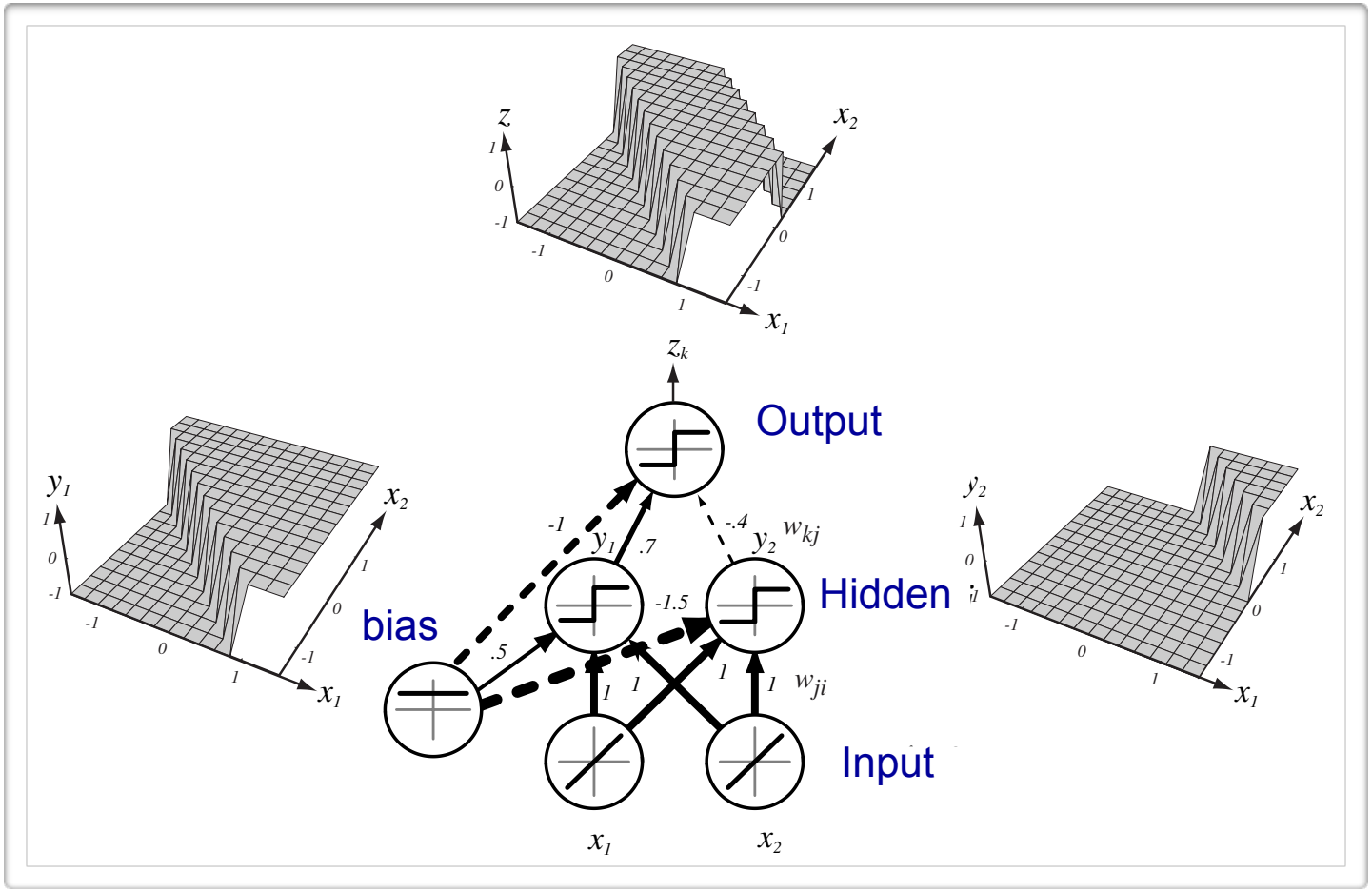
$$\mathbf{h}^{(L+1)}(\mathbf{x}) = \mathbf{o}(\mathbf{a}^{(L+1)}(\mathbf{x})) = \mathbf{f}(\mathbf{x})$$



$$(\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x})$$

Capacity of Neural Nets

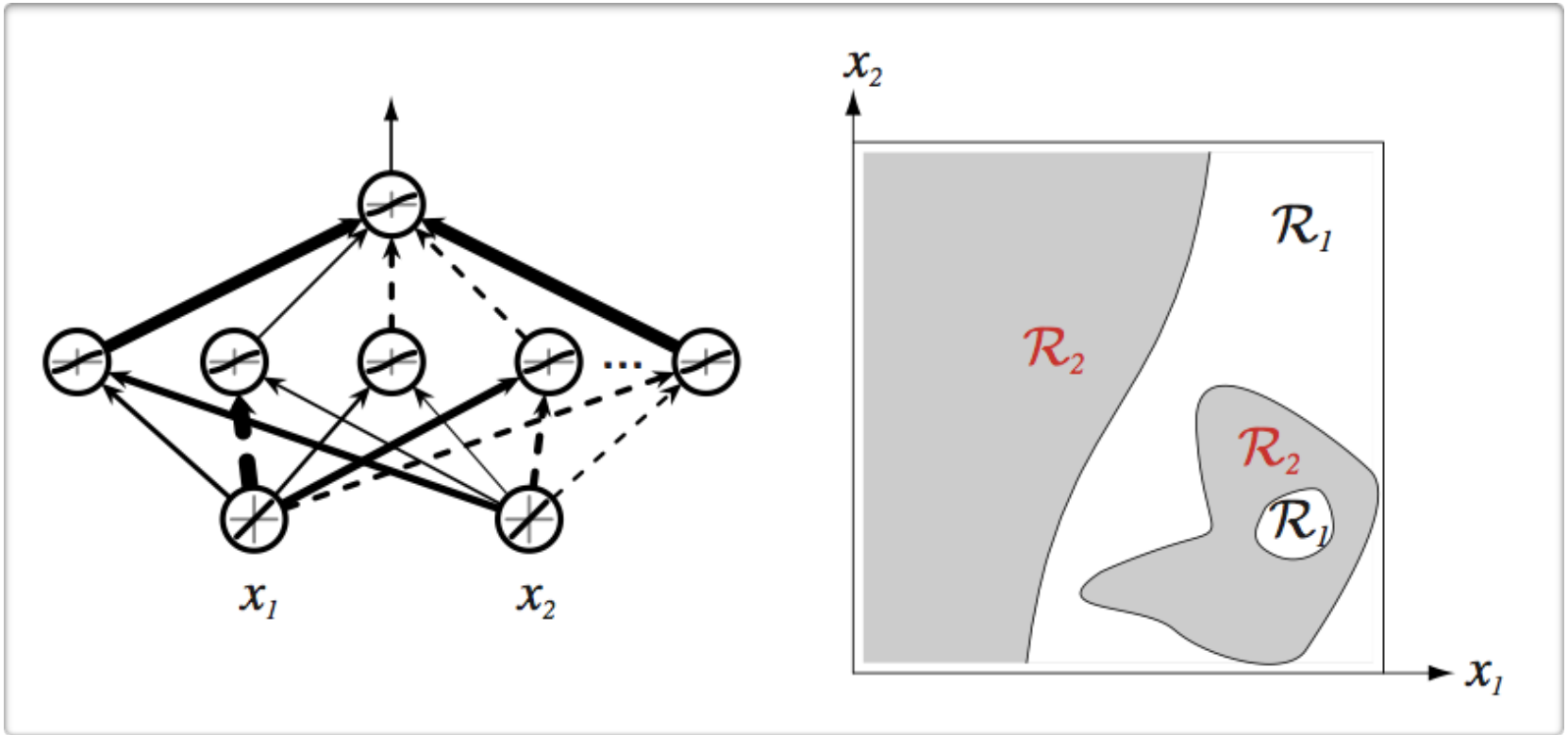
- Consider a single layer neural network



(from Pascal Vincent's slides)

Capacity of Neural Nets

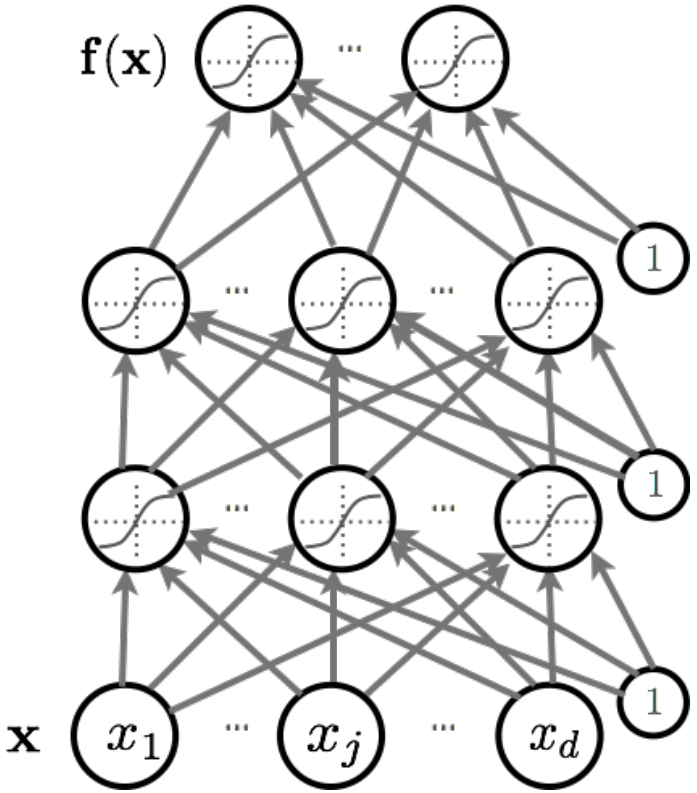
- Consider a single layer neural network



(from Pascal Vincent's slides)

Feedforward Neural Networks

- How neural networks predict f(x) given an input x:
  - Forward propagation
  - Types of units
  - Capacity of neural networks
- How to train neural nets:
  - Loss function
  - Backpropagation with gradient descent
- More recent techniques:
  - Dropout
  - Batch normalization
  - Unsupervised Pre-training



Training

- Empirical Risk Minimization:

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{T} \sum_{t} \underbrace{l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})}_{\text{Loss function}} + \underbrace{\lambda \Omega(\boldsymbol{\theta})}_{\text{Regularizer}}$$

- To train a neural net, we need:
  - Loss function: $l(\mathbf{f}(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$
  - A procedure to compute gradients: $\nabla_{\boldsymbol{\theta}} l(\mathbf{f}(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$
  - Regularizer and its gradient: $\Omega(\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$

Stochastic Gradient Descend

- Perform updates after seeing each example:
  - Initialize: $\boldsymbol{\theta} \equiv \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \ldots, \mathbf{W}^{(L+1)}, \mathbf{b}^{(L+1)}\}$
  - For t=1:T
    - for each training example $(\mathbf{x}^{(t)}, y^{(t)})$

$$\Delta = -\nabla_{\boldsymbol{\theta}} l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)}) - \lambda \nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \, \Delta$$

Training epoch
=
Iteration of all examples

- To train a neural net, we need:
  - Loss function: $l(\mathbf{f}(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$
  - A procedure to compute gradients: $\nabla_{\boldsymbol{\theta}} l(\mathbf{f}(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)})$
  - Regularizer and its gradient: $\Omega(\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$

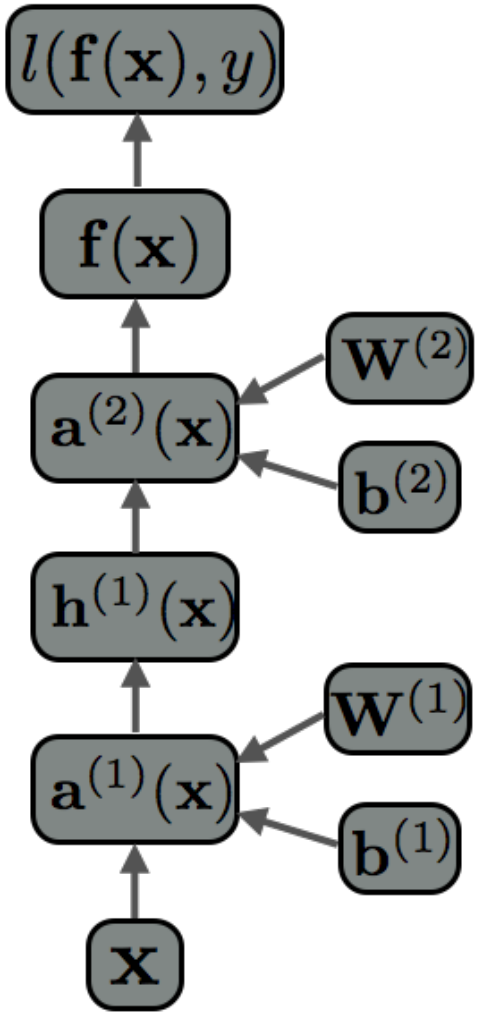
Backpropagation Algorithm: Computational Flow Graph

• Forward propagation can be represented as an acyclic flow graph

• Forward propagation can be implemented in a modular way:

- Each box can be an object with an fprop method, that computes the value of the box given its children

- Calling the fprop method of each box in the right order yields forward propagation

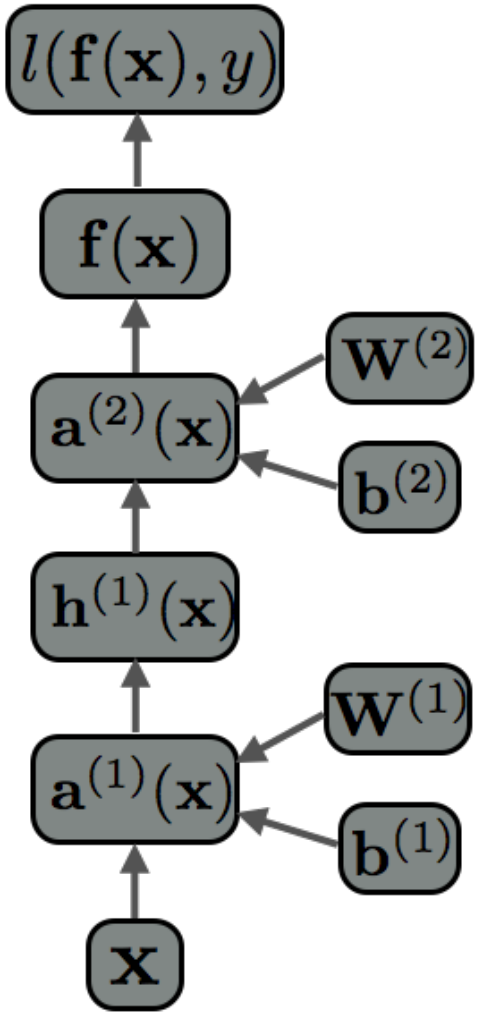


Backpropagation Algorithm: Computational Flow Graph

- Each object also has a bprop method

  – it computes the gradient of the loss with respect to each child box.

- By calling bprop in the reverse order, we obtain backpropagation



Weight Decay

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{T} \sum_t l(f(\mathbf{x}^{(t)}; \boldsymbol{\theta}), y^{(t)}) + \lambda \Omega(\boldsymbol{\theta})$$

- L2 regularization:

$$\Omega(\boldsymbol{\theta}) = \sum_k \sum_i \sum_j \left(W_{i,j}^{(k)}\right)^2 = \sum_k ||\mathbf{W}^{(k)}||_F^2$$

- L1 regularization:

$$\Omega(\boldsymbol{\theta}) = \sum_k \sum_i \sum_j |W_{i,j}^{(k)}|$$

  - Only applies to weights, not biases (weigh decay)

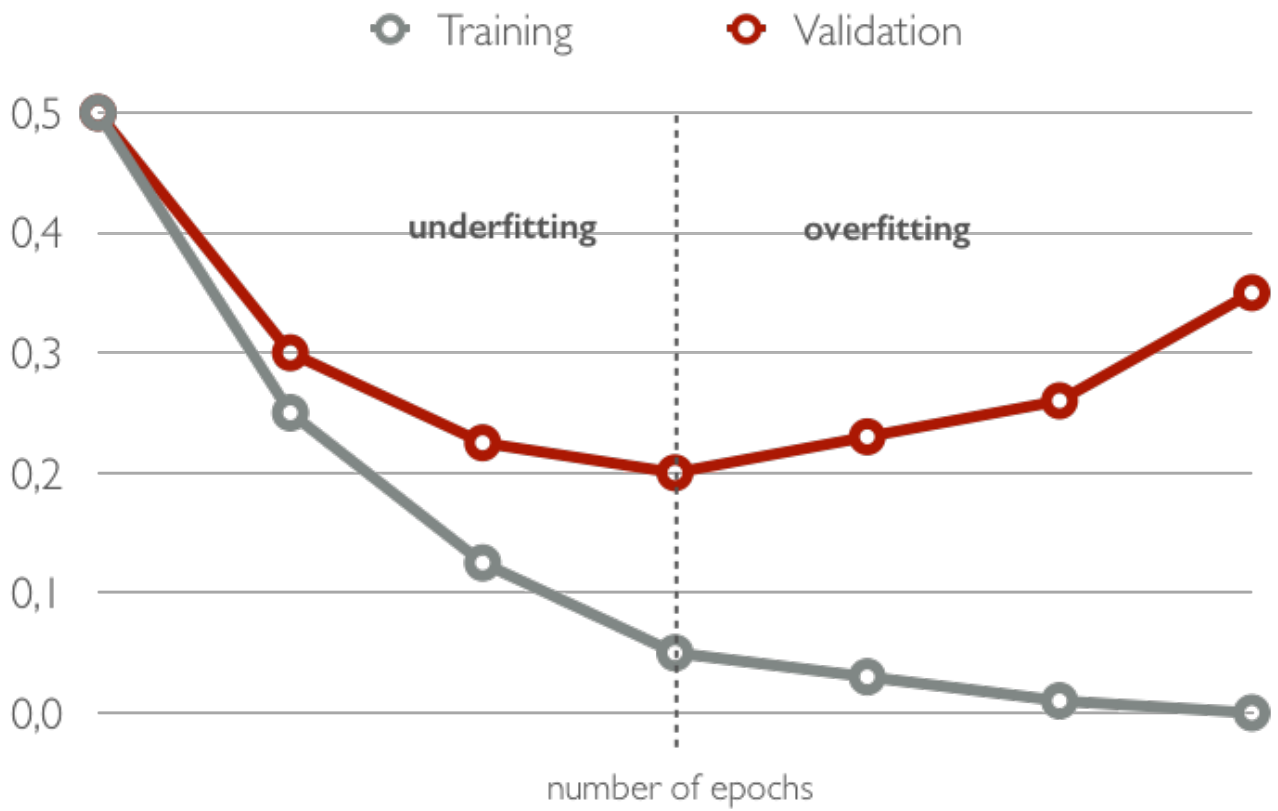
Model Selection

- Training Protocol:
  - Train your model on the Training Set $\mathcal{D}^{\text{train}}$
  - For model selection, use Validation Set $\mathcal{D}^{\text{valid}}$
    - Hyper-parameter search: hidden layer size, learning rate, number of iterations/epochs, etc.
  - Estimate generalization performance using the Test Set $\mathcal{D}^{\text{test}}$

- Generalization is the behavior of the model on **unseen examples**.

Early Stopping

• To select the number of epochs, stop training when validation set error increases (with some look ahead).



Mini-batch, Momentum

• Make updates based on a mini-batch of examples (instead of a single example):

- the gradient is the average regularized loss for that mini-batch
- can give a more accurate estimate of the gradient
- can leverage matrix/matrix operations, which are more efficient

• Momentum: Can use an exponential average of previous gradients:

$$\overline{\nabla}_{\boldsymbol{\theta}}^{(t)} = \nabla_{\boldsymbol{\theta}} l(\mathbf{f}(\mathbf{x}^{(t)}), y^{(t)}) + \beta \overline{\nabla}_{\boldsymbol{\theta}}^{(t-1)}$$

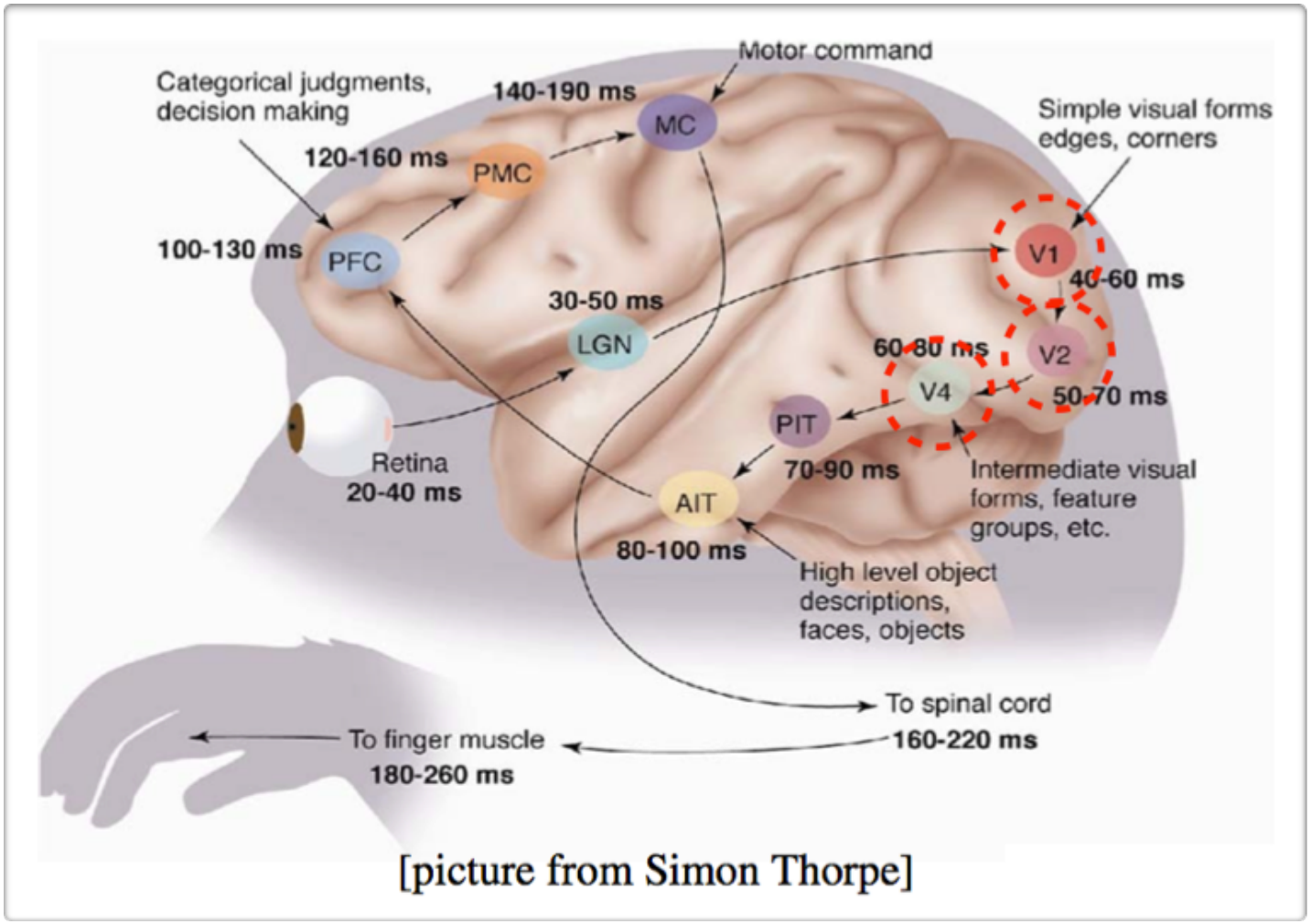
- can get pass plateaus more quickly, by “gaining momentum”

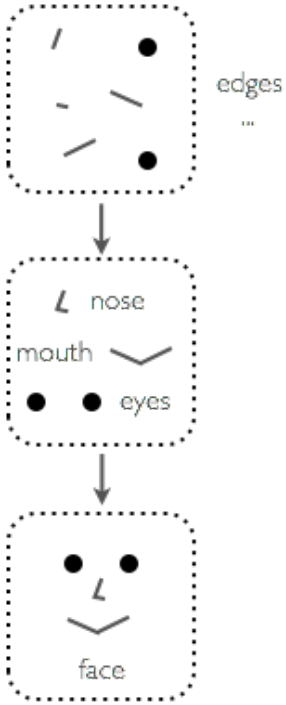
Learning Distributed Representations

- Deep learning: learning models with multilayer representations
  - multilayer (feed-forward) neural networks
  - multilayer graphical model (deep belief network, deep Boltzmann machine)

- Each layer learns "distributed representation"
  - Units in a layer are not mutually exclusive
    - each unit is a separate feature of the input
    - two units can be "active" at the same time
  - Units do not correspond to a partitioning (clustering) of the inputs
    - in clustering, an input can only belong to a single cluster

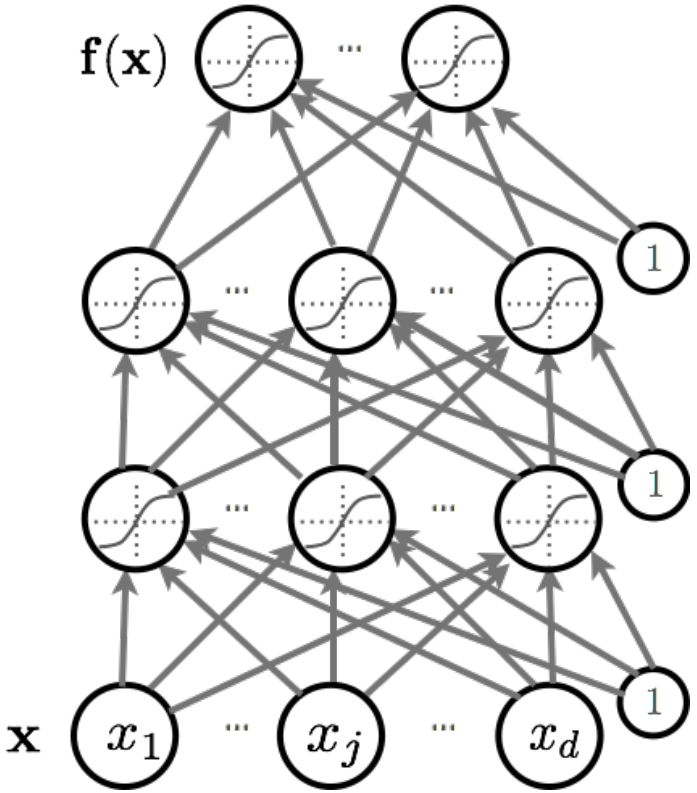
Inspiration from Visual Cortex





Feedforward Neural Networks

- How neural networks predict f(x) given an input x:
  - Forward propagation
  - Types of units
  - Capacity of neural networks
- How to train neural nets:
  - Loss function
  - Backpropagation with gradient descent
- More recent techniques:
  - Dropout
  - Batch normalization



Best Practice

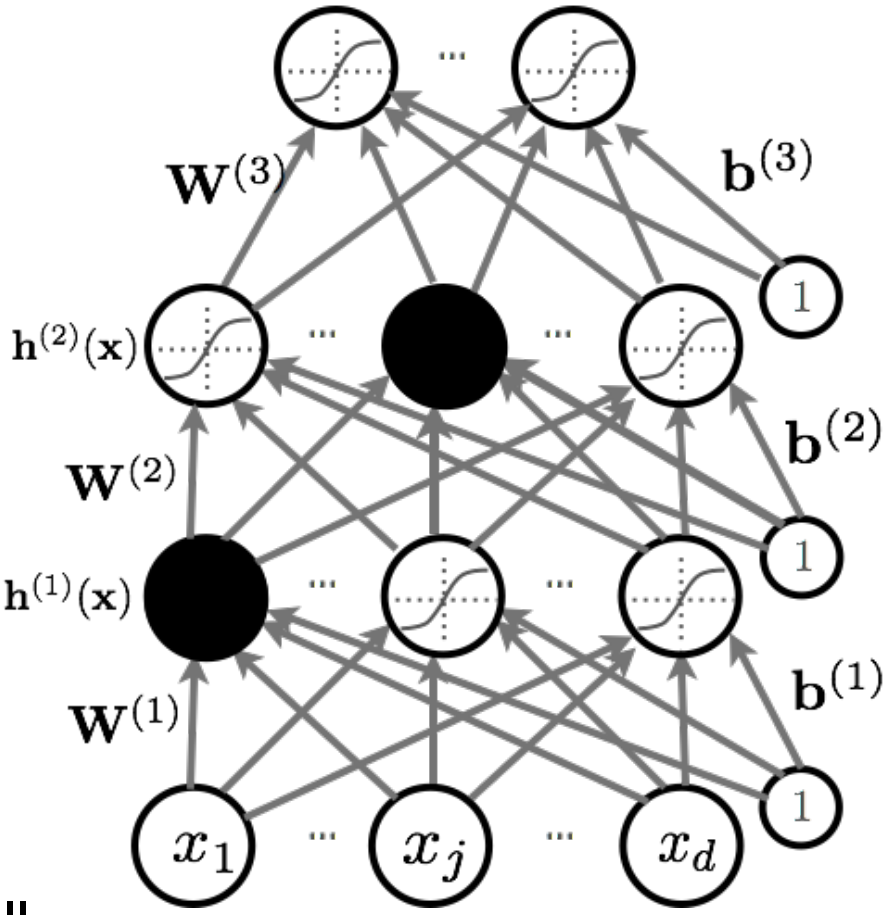
- Given a dataset D, pick a model so that:
  - You can achieve 0 training error– Overfit on the training set
- Regularize the model (e.g. using Dropout).
- SGD with momentum, batch-normalization, and dropout usually works very well.

Dropout

• Key idea: Cripple neural network by removing hidden units stochastically

- each hidden unit is set to 0 with probability 0.5
- hidden units cannot co-adapt to other units
- hidden units must be more generally useful

• Could use a different dropout probability, but 0.5 usually works well



(Srivastava, Hinton, Krizhevsky, Sutskever, Salakhutdinov, JMLR 2014)

Dropout

- Use random binary masks $\mathbf{m}^{(k)}$
  - layer pre-activation for k>0

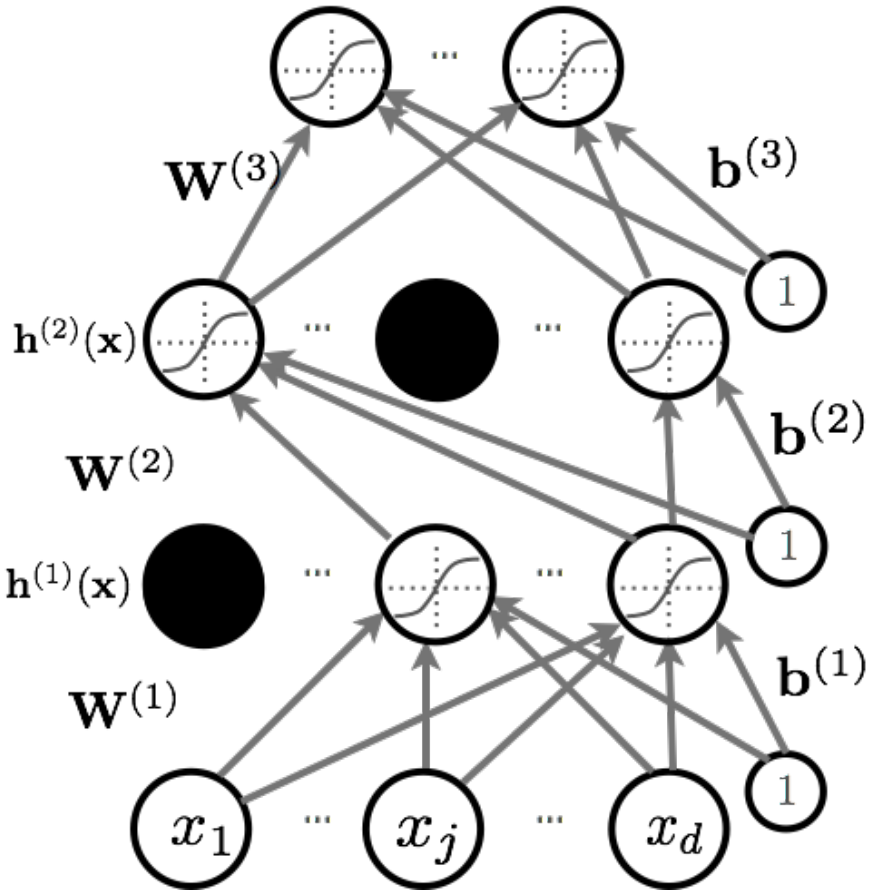
$$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{b}^{(k)} + \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}(\mathbf{x})$$

  - hidden layer activation (k=1 to L):

$$\mathbf{h}^{(k)}(\mathbf{x}) = \mathbf{g}(\mathbf{a}^{(k)}(\mathbf{x})) \odot \mathbf{m}^{(k)}$$

  - Output activation (k=L+1)

$$\mathbf{h}^{(L+1)}(\mathbf{x}) = \mathbf{o}(\mathbf{a}^{(L+1)}(\mathbf{x})) = \mathbf{f}(\mathbf{x})$$



(Srivastava, Hinton, Krizhevsky, Sutskever, Salakhutdinov, JMLR 2014)

Dropout at Test Time

- At test time, we replace the masks by their expectation
  - This is simply the constant vector 0.5 if dropout probability is 0.5
  - For single hidden layer: equivalent to taking the geometric average of all neural networks, with all possible binary masks
- Can be combined with unsupervised pre-training
- Beats regular backpropagation on many datasets
- Ensemble: Can be viewed as a geometric average of exponential number of networks.

Batch Normalization

- Normalizing the inputs will speed up training (Lecun et al. 1998)
  - could normalization be useful at the level of the hidden layers?

- Batch normalization is an attempt to do that (Ioffe and Szegedy, 2015)
  - each unit's pre-activation is normalized (mean subtraction, stddev division)
  - during training, mean and stddev is computed for each minibatch
  - backpropagation takes into account the normalization
  - at test time, the global mean / stddev is used

(Ioffe and Szegedy, ICML 2015)

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Learned linear transformation to adapt to non-linear activation function (γ and β are trained)

(Ioffe and Szegedy, ICML 2015)

Batch Normalization

- Why normalize the pre-activation?
  - can help keep the pre-activation in a non-saturating regime (though the linear transform $y_i \leftarrow \gamma \widehat{x}_i + \beta$ could cancel this effect)

- Use the global mean and stddev at test time.
  - removes the stochasticity of the mean and stddev
  - requires a final phase where, from the first to the last hidden layer
    - propagate all training data to that layer
    - compute and store the global mean and stddev of each unit
  - for early stopping, could use a running average

(Ioffe and Szegedy, ICML 2015)

Optimization Tricks

• SGD with momentum, batch-normalization, and dropout usually works very well

• Pick learning rate by running on a subset of the data

- Start with large learning rate & divide by 2 until loss does not diverge
- Decay learning rate by a factor of ~100 or more by the end of training

• Use ReLU nonlinearity

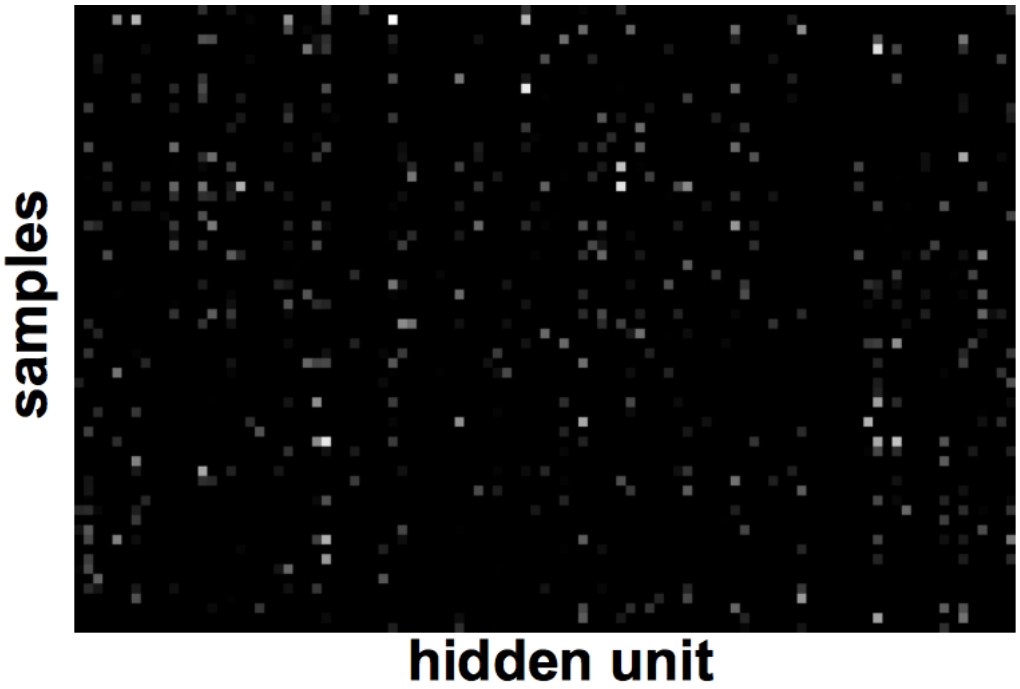
• Initialize parameters so that each feature across layers has similar variance. Avoid units in saturation.

[From Marc'Aurelio Ranzato, CVPR 2014 tutorial]

Visualization

- Check gradients numerically by finite differences

- Visualize features (features need to be uncorrelated) and have high variance



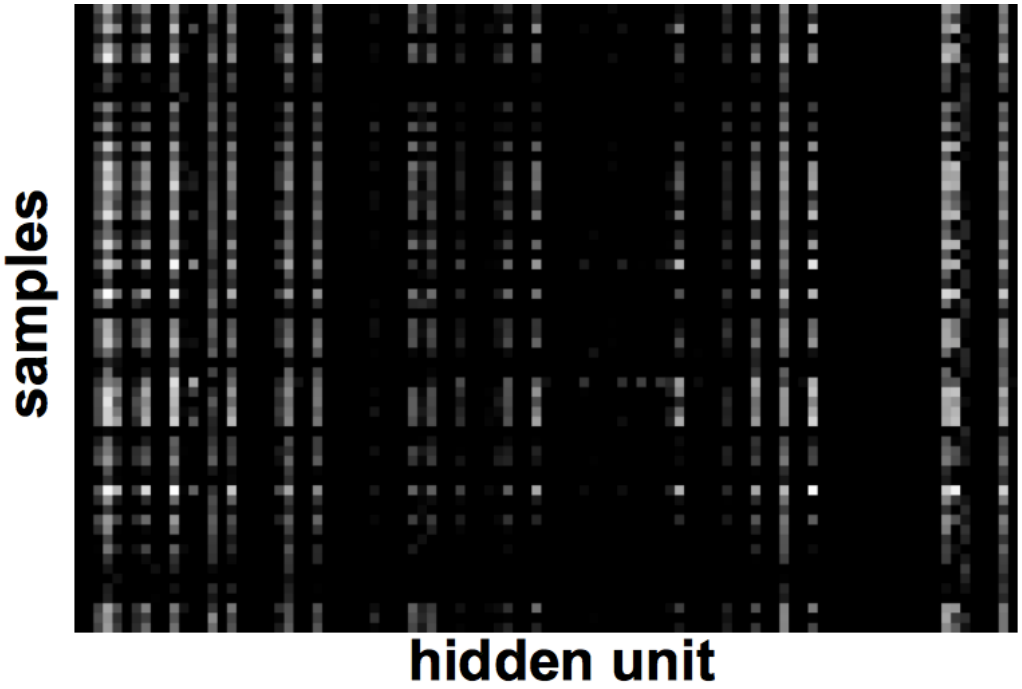
- Good training: hidden units are sparse across samples

[From Marc'Aurelio Ranzato, CVPR 2014 tutorial]

Visualization

- Check gradients numerically by finite differences

- Visualize features (features need to be uncorrelated) and have high variance



- Bad training: many hidden units ignore the input and/or exhibit strong correlations

Debugging on Small Dataset

- Next, make sure your model can overfit on a smaller dataset (~ 500-1000 examples)
- If not, investigate the following situations:
  - Are some of the units saturated, even before the first update?
    - scale down the initialization of your parameters for these units
    - properly normalize the inputs
  - Is the training error bouncing up and down?
    - decrease the learning rate
- This does not mean that you have computed gradients correctly:
  - You could still overfit with some of the gradients being wrong

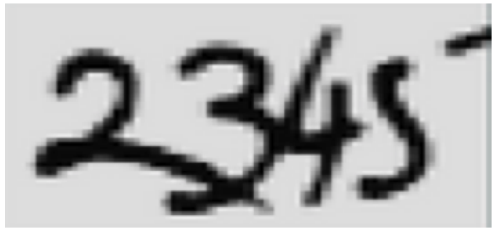
Computer Vision

- Design algorithms that can process visual data to accomplish a given task:

  - For example, object recognition: Given an input image, identify which object it contains

ConvNets: Examples

- Optical Character Recognition, House Number and Traffic Sign classification



Ciresan et al. “MCDNN for image classification” CVPR 2012
Wan et al. “Regularization of neural networks using dropconnect” ICML 2013
Goodfellow et al. “Multi-digit nuber recognition from StreetView...” ICLR 2014
Jaderberg et al. “Synthetic data and ANN for natural scene text recognition” arXiv 2014

Architecture

- How can we select the right architecture:
  - Manual tuning of features is now replaced with the manual tuning of architechtures

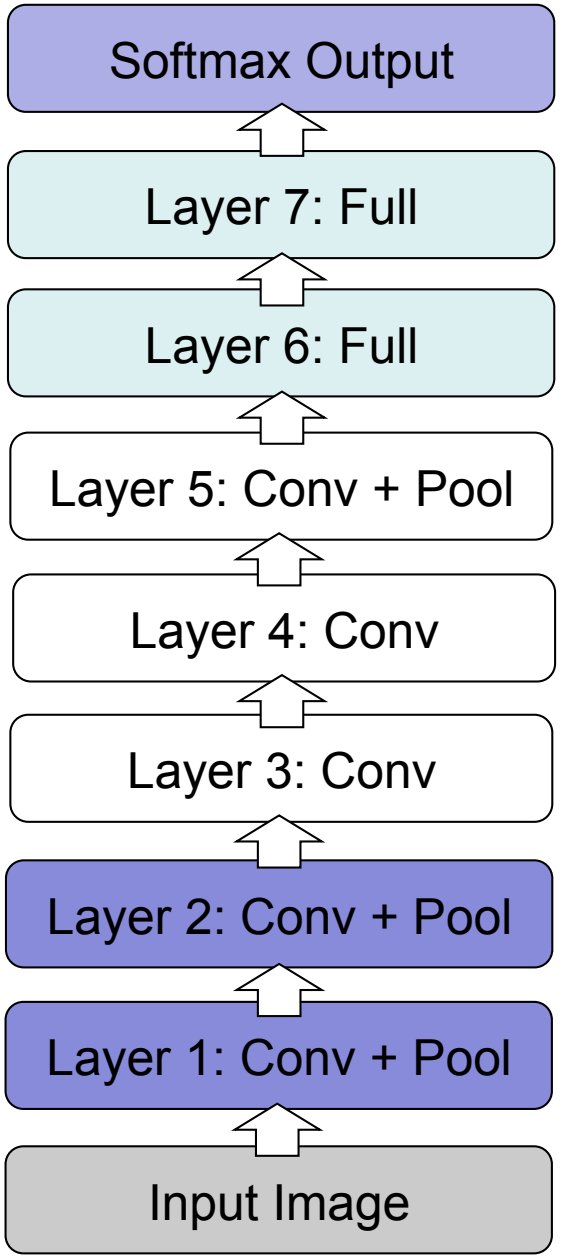
- Depth
- Width
- Parameter count

How to Choose Architecture

- Many hyper-parameters:
  - Number of layers, number of feature maps
- Cross Validation
- Grid Search (need lots of GPUs)
- Smarter Strategies
  - Random search
  - Bayesian Optimization

AlexNet

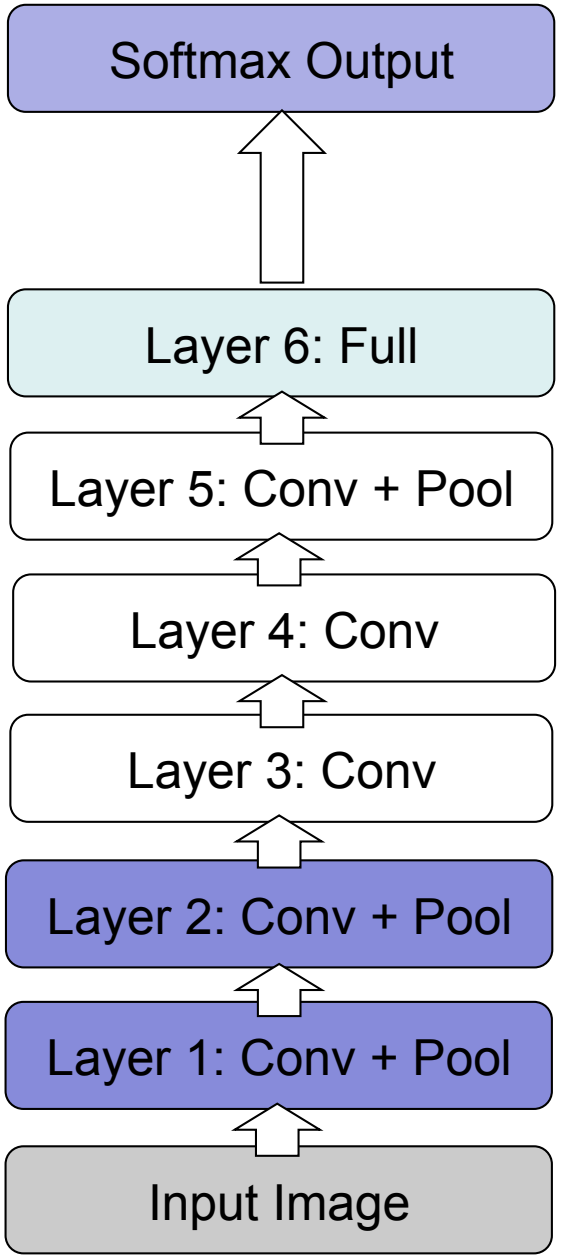
- 8 layers total
- Trained on Imagenet dataset [Deng et al. CVPR'09]
- 18.2% top-5 error



[From Rob Fergus' CIFAR 2016 tutorial]

AlexNet

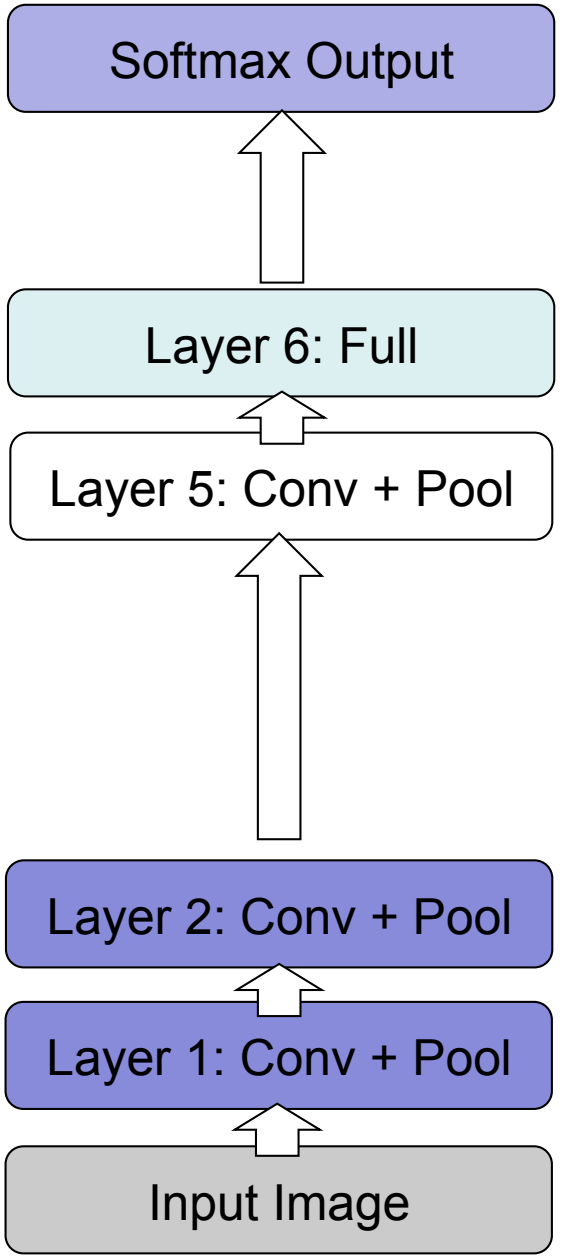
- Remove top fully connected layer 7
- Drop ~16 million parameters
- Only 1.1% drop in performance!



[From Rob Fergus' CIFAR 2016 tutorial]

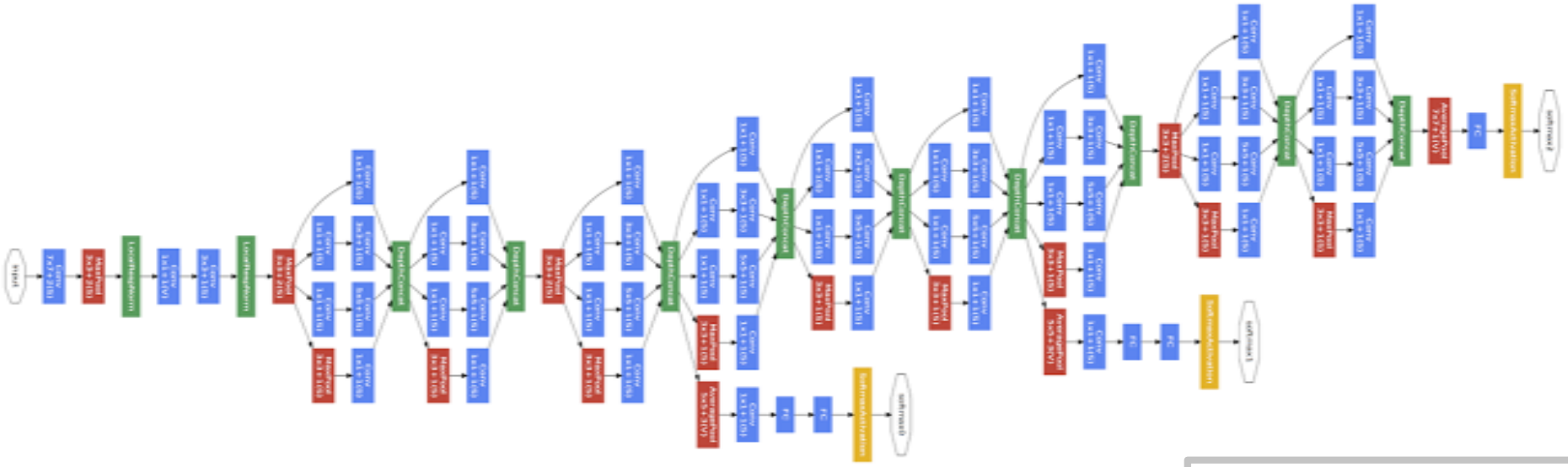
AlexNet

- Let us remove upper feature extractor layers and fully connected:
  - Layers 3,4, 6 and 7
- Drop ~50 million parameters
- **33.5 drop in performance!**
- Depth of the network is the key.



[From Rob Fergus' CIFAR 2016 tutorial]

GoogLeNet

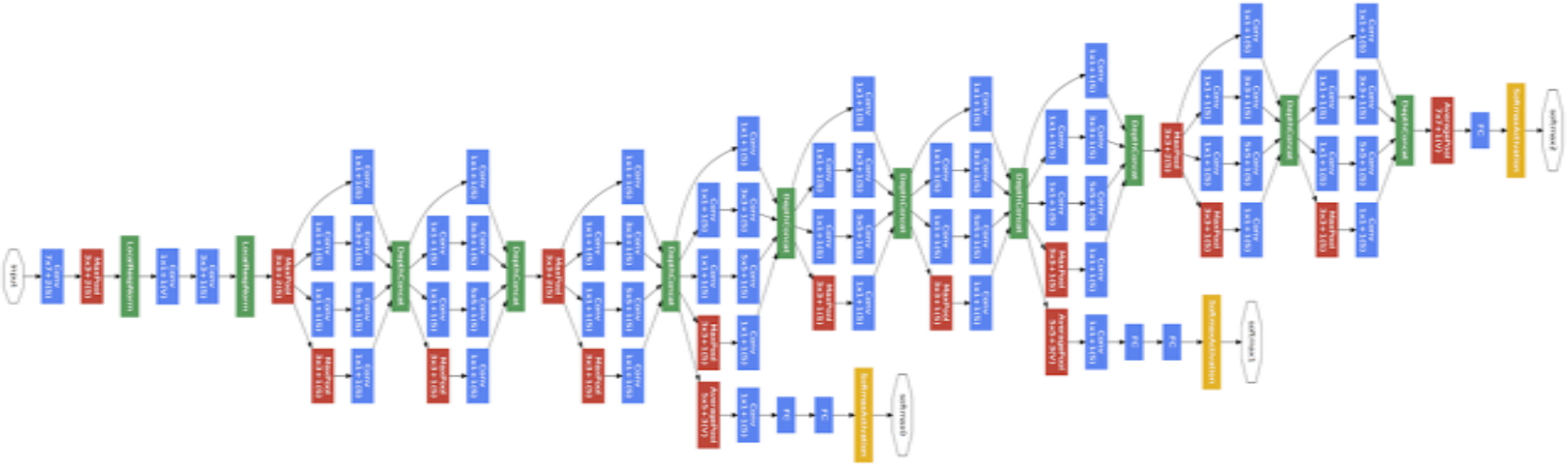


- 24 layer model that uses so-called inception module.

Convolution
Pooling
Softmax
Other

(Szegedy et al., Going Deep with Convolutions, 2014)

GoogLeNet

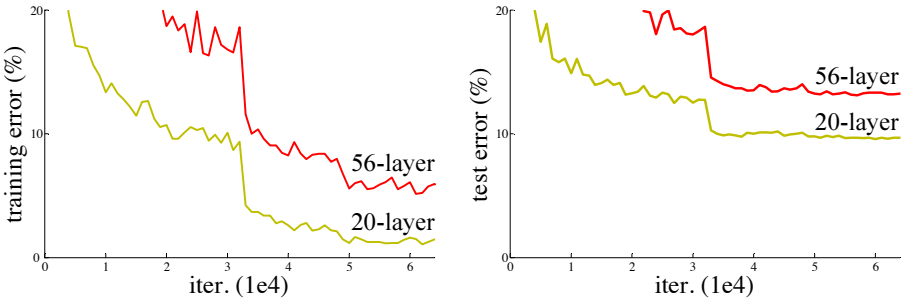


- Width of inception modules ranges from 256 filters (in early modules) to 1024 in top inception modules.
- Can remove fully connected layers on top completely
- Number of parameters is reduced to 5 million
- 6.7% top-5 validation error on Imagnet

(Szegedy et al., Going Deep with Convolutions, 2014)

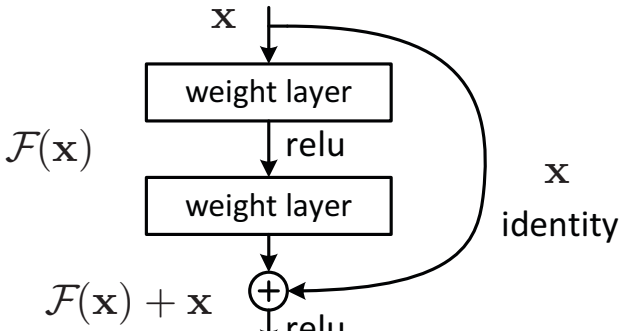
Residual Networks

Really, really deep convnets do not train well,
E.g. CIFAR10:



Key idea: introduce “pass through” into each layer

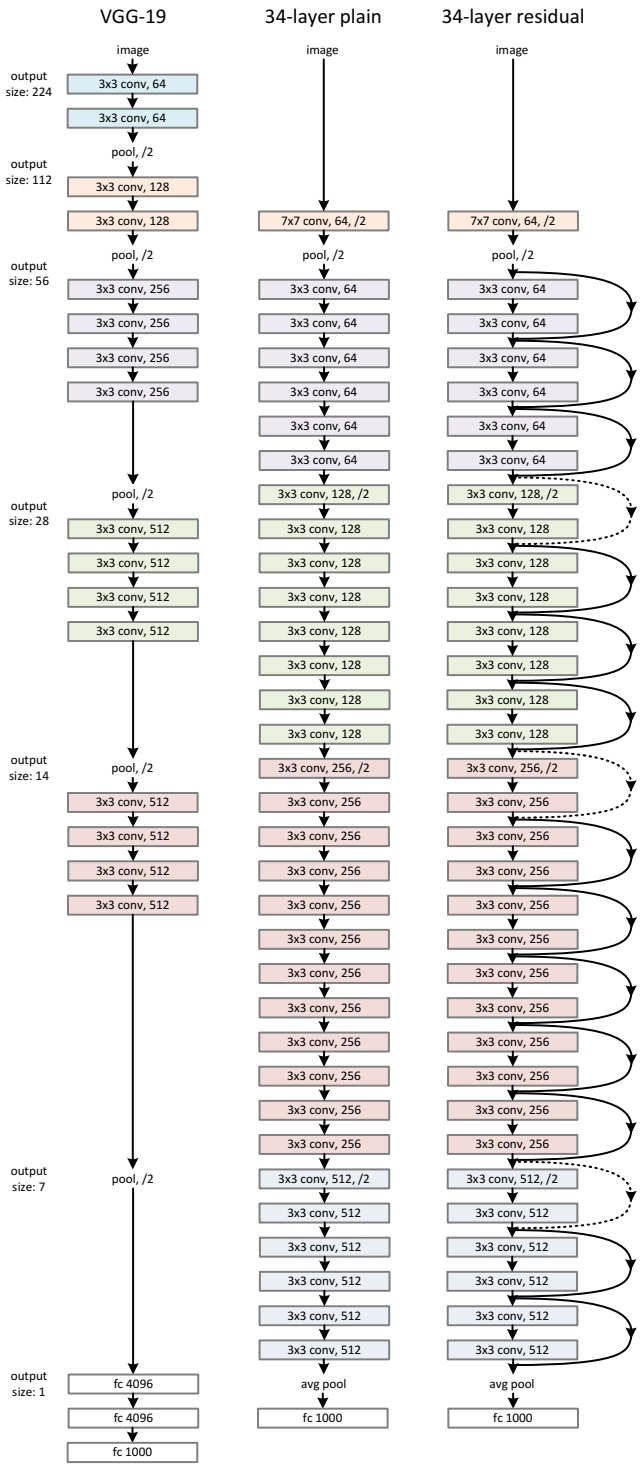
Thus only residual now needs to be learned



method	top-1 err.	top-5 err.
VGG [41] (ILSVRC’14)	-	8.43†
GoogLeNet [44] (ILSVRC’14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

With ensembling, 3.57% top-5 test error on ImageNet



(He, Zhang, Ren, Sun, CVPR 2016)

End of Part 1

Deep Learning
Unsupervised Learning

Russ Salakhutdinov

Machine Learning Department
Carnegie Mellon University
Canadian Institute for Advanced Research

Carnegie Mellon University

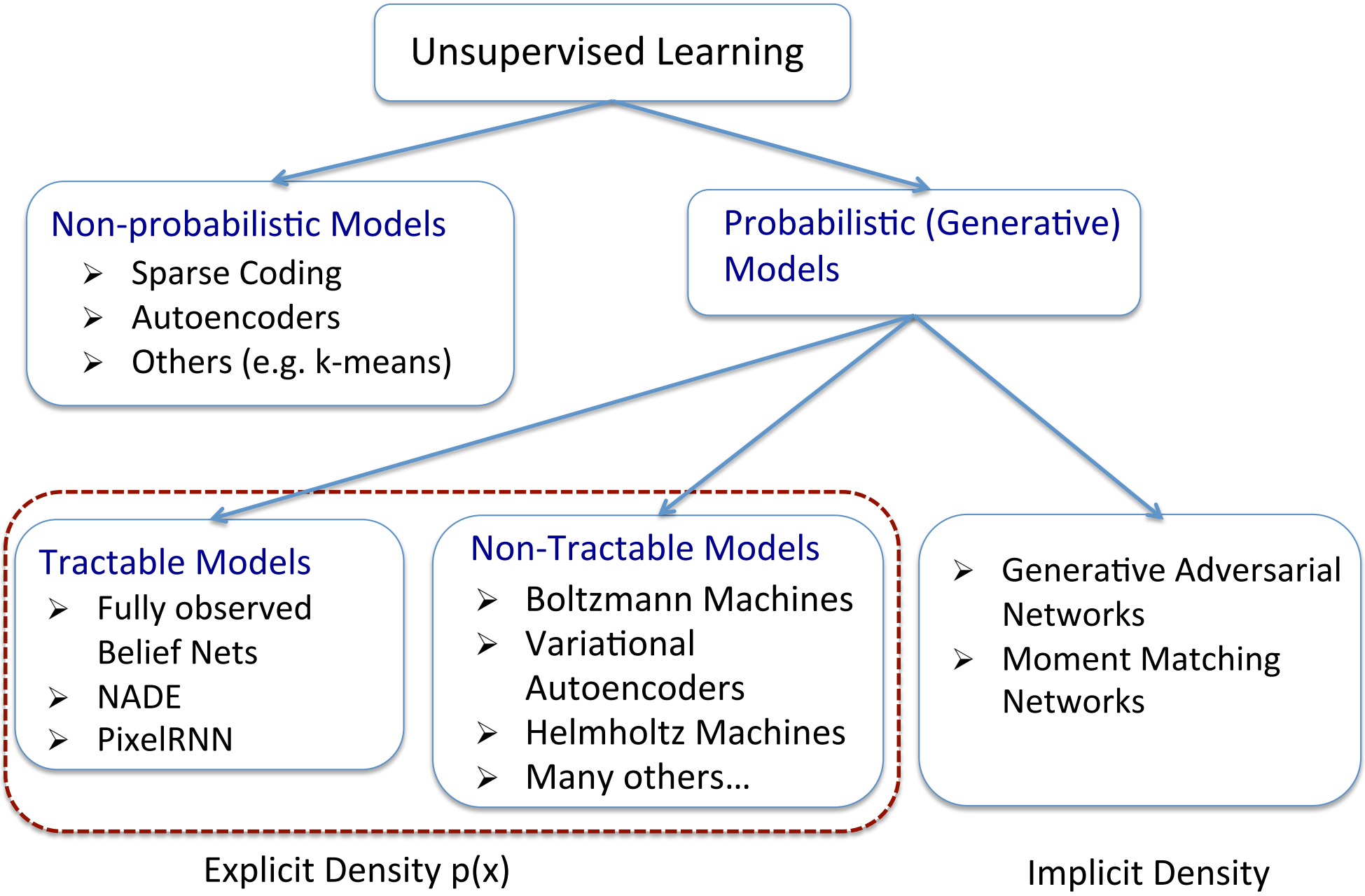
MACHINE LEARNING DEPARTMENT

Tutorial Roadmap

Part 1: Supervised (Discriminative) Learning: Deep Networks

Part 2: Unsupervised Learning: Deep Generative Models

Part 3: Open Research Questions


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

Tutorial Roadmap

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

Tutorial Roadmap

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

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:

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

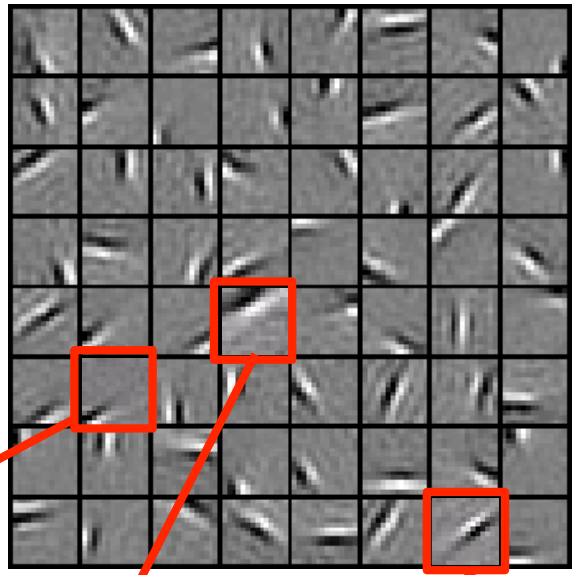
Sparse: mostly zeros

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

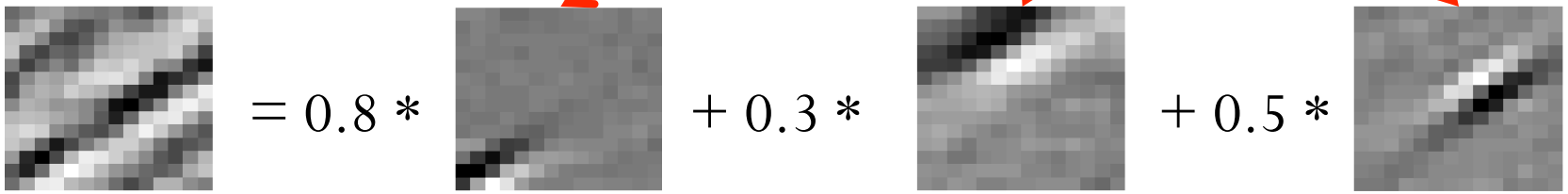
Sparse Coding

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

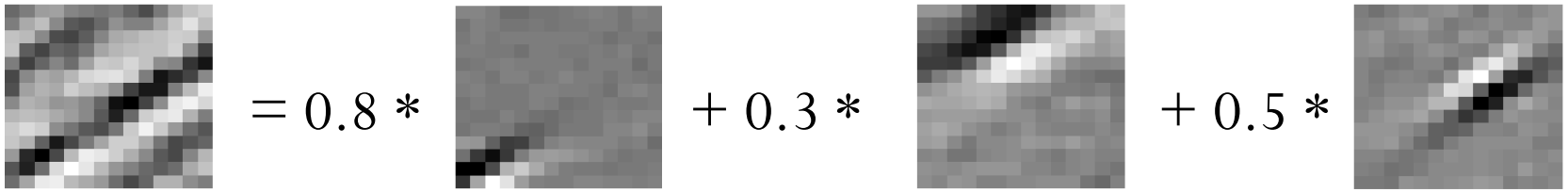
- Alternating Optimization:

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*.

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

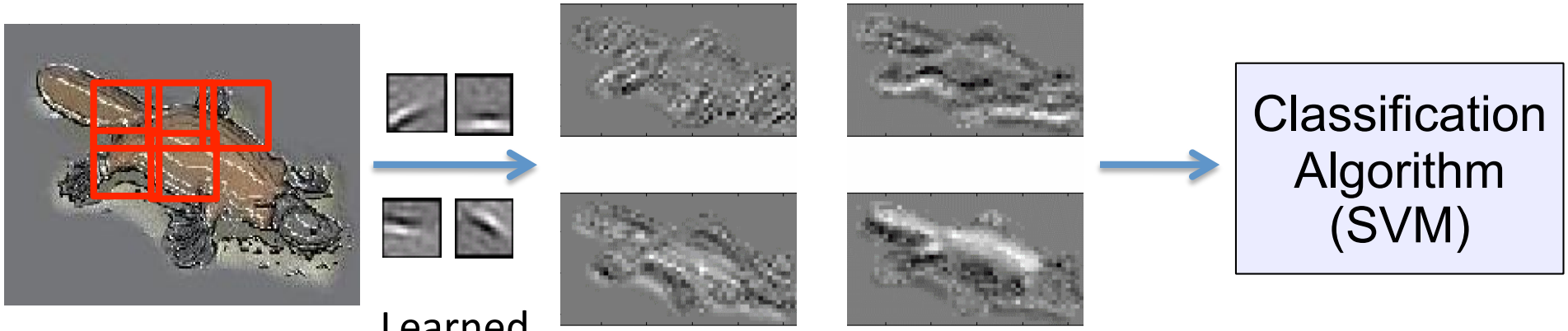


x^* $= 0.8 *$ ϕ_{36} $+ 0.3 *$ ϕ_{42} $+ 0.5 *$ ϕ_{65}

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

Image Classification

Evaluated on Caltech101 object category dataset.

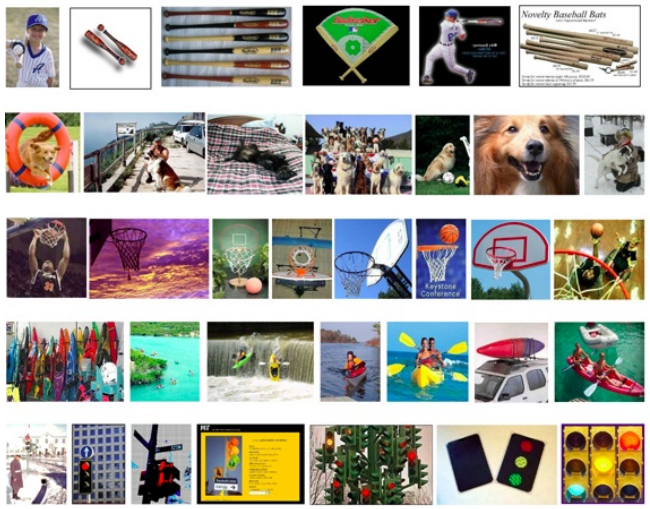


Input Image

Learned bases

Features (coefficients)

9K images, 101 classes



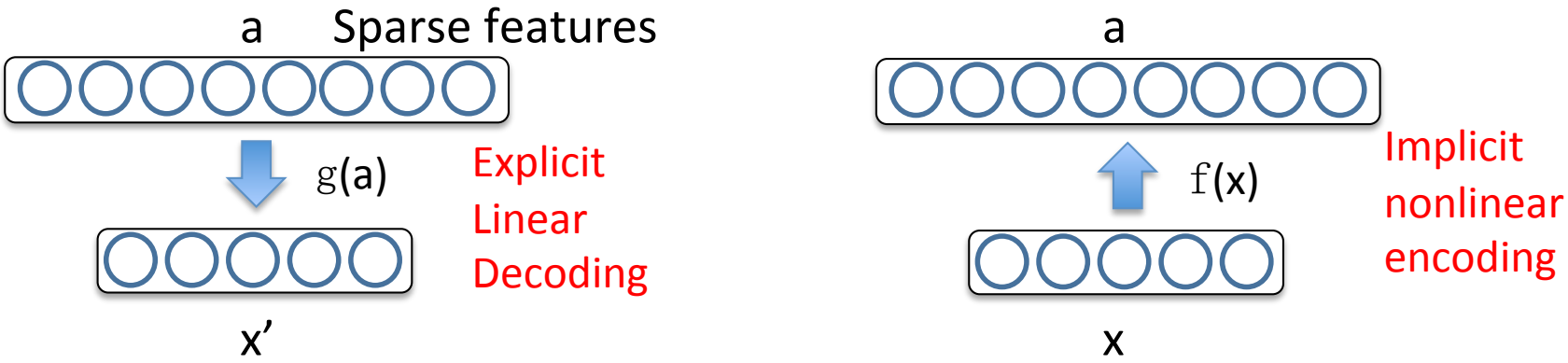
Algorithm	Accuracy
Baseline (Fei-Fei et al., 2004)	16%
PCA	37%
Sparse Coding	**47%**

Slide Credit: Honglak Lee

Lee, Battle, Raina, Ng, 2006

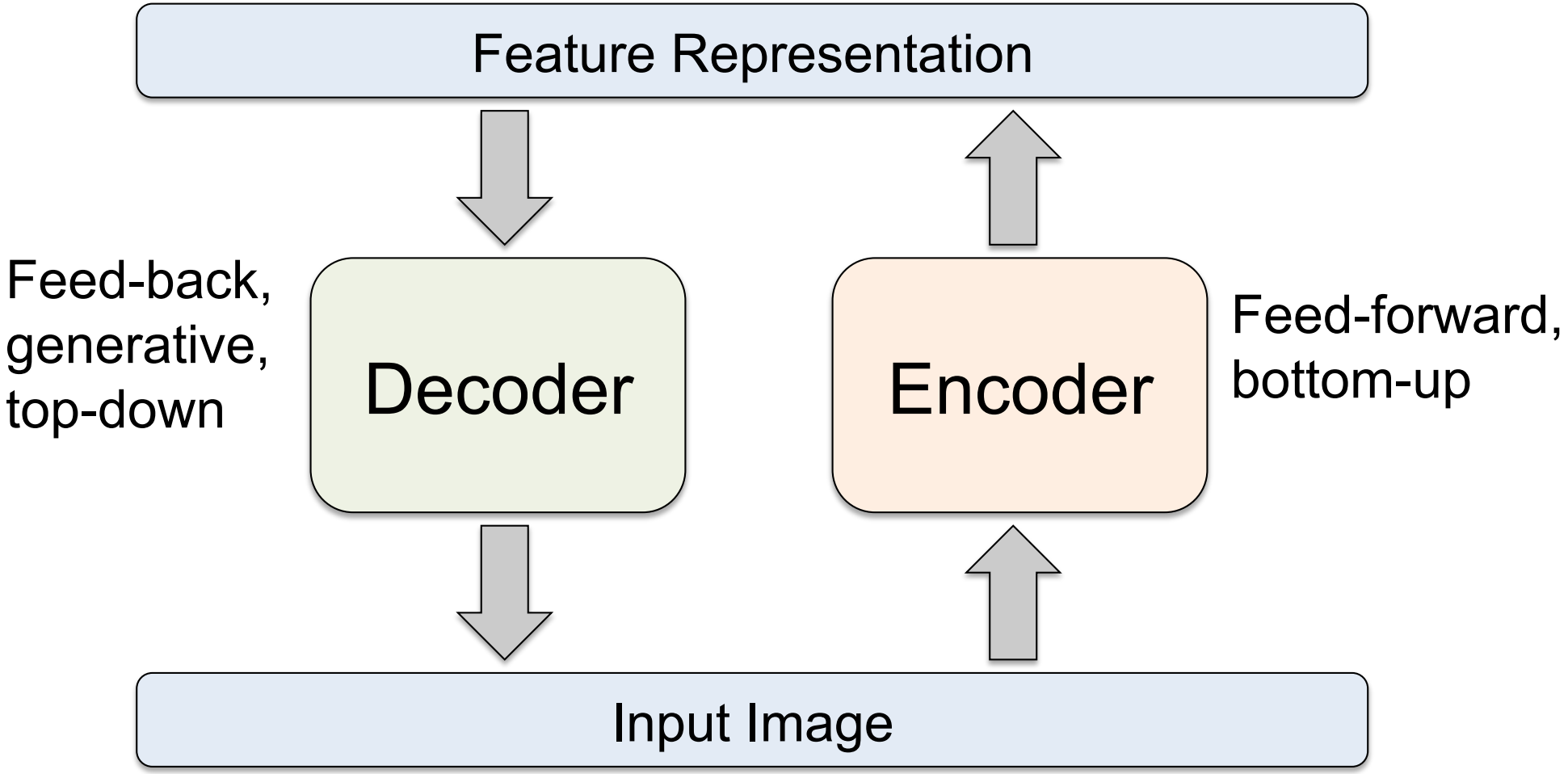
Interpreting Sparse Coding

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



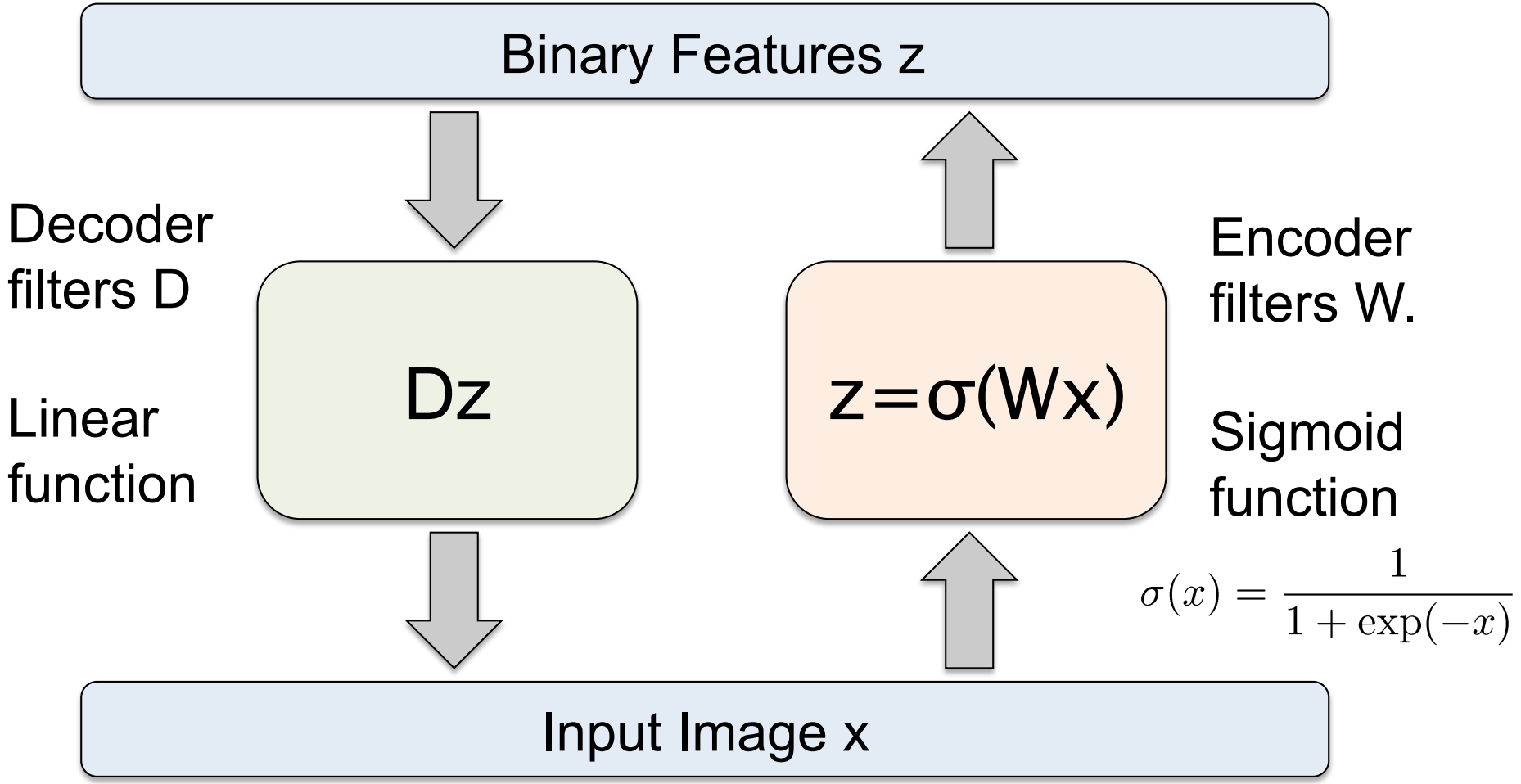
- Sparse, over-complete representation **a.**
- Encoding **a** = f(**x**) is implicit and nonlinear function of **x**.
- Reconstruction (or decoding) **x'** = g(**a**) is linear and explicit.

Autoencoder

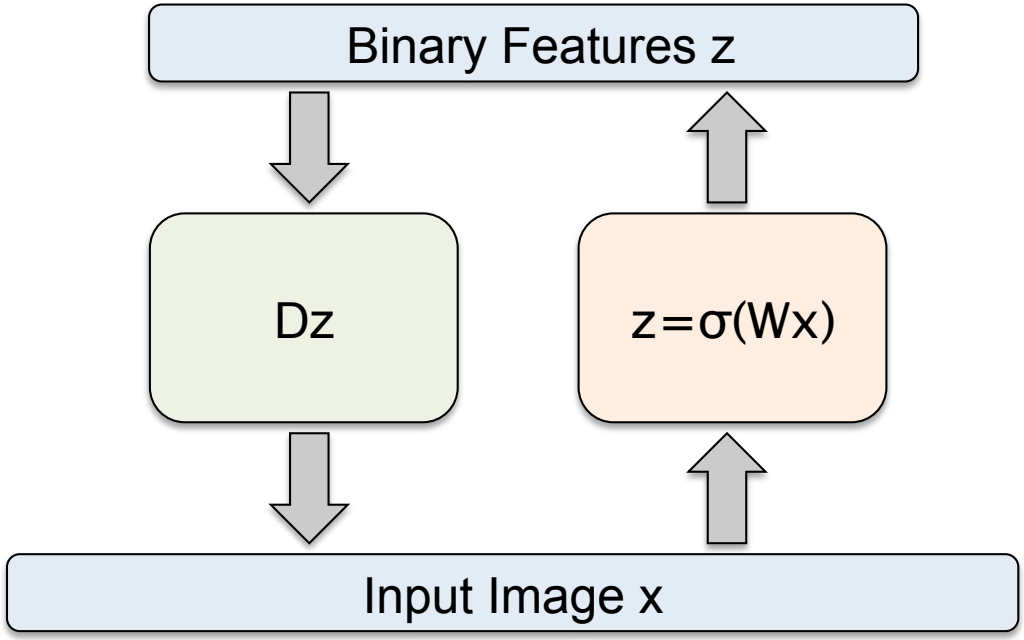


- Details of what goes insider the encoder and decoder matter!
- Need constraints to avoid learning an identity.

Autoencoder



Autoencoder



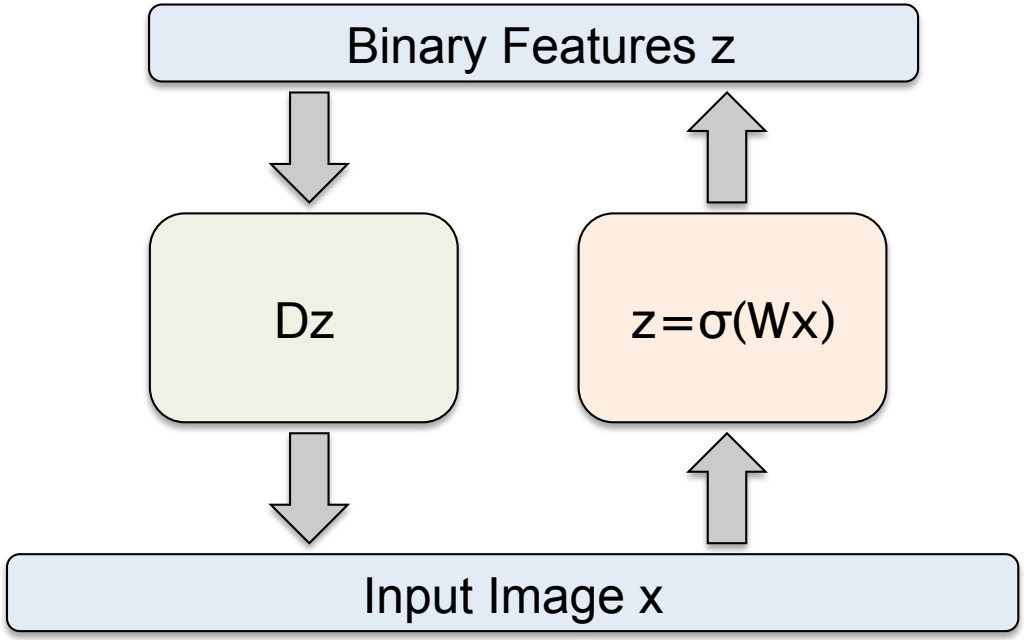
- An autoencoder with D inputs, D outputs, and K hidden units, with K<D.

- 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

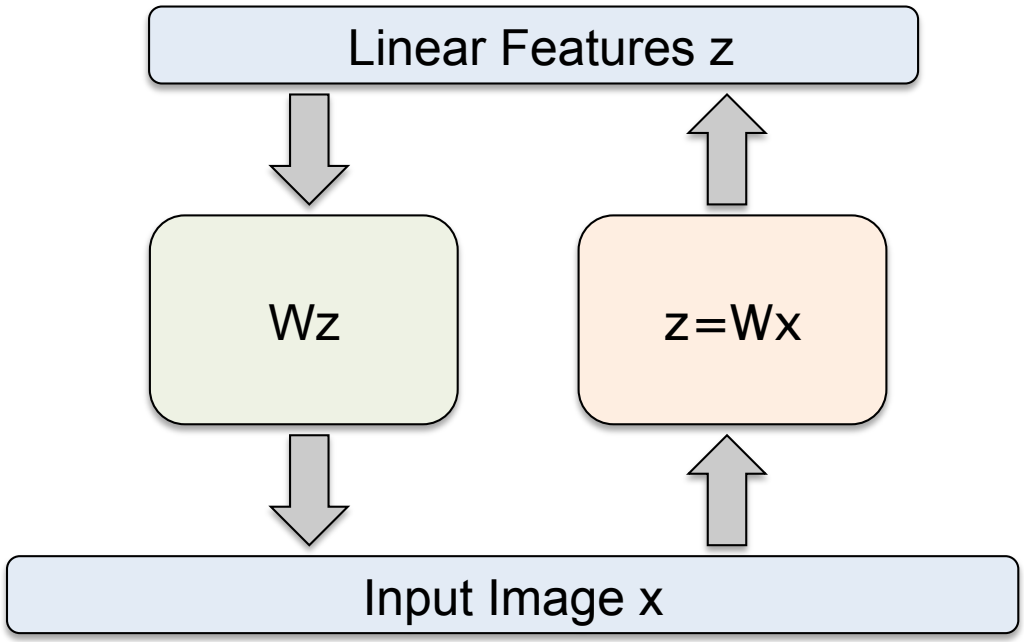


• An autoencoder with D inputs, D outputs, and K hidden units, with K<D.

• 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

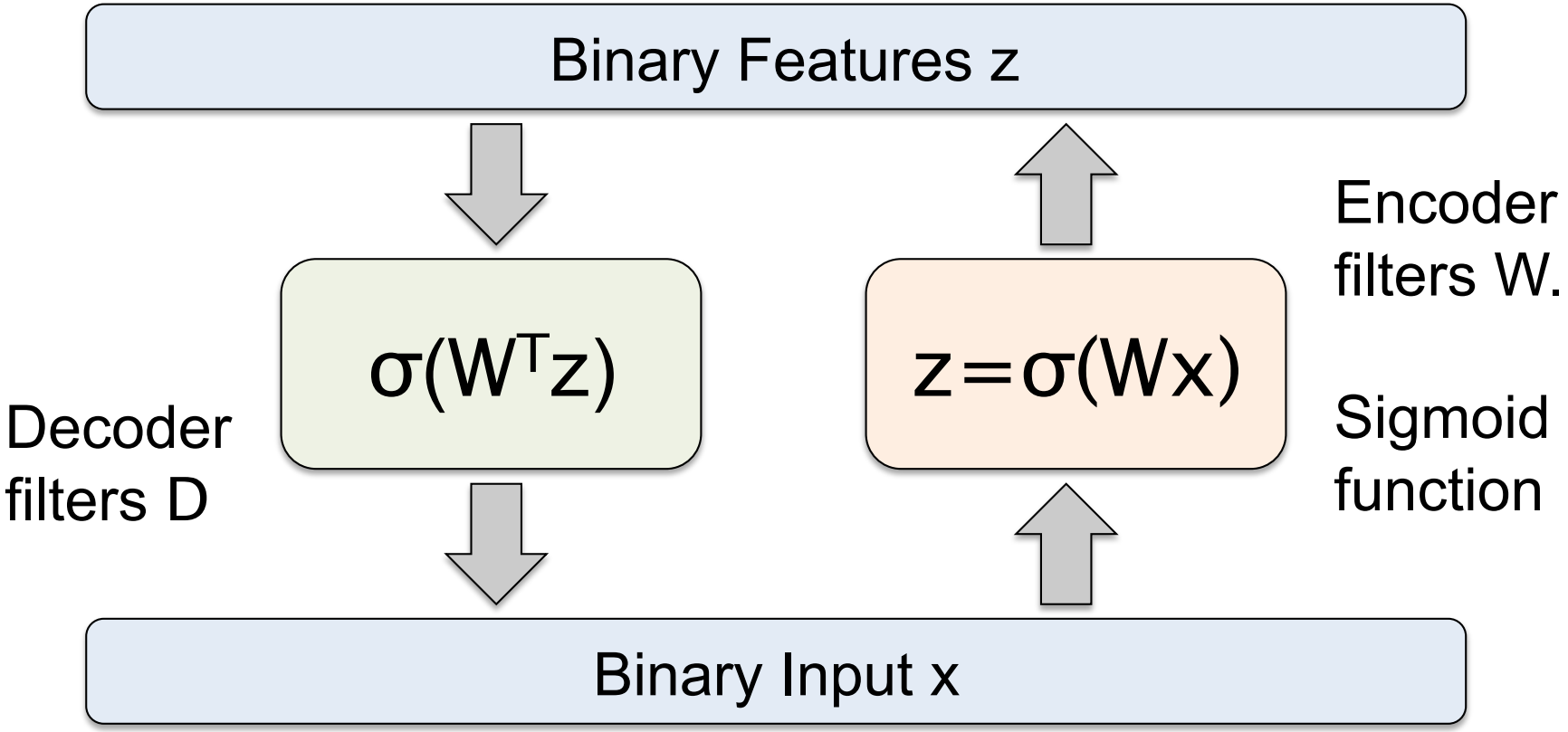


- If the hidden and output layers are linear, it will learn hidden units that are a linear function of the data and minimize the squared error.

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

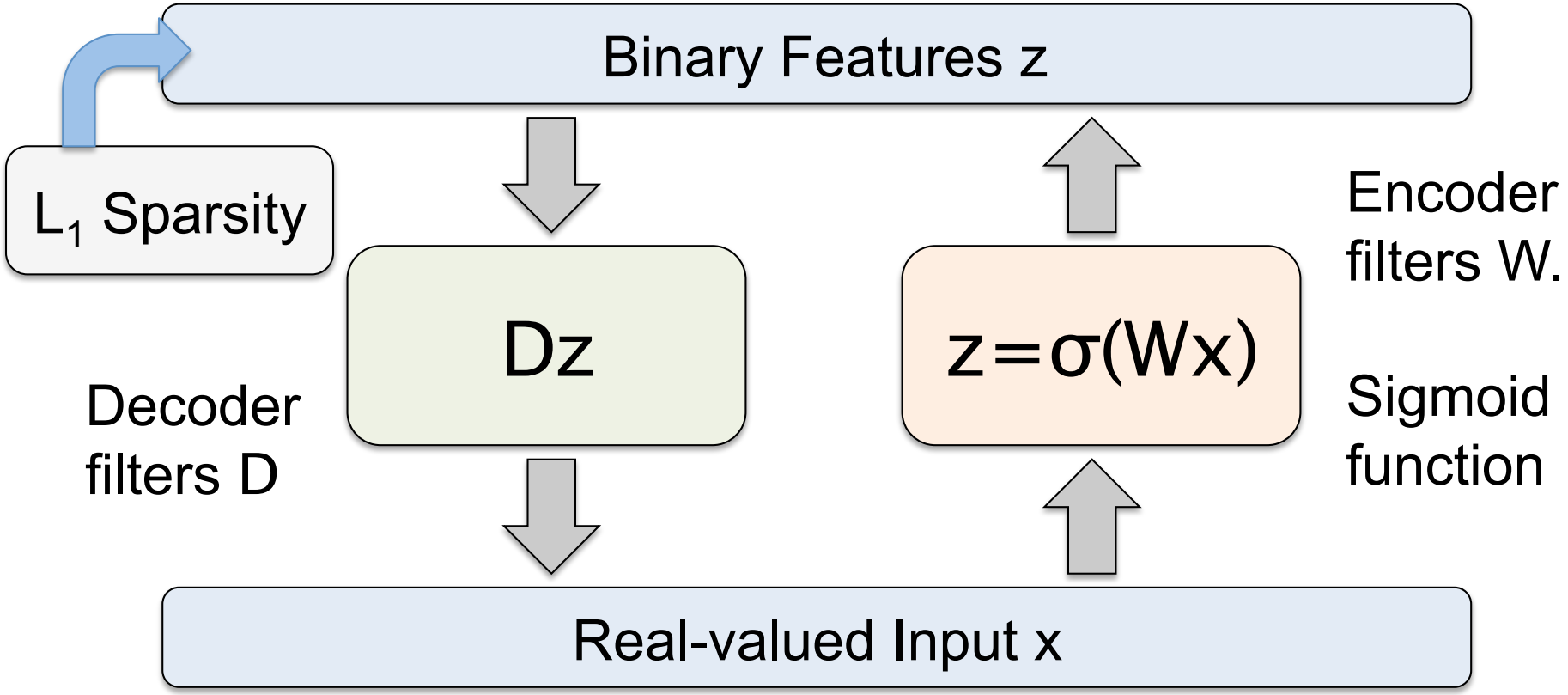
- With nonlinear hidden units, we have a nonlinear generalization of PCA.

Another Autoencoder Model



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

Predictive Sparse Decomposition

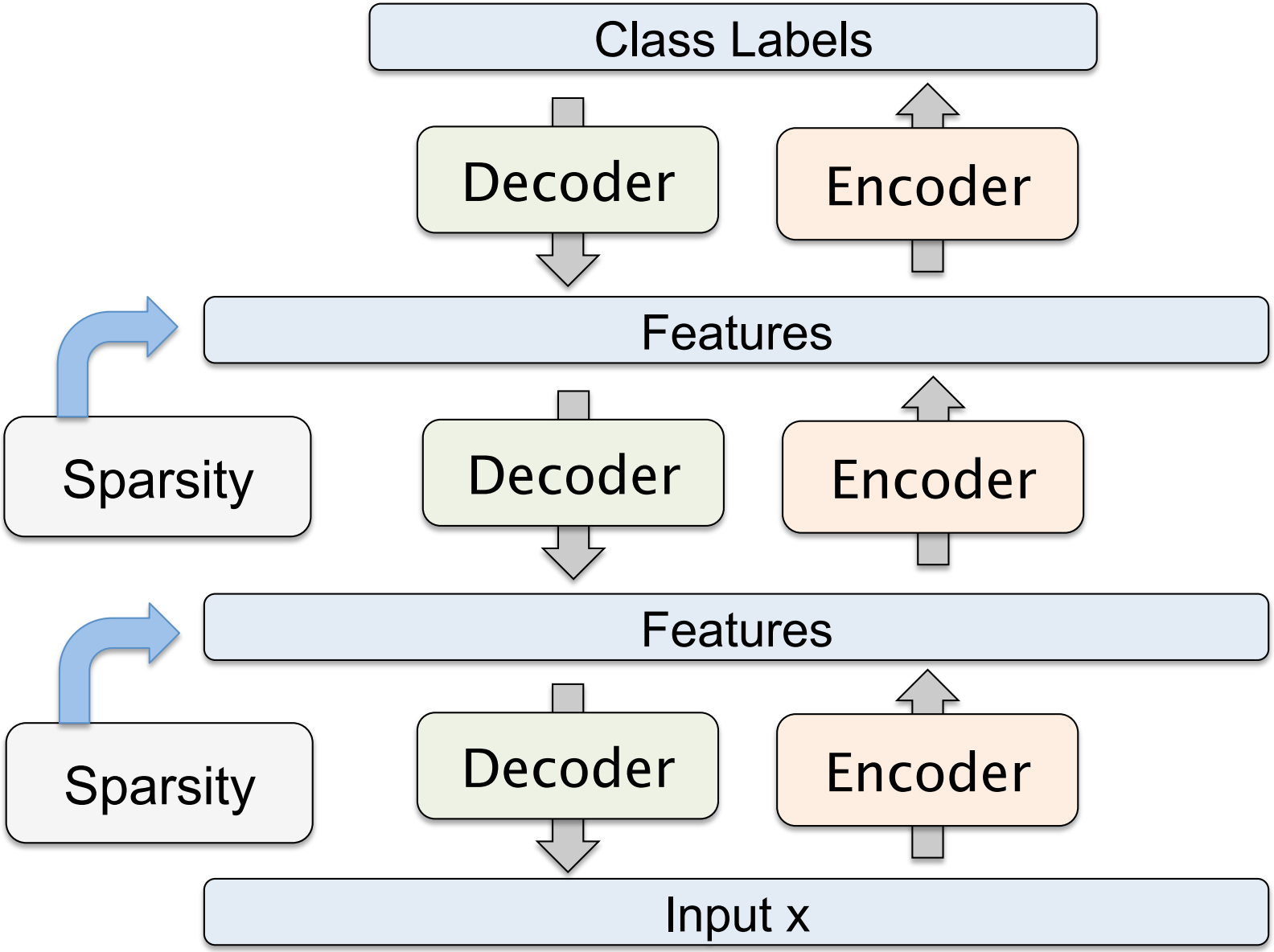


At training time

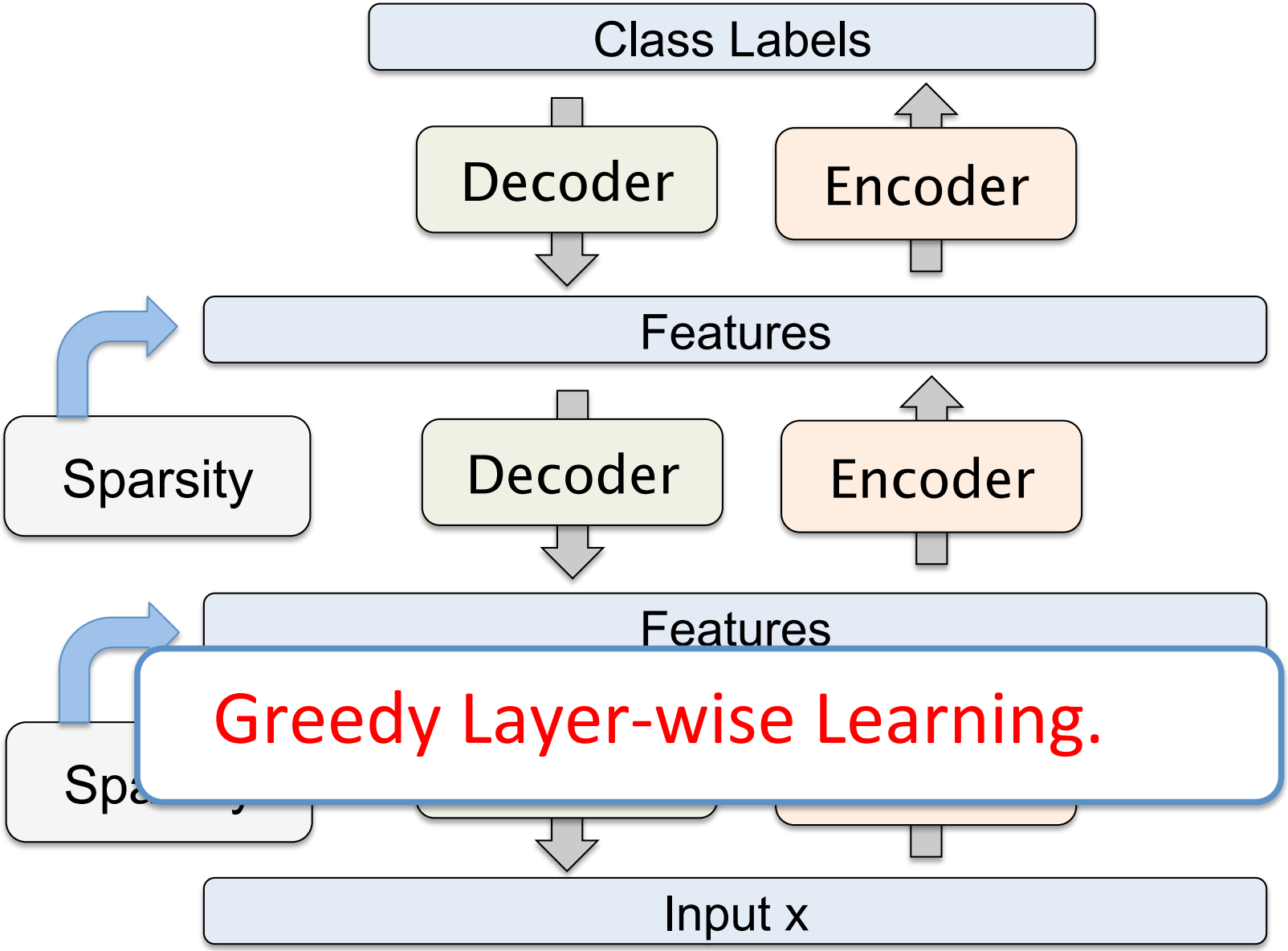
$$\min_{D,W,\mathbf{z}} \underbrace{||D\mathbf{z}-\mathbf{x}||_2^2+\lambda|\mathbf{z}|_1}_{\text{Decoder}} + \underbrace{||\sigma(W\mathbf{x})-\mathbf{z}||_2^2}_{\text{Encoder}}$$

Kavukcuoglu, Ranzato, Fergus, LeCun, 2009

Stacked Autoencoders

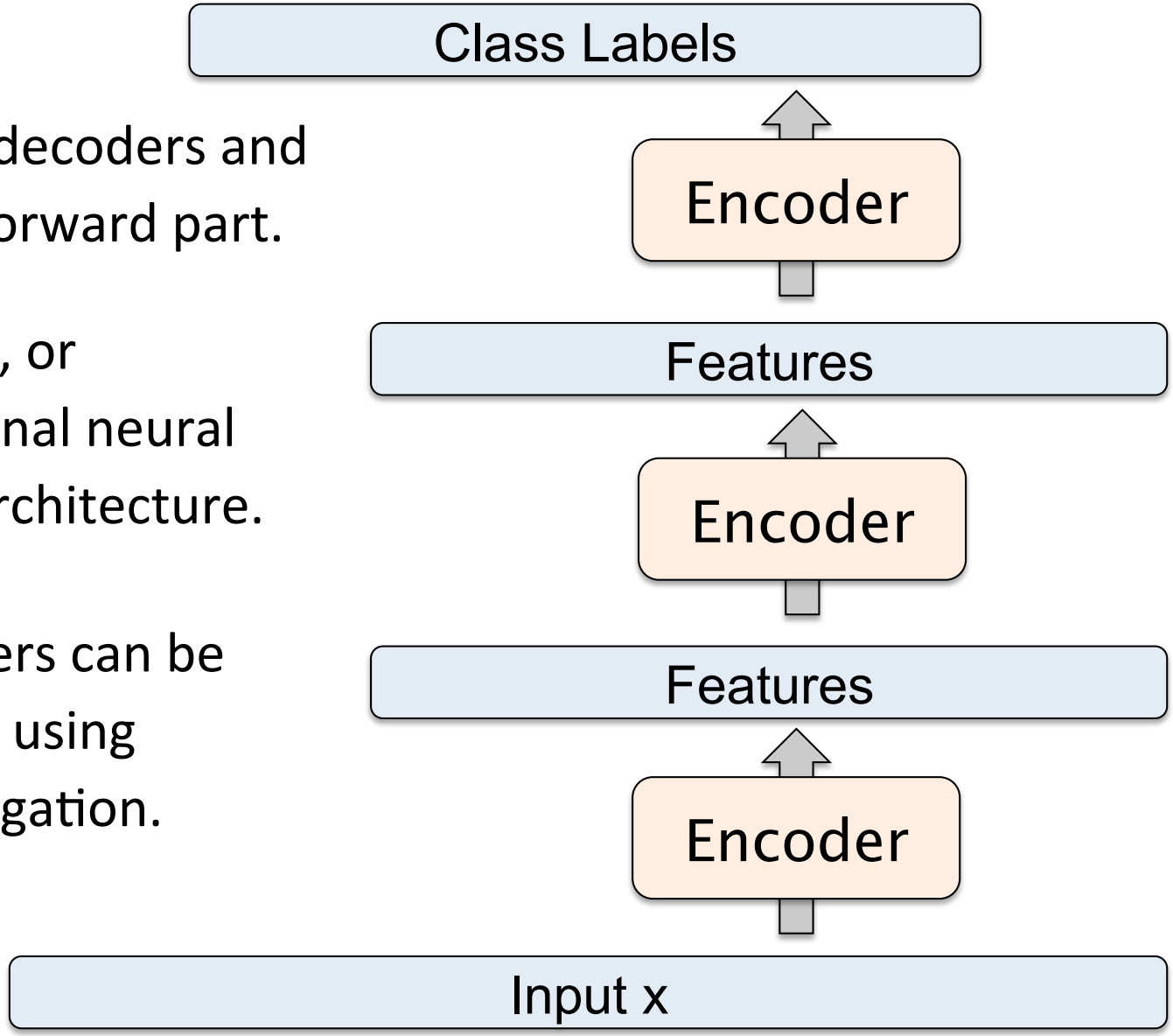


Stacked Autoencoders

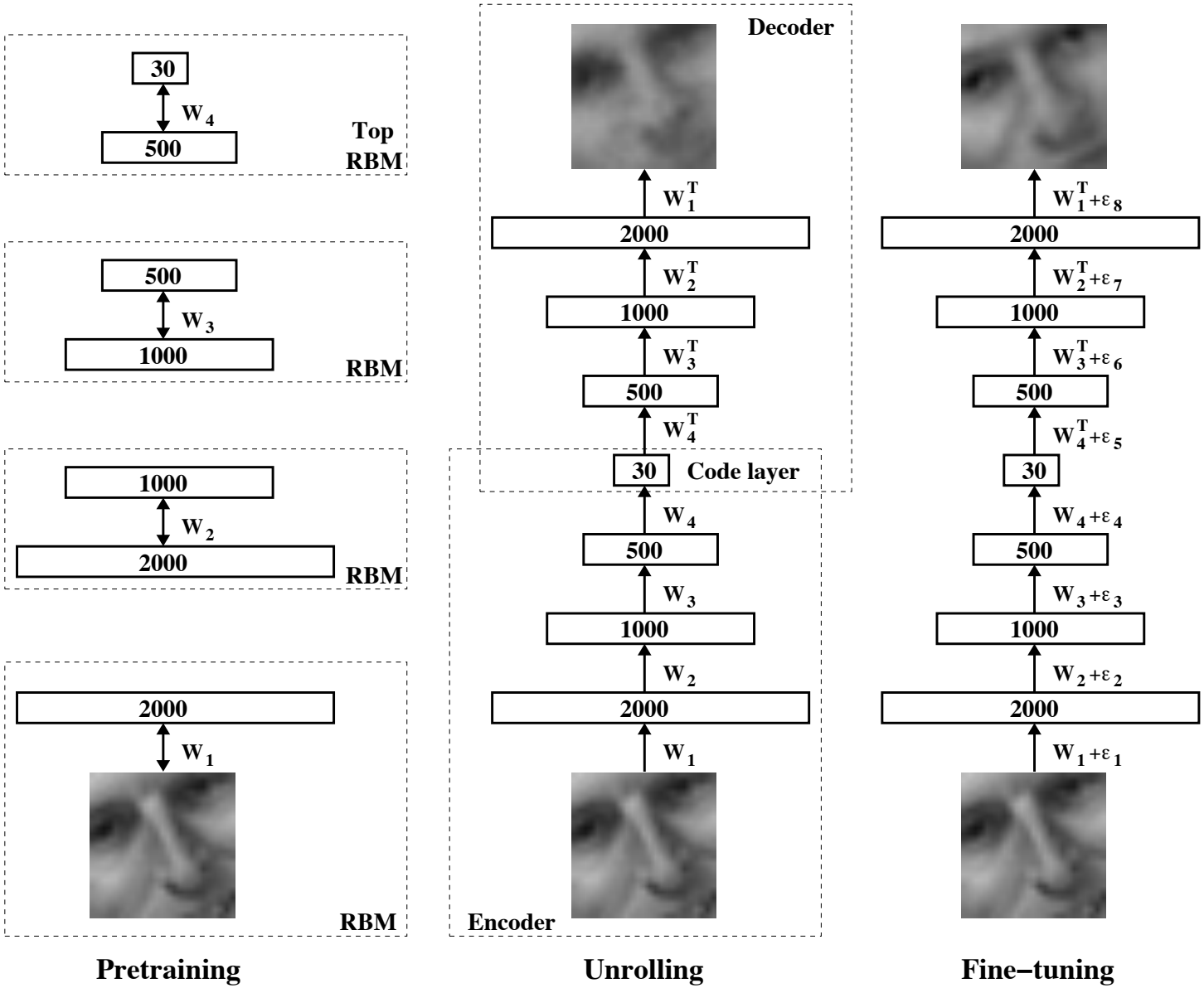


Stacked Autoencoders

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

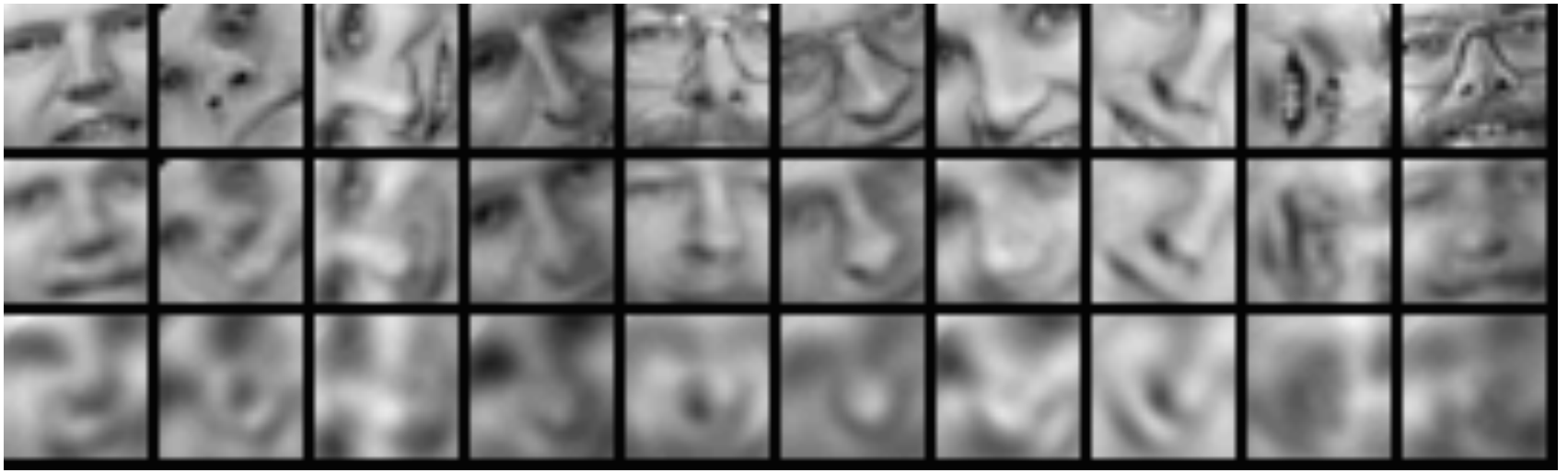


Deep Autoencoders



Deep Autoencoders

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

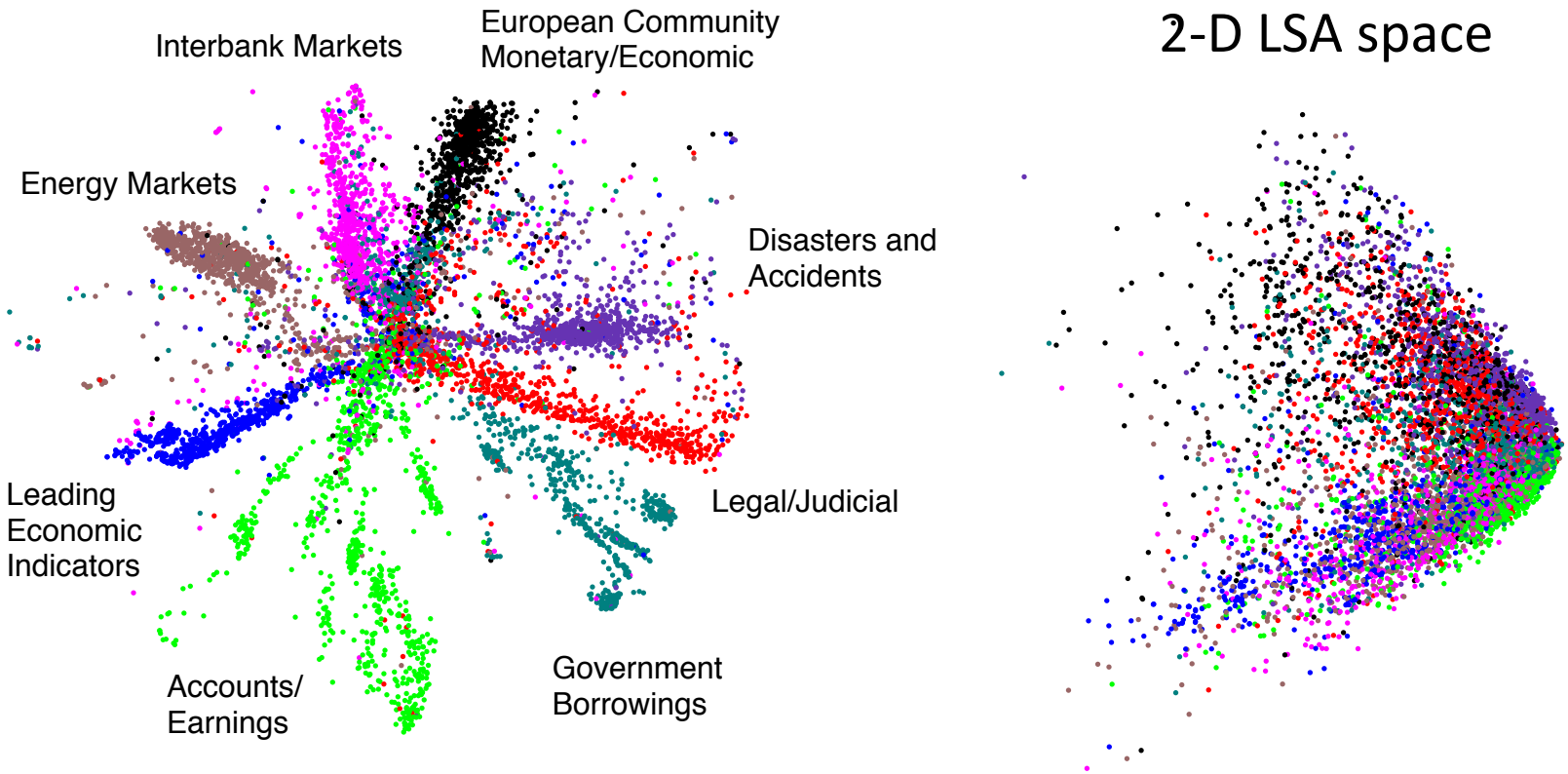


• **Top**: Random samples from the test dataset.

• **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.

(Hinton and Salakhutdinov, Science 2006)

Tutorial Roadmap

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

Fully Observed Models

- Explicitly model conditional probabilities:

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

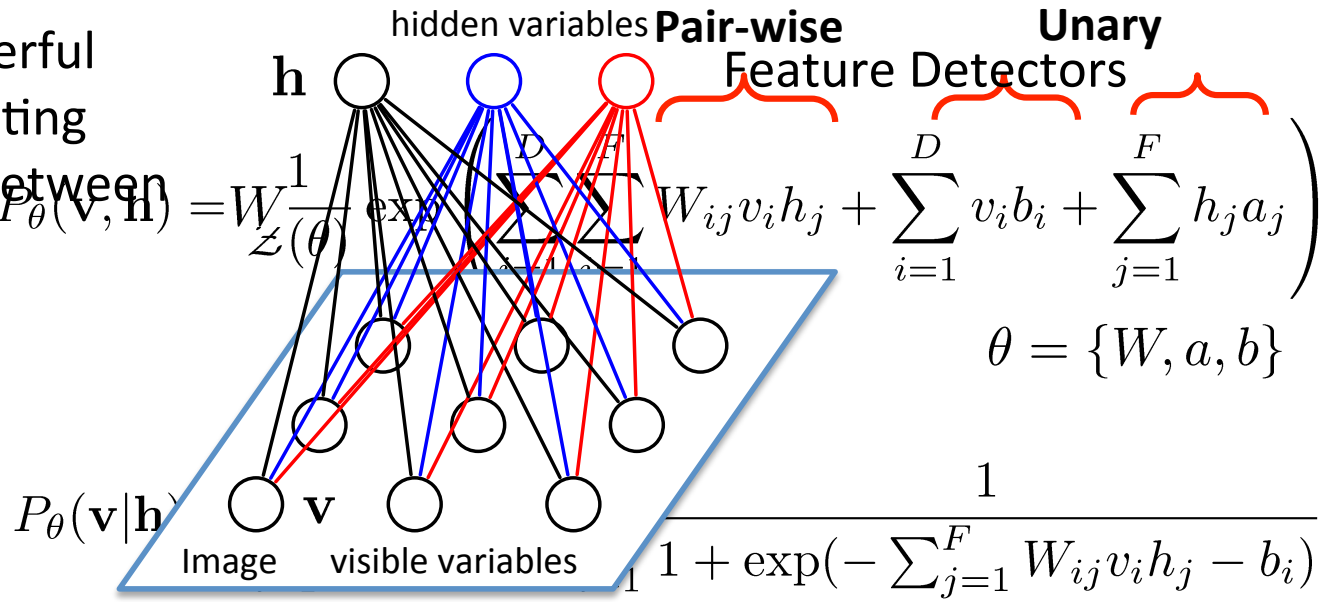
Each conditional can be a complicated neural network

- 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

Graphical Models: Powerful framework for representing dependency structure between random variables.



RBM is a Markov Random Field with:

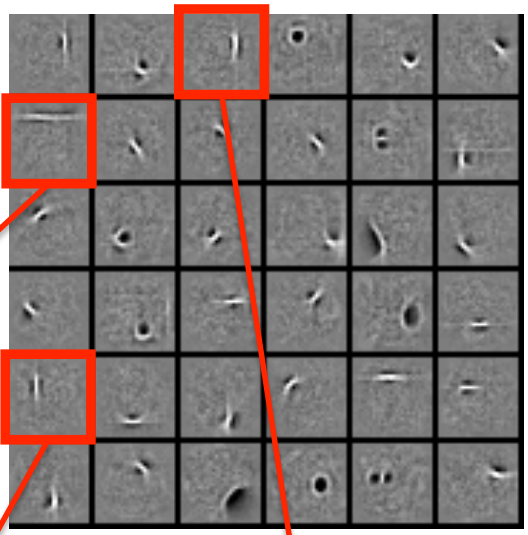
- Stochastic binary visible variables $\mathbf{v} \in \{0,1\}^D$.
- Stochastic binary hidden variables $\mathbf{h} \in \{0,1\}^F$.
- Bipartite connections.

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

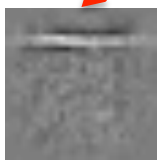
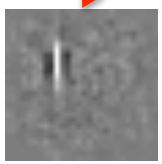
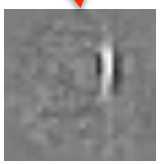
Learning Features

Observed Data
Subset of 25,000 characters

Learned W: "edges"
Subset of 1000 features



New Image:

$= \sigma\left(0.99 \times \right.$  $+ \quad 0.97 \times$  $+ \quad 0.82 \times$  $\left. \cdots \right)$

$p(h_7 = 1|v)$

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

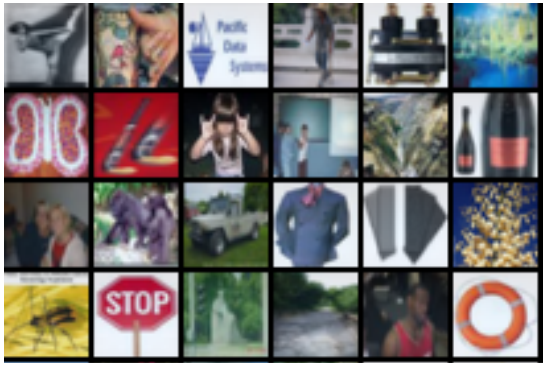
Sparse representations

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

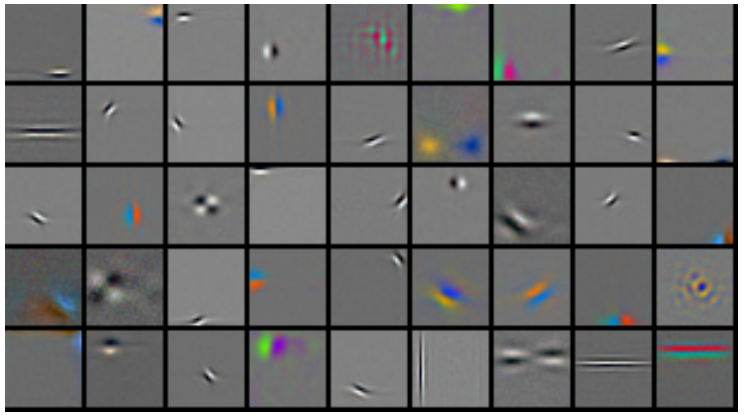
Logistic Function: Suitable for modeling binary images

RBMs for Real-valued & Count Data

4 million **unlabelled** images



Learned features (out of 10,000)



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

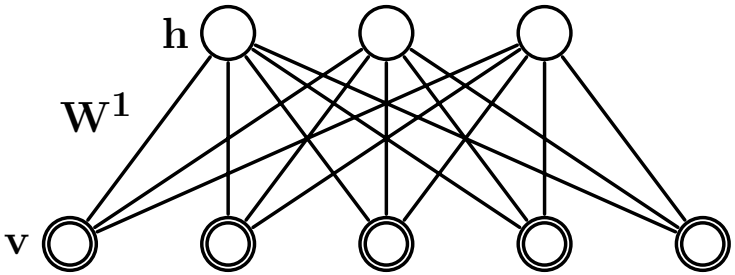
Learned features: ``topics''

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

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''

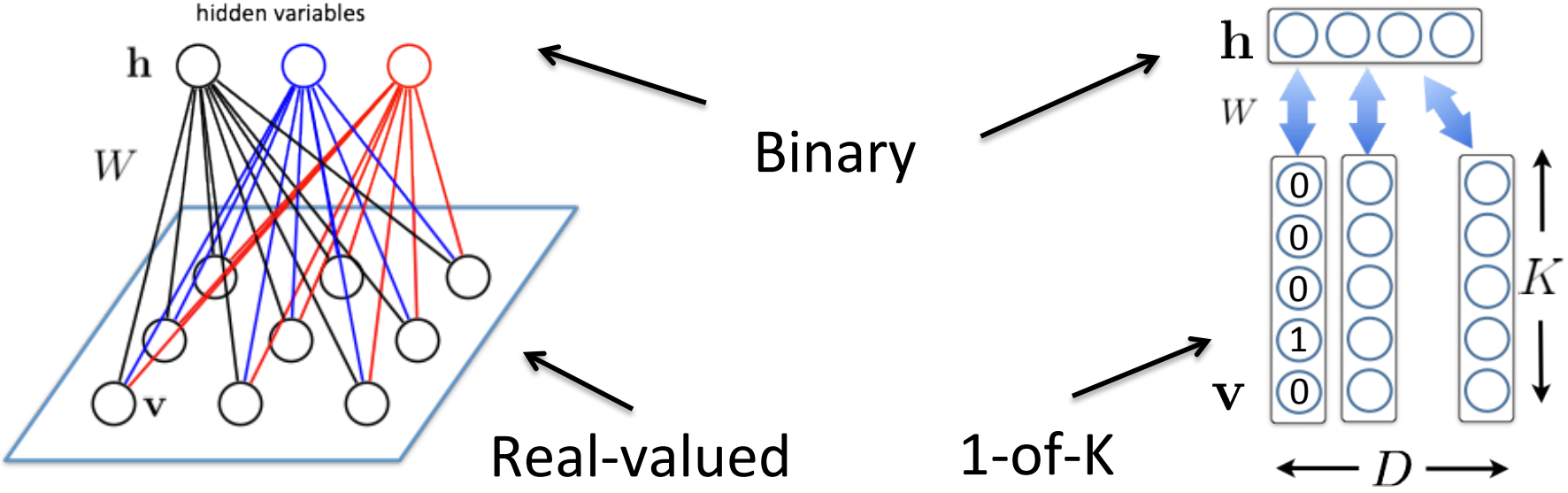
Fahrenheit 9/11	Independence Day
Bowling for Columbine	The Day After Tomorrow
The People vs. Larry Flynt	Con Air
Canadian Bacon	Men in Black II
La Dolce Vita	Men in Black
Friday the 13th	Scary Movie
The Texas Chainsaw Massacre	Naked Gun
Children of the Corn	Hot Shots!
Child's Play	American Pie
The Return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

State-of-the-art performance on the Netflix dataset.

(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:

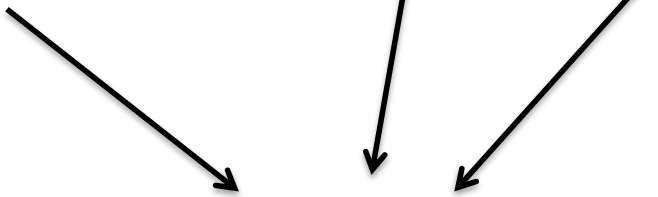
$$P_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\Big(\sum_{ij} W_{ij} v_i h_j + \sum_i b_i v_i + \sum_j a_j h_j\Big)$$

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:

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

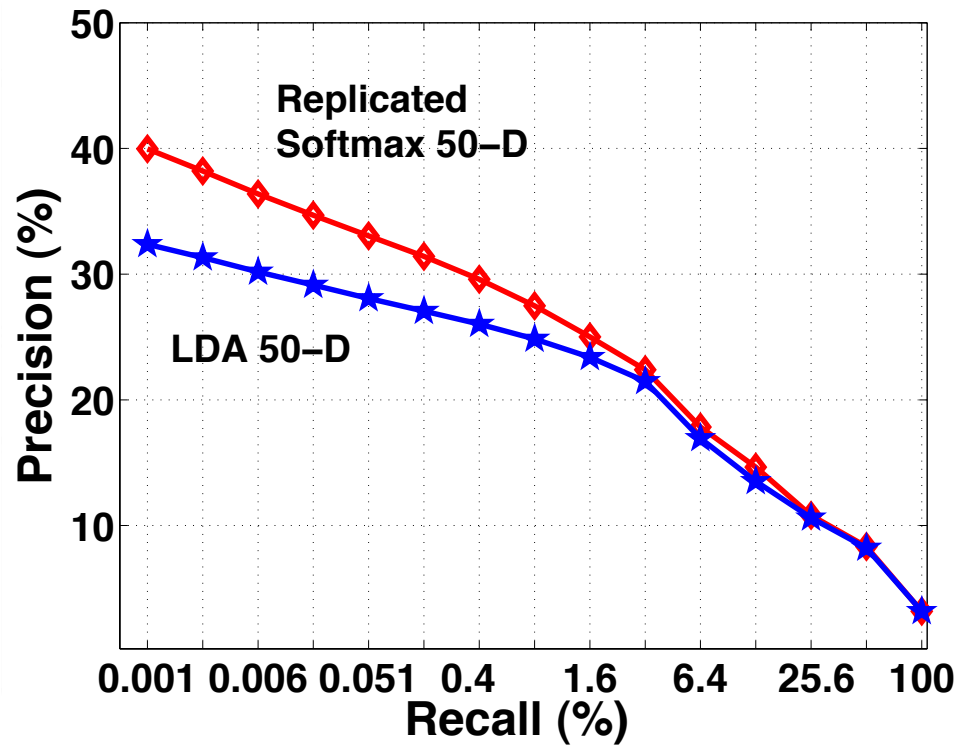
Marginalizing o

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

duct of Experts

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

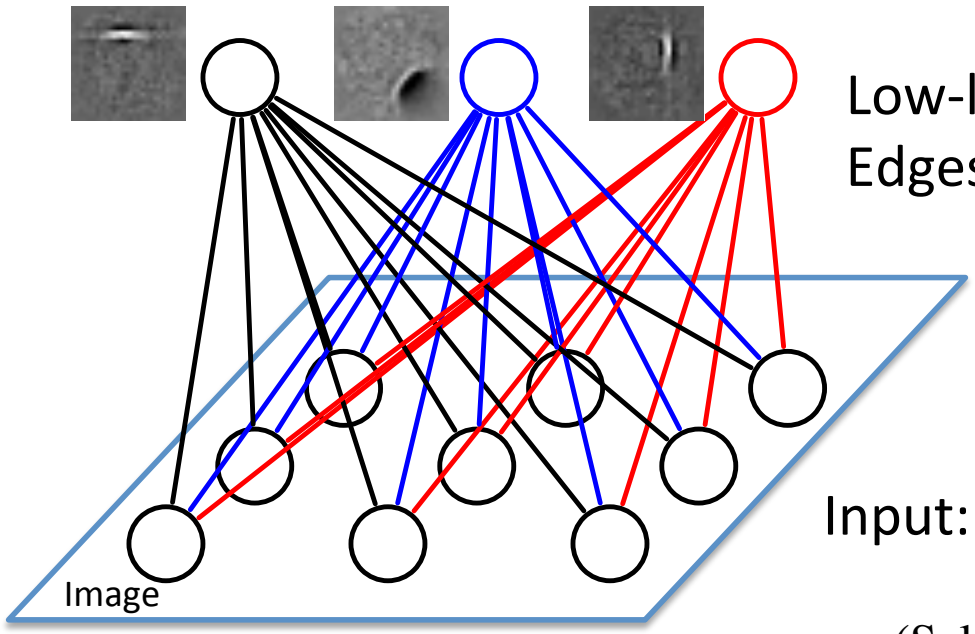
government	clint
authority	hou
power	pres
empire	bill
federation	cong



, "corruption"
bine to give very
word "Silvio

Berlusconi".

Deep Boltzmann Machines



Low-level features:
Edges

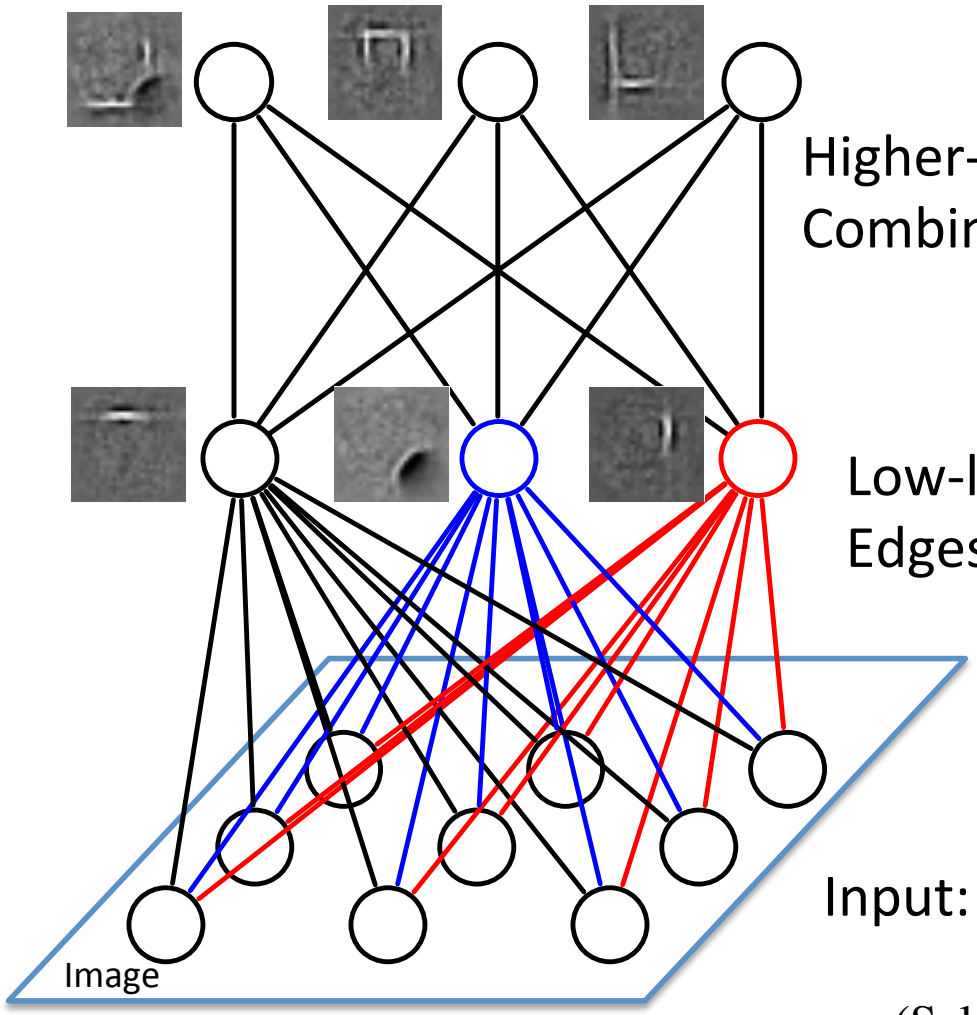
Built from **unlabeled** inputs.

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

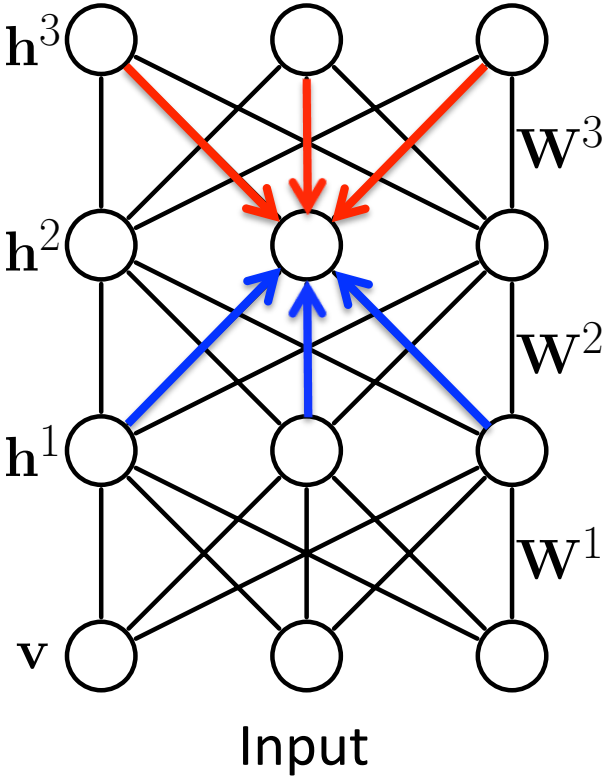
Built from **unlabeled** inputs.

(Salakhutdinov 2008, Salakhutdinov & Hinton 2009)

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:

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

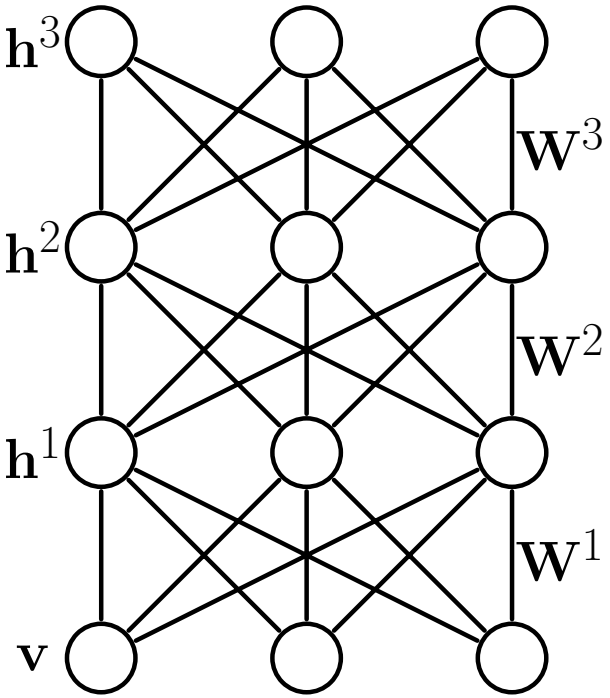
Top-down

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!

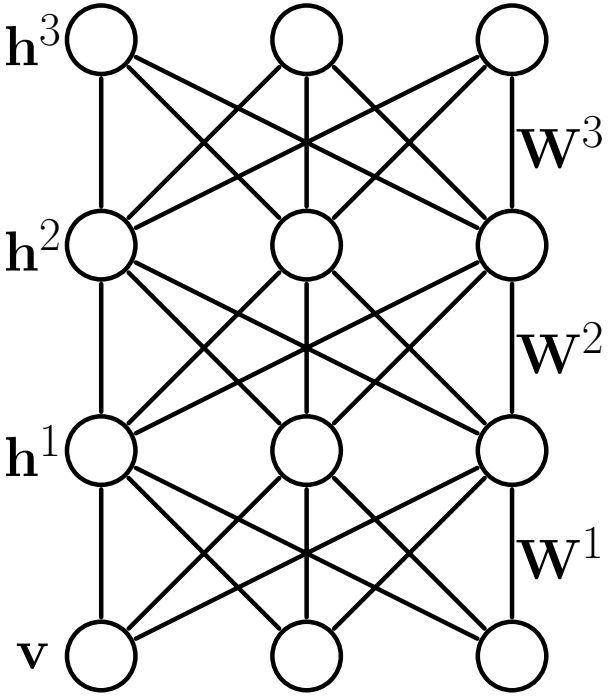
$$P_{data}(\mathbf{v}, \mathbf{h^1}) = P_\theta(\mathbf{h^1}|\mathbf{v}) P_{data}(\mathbf{v})$$

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

Not factorial any more!

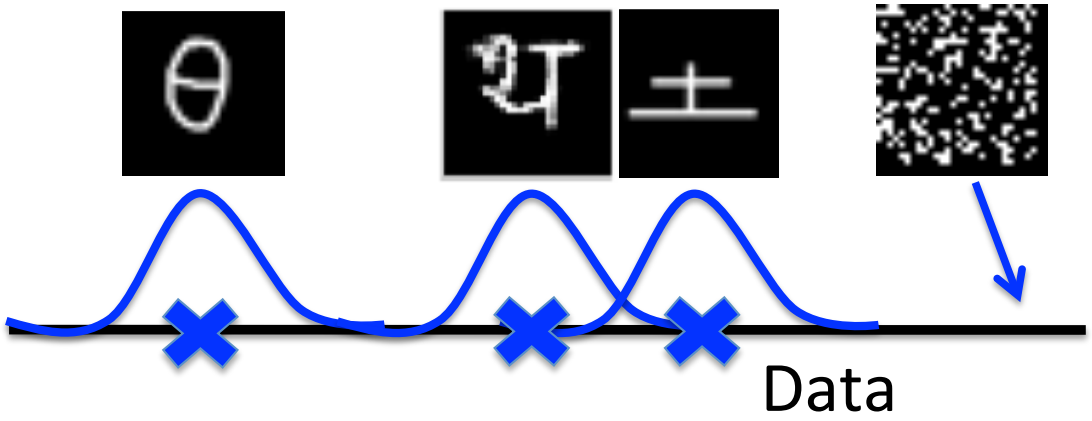
Approximate Learning

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



(Approximate) Maximum Likelihood:

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



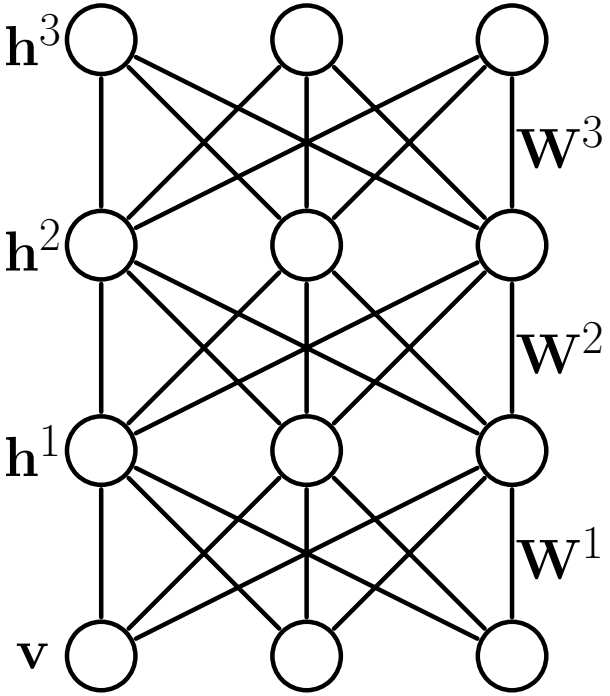
$$P_{data}(\mathbf{v}, \mathbf{h}^{\mathbf{1}}) = P_\theta(\mathbf{h}^{\mathbf{1}}|\mathbf{v}) P_{data}(\mathbf{v})$$

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

Not factorial any more!

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)

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

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

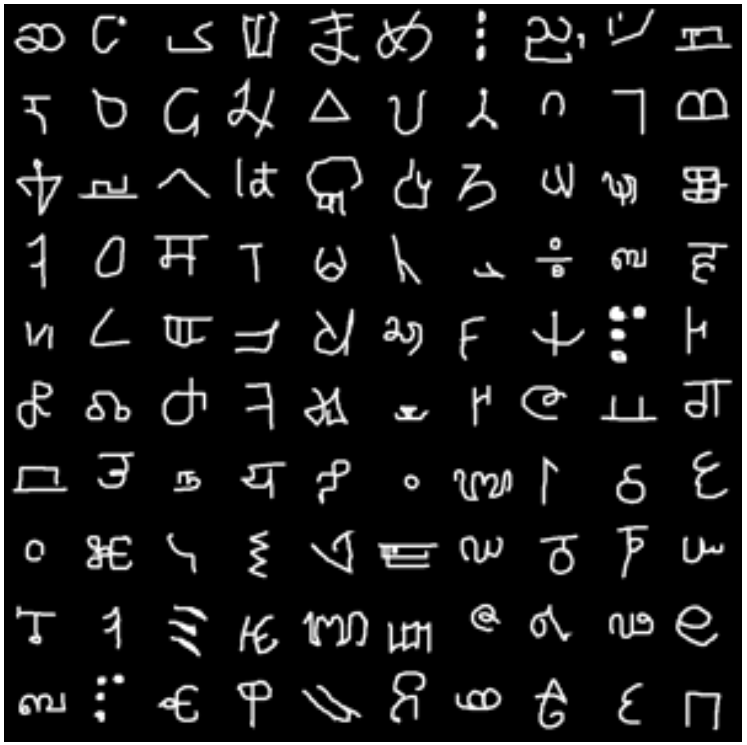
Not factorial any more!

Good Generative Model?

Handwritten Characters

Good Generative Model?

Handwritten Characters



Good Generative Model?

Handwritten Characters

Simulated

Real Data

Good Generative Model?

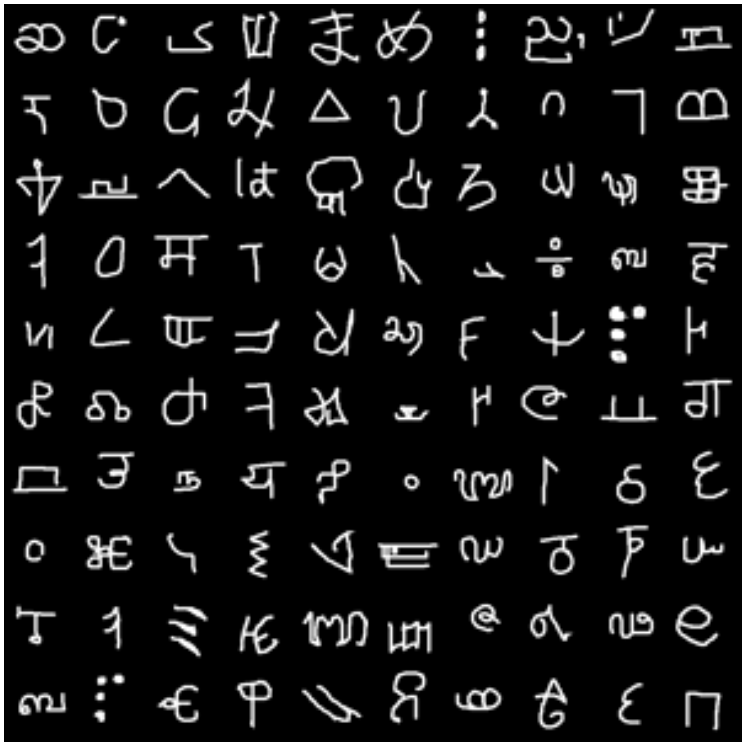
Handwritten Characters

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%**

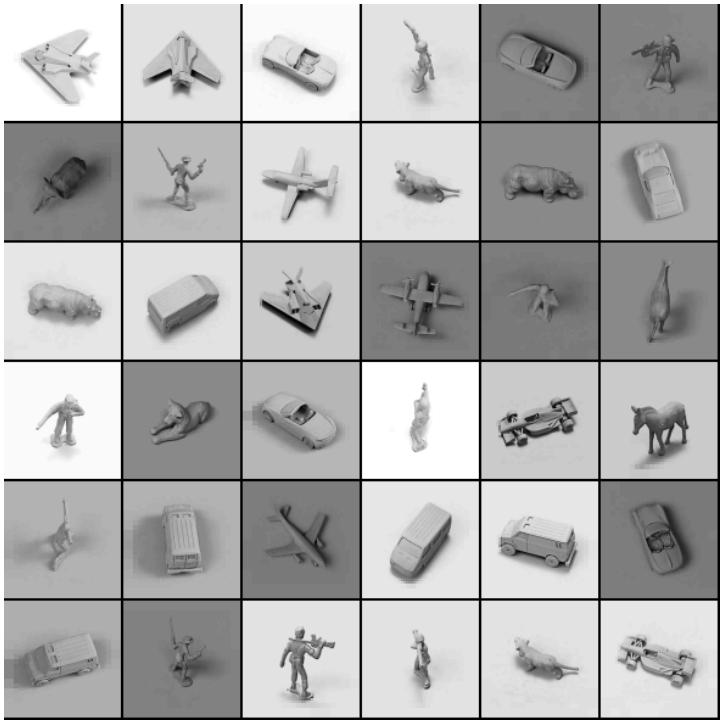
Optical Character Recognition
42,152 examples of 26 English letters

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.

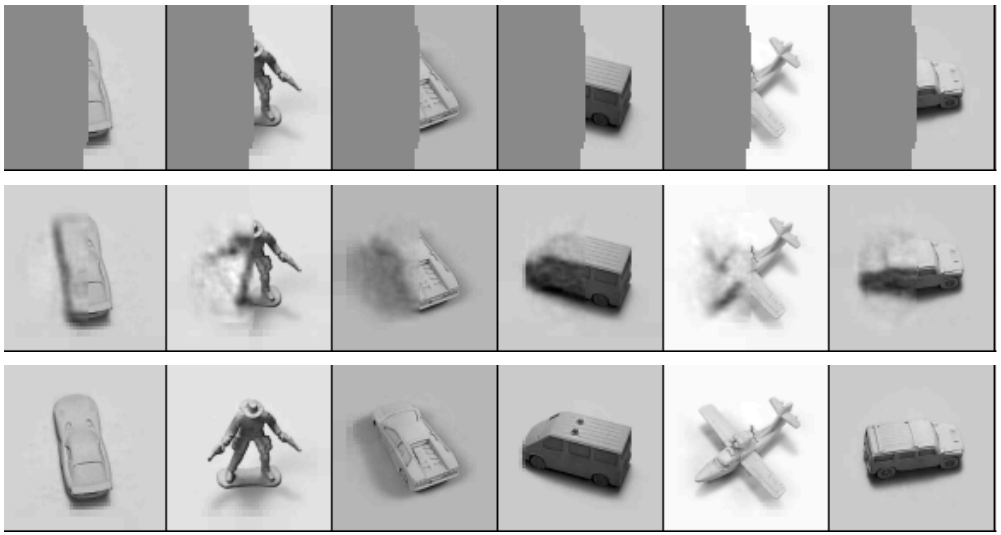
3-D object Recognition

NORB Dataset: 24,000 examples



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



Data – Collection of Modalities

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

car, automobile

sunset, pacificocean, bakerbeach, seashore, ocean

Touch sensors

Vision

Audio

Challenges - I

Image

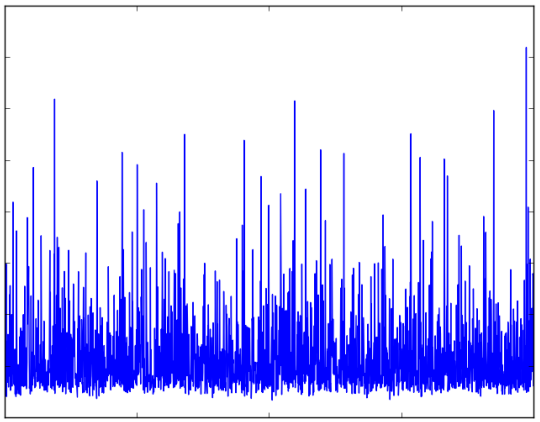
Text

sunset, pacific ocean, baker beach, seashore, ocean

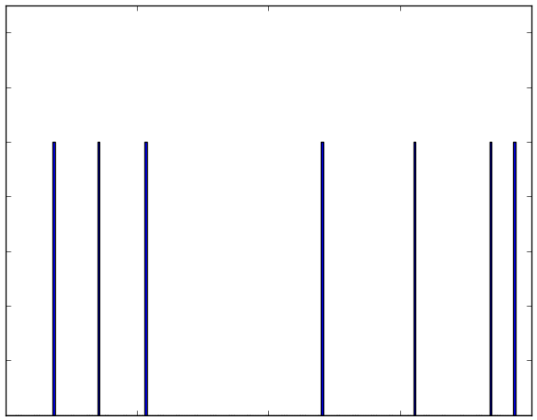
Very different input representations

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

Dense



Sparse



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

Challenges - II

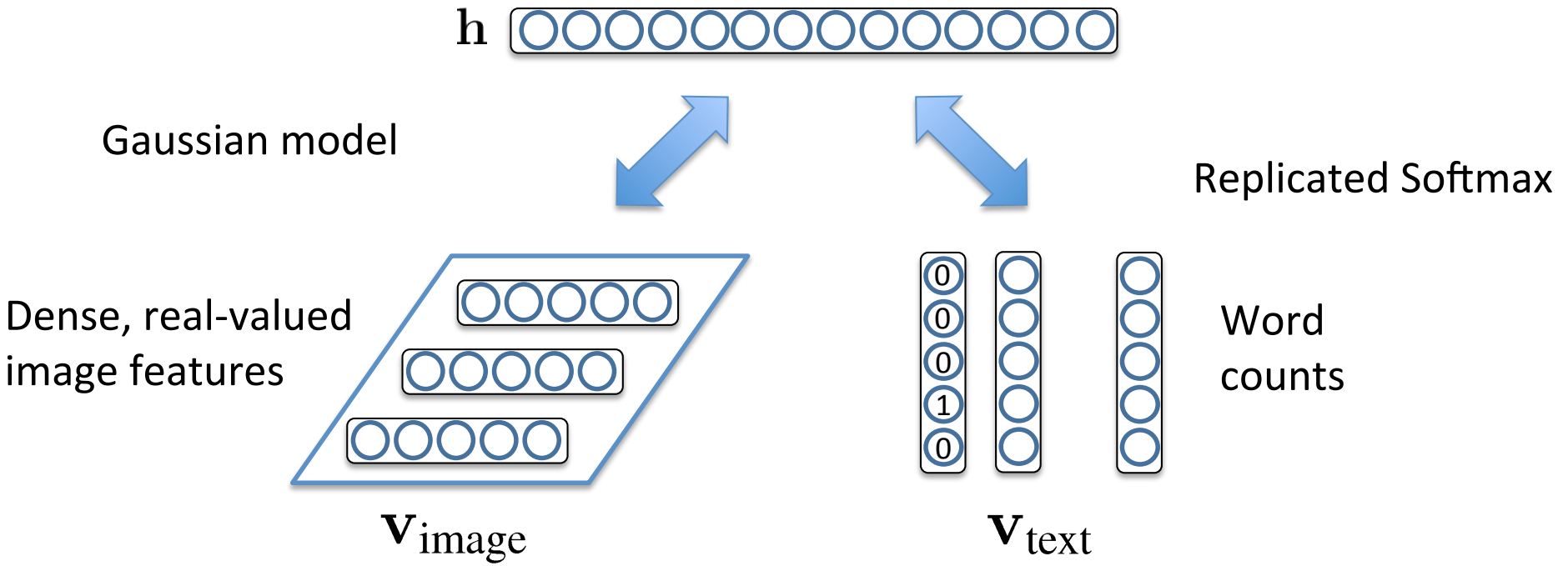
Image	Text
	pentax, k10d, pentaxda50200, kangarooisland, sa, australiansealion
	mickikrimmel, mickipedia, headshot
	< no text>
	unseulpixel, naturey

Noisy and missing data

Challenges - II

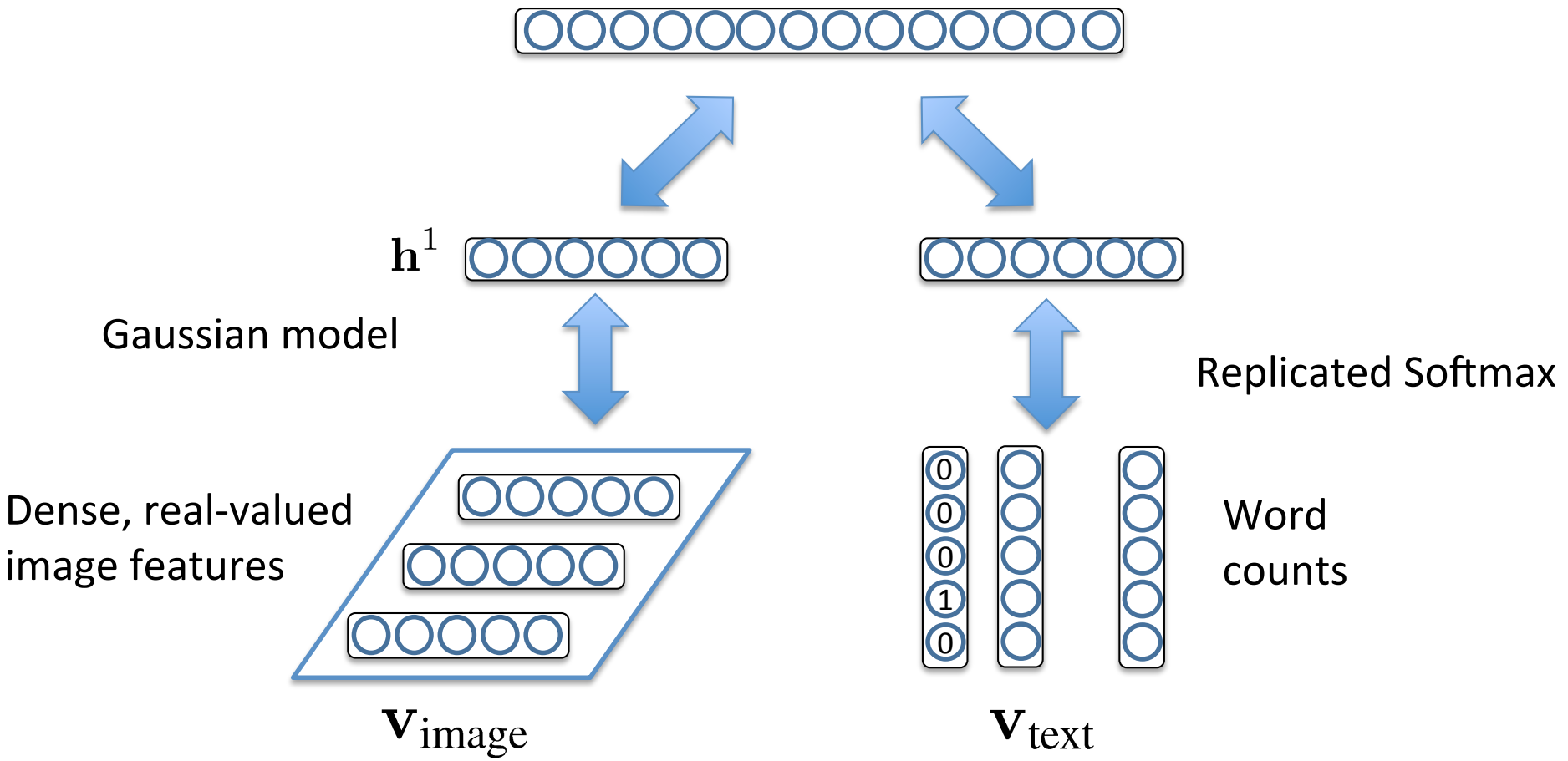
Image	Text	Text generated by the model
	pentax, k10d, pentaxda50200, kangarooisland, sa, australiansealion	beach, sea, surf, strand, shore, wave, seascape, sand, ocean, waves
	mickikrimmel, mickipedia, headshot	portrait, girl, woman, lady, blonde, pretty, gorgeous, expression, model
	< no text>	night,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

Multimodal DBM



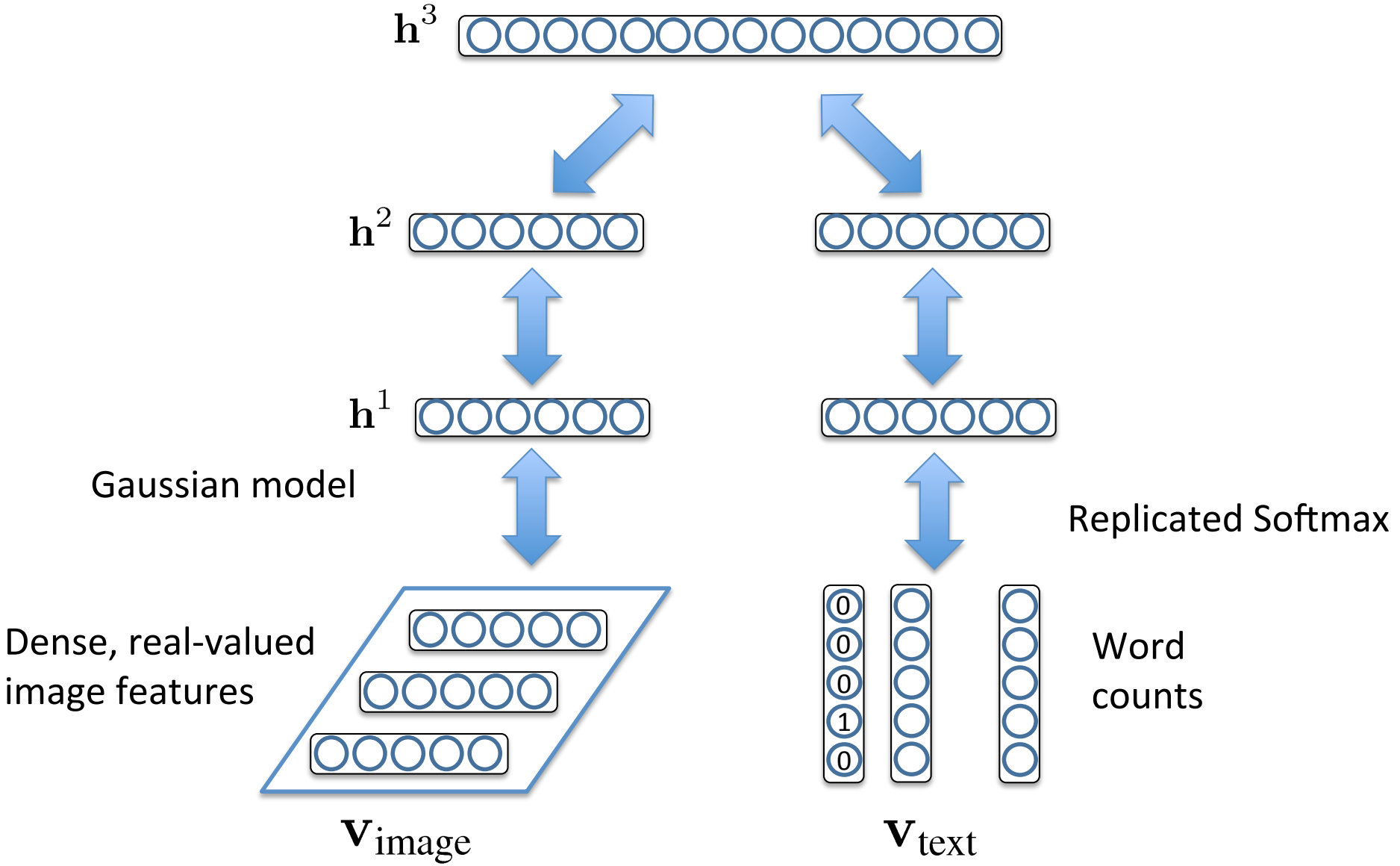
(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)

Multimodal DBM



(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)

Multimodal DBM



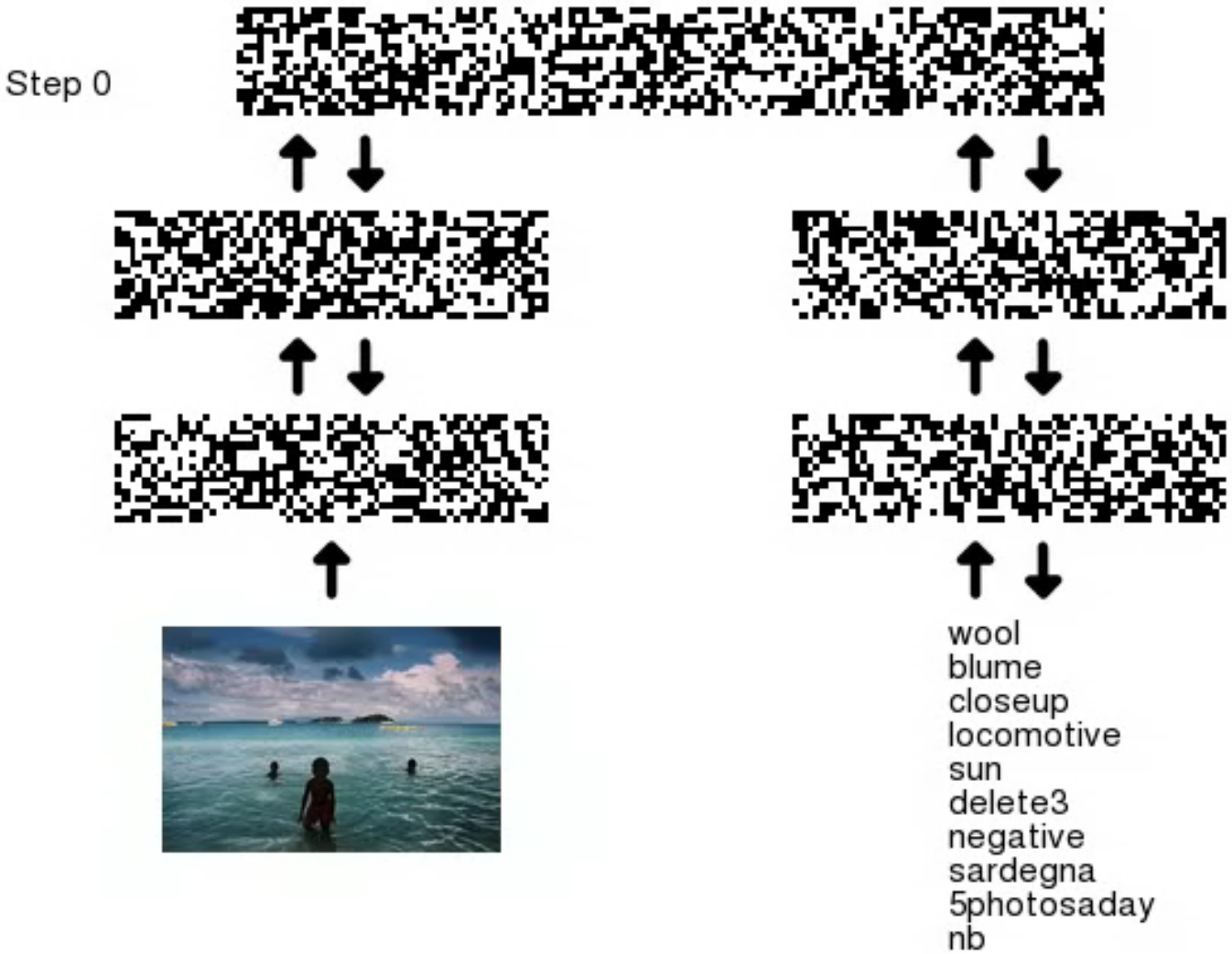
(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)

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

Generating Text from Images



Text Generated from Images

Given

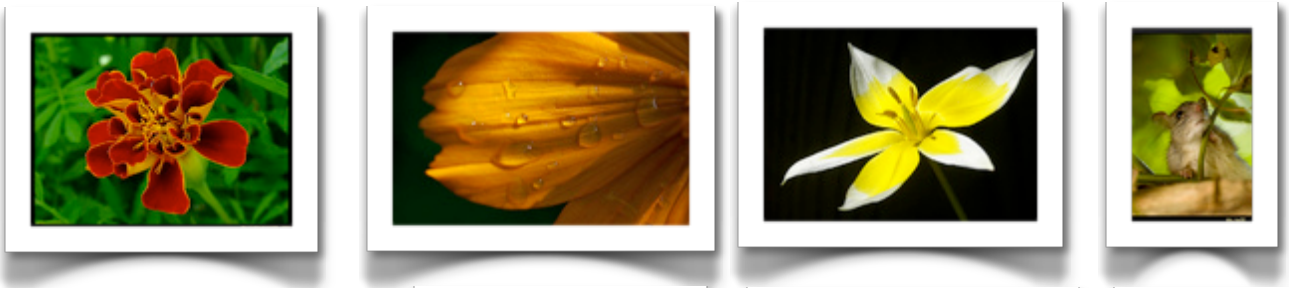
Generated

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

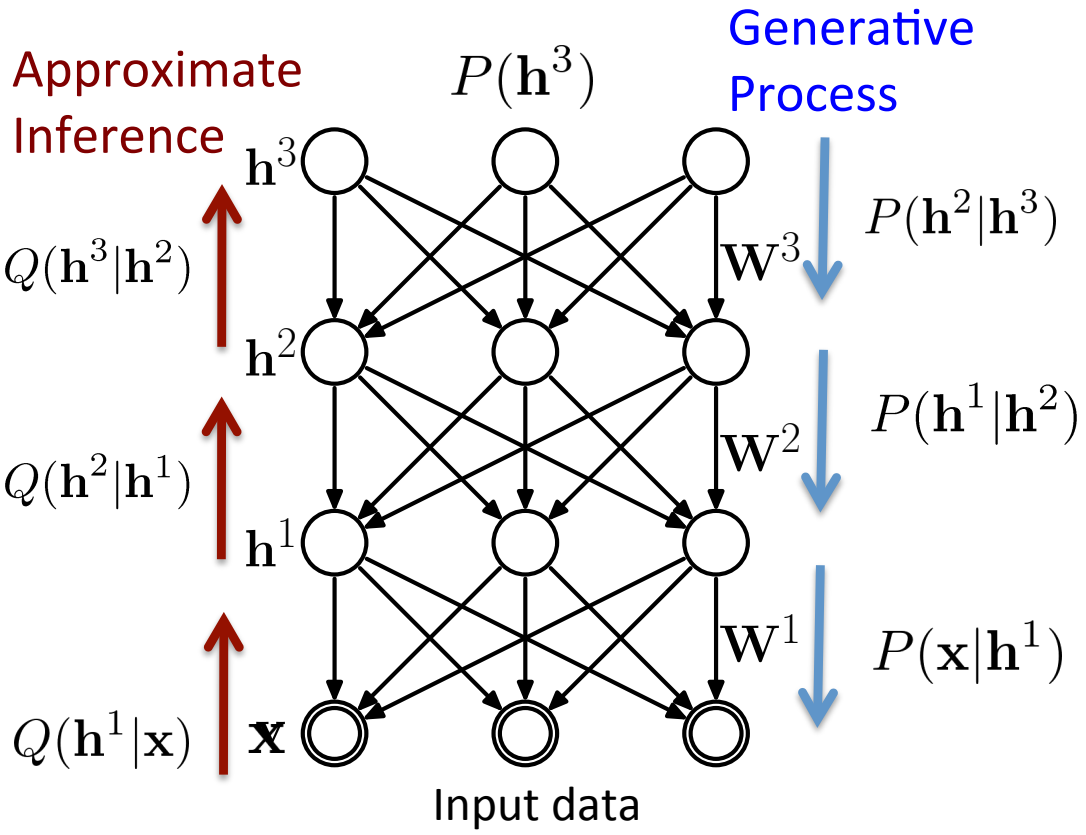
Learning Algorithm	MAP	Precision@50
Random	0.124	0.124
LDA [Huiskes et. al.]	0.492	0.754
SVM [Huiskes et. al.]	0.475	0.758
DBM-Labelled	0.526	0.791
Deep Belief Net	0.638	0.867
Autoencoder	0.638	0.875
DBM	0.641	0.873

Labeled 25K examples

+ 1 Million unlabelled

Helmholtz Machines

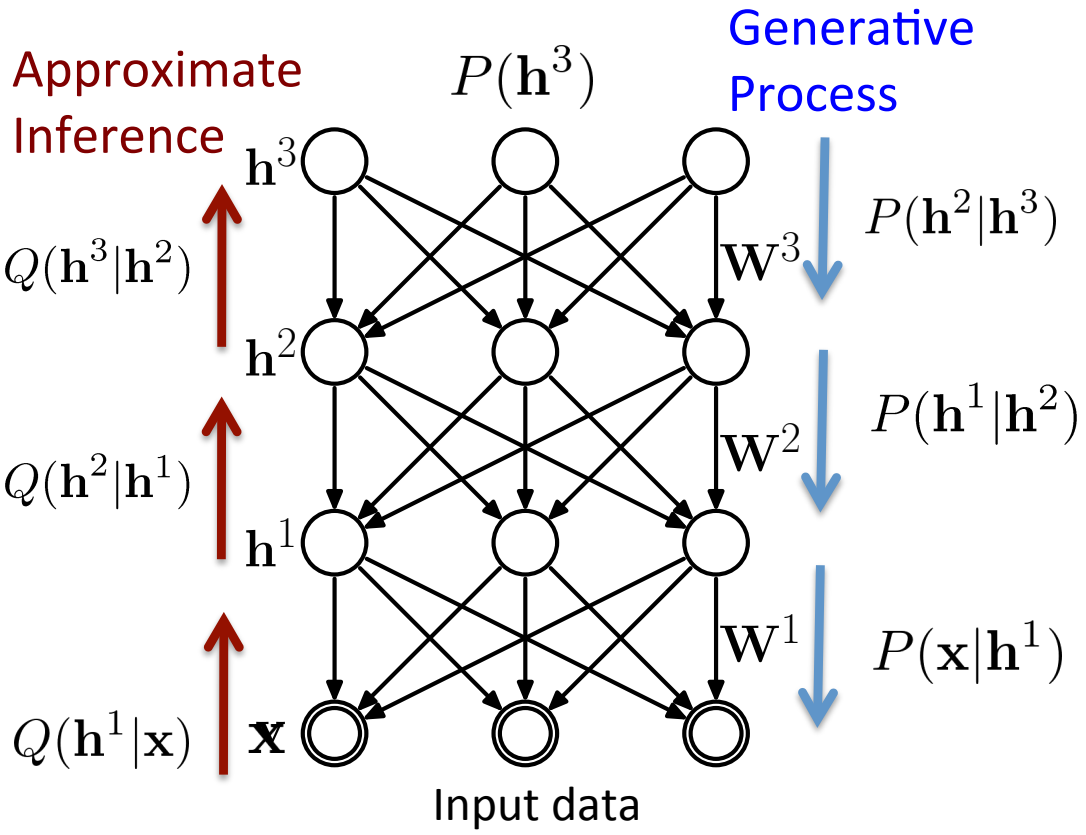
- Hinton, G. E., Dayan, P., Frey, B. J. and Neal, R., Science 1995



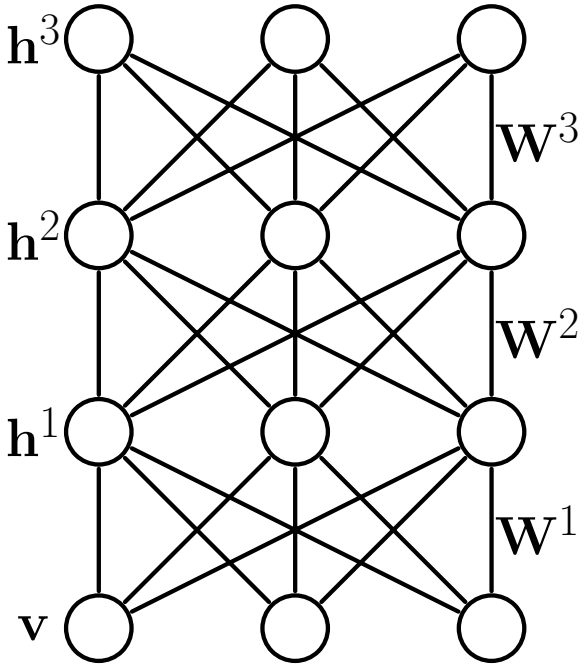
- Kingma & Welling, 2014
- Rezende, Mohamed, Daan, 2014
- Mnih & Gregor, 2014
- Bornschein & Bengio, 2015
- Tang & Salakhutdinov, 2013

Helmholtz Machines vs. DBMs

Helmholtz Machine



Deep Boltzmann Machine

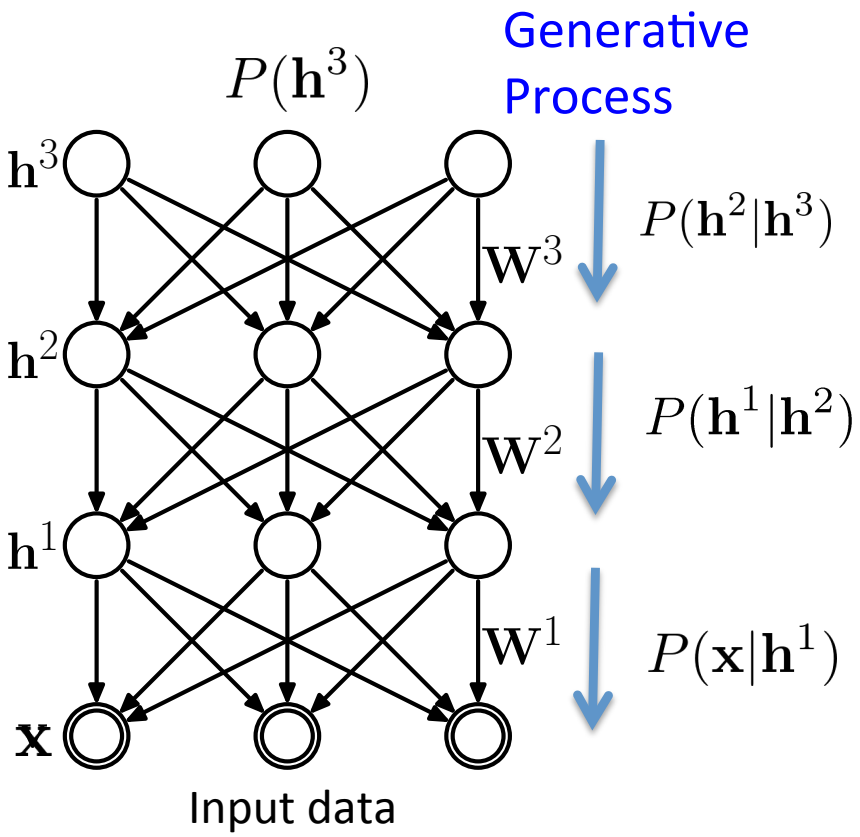


Variational Autoencoders (VAEs)

• 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



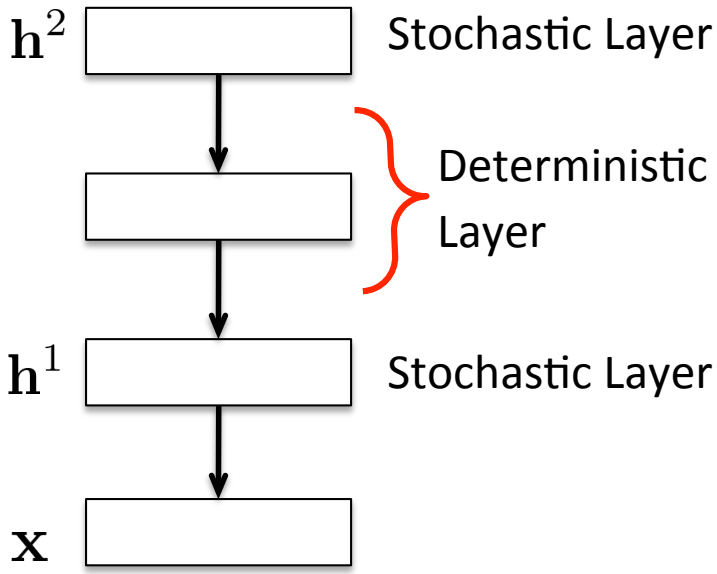
- $\boldsymbol{\theta}$ denotes parameters of VAE.
- L is the number of **stochastic** layers.
- Sampling and probability evaluation is tractable for each $p(\mathbf{h}^\ell|\mathbf{h}^{\ell+1})$.

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 $p(\mathbf{h}^\ell|\mathbf{h}^{\ell+1})$.

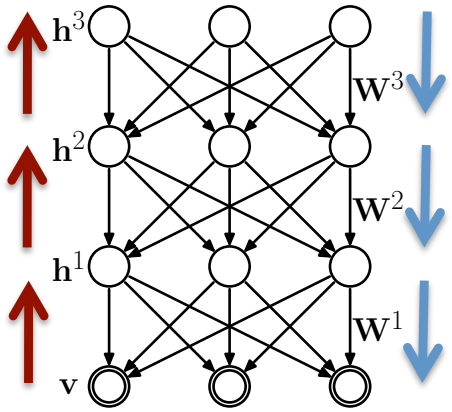
Variational Bound

- 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 $\boldsymbol{\epsilon}$.

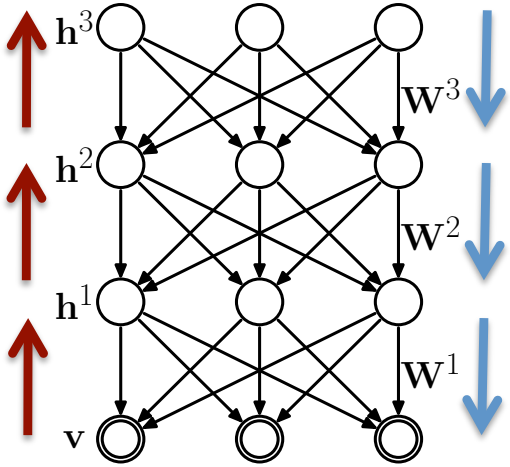
Importance Weighted Autoencoders

- 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

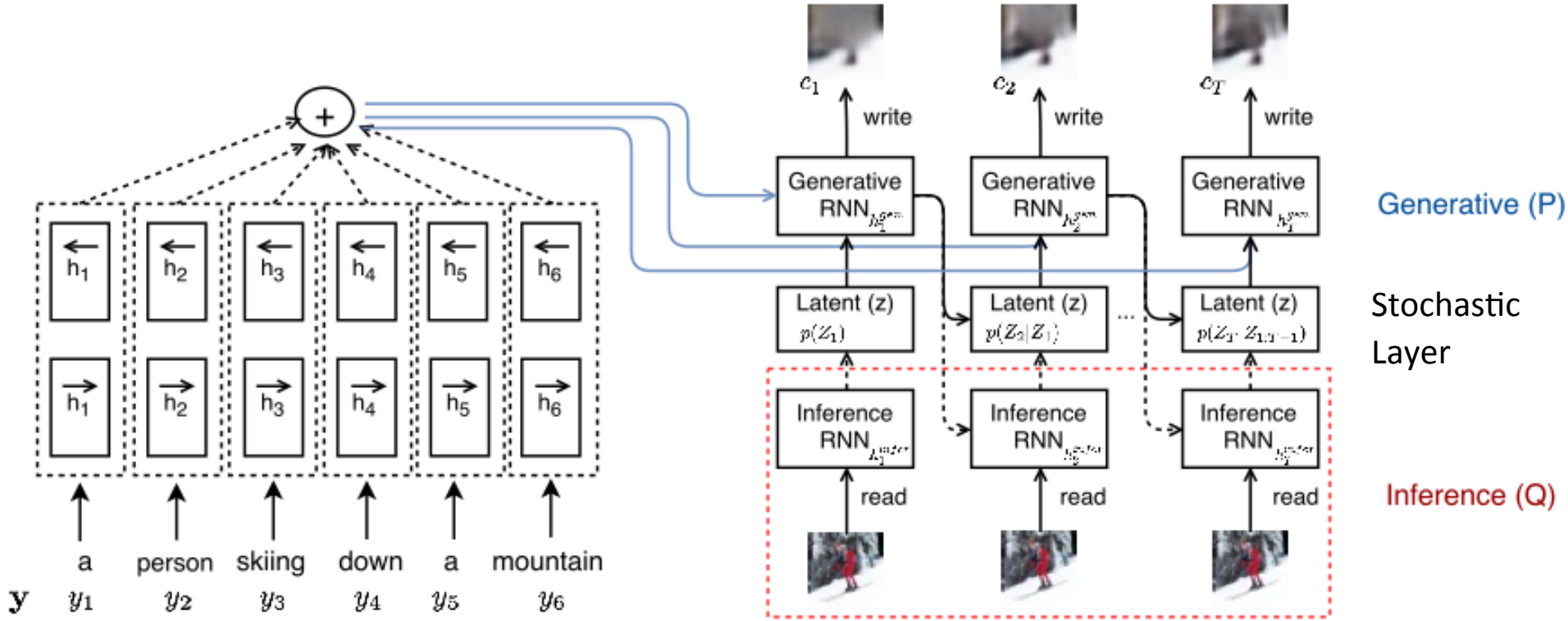


Input data

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.

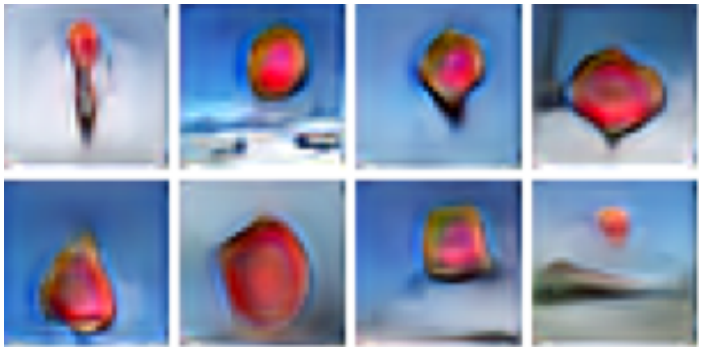
Gregor et. al. 2015

(Mansimov, Parisotto, Ba, Salakhutdinov, 2015)

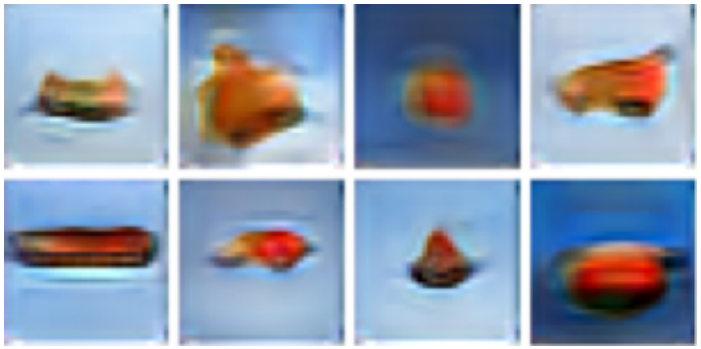
Motivating Example

- Can we generate images from natural language descriptions?

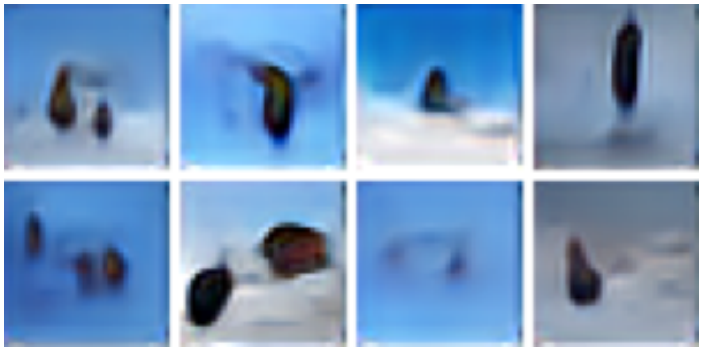
A **stop sign** is flying in blue skies



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)

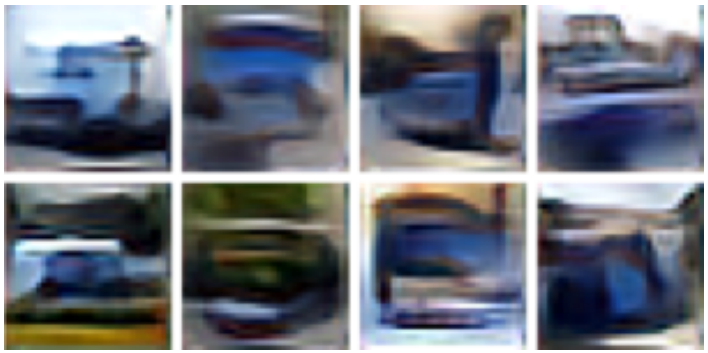
Flipping Colors

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)

Novel Scene Compositions

A toilet seat sits open in the bathroom

A toilet seat sits open in the grass field

Ask Google?

(Some) Open Problems

- Reasoning, Attention, and Memory
- Natural Language Understanding
- Deep Reinforcement Learning
- Unsupervised Learning / Transfer Learning / One-Shot Learning

(Some) Open Problems

- Reasoning, Attention, and Memory
- Natural Language Understanding
- Deep Reinforcement Learning
- Unsupervised Learning / Transfer Learning / One-Shot Learning

Who-Did-What Dataset

- **Document**: "…arrested Illinois governor **Rod Blagojevich** and his chief of staff John Harris on corruption charges … included **Blogojevich** allegedly conspiring to sell or trade the senate seat left vacant by President-elect Barack Obama…"

- **Query**: President-elect Barack Obama said Tuesday he was not aware of alleged corruption by **X** who was arrested on charges of trying to sell Obama's senate seat.

- **Answer**: Rod Blagojevich

Onishi, Wang, Bansal, Gimpel, McAllester, EMNLP, 2016

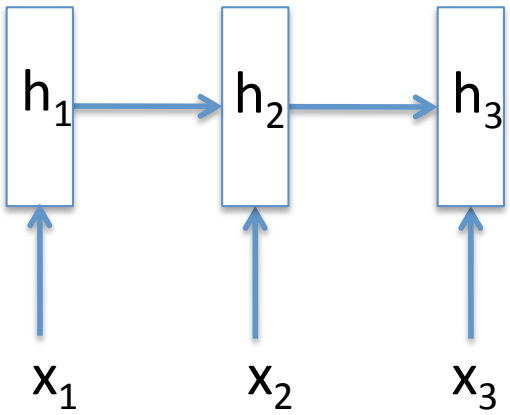
Recurrent Neural Network

$$\mathbf{h_t} = \phi\left(\mathbf{U}\mathbf{h_{t-1}} + \mathbf{W}\mathbf{x_t} + \mathbf{b}\right)$$

Nonlinearity

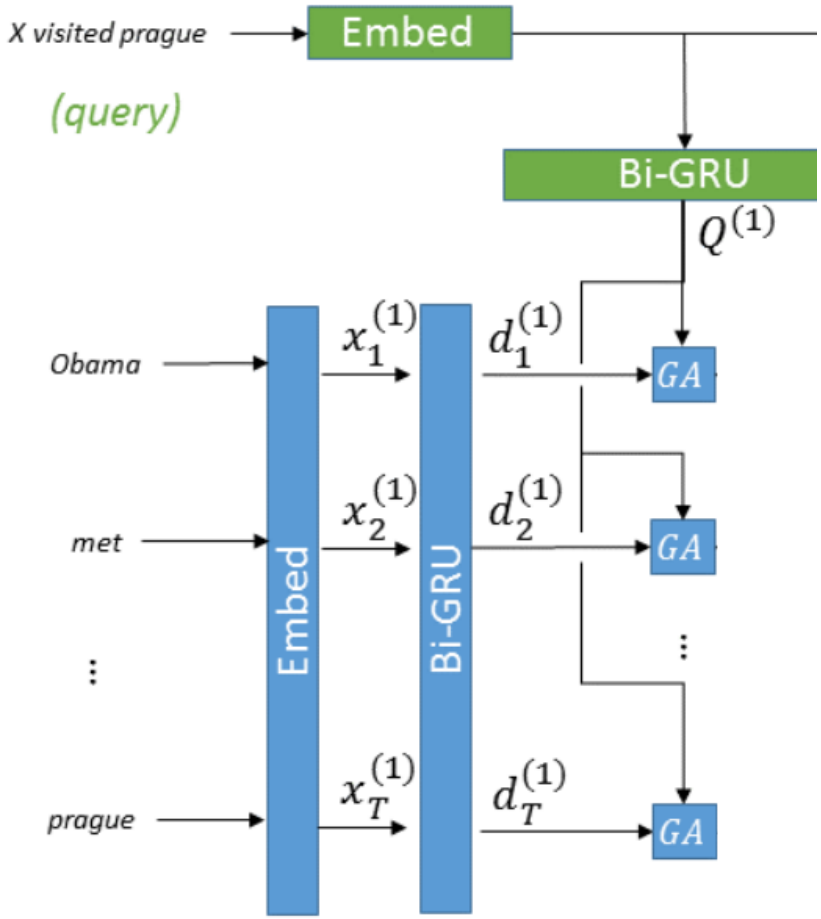
Hidden State at previous time step

Input at time step t



Gated Attention Mechanism

• Use Recurrent Neural Networks (RNNs) to encode a document and a query:



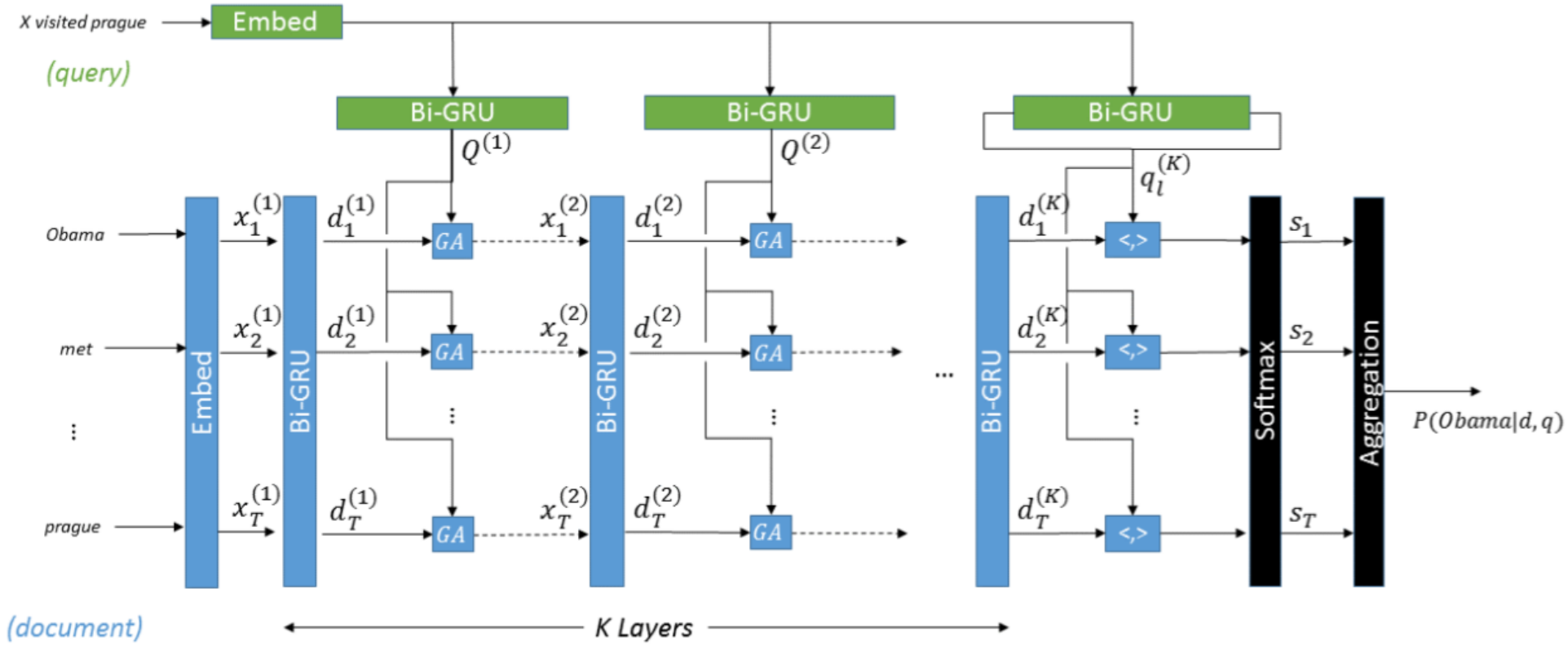
- Use element-wise multiplication to model the interactions between document and query:

$$x_i = d_i \odot q_i$$

(Dhingra, Liu, Yang, Cohen, Salakhutdinov, ACL 2017)

Multi-hop Architecture

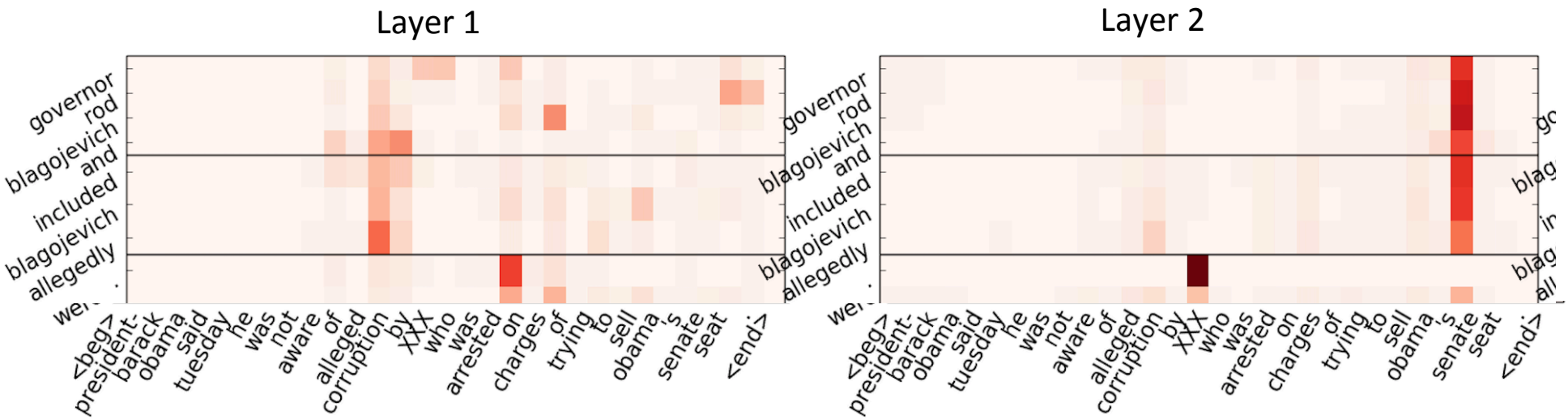
- Reasoning requires several passes over the context



(Dhingra, Liu, Yang, Cohen, Salakhutdinov, ACL 2017)

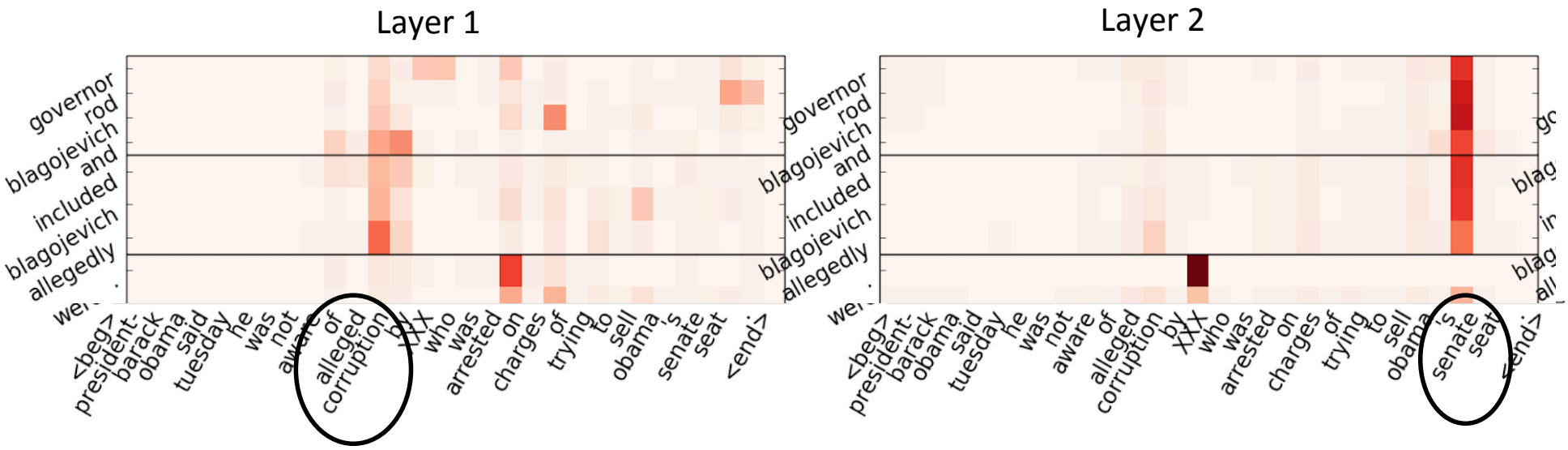
Analysis of Attention

- **Context**: "…arrested Illinois governor **Rod Blagojevich** and his chief of staff John Harris on corruption charges … included **Blogojevich** allegedly conspiring to sell or trade the senate seat left vacant by President-elect Barack Obama…"
- **Query**: "President-elect Barack Obama said Tuesday he was not aware of alleged corruption by **X** who was arrested on charges of trying to sell Obama's senate seat."
- **Answer**: **Rod Blagojevich**



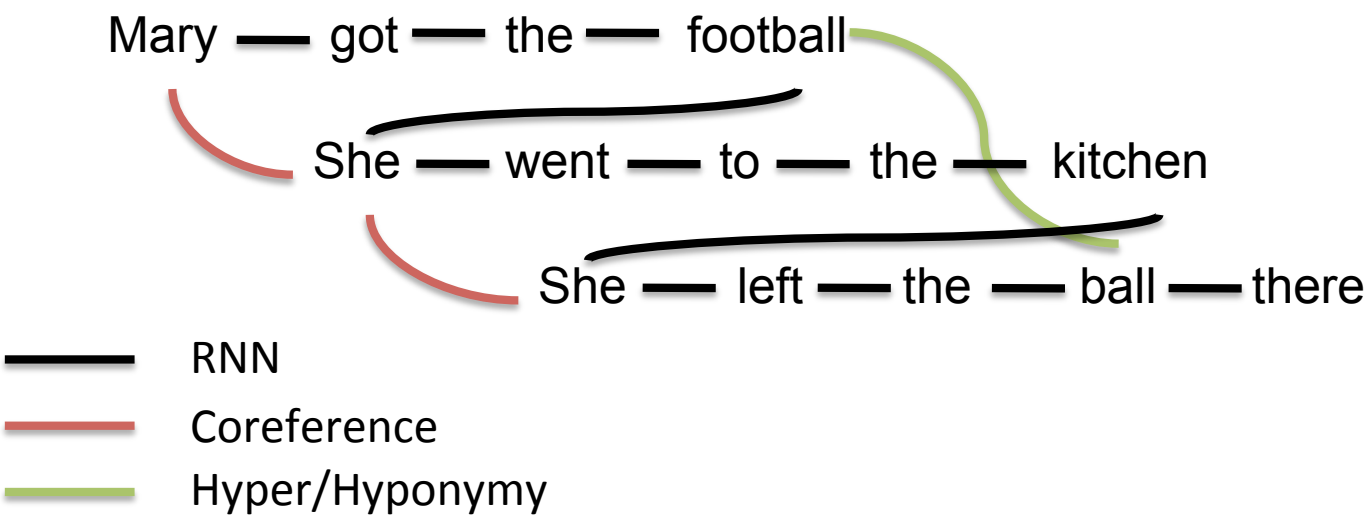
Analysis of Attention

- **Context**: "…arrested Illinois governor **Rod Blagojevich** and his chief of staff John Harris on corruption charges … included **Blogojevich** allegedly conspiring to sell or trade the senate seat left vacant by President-elect Barack Obama…"
- **Query**: "President-elect Barack Obama said Tuesday he was not aware of alleged corruption by **X** who was arrested on charges of trying to sell Obama's senate seat."
- **Answer**: **Rod Blagojevich**



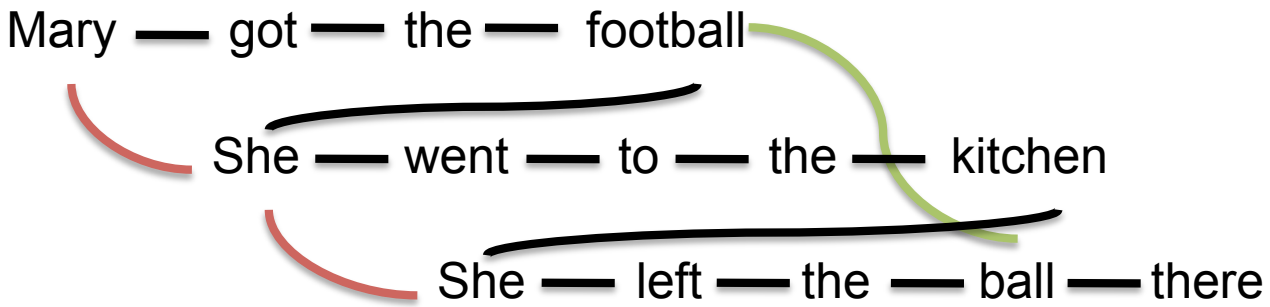
Code + Data: https://github.com/bdhingra/ga-reader

Incorporating Prior Knowledge



Dhingra, Yang, Cohen, Salakhutdinov 2017

Incorporating Prior Knowledge

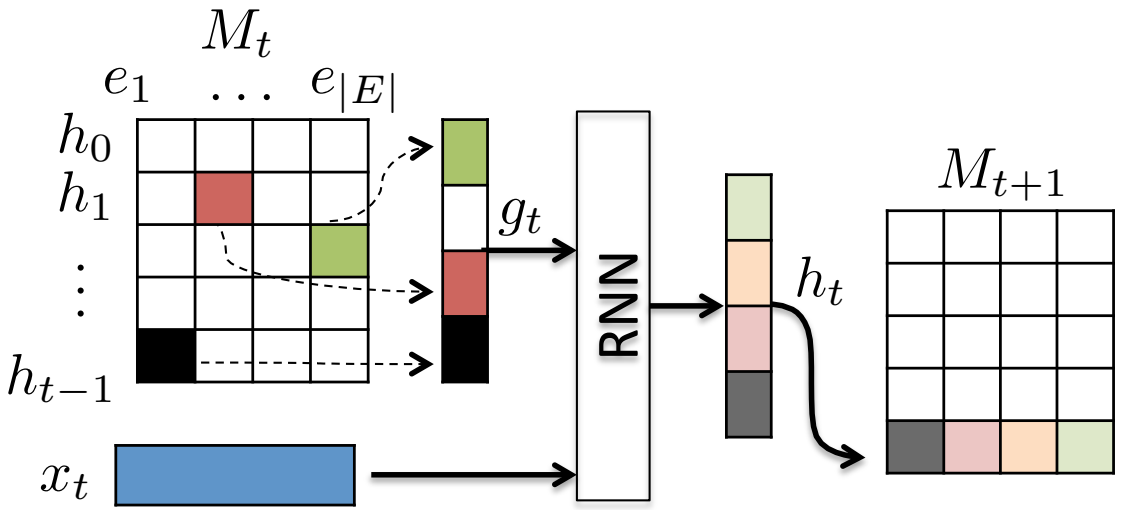


RNN

Coreference

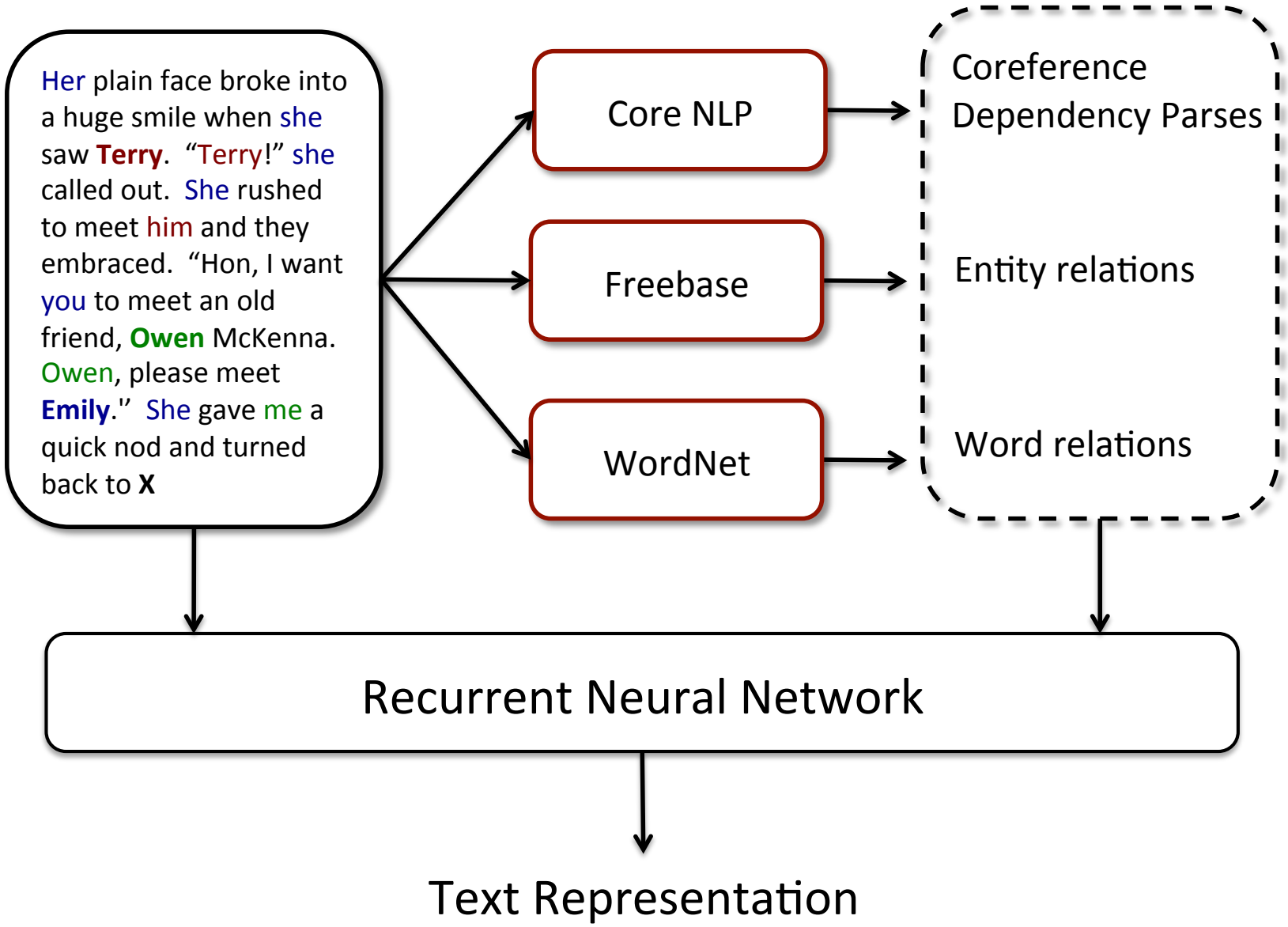
Hyper/Hyponymy

Memory as **A**cyclic **G**raph **E**ncoding (MAGE) - RNN



Dhingra, Yang, Cohen, Salakhutdinov 2017

Incorporating Prior Knowledge



Neural Story Telling

Sample from the Generative Model (recurrent neural network):

She was in love with him for the first time in months, so she had no intention of escaping.

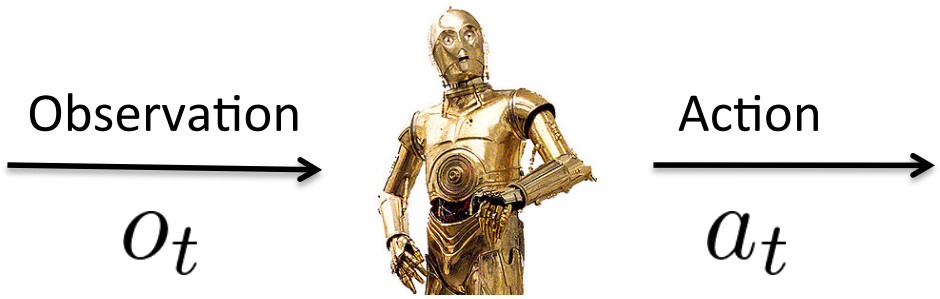
The sun had risen from the ocean, making her feel more alive than normal. She is beautiful, but the truth is that I do not know what to do. The sun was just starting to fade away, leaving people scattered around the Atlantic Ocean.

(Kiros et al., NIPS 2015)

(Some) Open Problems

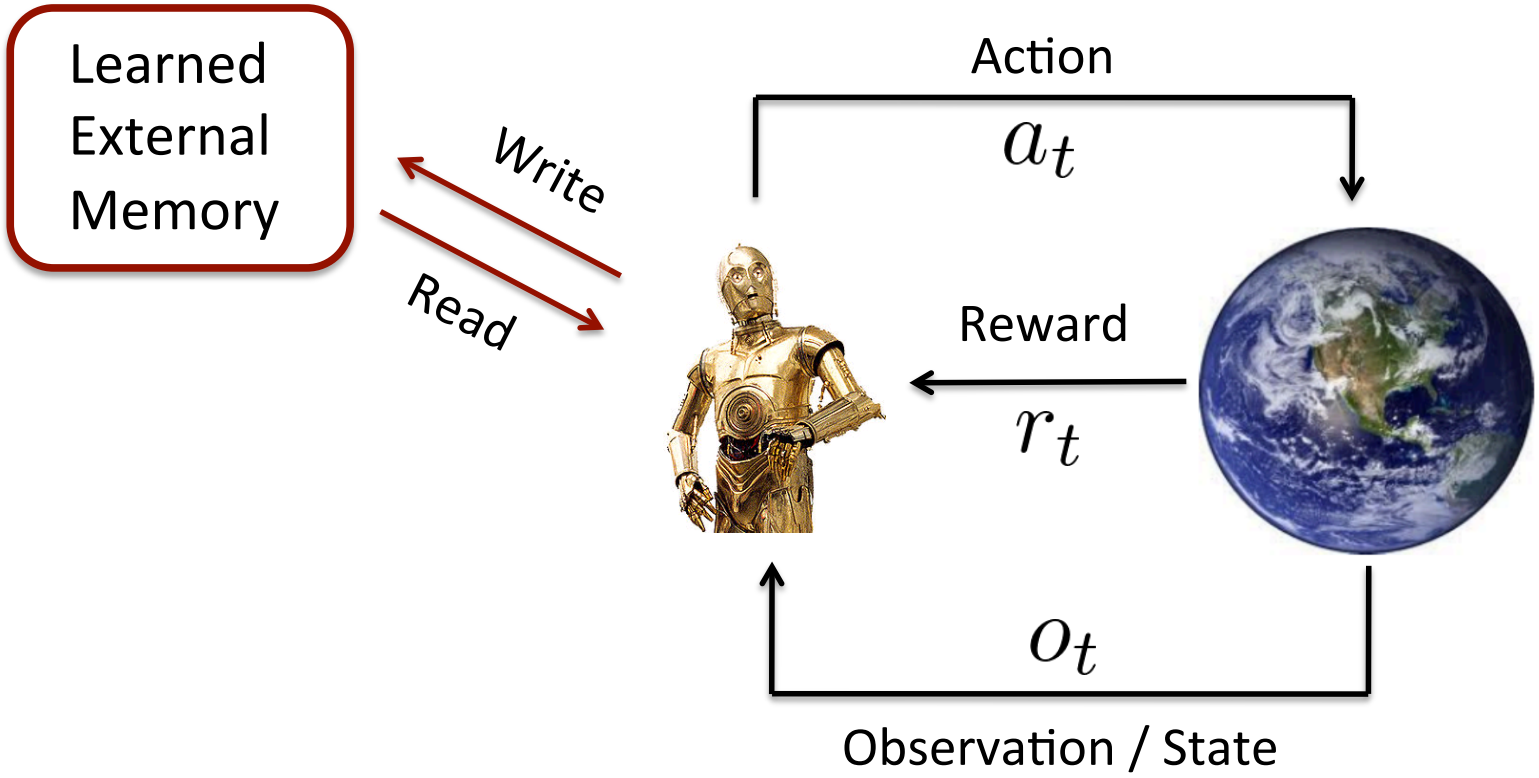
- Reasoning, Attention, and Memory
- Natural Language Understanding
- Deep Reinforcement Learning
- Unsupervised Learning / Transfer Learning / One-Shot Learning

Learning Behaviors



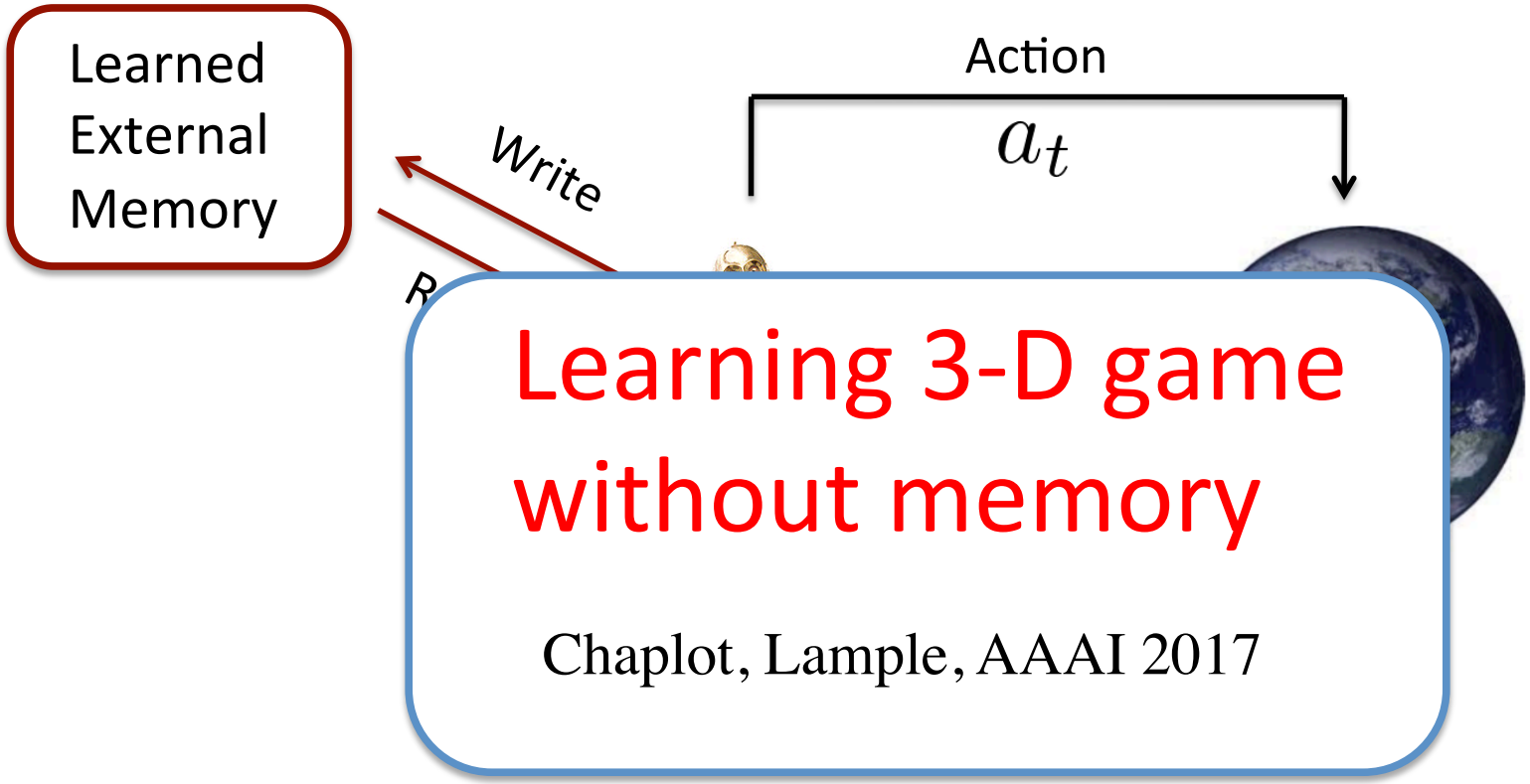
Learning to map sequences of observations to actions, for a particular goal

Reinforcement Learning with Memory



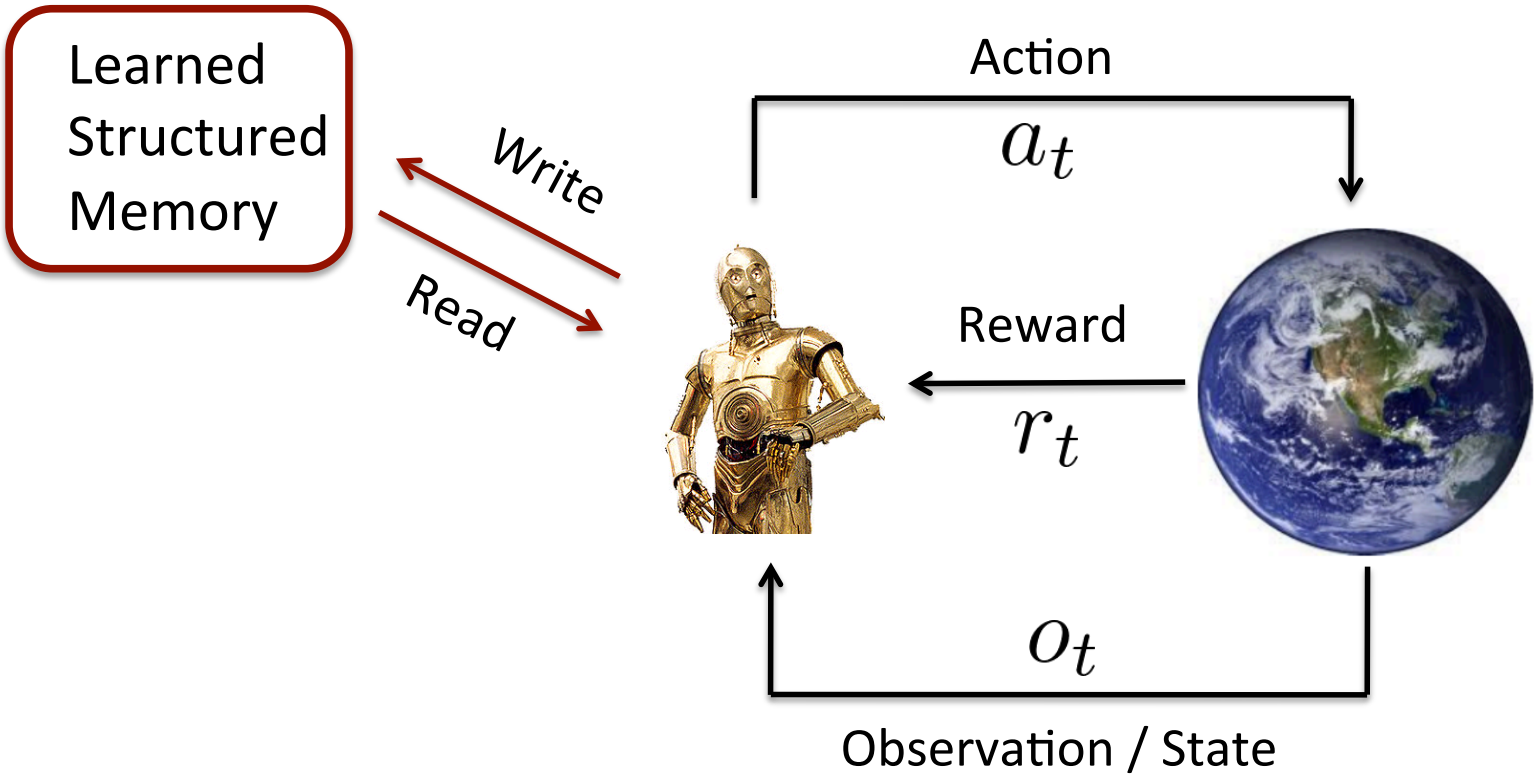
Differentiable Neural Computer, Graves et al., Nature, 2016;
Neural Turing Machine, Graves et al., 2014

Reinforcement Learning with Memory



Differentiable Neural Computer, Graves et al., Nature, 2016;
Neural Turing Machine, Graves et al., 2014

Deep RL with Memory

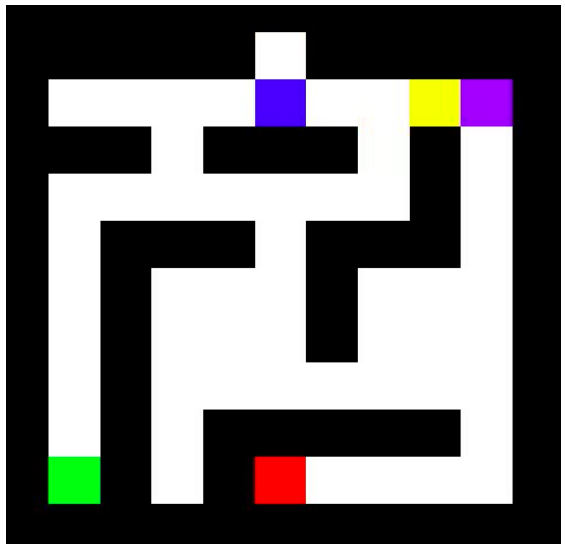


Parisotto, Salakhutdinov, 2017

Random Maze with Indicator

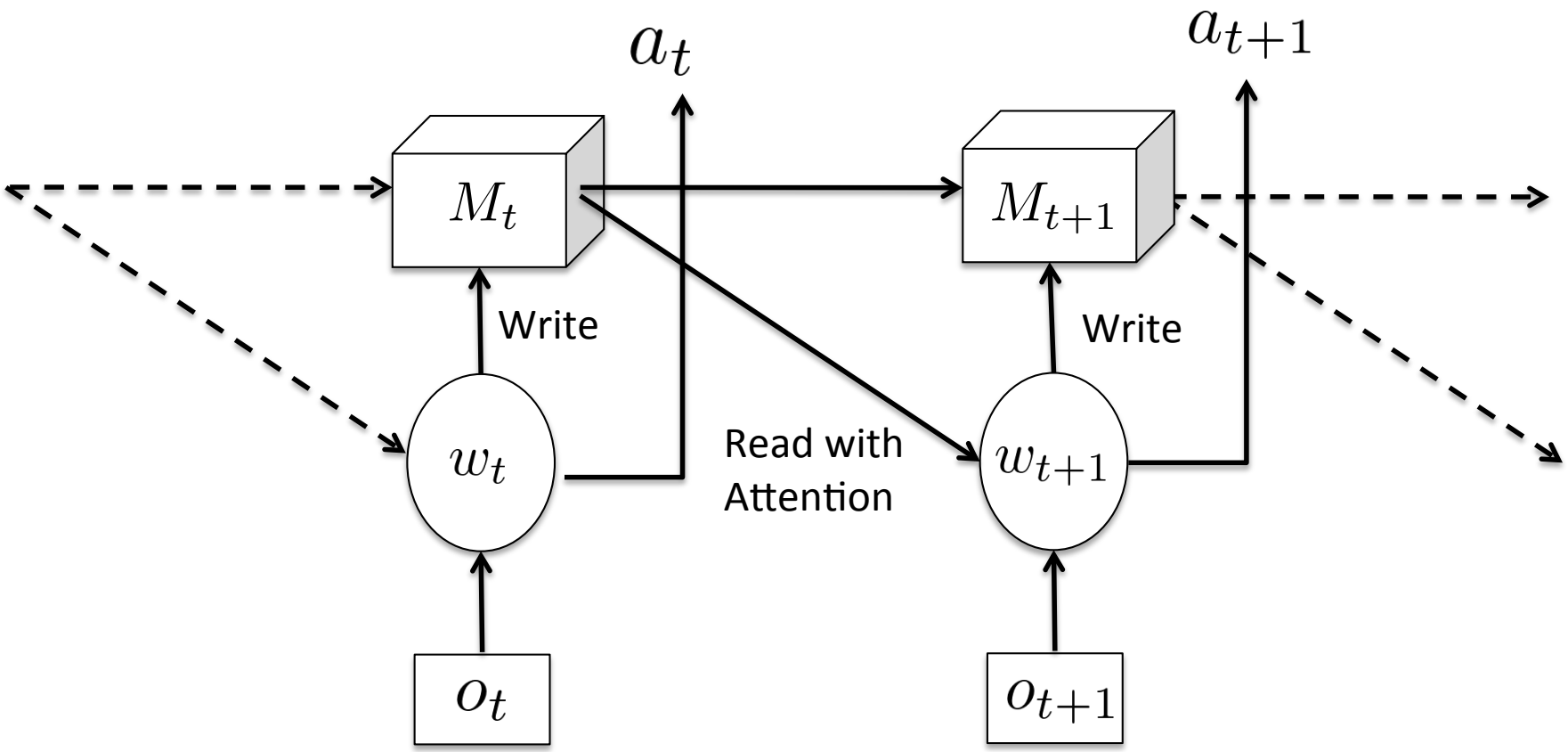
- Indicator: Either blue or pink
  - If blue, find the green block
  - If pink, find the red block

- Negative reward if agent does not find correct block in N steps or goes to wrong block.



Parisotto, Salakhutdinov, 2017

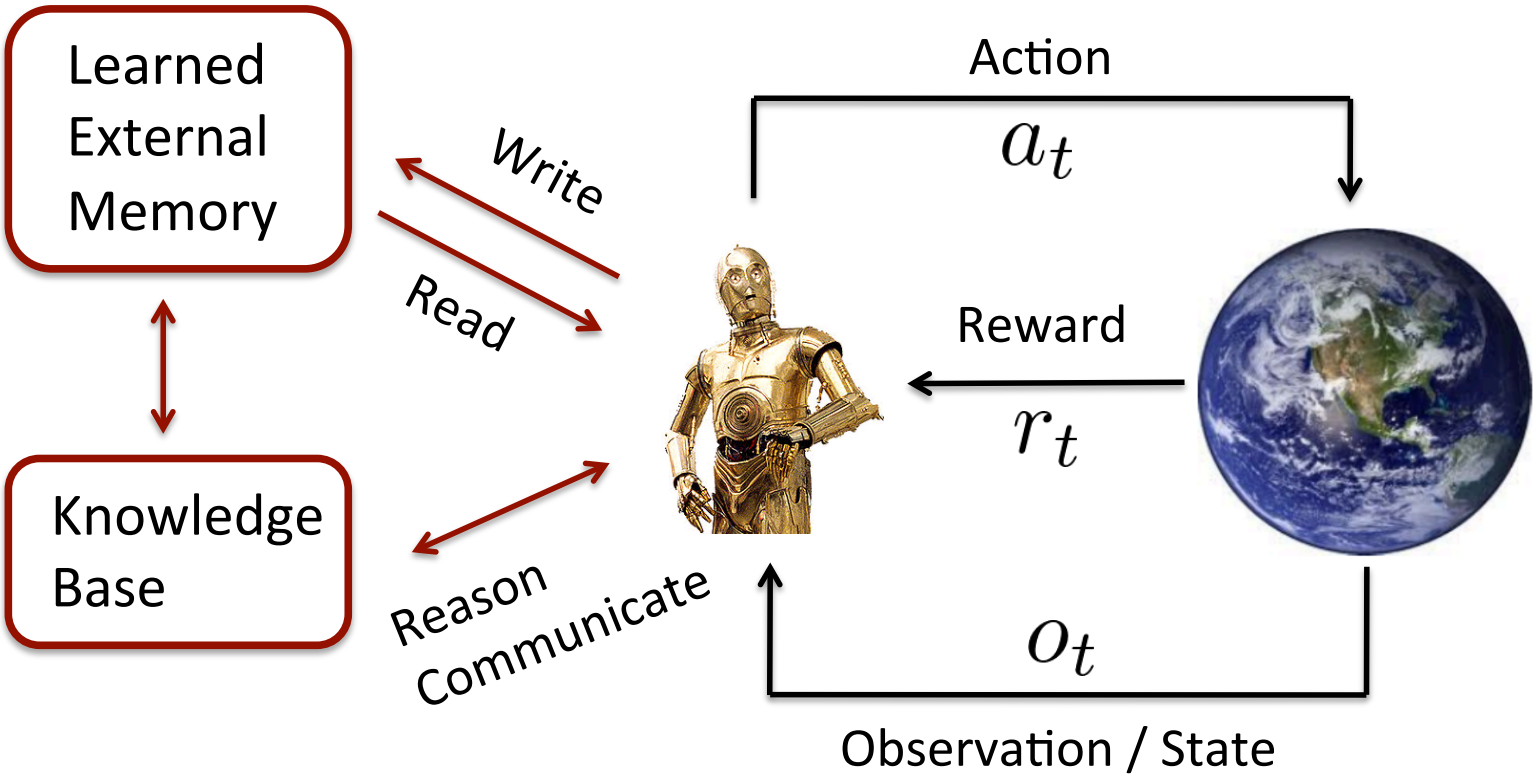
Random Maze with Indicator



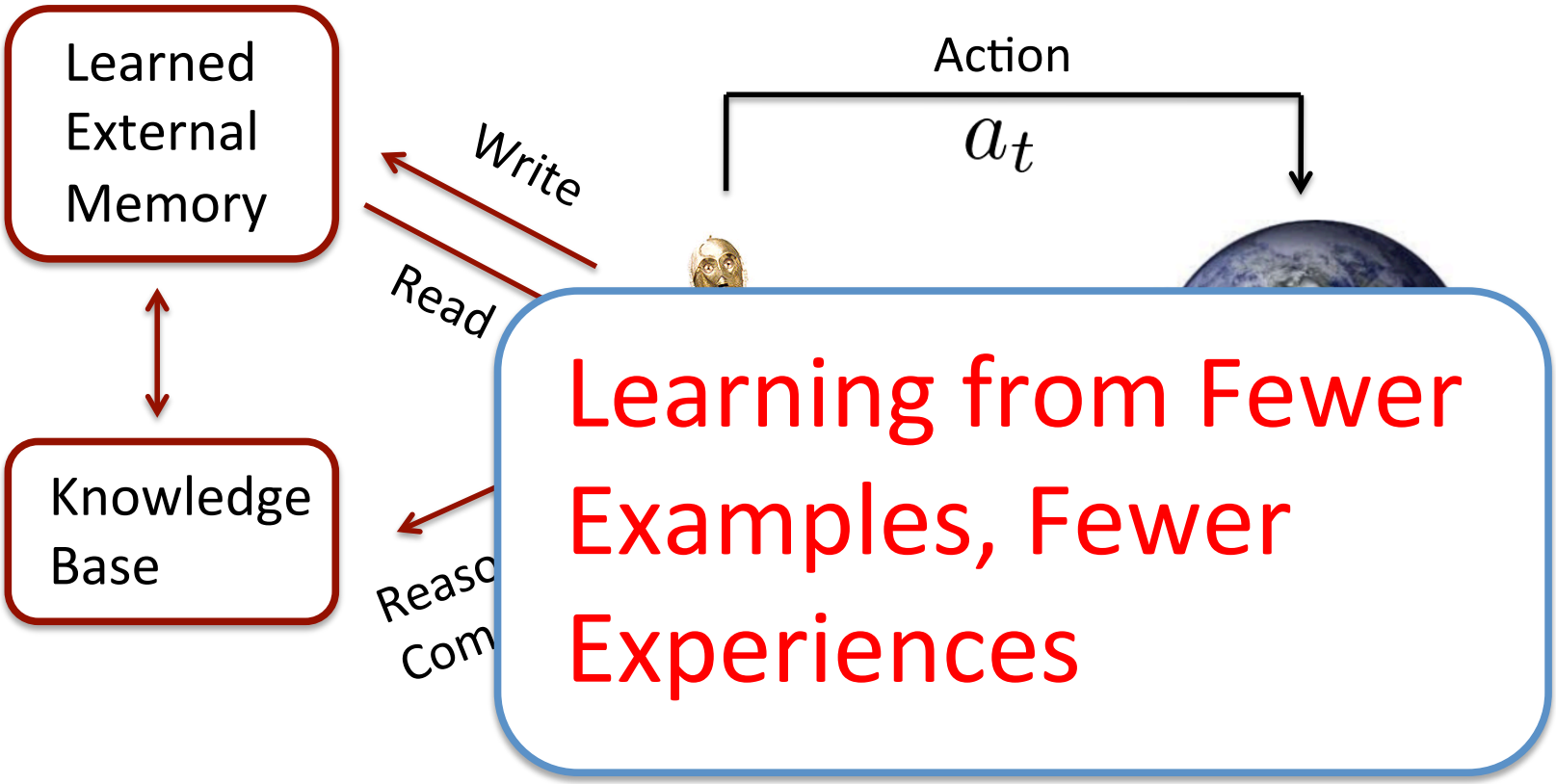
Parisotto, Salakhutdinov, 2017

Random Maze with Indicator

Building Intelligent Agents



Building Intelligent Agents

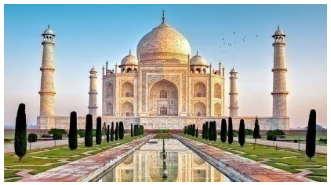


Summary

- Efficient learning algorithms for Deep Unsupervised Models

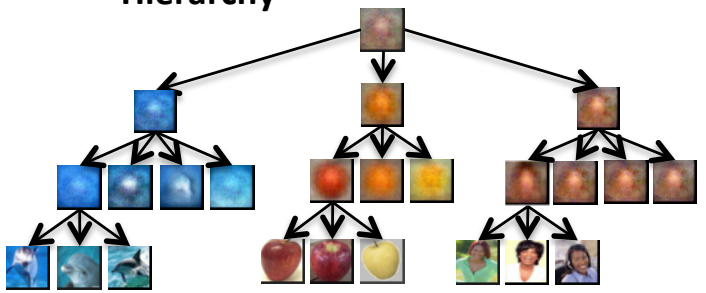
Text & image retrieval / Object recognition

Image Tagging

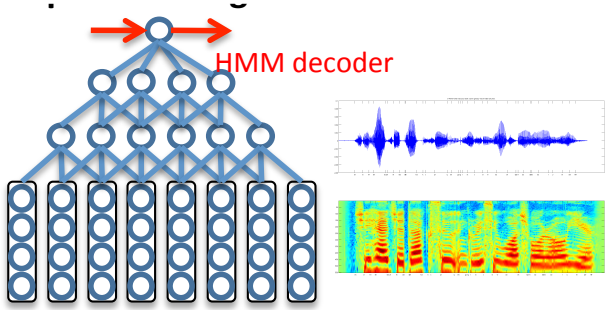


mosque, tower, building, cathedral, dome, castle

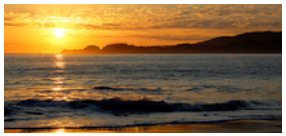
Learning a Category Hierarchy



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