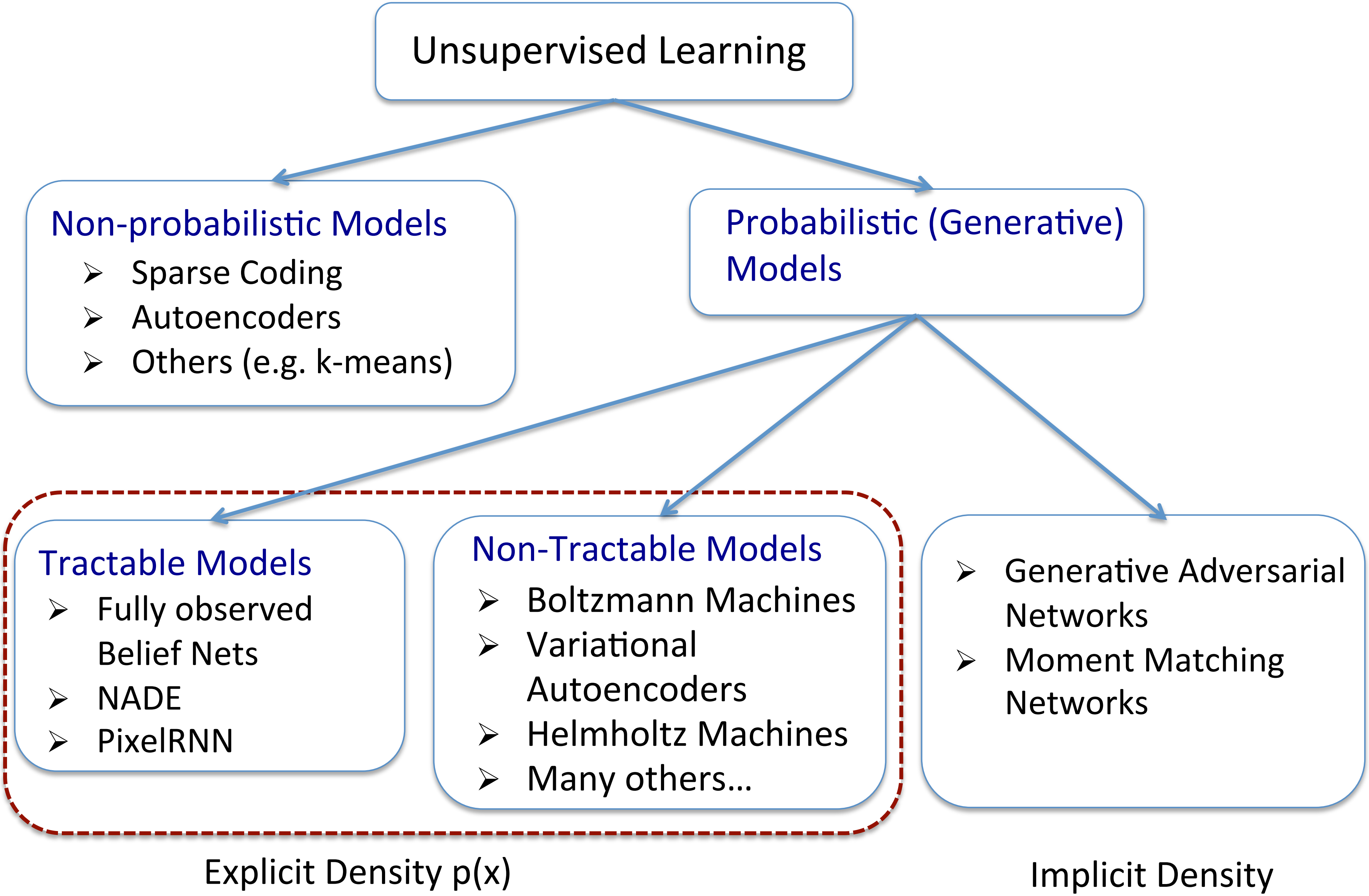
Deep Learning III
Unsupervised Learning

Russ Salakhutdinov

Machine Learning Department
Carnegie Mellon University
Canadian Institute of Advanced Research

Carnegie Mellon University

CIFAR
CANADIAN INSTITUTE
for ADVANCED RESEARCH

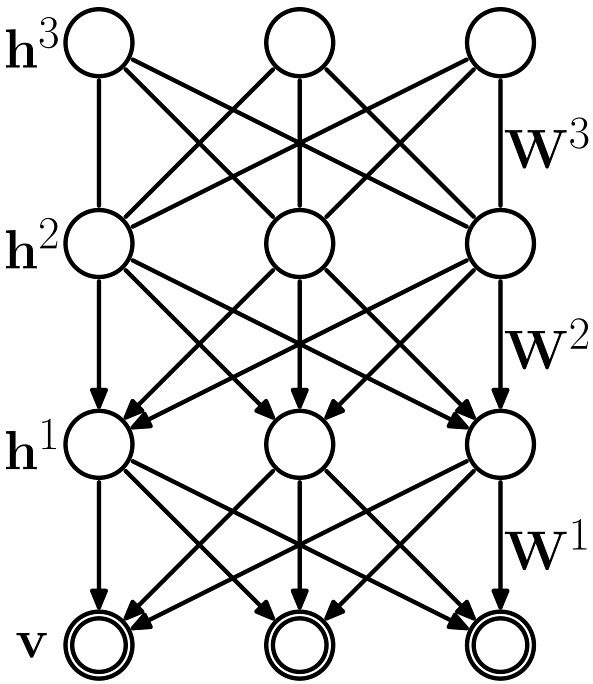

Unsupervised Learning
Non-probabilistic Models
Sparse Coding
Autoencoders
Others (e.g. k-means)
Probabilistic (Generative) Models
Tractable Models
Fully observed Belief Nets
NADE
PixelRNN
Non-Tractable Models
Boltzmann Machines
Variational Autoencoders
Helmholtz Machines
Many others…
Generative Adversarial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

Talk Roadmap

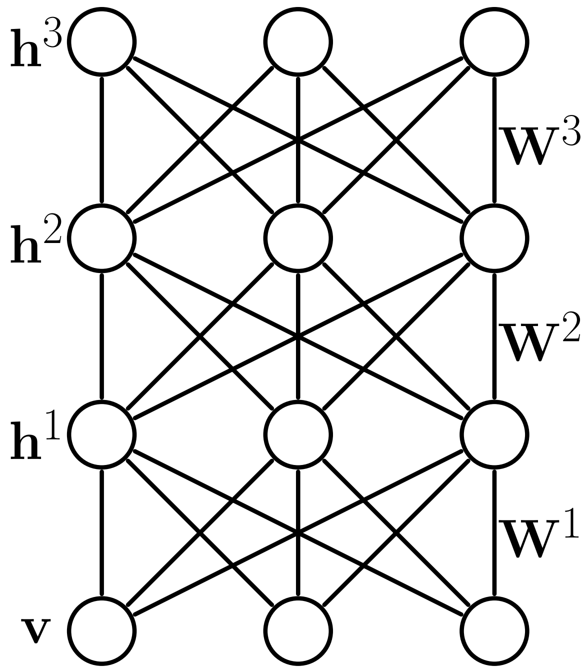
- Basic Building Blocks:
  - Sparse Coding
  - Autoencoders
- Deep Generative Models
  - Restricted Boltzmann Machines
  - Deep Belief Networks and Deep Boltzmann Machines
  - Helmholtz Machines / Variational Autoencoders
- Generative Adversarial Networks
- Model Evaluation

DBNs vs. DBMs

Deep Belief Network



Deep Boltzmann Machine



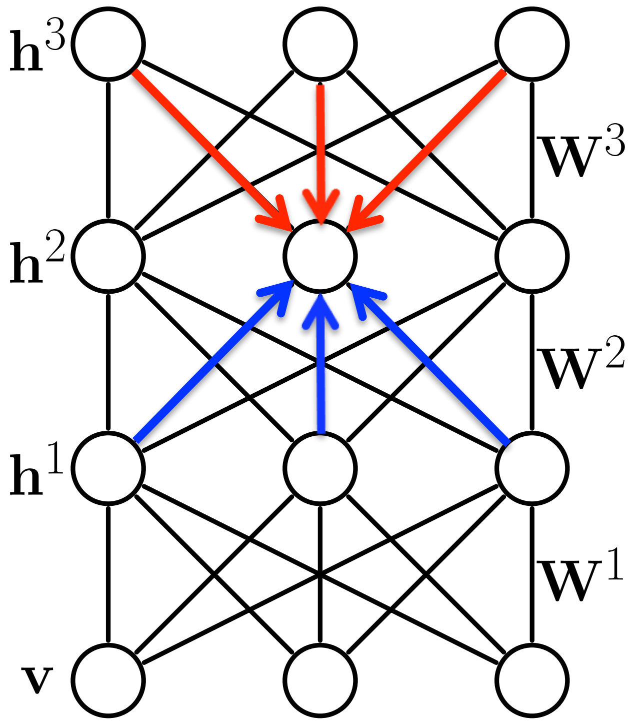
DBNs are hybrid models:

- Inference in DBNs is problematic due to **explaining away**.
- Only greedy pretrainig, **no joint optimization over all layers**.
- Approximate inference is feed-forward: **no bottom-up and top-down**.

Mathematical Formulation

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

Deep Boltzmann Machine



Input

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

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

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

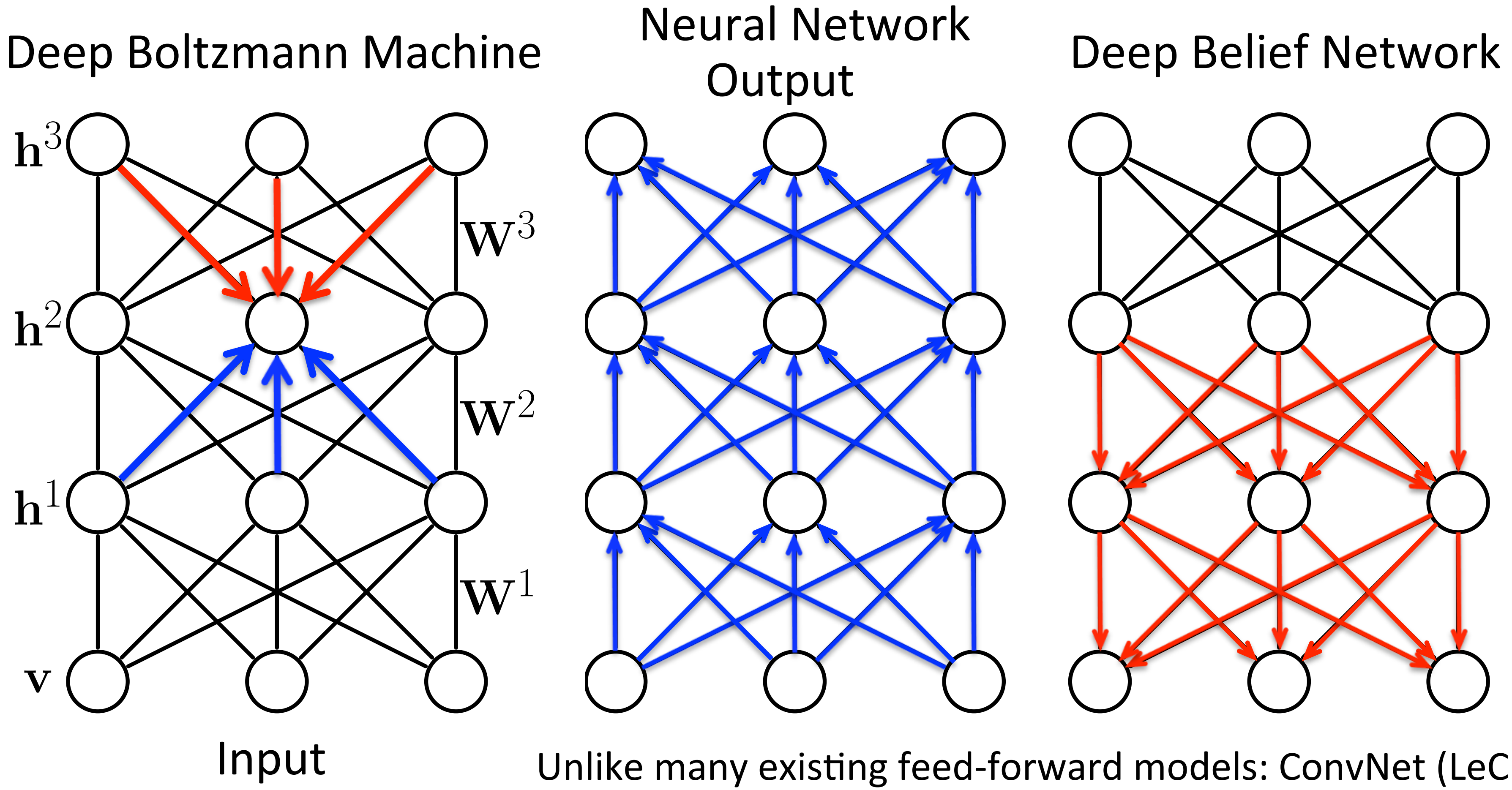
Bottom-up

Top-Down

Unlike many existing feed-forward models: ConvNet (LeCun), HMAX (Poggio et.al.), Deep Belief Nets (Hinton et.al.)

Mathematical Formulation

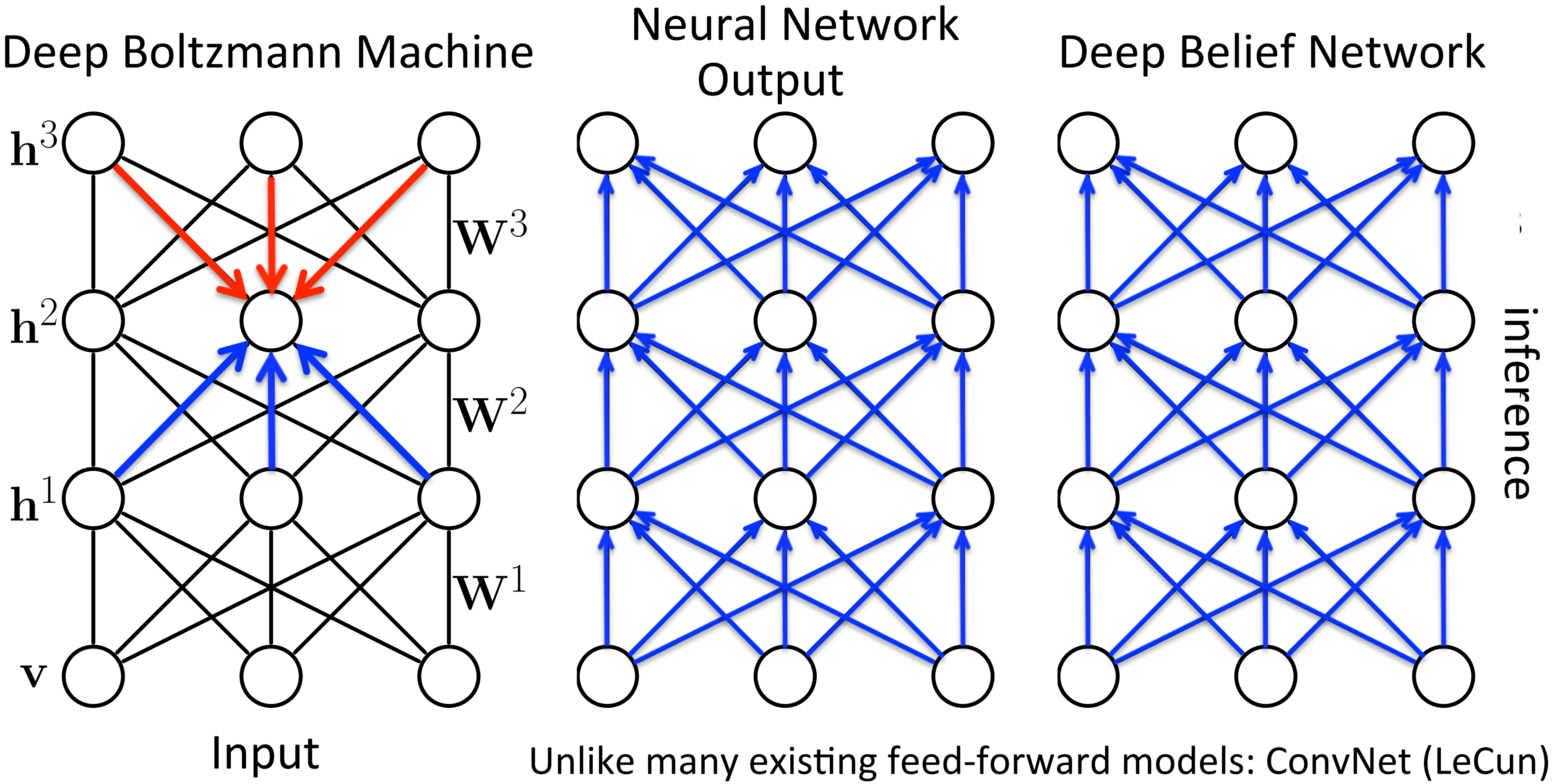
$$P_\theta(\mathbf{v}) = \frac{P^*(\mathbf{v})}{\mathcal{Z}(\theta)} = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}^1,\mathbf{h}^2,\mathbf{h}^3} \exp\left[\mathbf{v}^\top W^1 \mathbf{h}^1 + \mathbf{h}^{1\top} W^2 \mathbf{h}^2 + \mathbf{h}^{2\top} W^3 \mathbf{h}^3\right]$$



Unlike many existing feed-forward models: ConvNet (LeCun), HMAX (Poggio), Deep Belief Nets (Hinton)

Mathematical Formulation

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

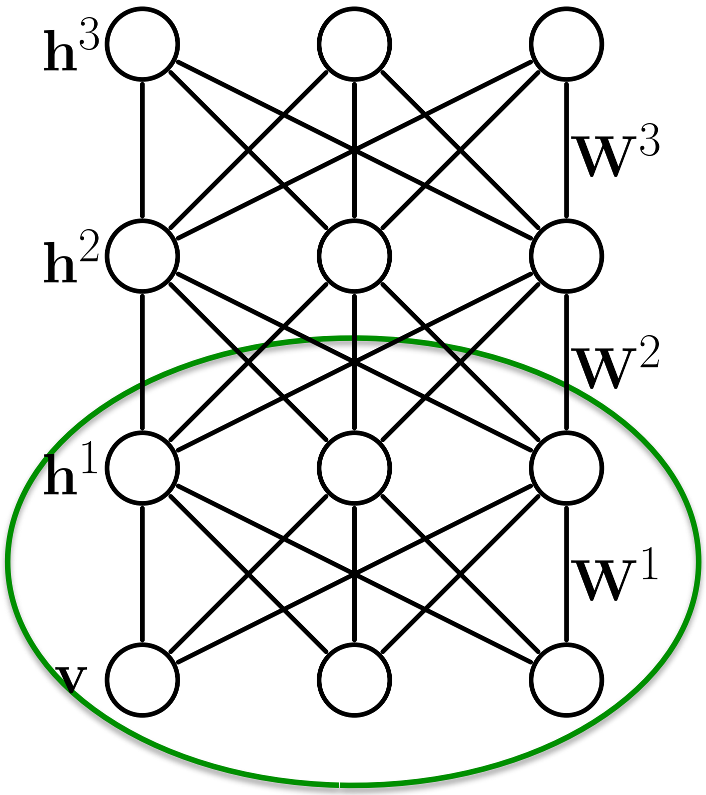


Unlike many existing feed-forward models: ConvNet (LeCun), HMAX (Poggio), Deep Belief Nets (Hinton)

Mathematical Formulation

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

Deep Boltzmann Machine



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

- Dependencies between hidden variables.

Maximum likelihood learning:

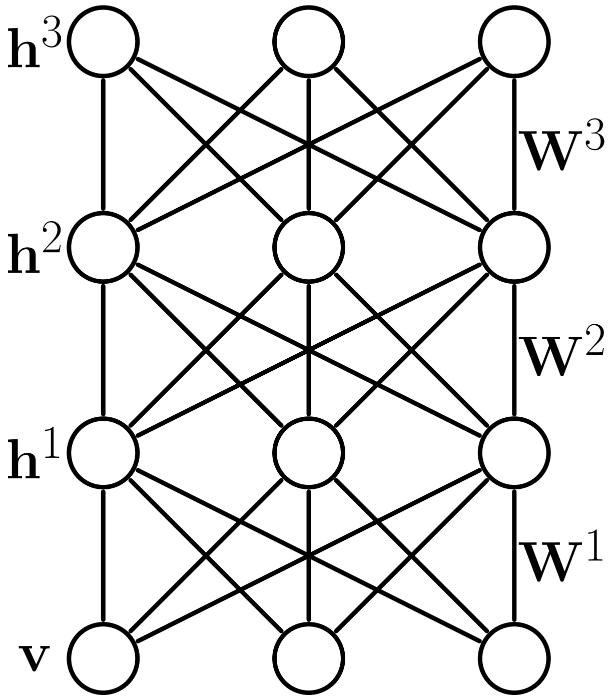
$$\frac{\partial \log P_\theta(\mathbf{v})}{\partial W^1} = \mathrm{E}_{P_{data}}[\mathbf{v}{\mathbf{h}^1}^\top] - \mathrm{E}_{P_\theta}[\mathbf{v}{\mathbf{h}^1}^\top]$$

Problem: Both expectations are intractable!

Learning rule for undirected graphical models: MRFs, CRFs, Factor graphs.

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:

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

- Both expectations are intractable!

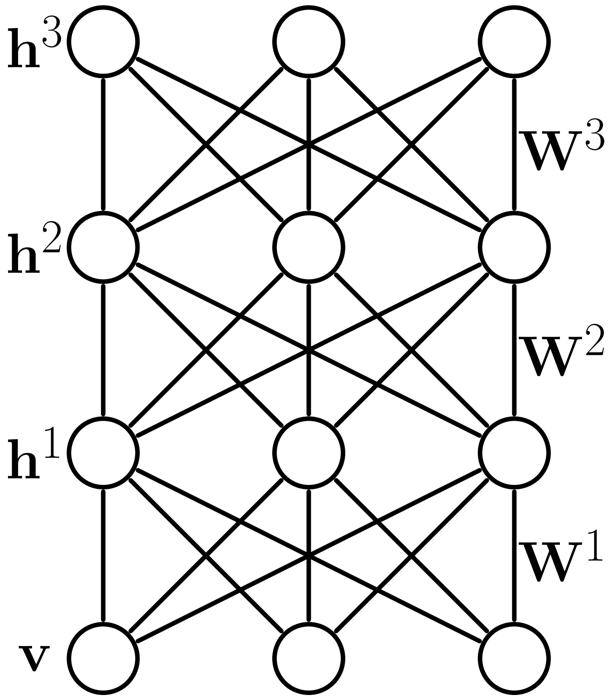
$$P_{data}(\mathbf{v}, \mathbf{h}^{\mathbf{1}}) = P_\theta(\mathbf{h}^{\mathbf{1}}|\mathbf{v}) P_{data}(\mathbf{v})$$

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

Not factorial any more!

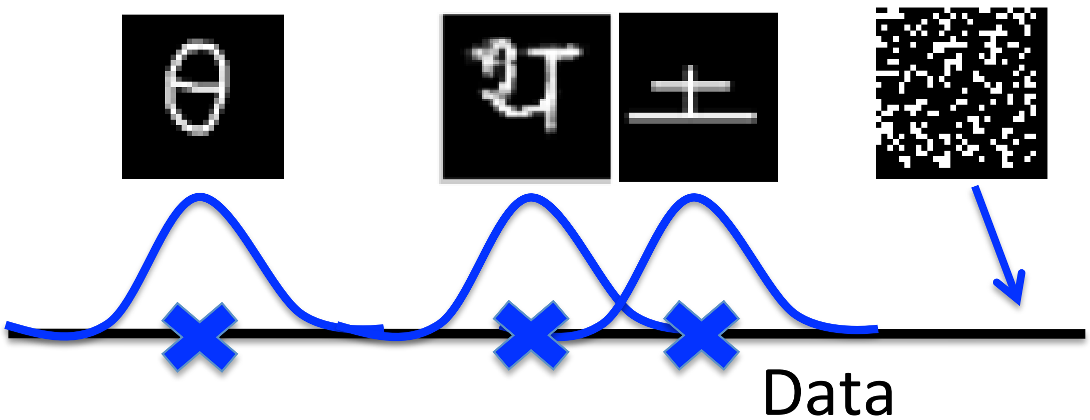
Approximate Learning

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



(Approximate) Maximum Likelihood:

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



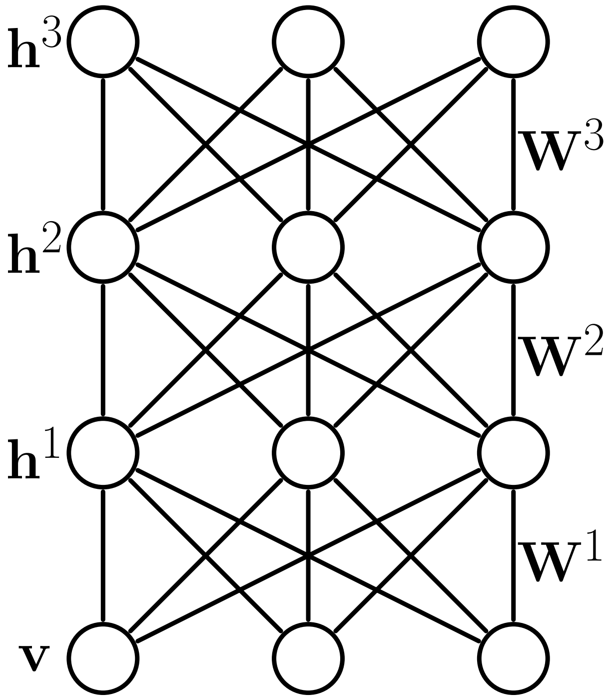
$$P_{data}(\mathbf{v},\mathbf{h}^{\mathbf{1}}) = P_\theta(\mathbf{h}^{\mathbf{1}}|\mathbf{v})P_{data}(\mathbf{v})$$

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

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!

Previous Work

Many approaches for learning Boltzmann machines have been proposed over the last 20 years:

- Hinton and Sejnowski (1983),
- Peterson and Anderson (1987)
- Galland (1991)
- Kappen and Rodriguez (1998)
- Lawrence, Bishop, and Jordan (1998)
- Tanaka (1998)
- Welling and Hinton (2002)
- Zhu and Liu (2002)
- Welling and Teh (2003)
- Yasuda and Tanaka (2009)

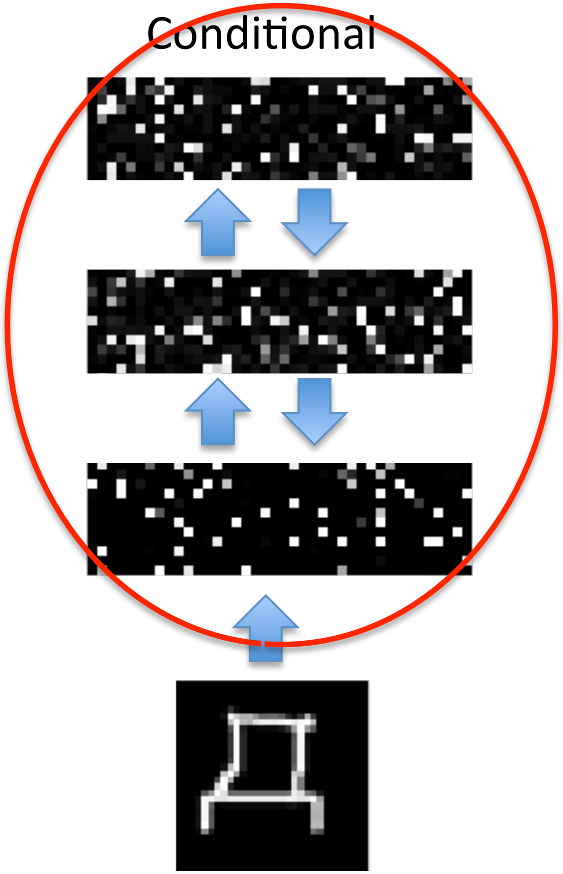
Real-world applications – thousands of hidden and observed variables with millions of parameters.

Many of the previous approaches were not successful for learning general Boltzmann machines with **hidden variables.**

Algorithms based on Contrastive Divergence, Score Matching, Pseudo-Likelihood, Composite Likelihood, MCMC-MLE, Piecewise Learning, cannot handle multiple layers of hidden variables.

New Learning Algorithm

Posterior Inference



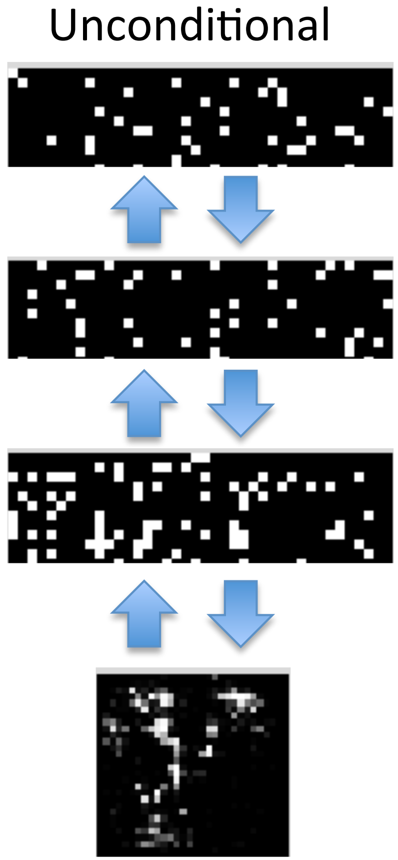
Approximate conditional

$$P_{data}(\mathbf{h}|\mathbf{v})$$

Simulate from the Model

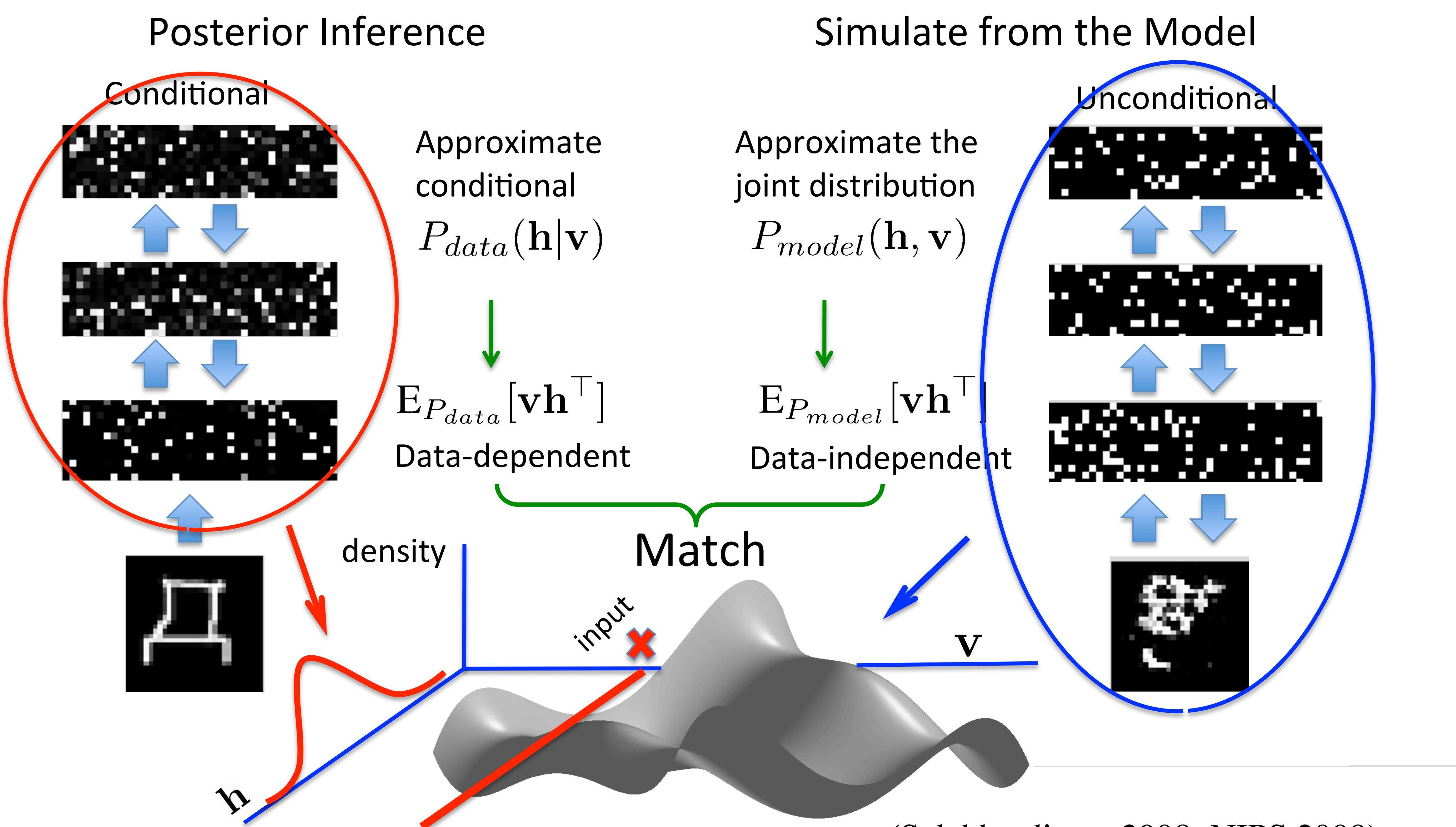
Approximate the joint distribution

$$P_{model}(\mathbf{h},\mathbf{v})$$



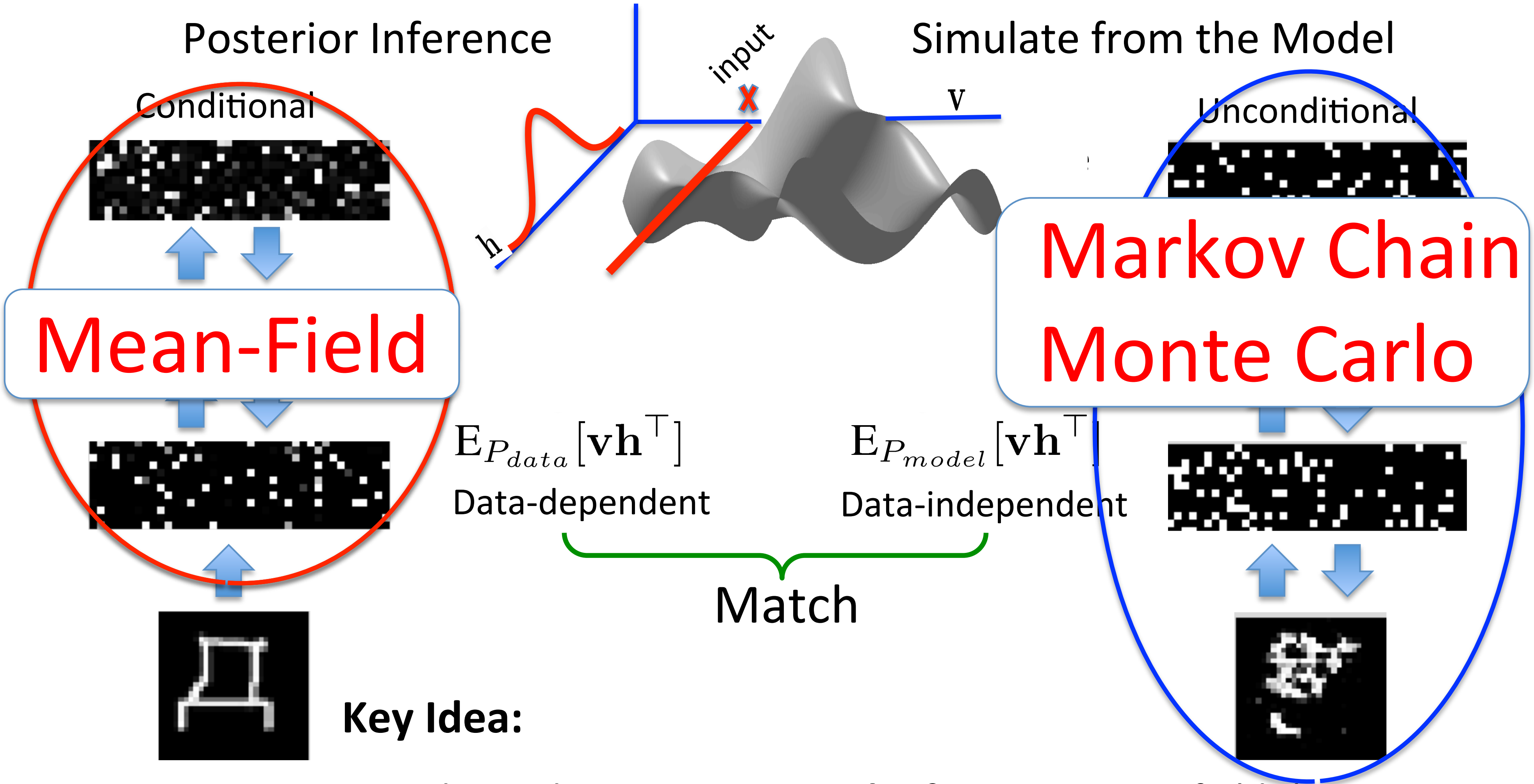
(Salakhutdinov, 2008; NIPS 2009)

New Learning Algorithm



(Salakhutdinov, 2008; NIPS 2009)

New Learning Algorithm

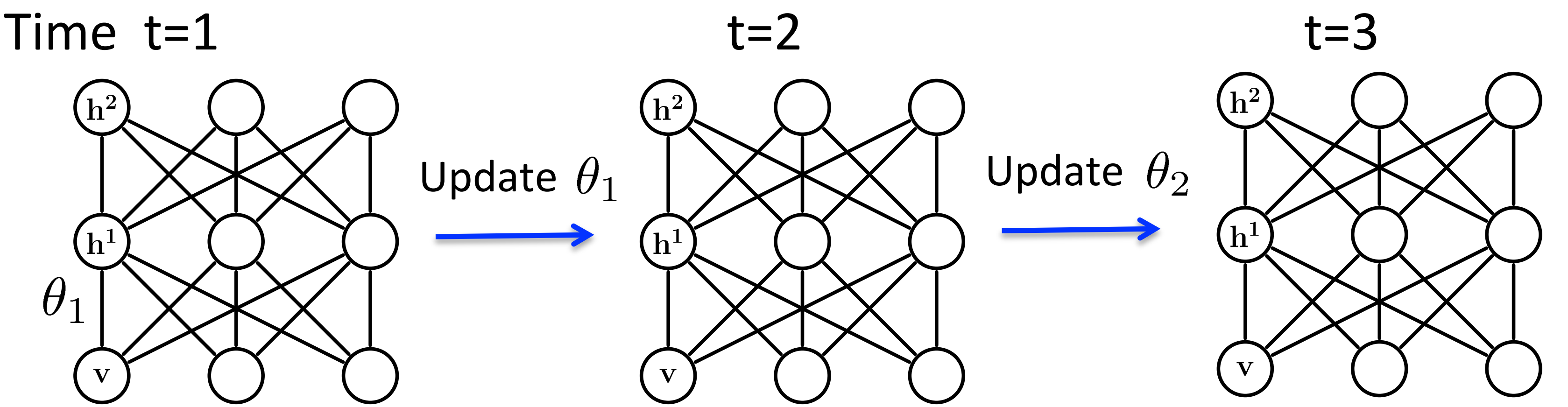


Key Idea:

Data-dependent: **Variational Inference**, mean-field theory

Data-independent: **Stochastic Approximation**, MCMC based

Stochastic Approximation



$\mathbf{x}_1 \sim \mathrm{T}_{\theta_1}(\mathbf{x}_1 \leftarrow \mathbf{x}_0)$ $\quad$ $\mathbf{x}_2 \sim \mathrm{T}_{\theta_2}(\mathbf{x}_2 \leftarrow \mathbf{x}_1)$ $\quad$ $\mathbf{x}_3 \sim \mathrm{T}_{\theta_3}(\mathbf{x}_3 \leftarrow \mathbf{x}_2)$

Update θ_t and $\mathbf{x}_t$ sequentially, where $\mathbf{x} = \{\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2\}$

- Generate $\mathbf{x}_t \sim \mathrm{T}_{\theta_t}(\mathbf{x}_t \leftarrow \mathbf{x}_{t-1})$ by simulating from a Markov chain that leaves P_{θ_t} invariant (e.g. Gibbs or M-H sampler)
- Update θ_t by replacing intractable $\mathrm{E}_{P_{\theta_t}}[\mathbf{v}\mathbf{h}^\top]$ with a point estimate $[\mathbf{v}_t\mathbf{h}_t^\top]$

In practice we simulate several Markov chains in parallel.

(Robbins and Monro, Ann. Math. Stats, 1957;
L. Younes, Probability Theory 1989)

Learning Algorithm

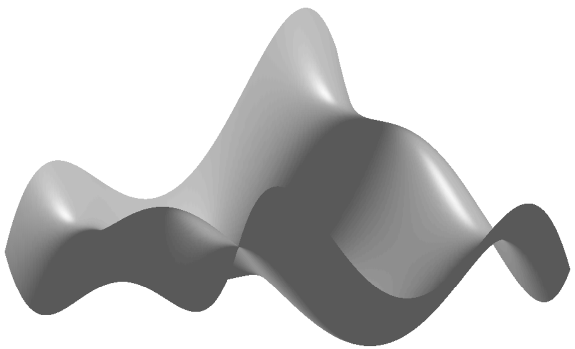
Update rule decomposes:

$$\theta_{t+1} = \theta_t + \alpha_t \Big(\mathbb{E}_{P_{data}}[\mathbf{v}\mathbf{h}^\top] - \frac{1}{M}\sum_{m=1}^{M} \mathbf{v}_t^{(m)} \mathbf{h}_t^{(m)\top} \Big)_{P_{\theta_t}}[\mathbf{v}\mathbf{h}^\top]$$

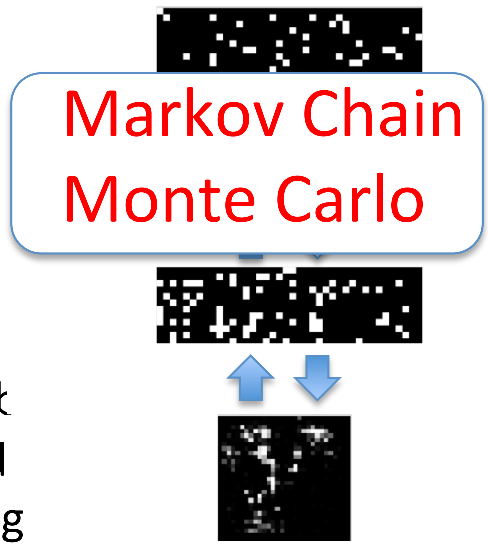
True gradient

Perturbation term ϵ_t

Almost sure convergence guarantees as learning rate $\alpha_t \to 0$



Problem: High-dimensional data: the probability landscape is highly multimodal.



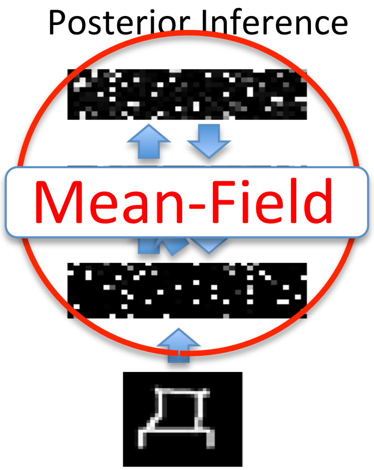
Key insight: The transition operator can
any valid transition operator – Tempered
Transitions, Parallel/Simulated Tempering

Connections to the theory of stochastic approximation and adaptive MCMC.

Variational Inference

Approximate intractable distribution $P_\theta(\mathbf{h}|\mathbf{v})$ with simpler, tractable distribution $Q_\mu(\mathbf{h}|\mathbf{v})$:

$$\log P_\theta(\mathbf{v}) = \log \sum_{\mathbf{h}} P_\theta(\mathbf{h},\mathbf{v}) = \log \sum_{\mathbf{h}} Q_\mu(\mathbf{h}|\mathbf{v}) \frac{P_\theta(\mathbf{h},\mathbf{v})}{Q_\mu(\mathbf{h}|\mathbf{v})}$$



$$\geq \sum_{\mathbf{h}} Q_\mu(\mathbf{h}|\mathbf{v}) \log \frac{P_\theta(\mathbf{h},\mathbf{v})}{Q_\mu(\mathbf{h}|\mathbf{v})}$$

$$= \underbrace{\sum_{\mathbf{h}} Q_\mu(\mathbf{h}|\mathbf{v}) \log \underbrace{P^*_\theta(\mathbf{h},\mathbf{v})}_{\mathbf{v}^\top W^1 \mathbf{h}^1 + \mathbf{h}^{1\top} W^2 \mathbf{h}^2 + \mathbf{h}^{2\top} W^3 \mathbf{h}^3} - \log \mathcal{Z}(\theta) + \sum_{\mathbf{h}} Q_\mu(\mathbf{h}|\mathbf{v}) \log}_{\text{Variational Lower Bound}} \frac{1}{Q_\mu(\mathbf{h}|\mathbf{v})}$$

$$= \log P_\theta(\mathbf{v}) - \mathrm{KL}\big(Q_\mu(\mathbf{h}|\mathbf{v}) || P_\theta(\mathbf{h}|\mathbf{v})\big)$$

$$\mathrm{KL}(Q||P) = \int Q(x) \log \frac{Q(x)}{P(x)} dx$$

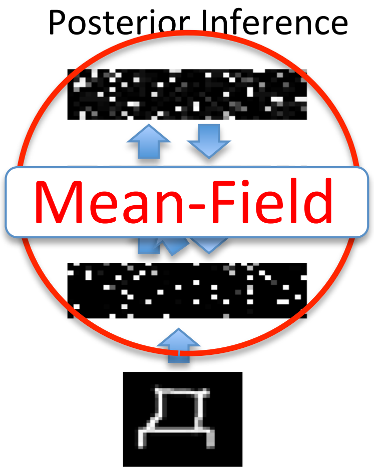
Minimize KL between approximating and true distributions with respect to variational parameters μ .

Variational Inference

Approximate intractable distribution $P_\theta(\mathbf{h}|\mathbf{v})$ with simpler, tractable distribution $Q_\mu(\mathbf{h}|\mathbf{v})$:

$$\mathrm{KL}(Q||P) = \int Q(x)\log\frac{Q(x)}{P(x)}dx$$

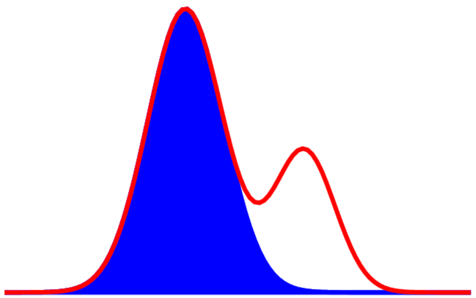
$$\log P_\theta(\mathbf{v}) \geq \underbrace{\log P_\theta(\mathbf{v}) - \mathrm{KL}\big(Q_\mu(\mathbf{h}|\mathbf{v})||P_\theta(\mathbf{h}|\mathbf{v})\big)}_{\text{Variational Lower Bound}}$$



Mean-Field: Choose a fully factorized distribution:

$$Q_\mu(\mathbf{h}|\mathbf{v}) = \prod_{j=1}^{F} q(h_j|\mathbf{v}) \text{ with } q(h_j = 1|\mathbf{v}) = \mu_j$$

Variational Inference: Maximize the lower bound w.r.t. Variational parameters μ .



Nonlinear fixed-point equations:

$$\mu_j^{(1)} = \sigma\left(\sum_i W_{ij}^1 v_i + \sum_k W_{jk}^2 \mu_k^{(2)}\right)$$

$$\mu_k^{(2)} = \sigma\left(\sum_j W_{jk}^2 \mu_j^{(1)} + \sum_m W_{km}^3 \mu_m^{(3)}\right)$$

$$\mu_m^{(3)} = \sigma\left(\sum_k W_{km}^3 \mu_k^{(2)}\right)$$

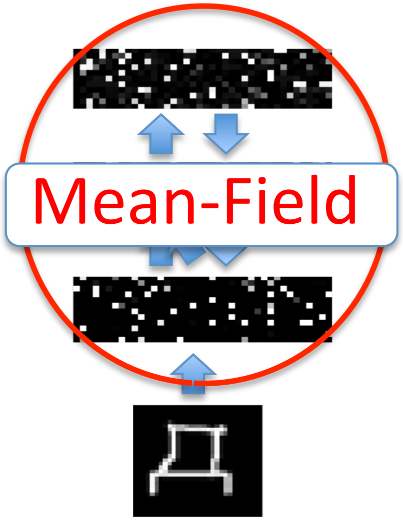
Variational Inference

Approximate intractable distribution $P_\theta(\mathbf{h}|\mathbf{v})$ with simpler, tractable distribution $Q_\mu(\mathbf{h}|\mathbf{v})$:

$$\mathrm{KL}(Q||P) = \int Q(x) \log \frac{Q(x)}{P(x)} dx$$

$$\log P_\theta(\mathbf{v}) \geq \underbrace{\log P_\theta(\mathbf{v}) - \mathrm{KL}\big(Q_\mu(\mathbf{h}|\mathbf{v})||P_\theta(\mathbf{h}|\mathbf{v})\big)}_{\text{Variational Lower Bound}}$$

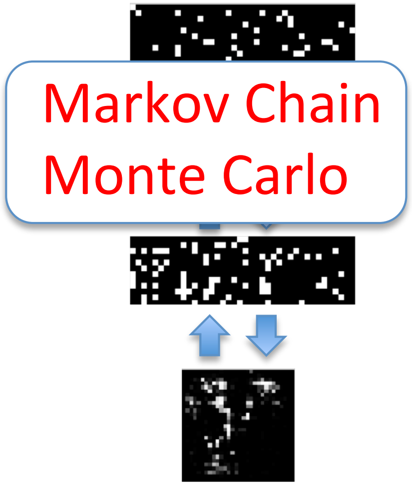
Posterior Inference



1. Variational Inference: Maximize the lower bound w.r.t. variational parameters

2. MCMC: Apply stochastic approximation to update model parameters

Unconditional Simulation



Almost sure convergence guarantees to an asymptotically stable point.

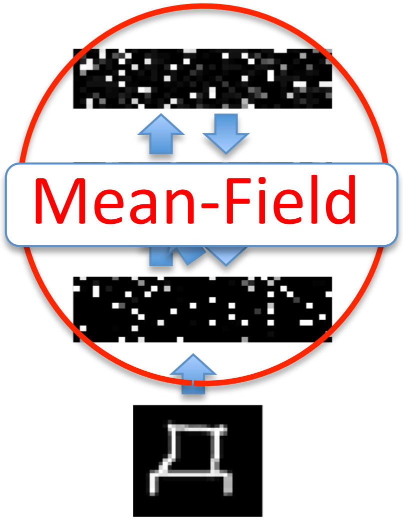
Variational Inference

Approximate intractable distribution $P_\theta(\mathbf{h}|\mathbf{v})$ with simpler, tractable distribution $Q_\mu(\mathbf{h}|\mathbf{v})$:

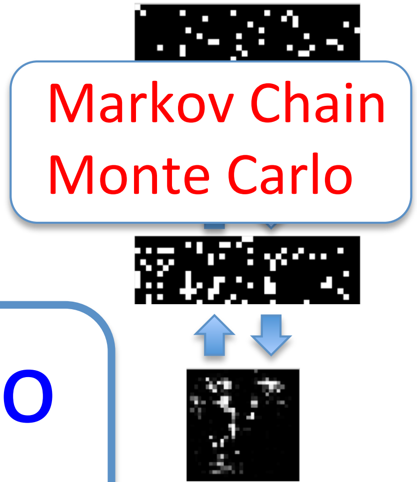
$$\mathrm{KL}(Q||P) = \int Q(x) \log \frac{Q(x)}{P(x)} dx$$

$$\log P_\theta(\mathbf{v}) \geq \underbrace{\log P_\theta(\mathbf{v}) - \mathrm{KL}(Q_\mu(\mathbf{h}|\mathbf{v})||P_\theta(\mathbf{h}|\mathbf{v}))}_{\text{Variational Lower Bound}}$$

Posterior Inference



Unconditional Simulation



1. V ... wer
bou

Fast Inference

2. M
to u

Learning can scale to millions of examples

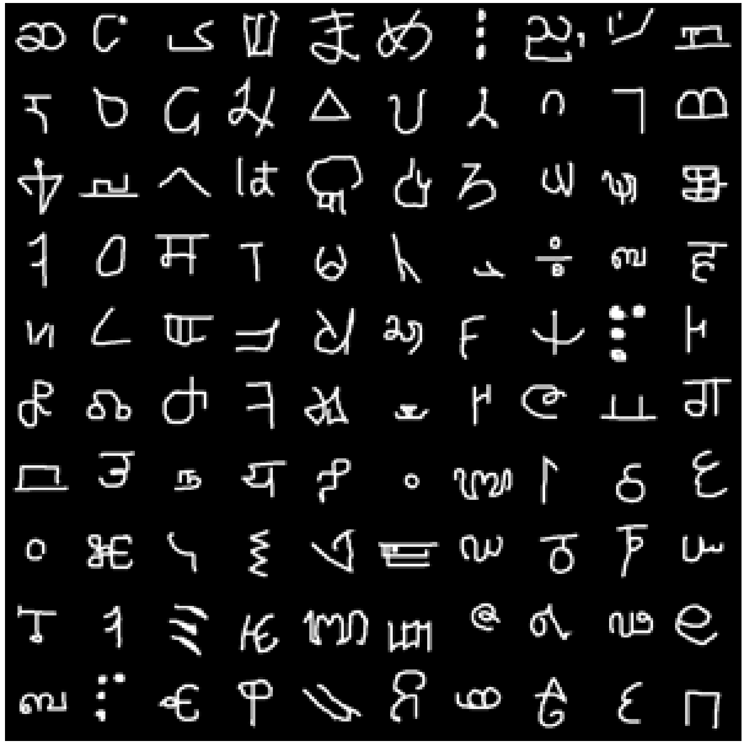
Almost sure convergence guarantees to an asymptotically stable point.

Good Generative Model?

Handwritten Characters

Good Generative Model?

Handwritten Characters



Good Generative Model?

Handwritten Characters

Simulated

Real Data

Good Generative Model?

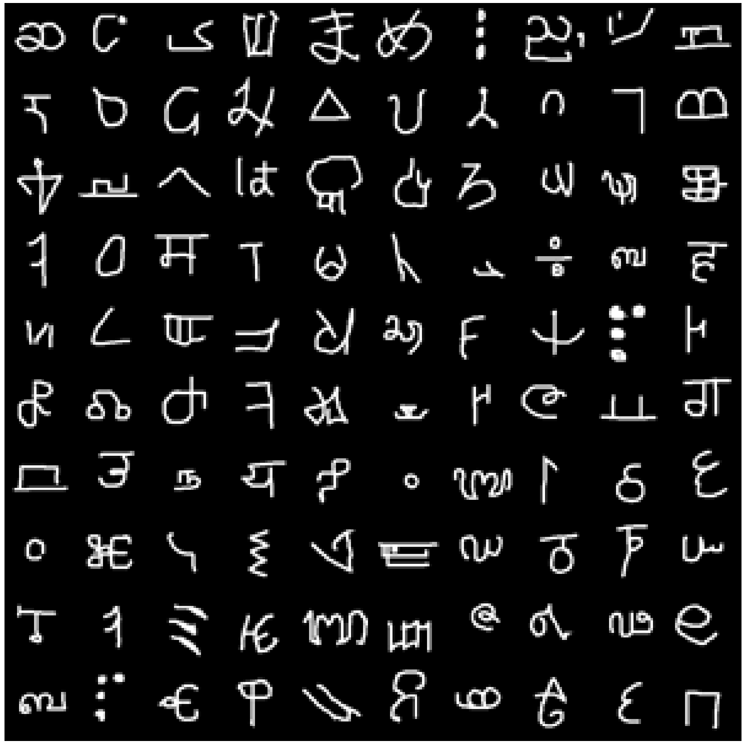
Handwritten Characters

Real Data

Simulated

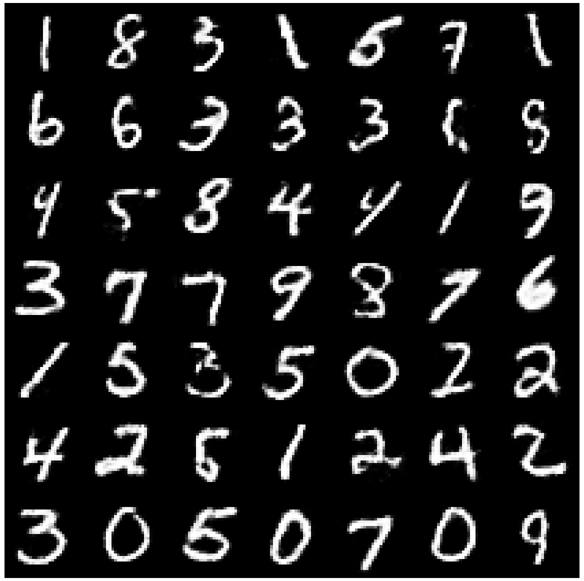
Good Generative Model?

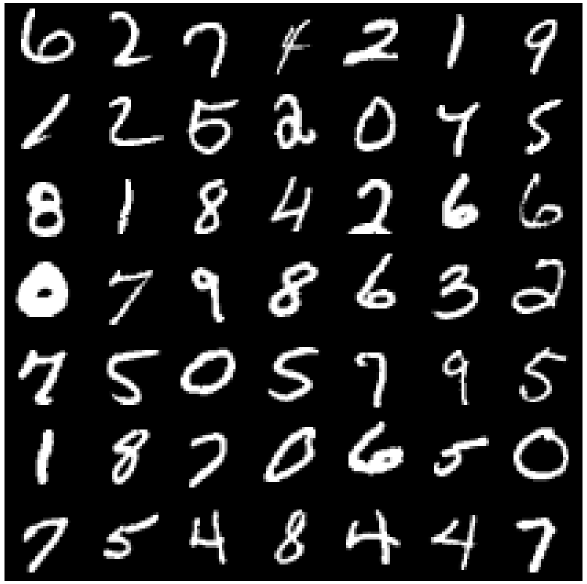
Handwritten Characters



Good Generative Model?

MNIST Handwritten Digit Dataset





Handwriting Recognition

MNIST Dataset
60,000 examples of 10 digits

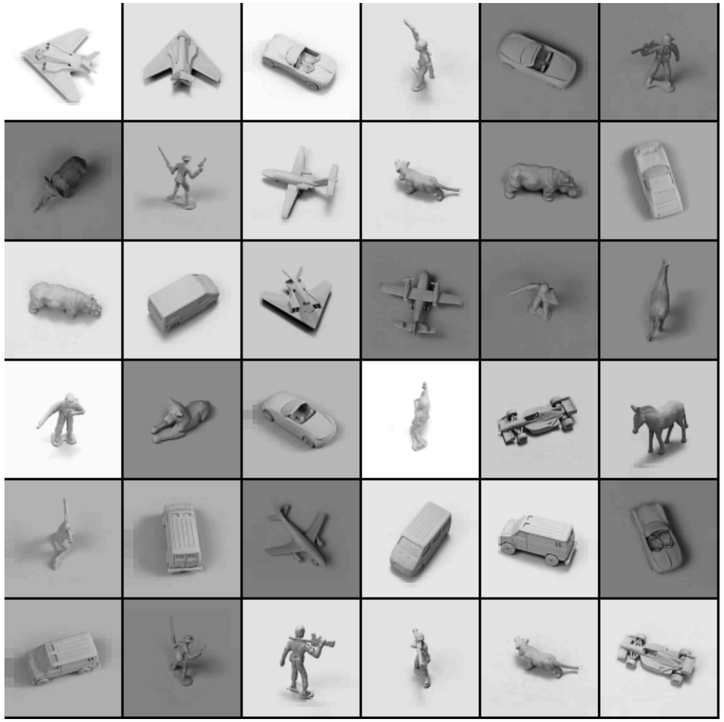
Learning Algorithm	Error
Logistic regression	12.0%
K-NN	3.09%
Neural Net (Platt 2005)	1.53%
SVM (Decoste et.al. 2002)	1.40%
Deep Autoencoder (Bengio et. al. 2007)	1.40%
Deep Belief Net (Hinton et. al. 2006)	1.20%
DBM	**0.95%**

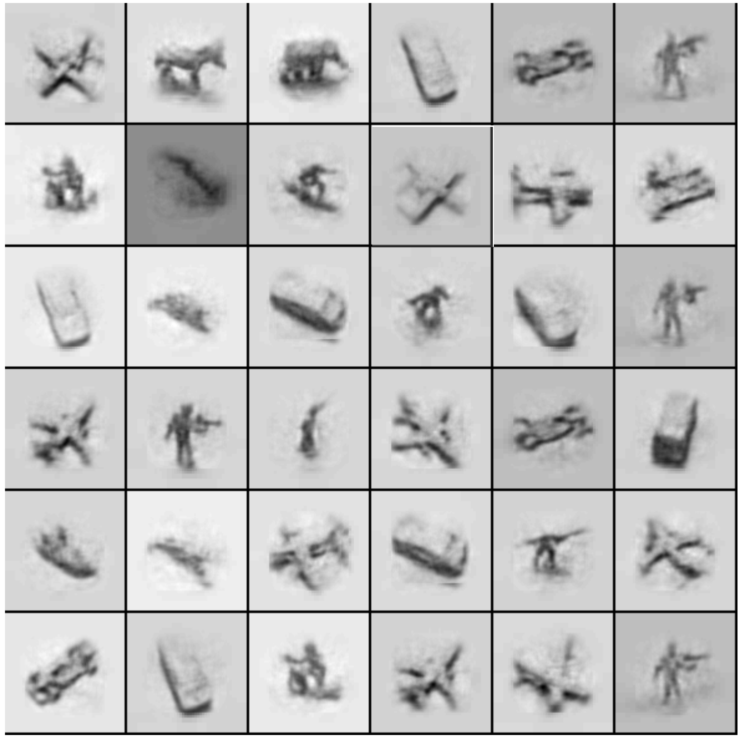
Optical Character Recognition
42,152 examples of 26 English letters

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



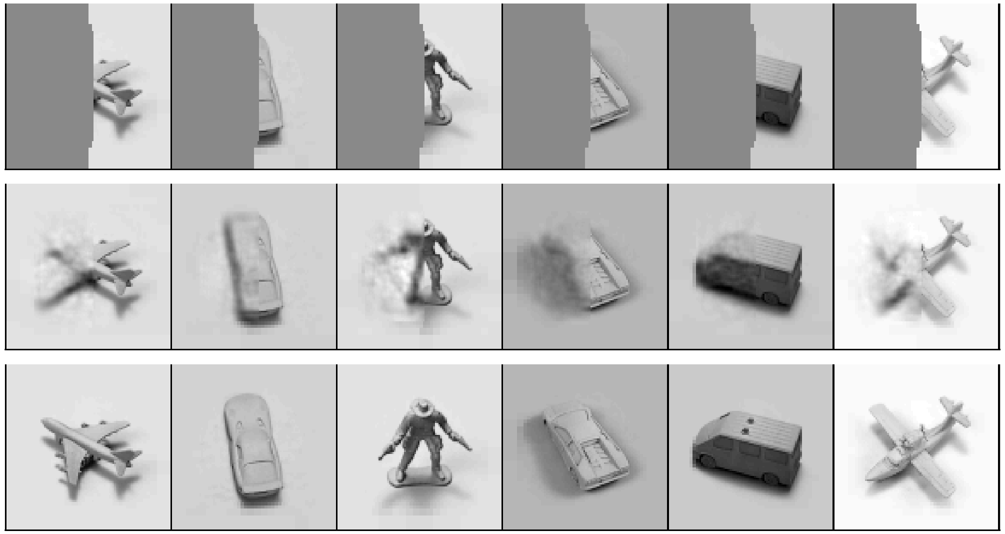


24,000 examples, 5 object categories, 5 different objects within each category, 6 lightning conditions, 9 elevations, 18 azimuths.

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

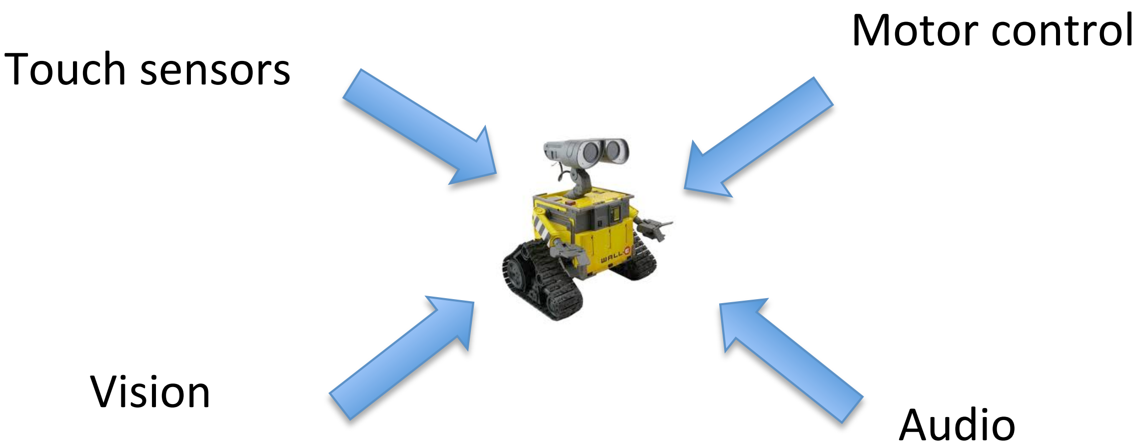
- Multimedia content on the web - image + text + audio.

car, automobile

sunset, pacificocean, bakerbeach, seashore, ocean

- Product recommendation systems.

- Robotics applications.



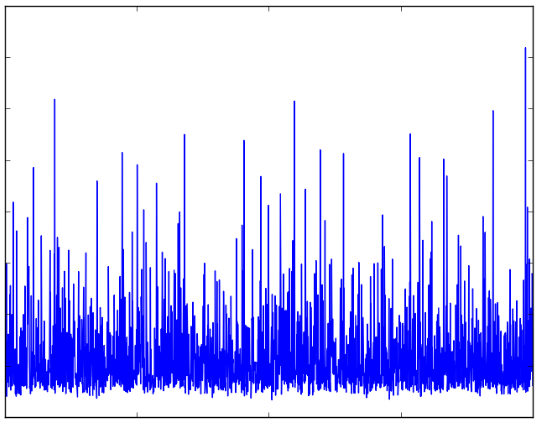
Challenges - I

Image

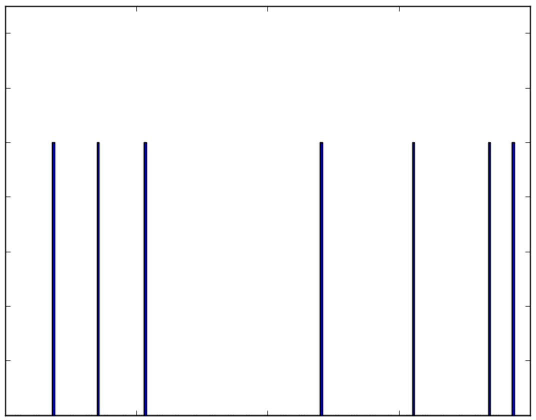
Text

sunset, pacific ocean, baker beach, seashore, ocean

Dense



Sparse



Very different input representations

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

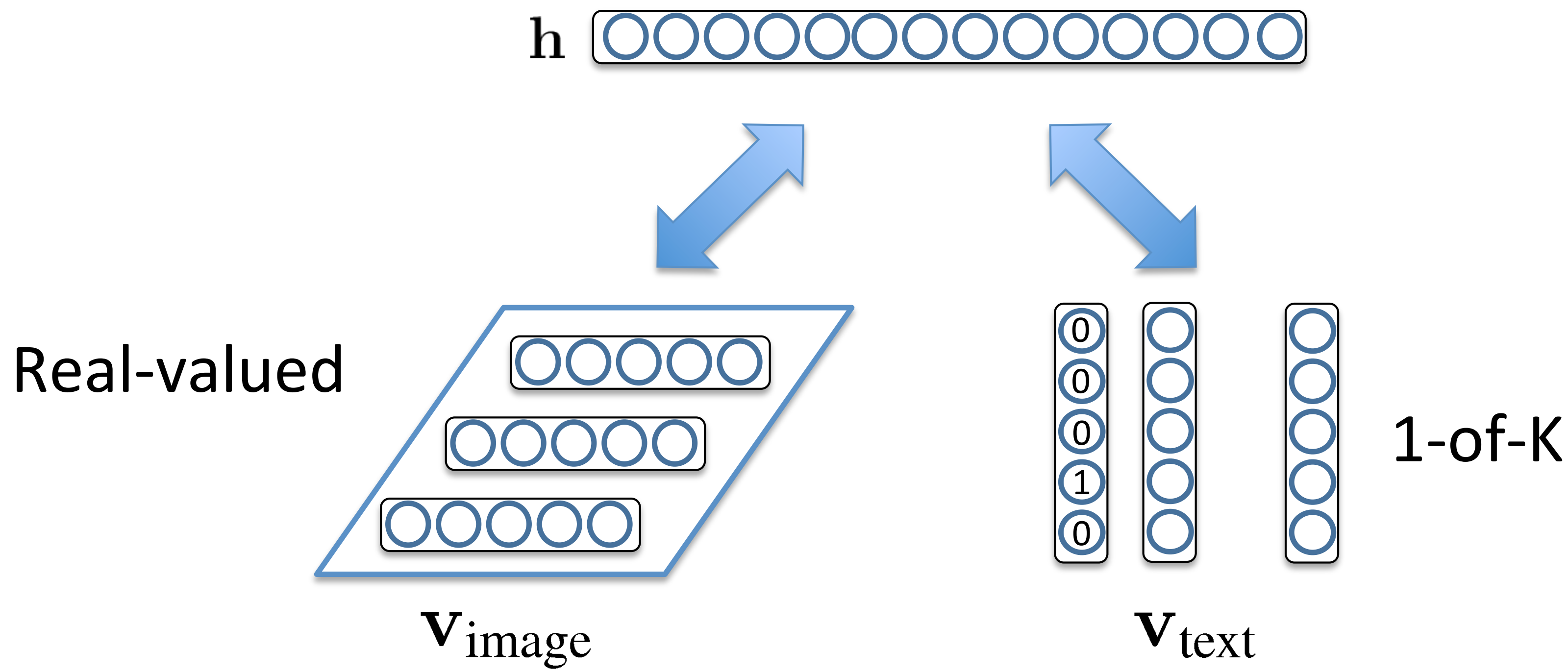
Noisy and missing data

Challenges - II

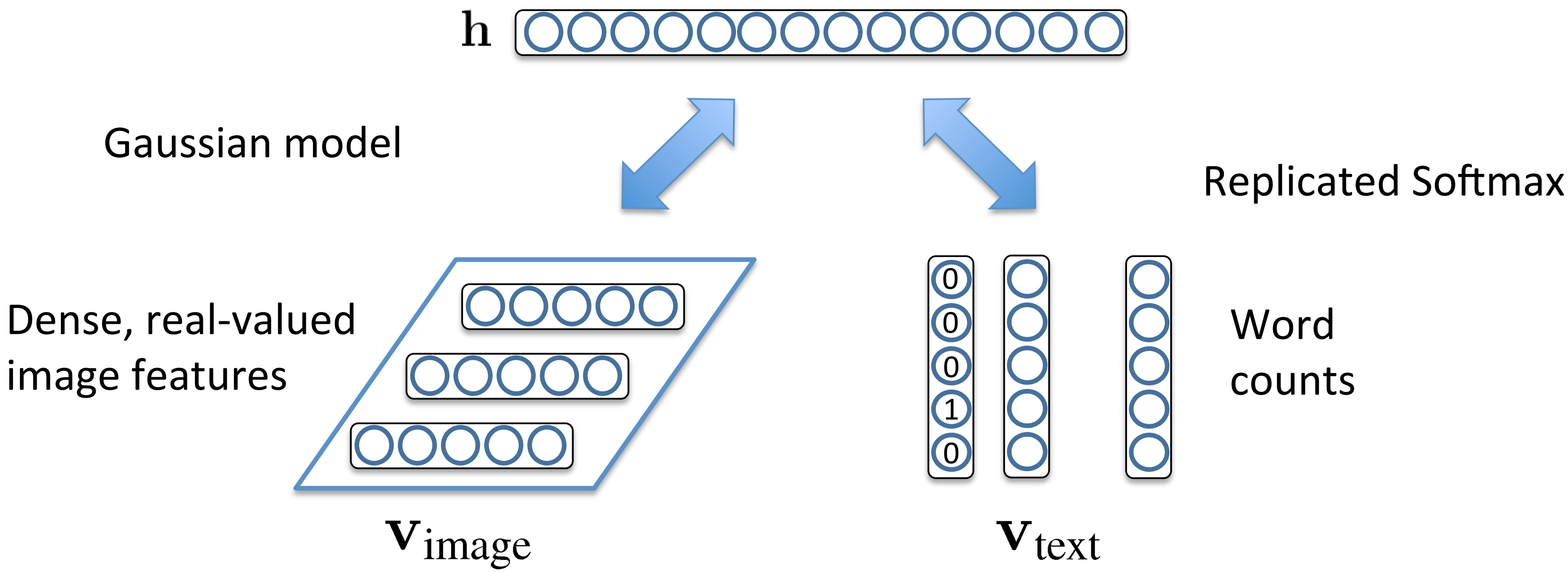
Image	Text	Text generated by the model
	pentax, k10d, pentaxda50200, kangarooisland, sa, australiansealion	beach, sea, surf, strand, shore, wave, seascape, sand, ocean, waves
	mickikrimmel, mickipedia, headshot	portrait, girl, woman, lady, blonde, pretty, gorgeous, expression, model
	< no text>	night, 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.

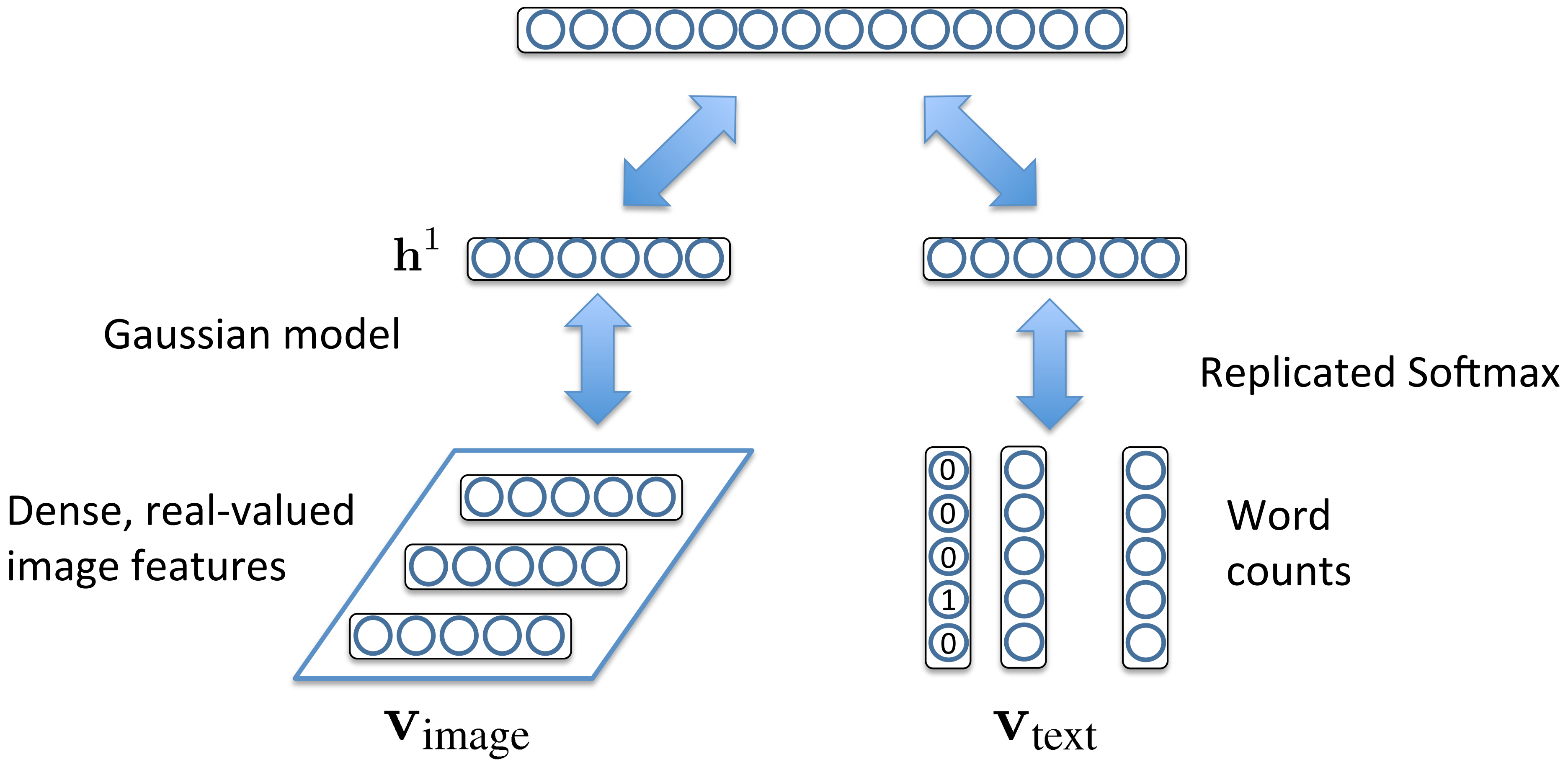


Multimodal DBM



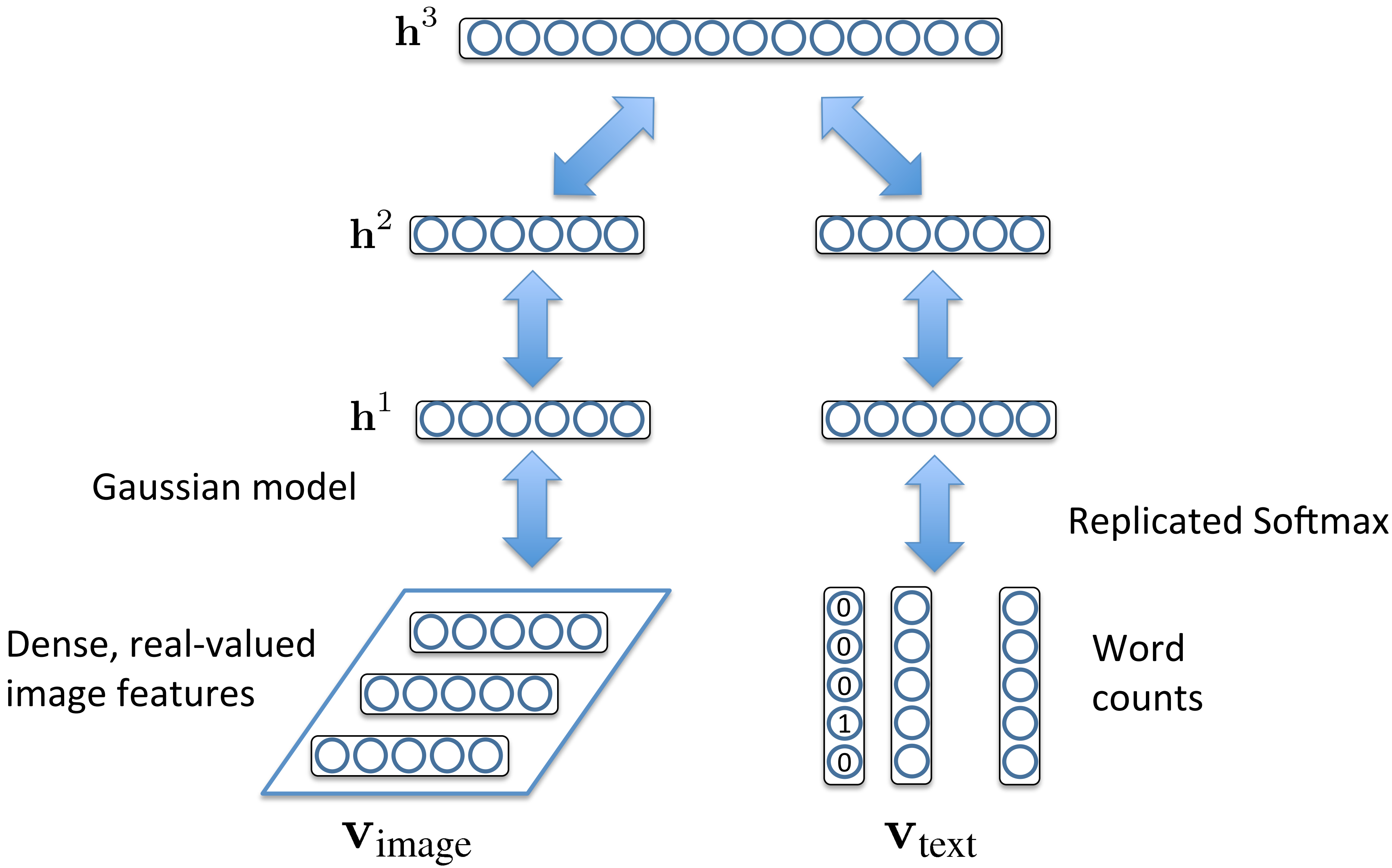
(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)

Multimodal DBM



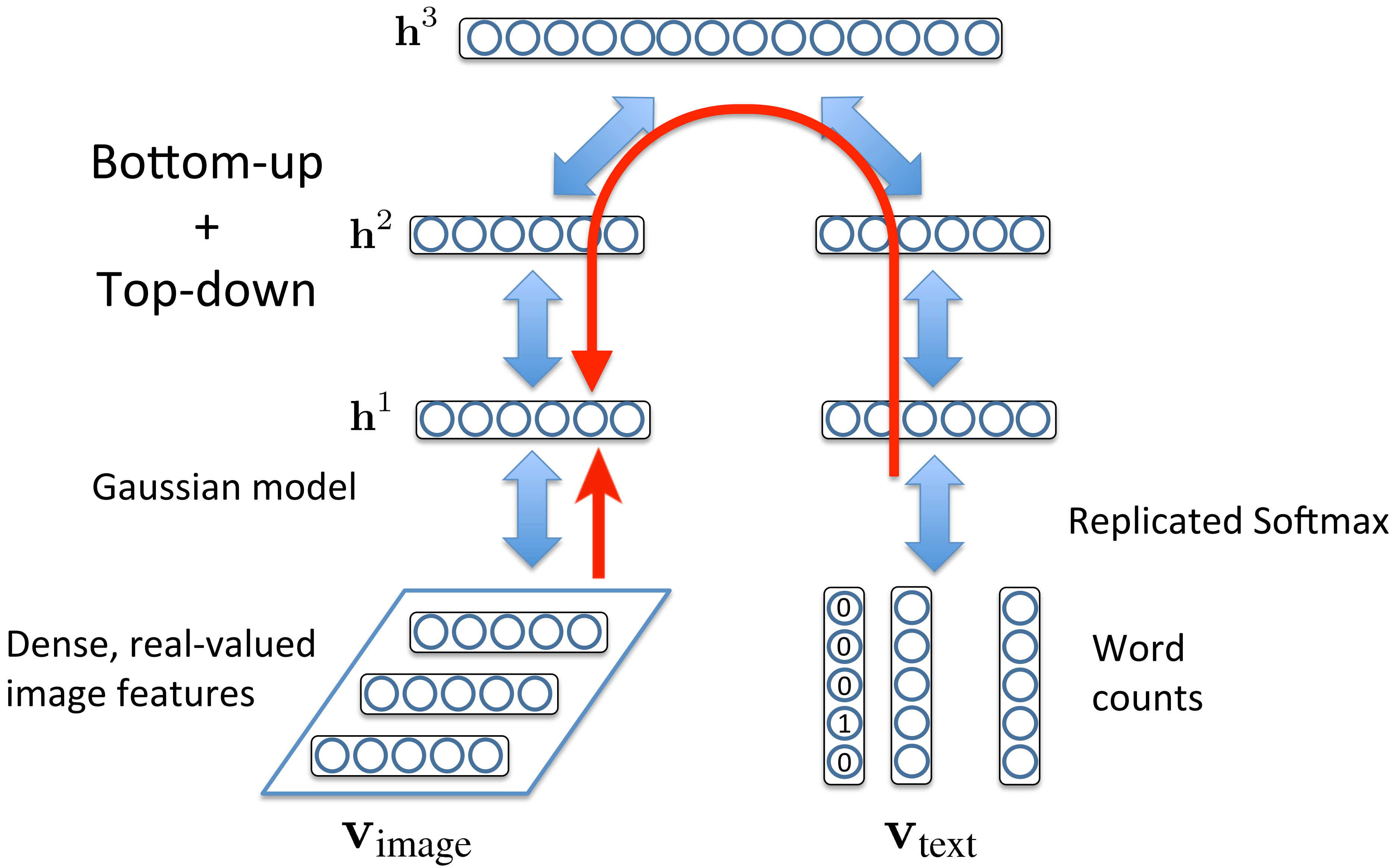
(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)

Multimodal DBM



(Srivastava & Salakhutdinov, NIPS 2012, JMLR 2014)

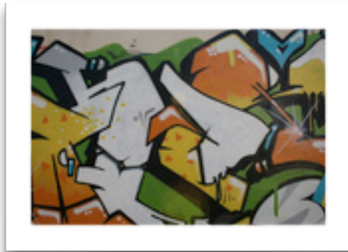
Multimodal DBM



(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

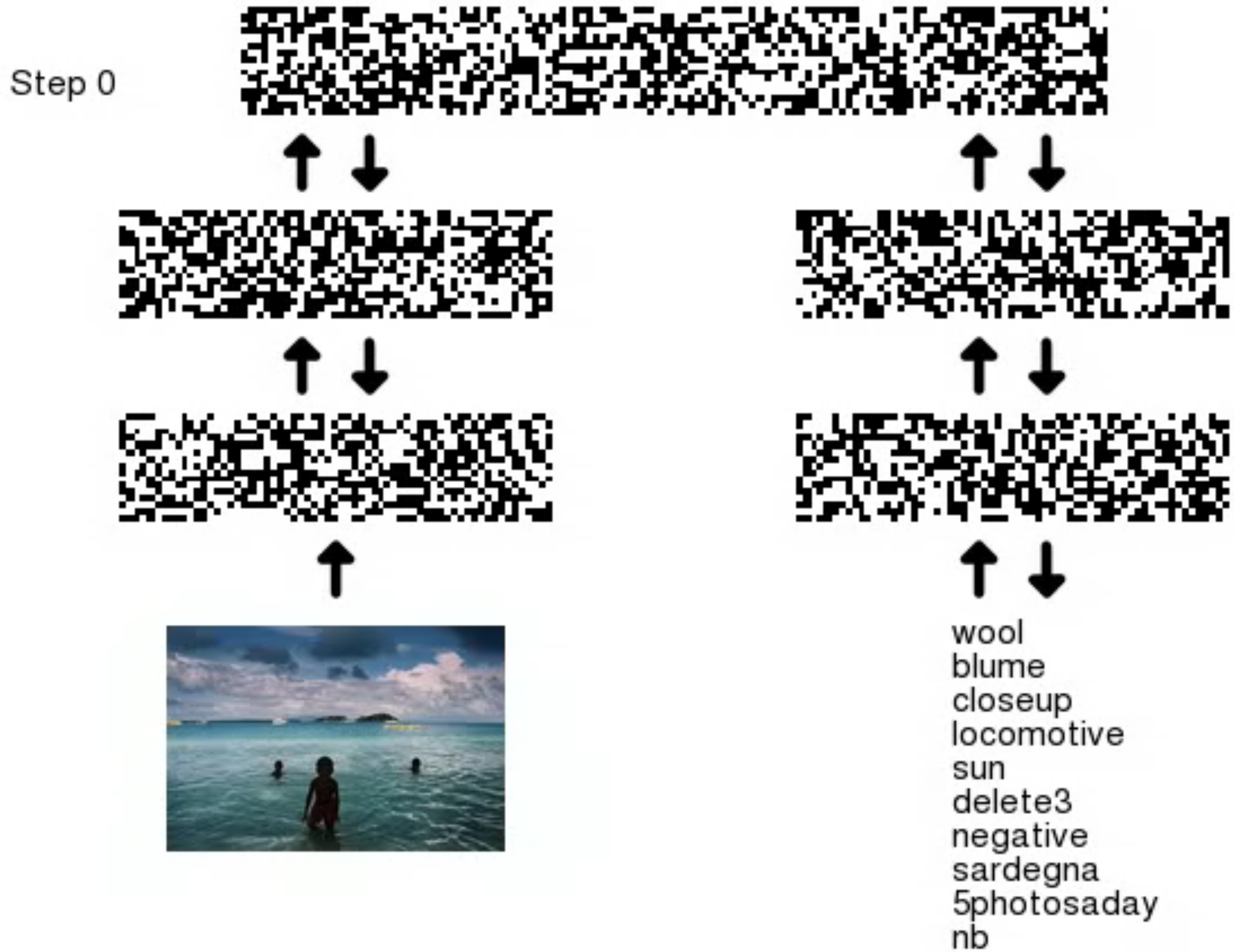
Given | Generated

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

Generating Text from Images



Samples drawn after every 50 steps of Gibbs updates

Sample at step 0
wool
blume
closeup
locomotive
sun
delete3
negative
sardegna
5photosaday
nb

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., 2010)

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

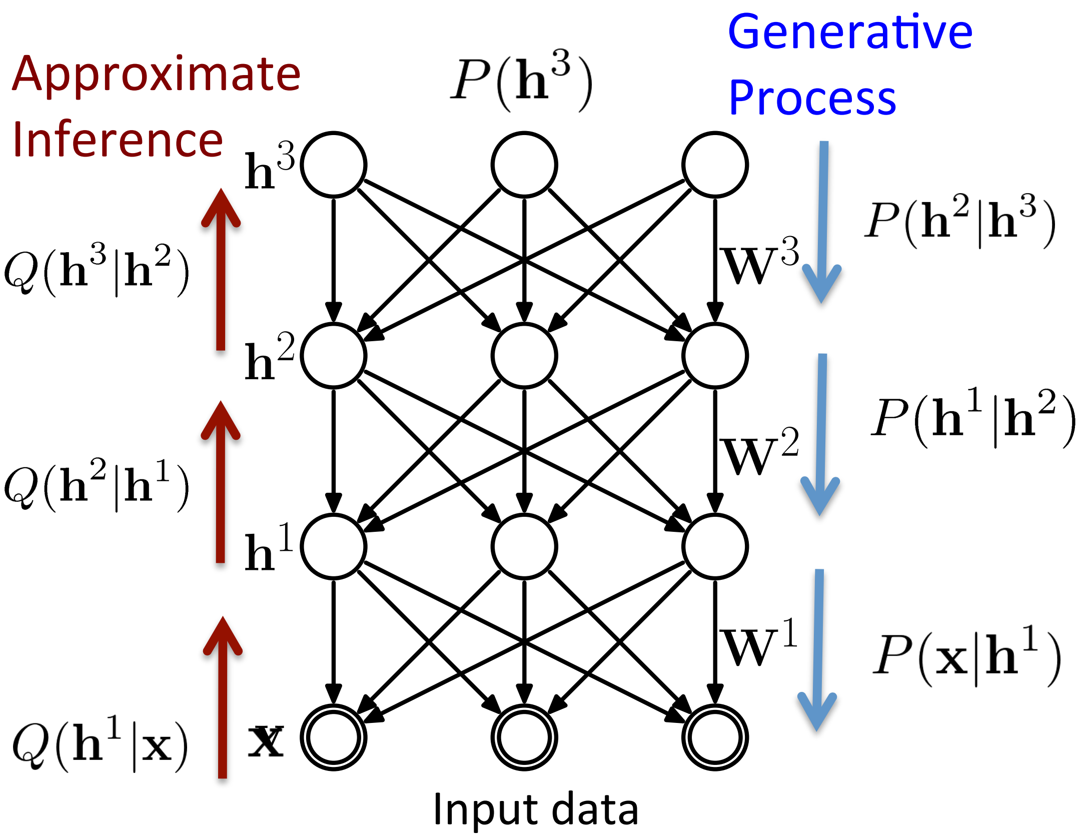
+ 1 Million unlabelled

Talk Roadmap

- Basic Building Blocks:
  - ➢ Sparse Coding
  - ➢ Autoencoders
- Deep Generative Models
  - ➢ Restricted Boltzmann Machines
  - ➢ Deep Belief Network, Deep Boltzmann Machines
  - ➢ Helmholtz Machines / Variational Autoencoders
- Generative Adversarial Networks
- Model Evaluation

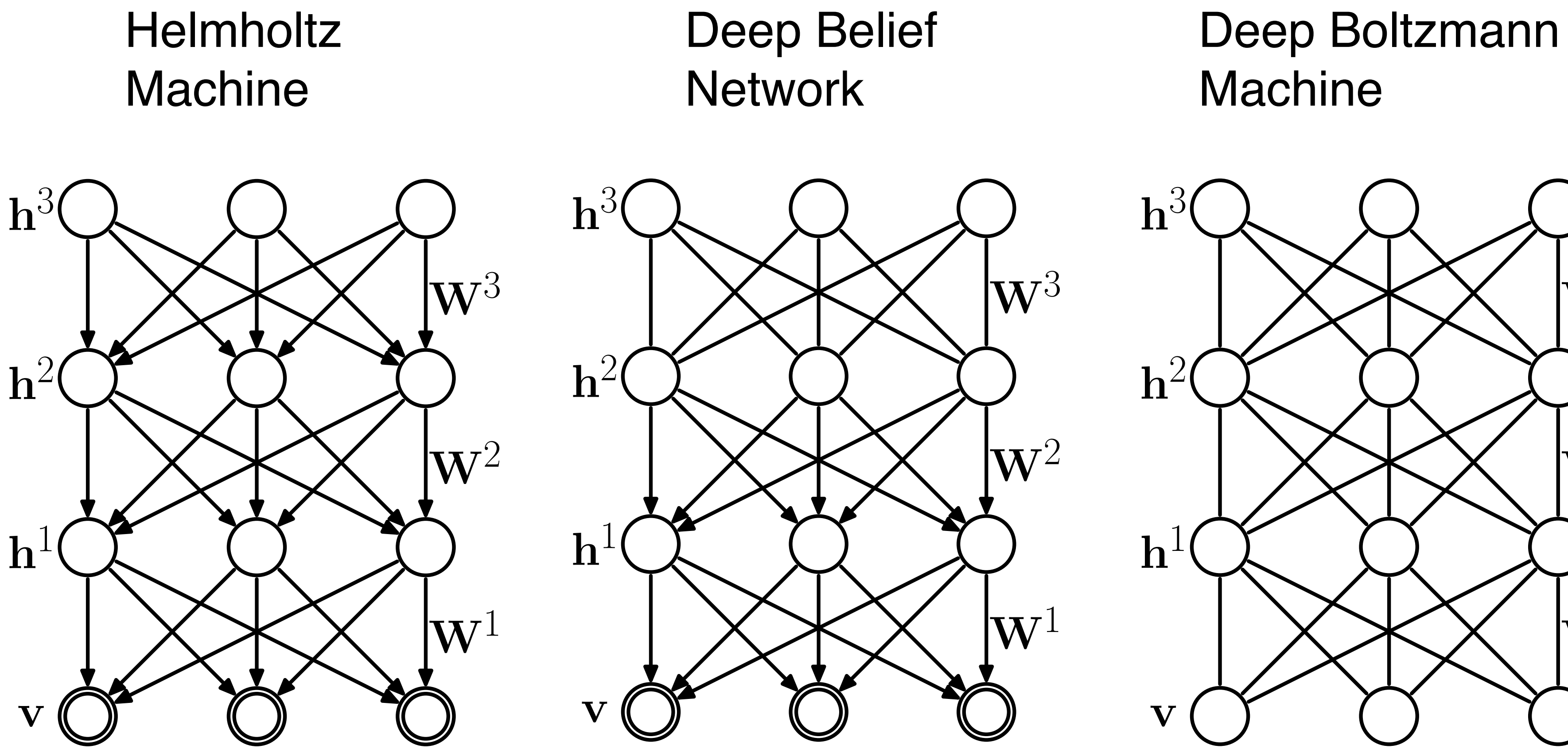
Helmholtz Machines

- 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, DBNs, DBMs

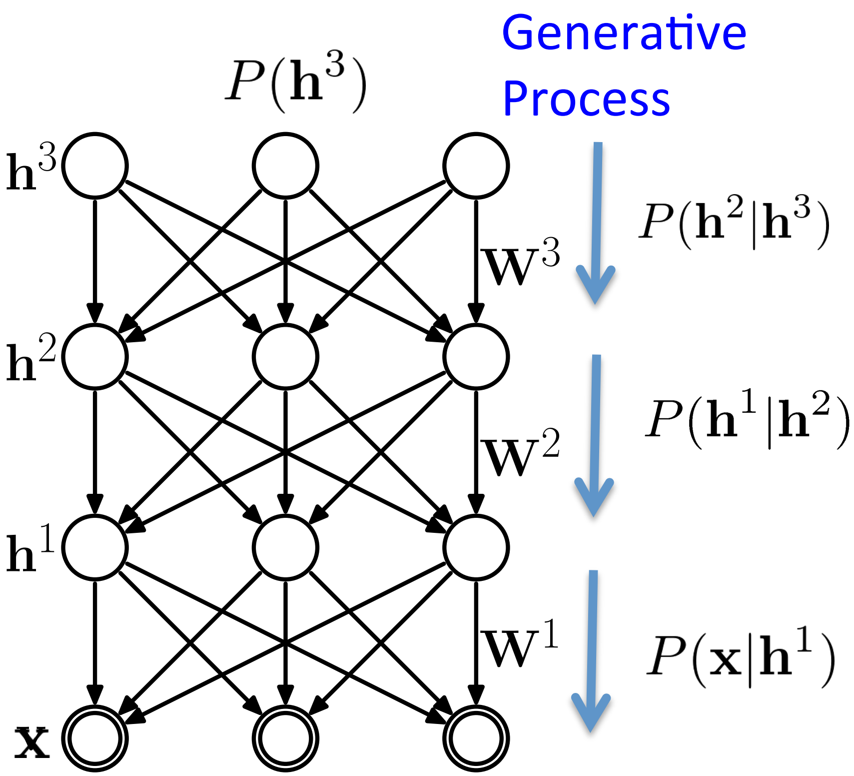


Variational Autoencoders (VAEs)

- 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



Input data

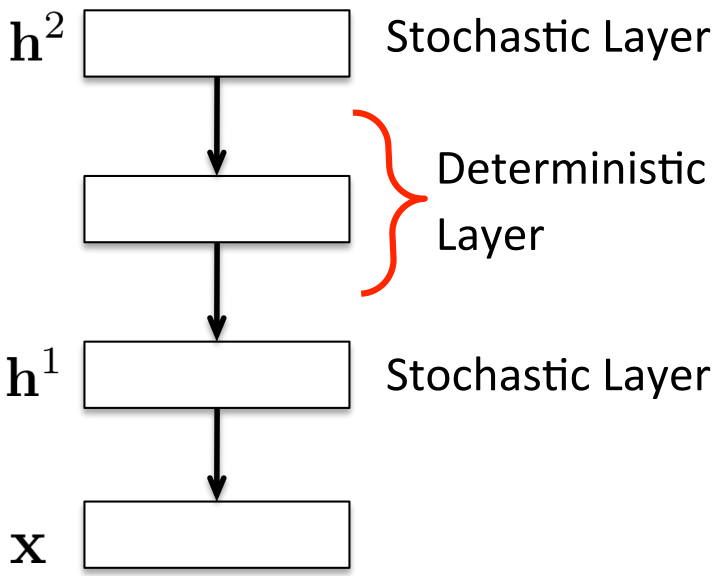
- $\boldsymbol{\theta}$ denotes parameters of VAE.
- L is the number of **stochastic** layers.
- Sampling and probability evaluation is tractable for each $p(\mathbf{h}^\ell|\mathbf{h}^{\ell+1})$.

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

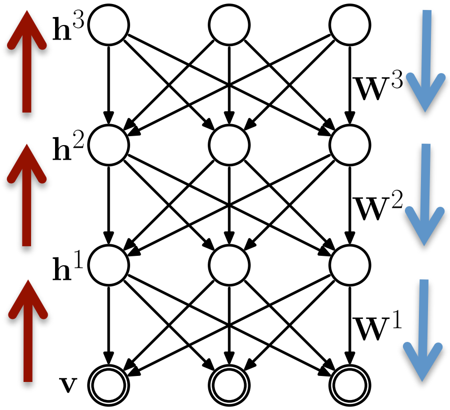
Variational Bound

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

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

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

- 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:

$$\underbrace{\mathbf{h}\left(\boldsymbol{\epsilon}, \mathbf{x}, \boldsymbol{\theta}\right)}_{\text{Deterministic Encoder}}, \quad \text{with} \quad \boldsymbol{\epsilon} = \underbrace{(\boldsymbol{\epsilon}^1, \ldots, \boldsymbol{\epsilon}^L)}_{\text{Distribution of } \boldsymbol{\epsilon} \text{ does not depend on } \boldsymbol{\theta}}$$

Computing the Gradients

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

Autoencoder

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

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

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

Gradients can be computed by backprop

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

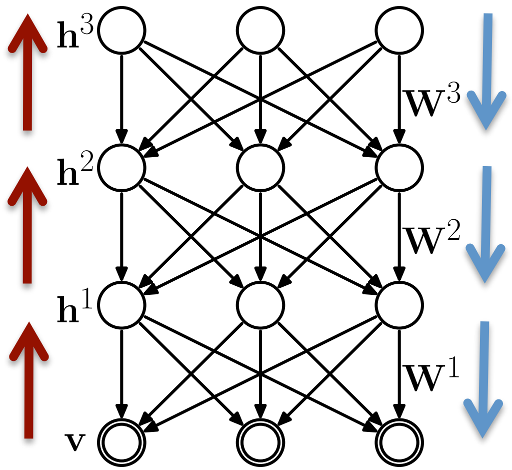
Importance Weighted Autoencoders

- 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, ICLR 2016)

Importance Weighted Autoencoders

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

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

- This is a lower bound on the marginal log-likelihood:

$$\mathcal{L}_k(\mathbf{x}) = \mathbb{E}\left[\log \frac{1}{k}\sum_{i=1}^{k} w_i\right] \leq \log \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} w_i\right] = \log p(\mathbf{x})$$

- Special Case of k=1: Same as standard VAE objective.

- Using more samples → Improves the tightness of the bound.

Tighter Lower Bound

- Using more samples can only improve the tightness of the bound.

- For all k, the lower bounds satisfy*:*

$$\log p(\mathbf{x}) \geq \mathcal{L}_{k+1}(\mathbf{x}) \geq \mathcal{L}_k(\mathbf{x})$$

- Moreover if $p(\mathbf{h}, \mathbf{x})/q(\mathbf{h}|\mathbf{x})$ is bounded, then:

$$\mathcal{L}_k(\mathbf{x}) \to \log p(\mathbf{x}), \quad \text{as} \quad k \to \infty$$

Computing the Gradients

- We can use the unbiased estimate of the gradient using reparameterization trick:

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

where we define normalized importance weights:

$$\widetilde{w}_i = w_i / \sum_{i=1}^{k} w_i, \quad \text{where } w_i = \frac{p(\mathbf{x}, \mathbf{h}_i)}{q(\mathbf{h}_i|\mathbf{x})}$$

IWAEs vs. VAEs

- Draw k-samples form the recognition network $q(\mathbf{h}|\mathbf{x})$
  - or k-sets of auxiliary variables $\boldsymbol{\epsilon}$.
- Obtain the following Monte Carlo estimate of the gradient:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_k(\mathbf{x}) \approx \sum_{i=1}^{k} \widetilde{w}_i \boxed{\nabla_{\boldsymbol{\theta}} \log w(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}_i, \mathbf{x}, \boldsymbol{\theta}), \boldsymbol{\theta})}$$

- Compare this to the VAE's estimate of the gradient:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathbf{x}) \approx \frac{1}{k} \sum_{i=1}^{k} \boxed{\nabla_{\boldsymbol{\theta}} \log w(\mathbf{x}, \mathbf{h}(\boldsymbol{\epsilon}_i, \mathbf{x}, \boldsymbol{\theta}), \boldsymbol{\theta})}$$

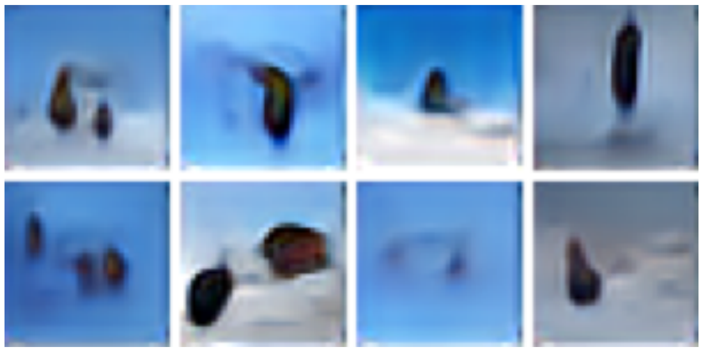
Motivating Example

- 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

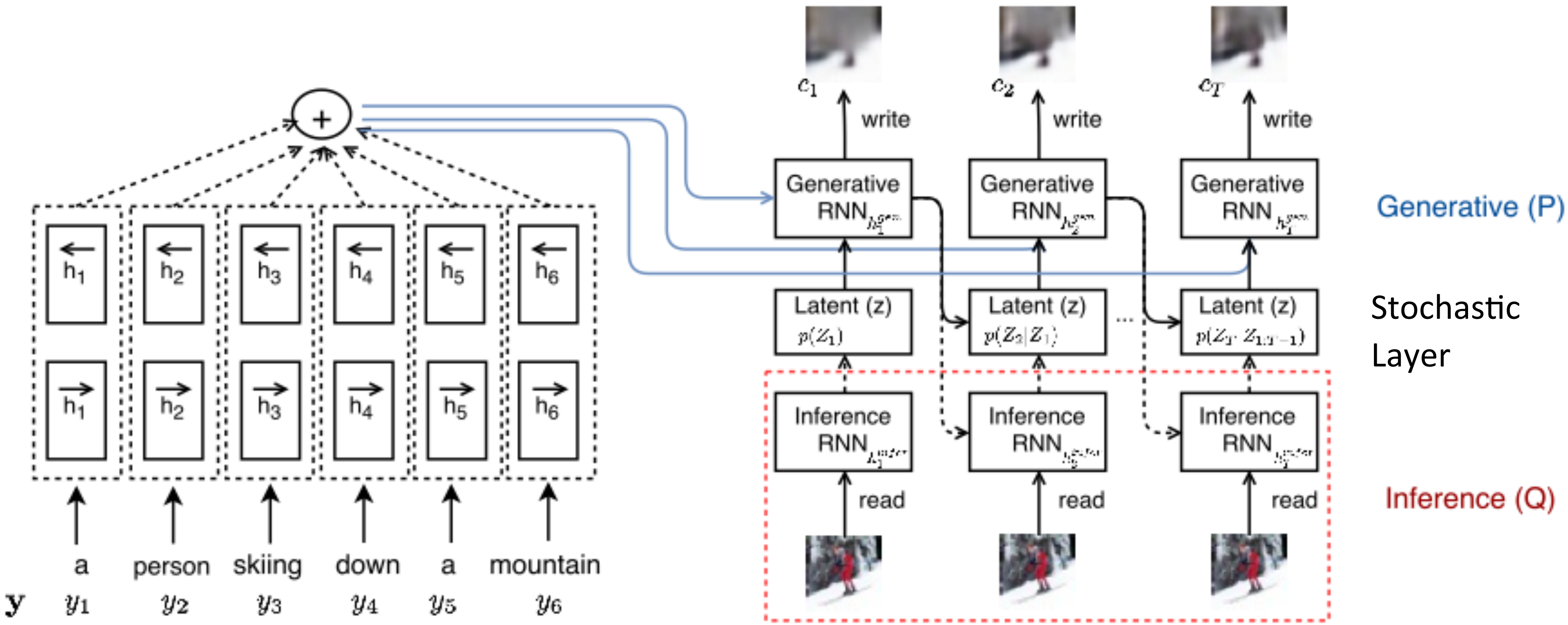
A **herd of elephants** is flying in blue skies



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

(Mansimov, Parisotto, Ba, Salakhutdinov, 2015)

Generating Images from Captions



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

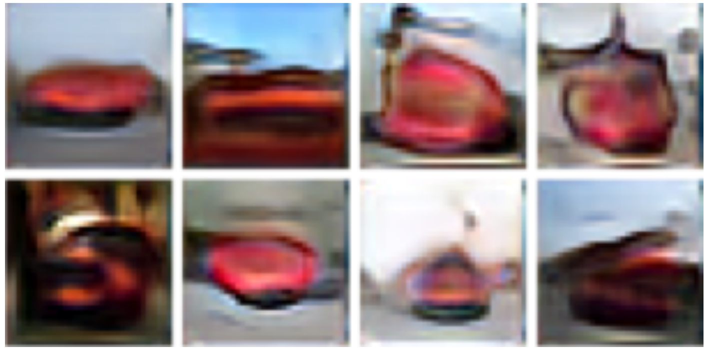
- Recognition Model: Deterministic Recurrent Network.

(Gregor et al., 2015)

Flipping Colors

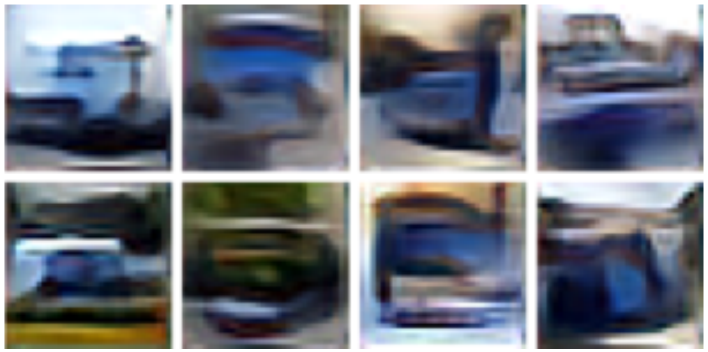
A **yellow school bus** parked in the parking lot

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



Novel Scene Compositions

A toilet seat sits open in the bathroom

A toilet seat sits open in the grass field

Ask Google?

Bloomberg News

A very large commercial plane flying in rainy skies. Source: University of Toronto

Ruslan Salakhutdinov, an assistant professor at the University of Toronto who worked on the toilet project, said this research had a side benefit of helping them learn more about how neural networks work. "We can better