

Learning Deep Generative Models

Russ Salakhutdinov

Deep Learning Summer School

Machine Learning Department
Carnegie Mellon University
Canadian Institute for Advanced Research

Carnegie
Mellon
University



Mining for Structure

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

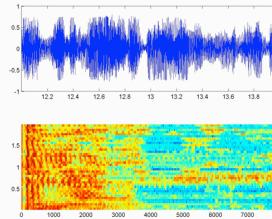
Images & Video



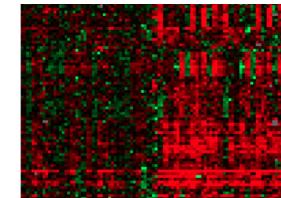
Text & Language



Speech & Audio



Gene Expression

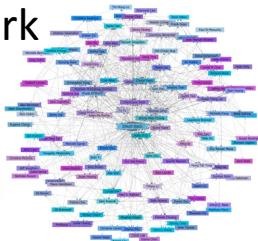


Product

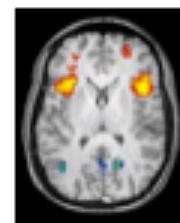
Recommendation



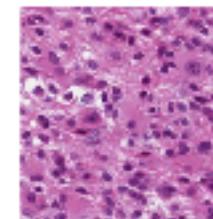
Relational Data/
Social Network



fMRI



Tumor region



Mostly Unlabeled

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

Mining for Structure

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

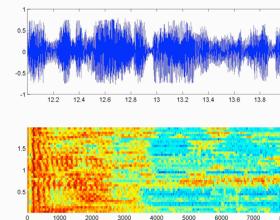
Images & Video



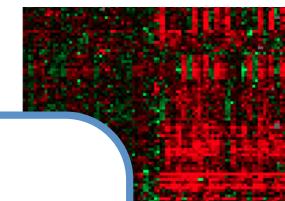
Text & Language



Speech & Audio



Gene Expression



Product

Recommen



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

Mostly Unlabeled

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

Example: Understanding Images



TAGS:

strangers, coworkers, conventioneers,
attendants, patrons

Nearest Neighbor Sentence:

people taking pictures of a crazy person

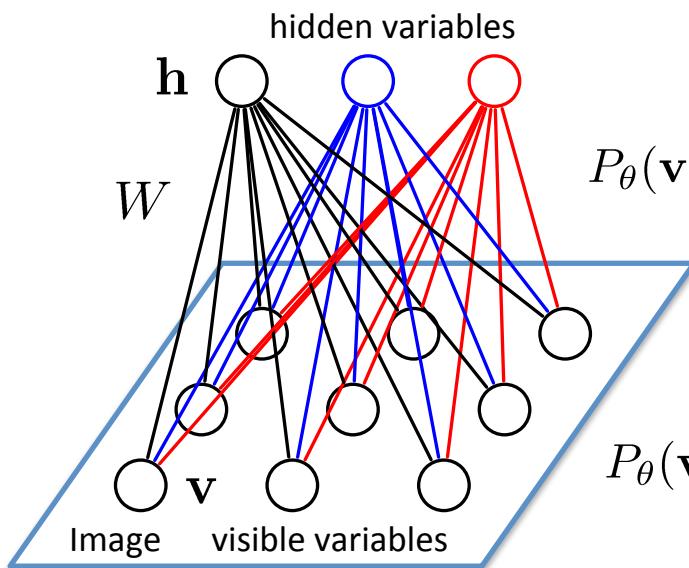
Model Samples

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

Talk Roadmap

- Learning Deep Undirected Models
 - Restricted Boltzmann Machines
 - Deep Boltzmann Machines
- Learning Deep Directed Models
 - Helmholtz Machines
 - Variational & Importance Weighted Autoencoders
 - Stochastic (Hard) Attention Models
- Applications and Some Open Problems

Restricted Boltzmann Machines



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

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

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

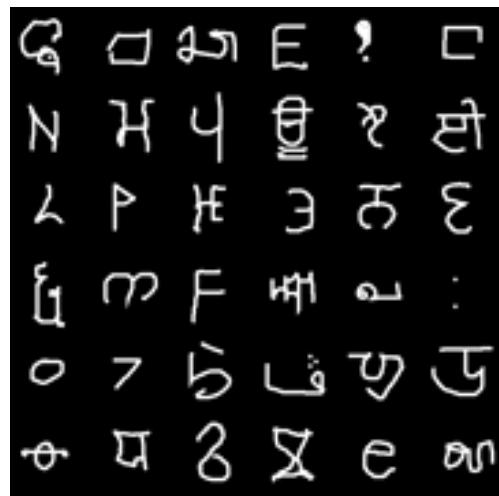
RBM is a Markov Random Field with:

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

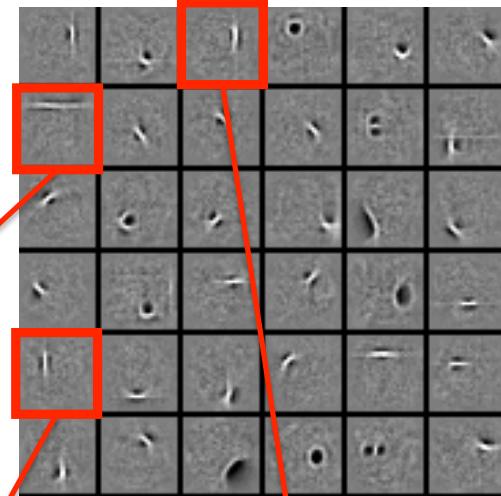
Markov random fields, Boltzmann machines, log-linear models.

Learning Features

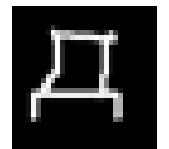
Observed Data
Subset of 25,000 characters



Learned W: "edges"
Subset of 1000 features



New Image: $p(h_7 = 1|v)$



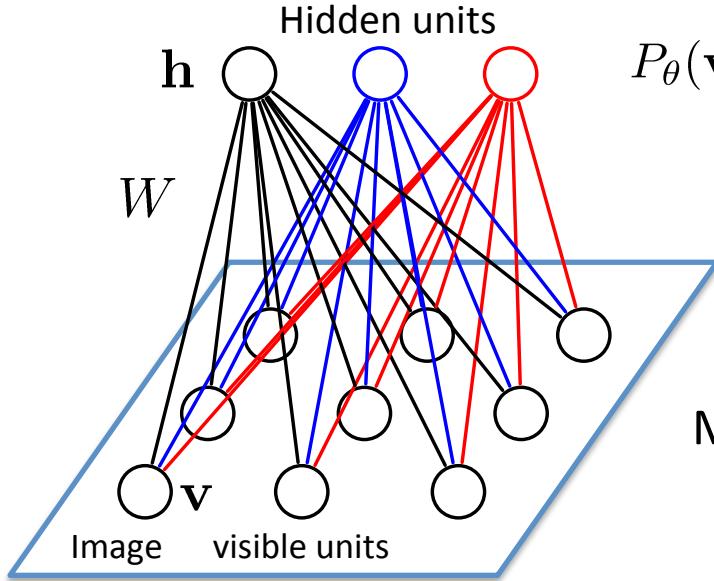
$$= \sigma\left(0.99 \times \begin{array}{c} \text{small image} \\ \text{with a thick vertical stroke} \end{array} + 0.97 \times \begin{array}{c} \text{small image} \\ \text{with a thick horizontal stroke} \end{array} + 0.82 \times \begin{array}{c} \text{small image} \\ \text{with a thick diagonal stroke} \end{array} \dots\right)$$

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

Logistic Function: Suitable for
modeling binary images

Sparse
representations

Model Learning



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

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

Maximize log-likelihood objective:

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

Derivative of the log-likelihood:

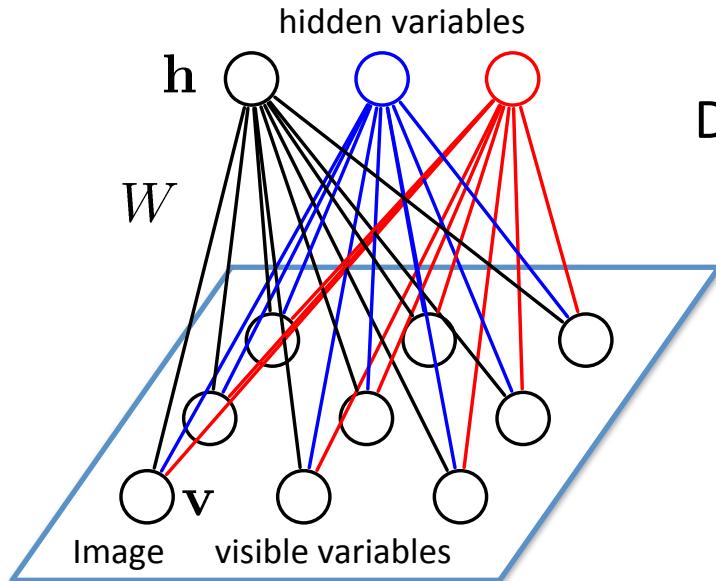
$$\begin{aligned} \frac{\partial L(\theta)}{\partial W_{ij}} &= \frac{1}{N} \sum_{n=1}^N \frac{\partial}{\partial W_{ij}} \log \left(\sum_{\mathbf{h}} \exp [\mathbf{v}^{(n)\top} W \mathbf{h} + \mathbf{a}^\top \mathbf{h} + \mathbf{b}^\top \mathbf{v}^{(n)}] \right) - \frac{\partial}{\partial W_{ij}} \log \mathcal{Z}(\theta) \\ &= \mathbb{E}_{P_{data}} [v_i h_j] - \underbrace{\mathbb{E}_{P_\theta} [v_i h_j]}_{\text{Difficult to compute: exponentially many configurations}} \end{aligned}$$

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

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

Difficult to compute: exponentially many configurations

Model Learning



Derivative of the log-likelihood:

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

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

Easy to
compute exactly

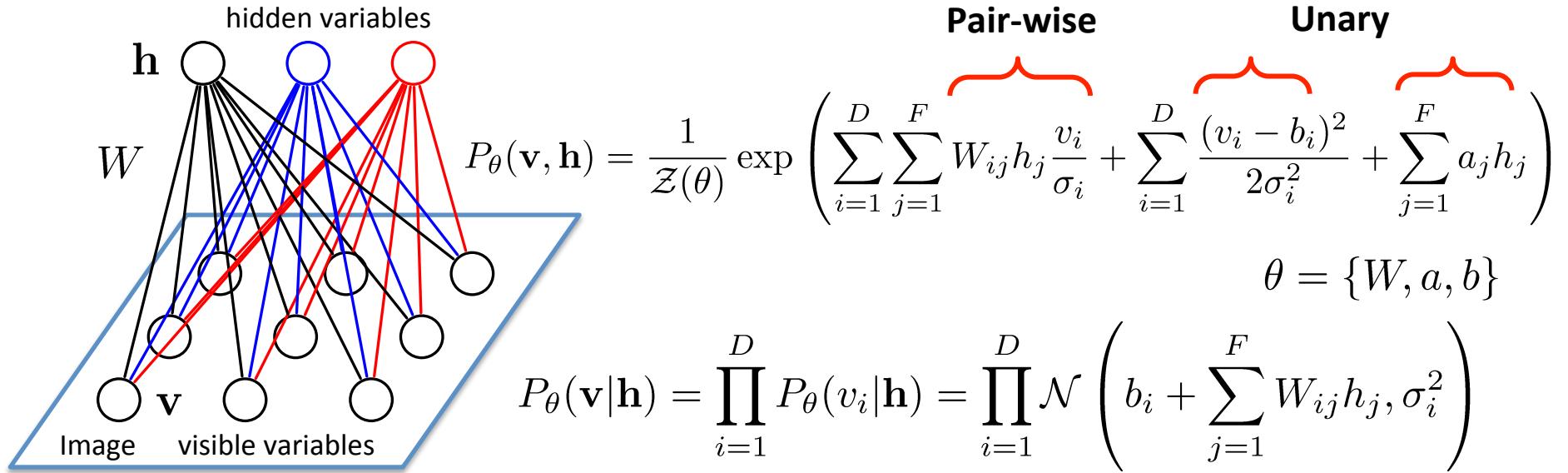
$$P_{data}(\mathbf{v}, \mathbf{h}; \theta) = P(\mathbf{h}|\mathbf{v}; \theta)P_{data}(\mathbf{v})$$

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

Difficult to compute:
exponentially many
configurations.
Use MCMC

Approximate maximum likelihood learning

RBMs for Real-valued Data

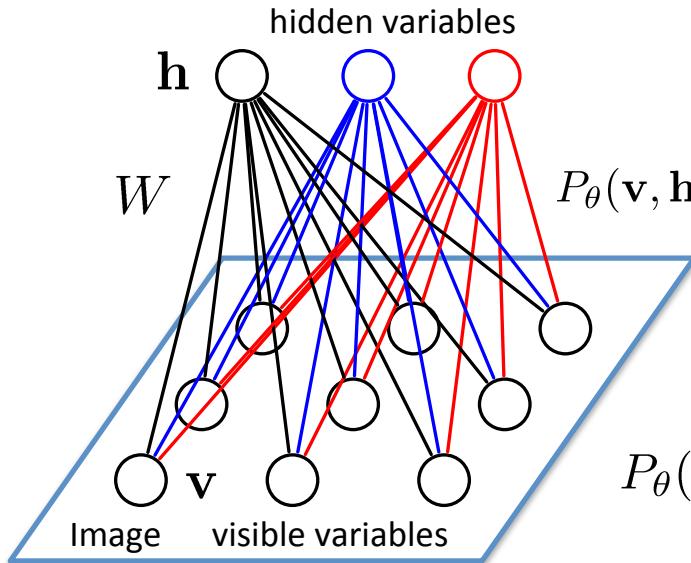


Gaussian-Bernoulli RBM:

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

(Salakhutdinov & Hinton, NIPS 2007; Salakhutdinov & Murray, ICML 2008)

RBM^s for Real-valued Data



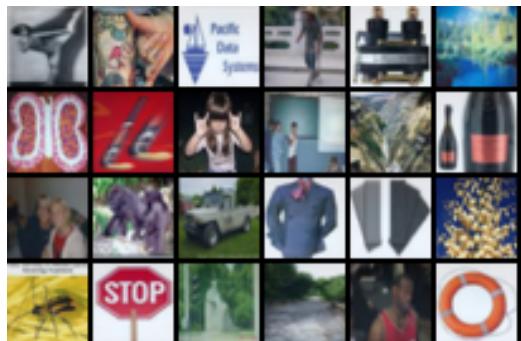
$$P_{\theta}(\mathbf{v}, \mathbf{h}) = \frac{1}{Z(\theta)} \exp \left(\sum_{i=1}^D \sum_{j=1}^F W_{ij} h_j \frac{v_i}{\sigma_i} + \sum_{i=1}^D \frac{(v_i - b_i)^2}{2\sigma_i^2} + \sum_{j=1}^F a_j h_j \right)$$

Pair-wise **Unary**

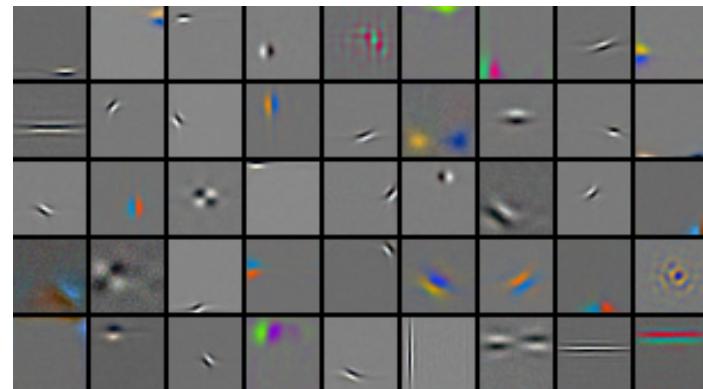
$$\theta = \{W, a, b\}$$

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

4 million **unlabelled** images



Learned features (out of 10,000)

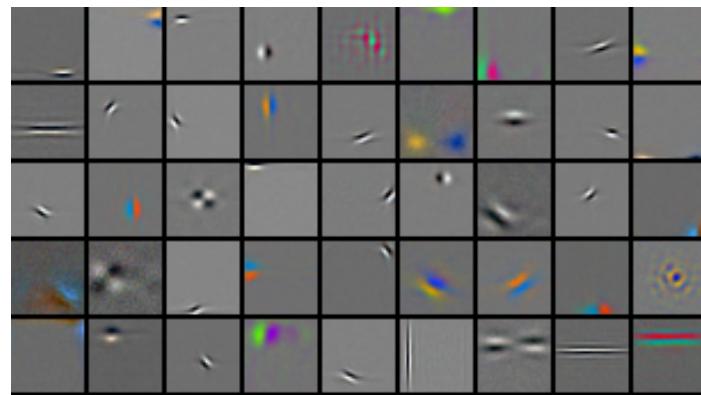


RBM^s for Real-valued Data

4 million **unlabelled** images

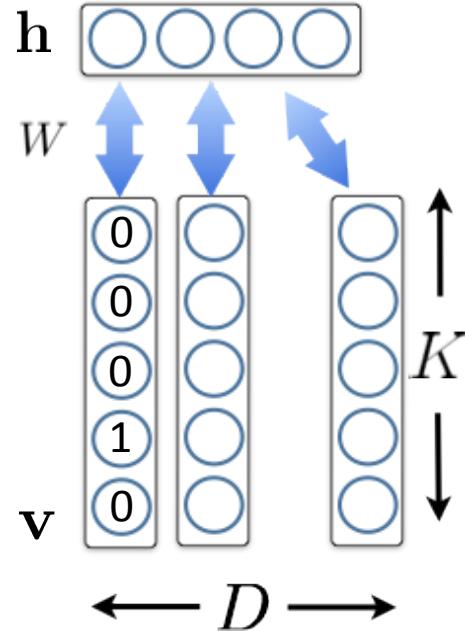


Learned features (out of 10,000)



$$\begin{aligned} p(h_7 = 1|v) & \quad p(h_{29} = 1|v) \\ \downarrow & \quad \downarrow \\ \text{New Image} & = 0.9 * \text{[feature 7]} + 0.8 * \text{[feature 29]} + 0.6 * \text{[feature 42]} \dots \end{aligned}$$

RBMs for Word Counts



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

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

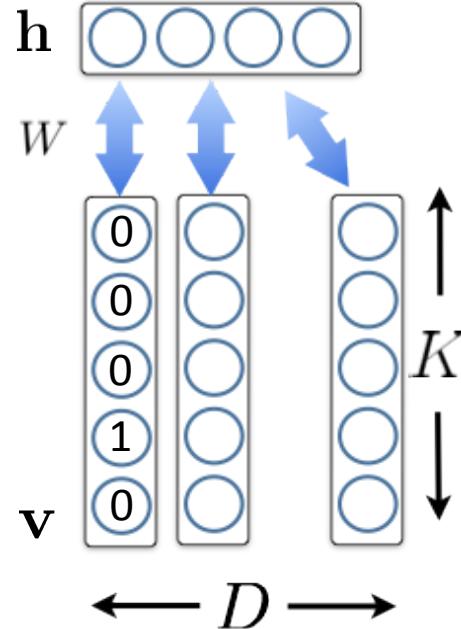
$$P_{\theta}(v_i^k = 1 | \mathbf{h}) = \frac{\exp(b_i^k + \sum_{j=1}^F h_j W_{ij}^k)}{\sum_{q=1}^K \exp(b_i^q + \sum_{j=1}^F h_j W_{ij}^q)}$$

Replicated Softmax Model: undirected topic model:

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

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

RBMs for Word Counts



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

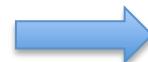
$$\theta = \{W, a, b\}$$

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



REUTERS
AP Associated Press

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

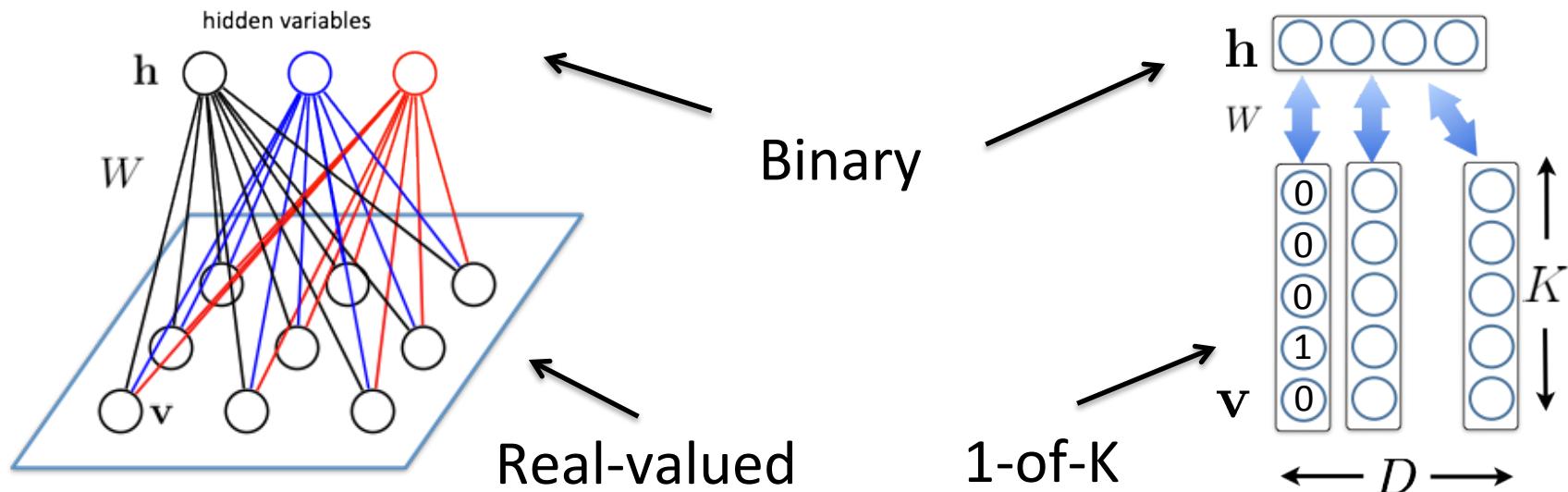


Learned features: ‘topics’

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

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:

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

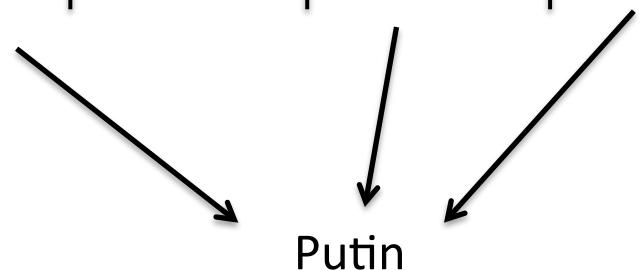
Product of Experts

The joint distribution is given by:

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

Marginalizing over hidden variables:

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



Topics “government”, “corruption” and “oil” can combine to give very high probability to a word “Putin”.

Product of Experts

The joint distribution is given by:

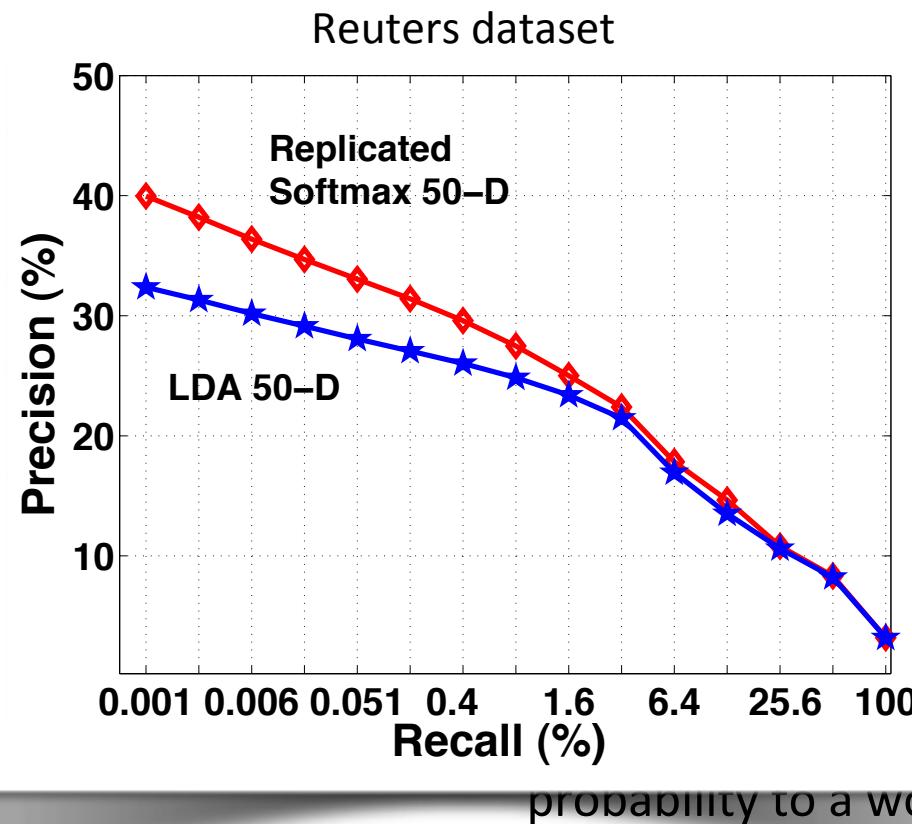
$$P_{\theta}(\mathbf{v}, \mathbf{h}) = \frac{1}{Z(\theta)} \exp \left(\sum_{ij} W_{ij} v_i h_j + \sum_i b_i v_i + \sum_j a_j h_j \right)$$

Marginalizing out \mathbf{h} :

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

government
auhority
power
empire
putin

clint
hou
pres
bill
cong

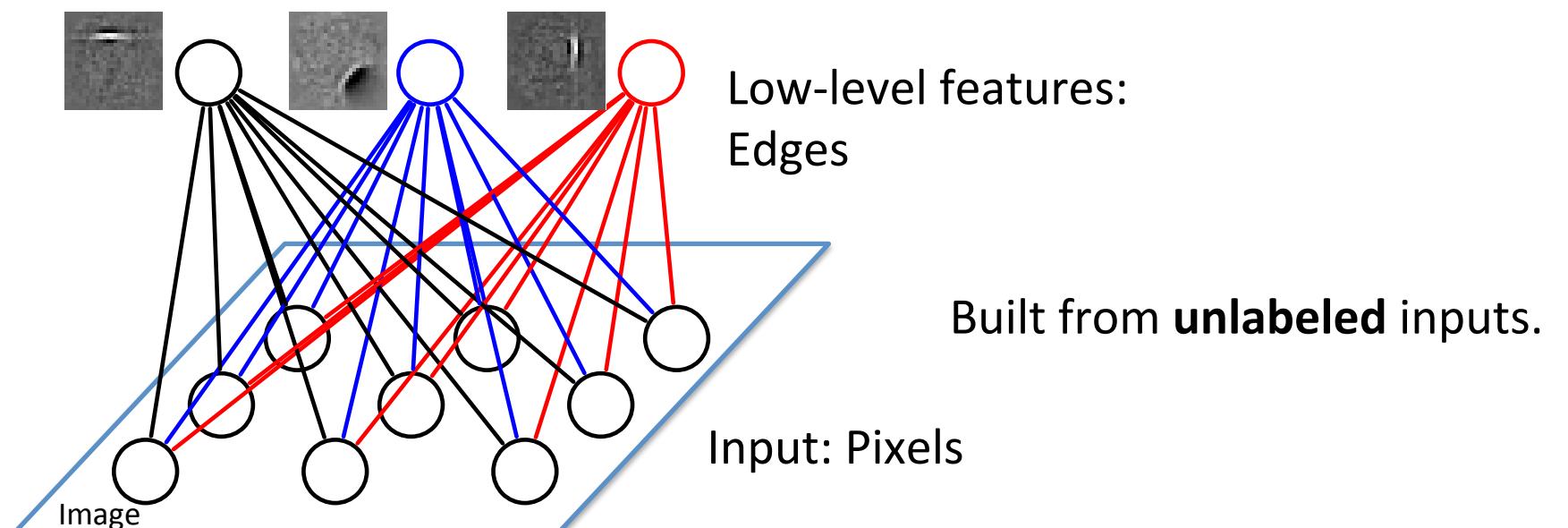


Product of Experts

$(W_{ij} v_i)$

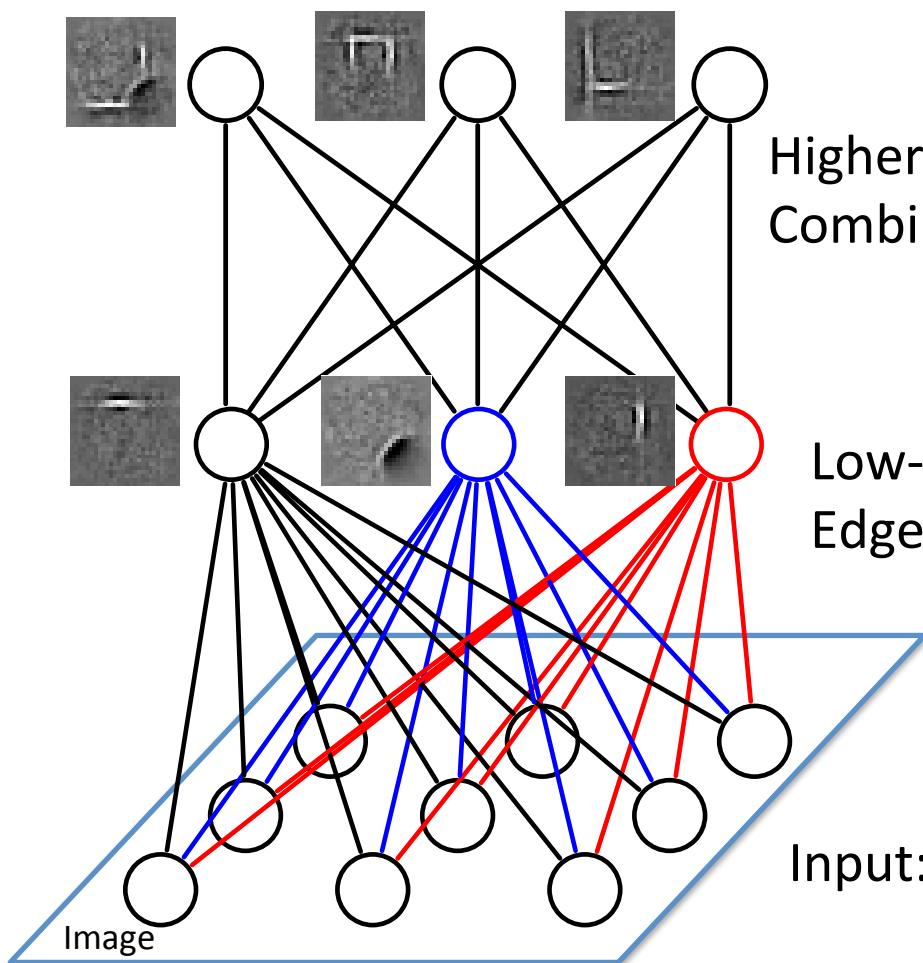
ations allow the
, "corruption" and
ive very high
probability to a word "Putin".

Deep Boltzmann Machines



(Salakhutdinov & Hinton, Neural Computation 2012)

Deep Boltzmann Machines



Learn simpler representations,
then compose more complex ones

Higher-level features:
Combination of edges

Low-level features:
Edges

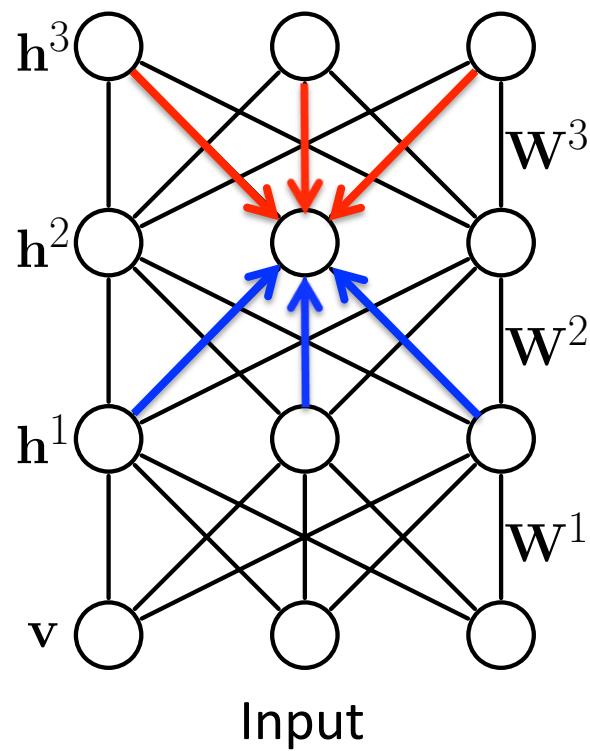
Built from **unlabeled** inputs.

Input: Pixels

(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)

Model Formulation

$$P_{\theta}(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = \frac{1}{Z(\theta)} \exp \left[\underbrace{\mathbf{v}^{\top} W^{(1)} \mathbf{h}^{(1)}}_{\text{Same as RBMs}} + \underbrace{\mathbf{h}^{(1)\top} W^{(2)} \mathbf{h}^{(2)}}_{\text{Same as RBMs}} + \underbrace{\mathbf{h}^{(2)\top} W^{(3)} \mathbf{h}^{(3)}}_{\text{Same as RBMs}} \right]$$



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

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

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

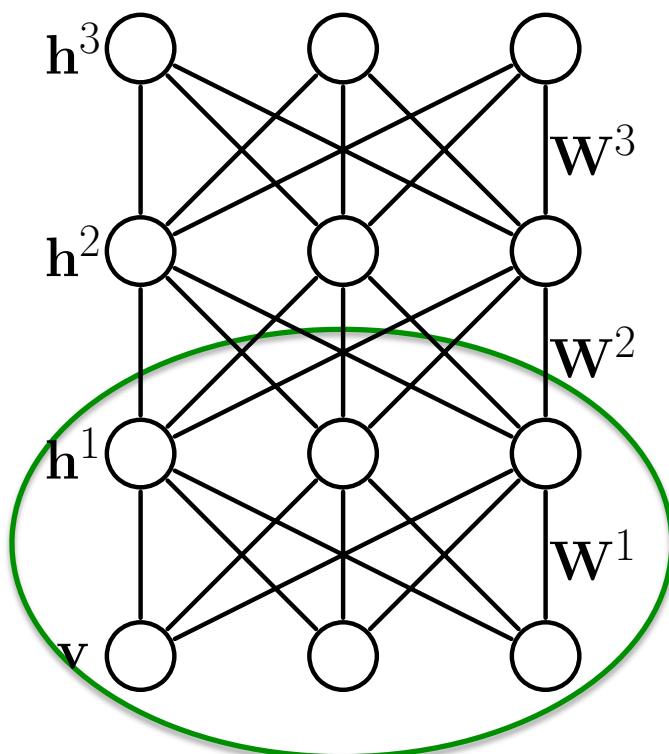
Top-down Bottom-up

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

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:

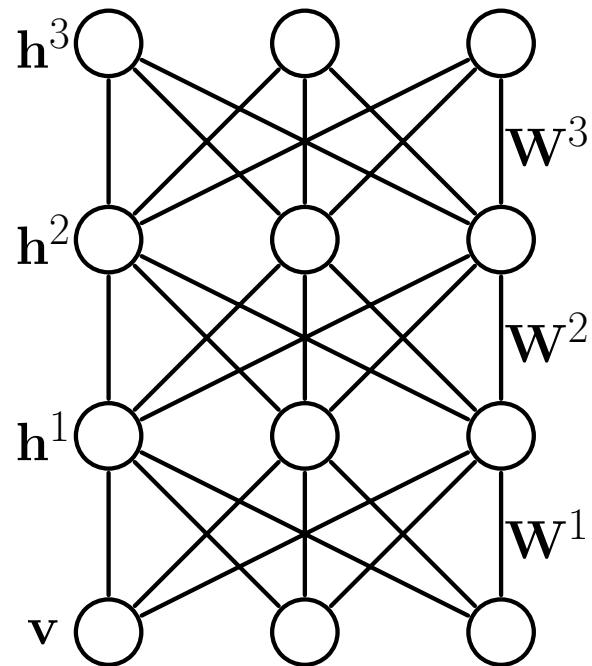
$$\frac{\partial \log P_{\theta}(\mathbf{v})}{\partial W^1} = \mathbb{E}_{P_{data}}[\mathbf{v}\mathbf{h}^{1\top}] - \mathbb{E}_{P_{\theta}}[\mathbf{v}\mathbf{h}^{1\top}]$$

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:

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

- Both expectations are intractable!

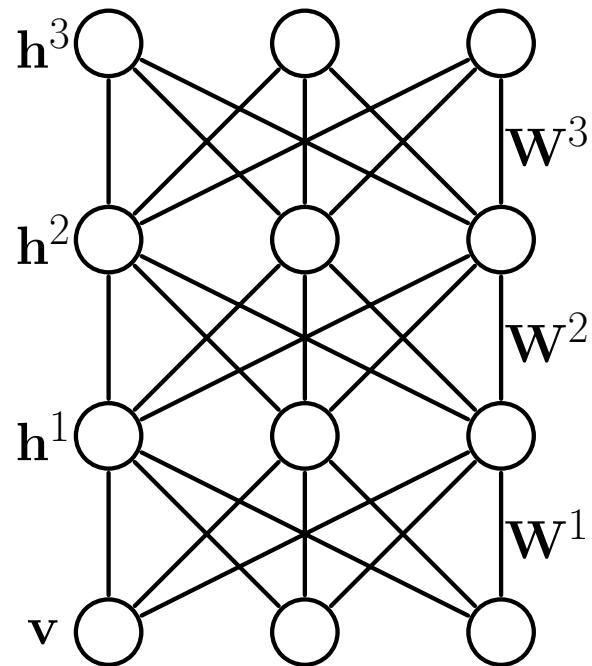
$$P_{data}(\mathbf{v}, \mathbf{h}^1) = P_\theta(\mathbf{h}^1 | \mathbf{v}) P_{data}(\mathbf{v})$$

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

Not factorial any more!

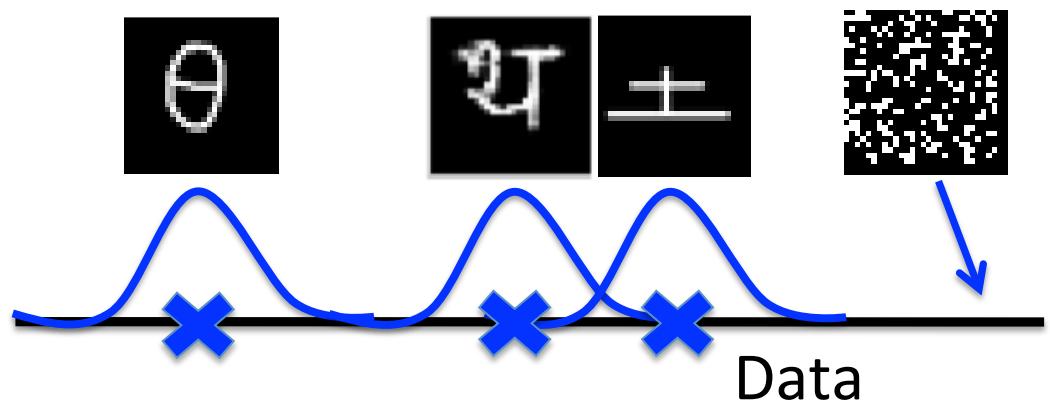
Approximate Learning

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



(Approximate) Maximum Likelihood:

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



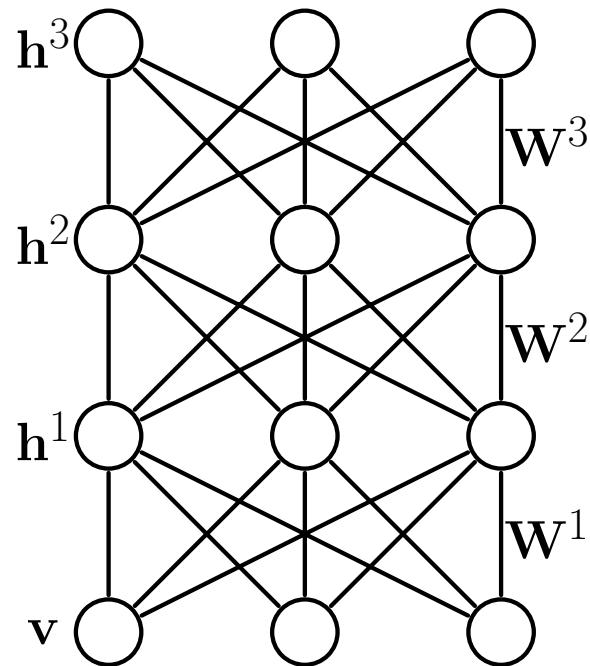
$$P_{data}(\mathbf{v}, \mathbf{h}^1) = P_{\theta}(\mathbf{h}^1 | \mathbf{v}) P_{data}(\mathbf{v})$$

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

Not factorial any more!

Approximate Learning

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



(Approximate) Maximum Likelihood:

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

Variational
Inference

Stochastic
Approximation
(MCMC-based)

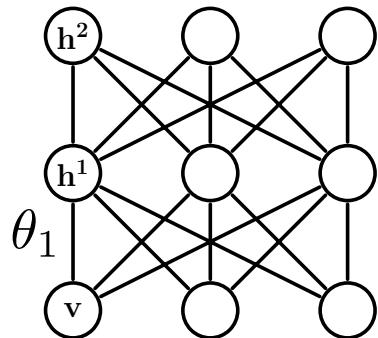
$$P_{data}(\mathbf{v}, \mathbf{h}^1) = P_\theta(\mathbf{h}^1 | \mathbf{v}) P_{data}(\mathbf{v})$$

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

Not factorial any more!

Stochastic Approximation

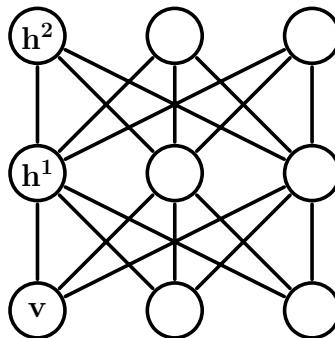
Time $t=1$



$$\mathbf{x}_1 \sim T_{\theta_1}(\mathbf{x}_1 \leftarrow \mathbf{x}_0)$$

Update θ_1

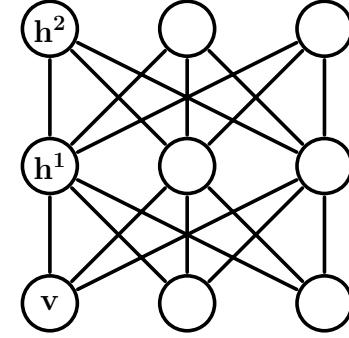
$t=2$



$$\mathbf{x}_2 \sim T_{\theta_2}(\mathbf{x}_2 \leftarrow \mathbf{x}_1)$$

Update θ_2

$t=3$



$$\mathbf{x}_3 \sim T_{\theta_3}(\mathbf{x}_3 \leftarrow \mathbf{x}_2)$$

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

- Generate $\mathbf{x}_t \sim T_{\theta_t}(\mathbf{x}_t \leftarrow \mathbf{x}_{t-1})$ 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 $E_{P_{\theta_t}}[\mathbf{v}\mathbf{h}^\top]$ with a point estimate $[\mathbf{v}_t \mathbf{h}_t^\top]$

In practice we simulate several Markov chains in parallel.

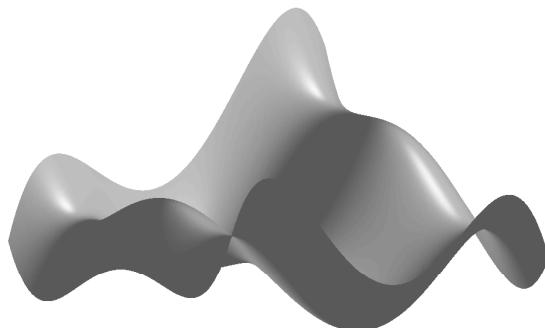
Robbins and Monro, Ann. Math. Stats, 1957
L. Younes, Probability Theory 1989

Learning Algorithm

Update rule decomposes:

$$\theta_{t+1} = \theta_t + \alpha_t \left(\underbrace{\mathbb{E}_{P_{data}}[\mathbf{v}\mathbf{h}^\top] - \mathbb{E}_{P_{\theta_t}}[\mathbf{v}\mathbf{h}^\top]}_{\text{True gradient}} \right) + \alpha_t \left(\underbrace{\mathbb{E}_{P_{\theta_t}}[\mathbf{v}\mathbf{h}^\top] - \frac{1}{M} \sum_{m=1}^M \mathbf{v}_t^{(m)} \mathbf{h}_t^{(m)\top}}_{\text{Perturbation term } \epsilon_t} \right)$$

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

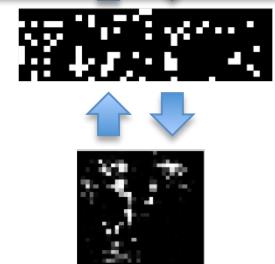


(Salakhutdinov, ICML 2010, NIPS 2011, Srivastava et al., NIPS 2012, Grosse et.al., 2013, Burda et.al., 2015);

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

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

**Markov Chain
Monte Carlo**



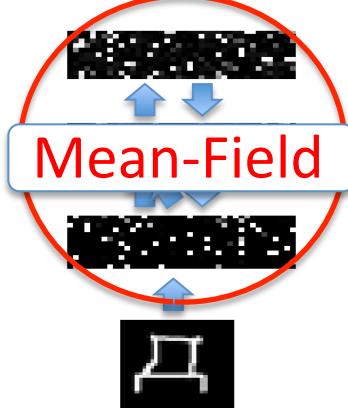
Connections to the theory of stochastic approximation and adaptive MCMC.

Variational Inference

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

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

Posterior Inference



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

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

Variational Lower Bound

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

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

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

(Salakhutdinov, 2008; Salakhutdinov & Larochelle, AI & Statistics 2010)

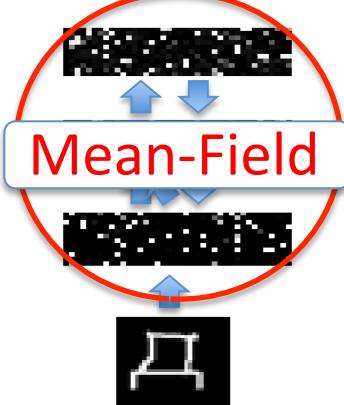
Variational Inference

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

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

$$\log P_\theta(\mathbf{v}) \geq \log P_\theta(\mathbf{v}) - \text{KL}(Q_\mu(\mathbf{h}|\mathbf{v}) || P_\theta(\mathbf{h}|\mathbf{v}))$$

Posterior Inference

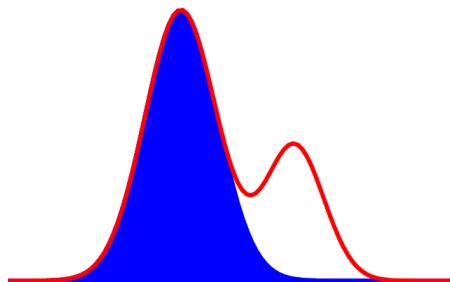


Variational Lower Bound

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:

$$\begin{aligned}\mu_j^{(1)} &= \sigma \left(\sum_i W_{ij}^1 v_i + \sum_k W_{jk}^2 \mu_k^{(2)} \right) \\ \mu_k^{(2)} &= \sigma \left(\sum_j W_{jk}^2 \mu_j^{(1)} + \sum_m W_{km}^3 \mu_m^{(3)} \right) \\ \mu_m^{(3)} &= \sigma \left(\sum_k W_{km}^3 \mu_k^{(2)} \right)\end{aligned}$$

Variational Inference

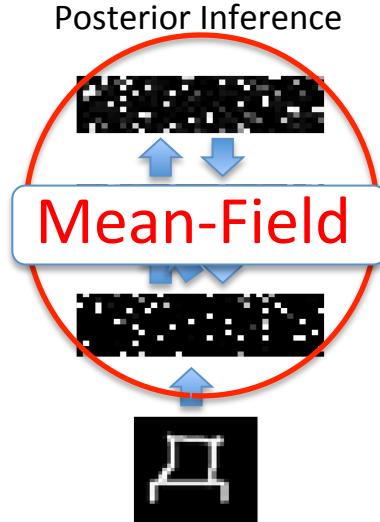
Approximate intractable distribution $P_\theta(\mathbf{h}|\mathbf{v})$ with simpler, tractable distribution $Q_\mu(\mathbf{h}|\mathbf{v})$:

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

$$\log P_\theta(\mathbf{v}) \geq \log P_\theta(\mathbf{v}) - \text{KL}(Q_\mu(\mathbf{h}|\mathbf{v}) || P_\theta(\mathbf{h}|\mathbf{v}))$$

Variational Lower Bound

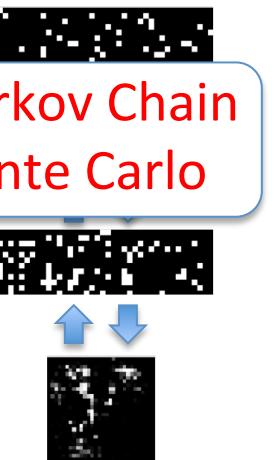
Unconditional Simulation



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

Markov Chain Monte Carlo

2. MCMC: Apply stochastic approximation to update model parameters



Almost sure convergence guarantees to an asymptotically stable point.

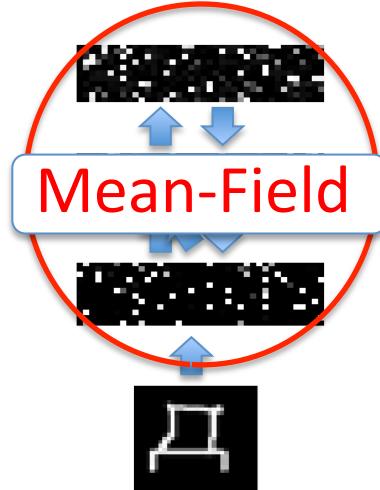
Variational Inference

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

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

$$\log P_\theta(\mathbf{v}) \geq \log P_\theta(\mathbf{v}) - \text{KL}(Q_\mu(\mathbf{h}|\mathbf{v}) || P_\theta(\mathbf{h}|\mathbf{v}))$$

Posterior Inference



Variational Lower Bound

Unconditional Simulation

1. V
bou

Fast Inference

2. M
to u

Learning can scale to
millions of examples

Markov Chain
Monte Carlo



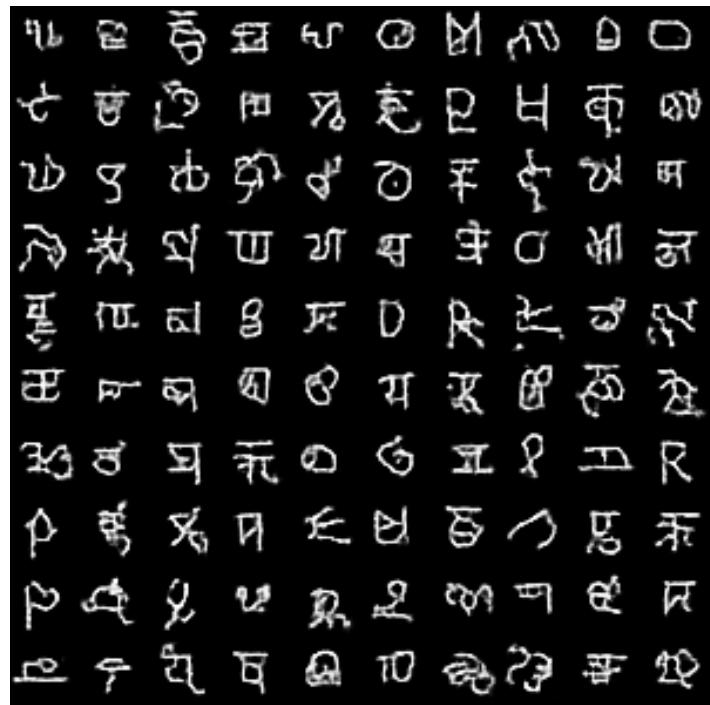
Almost sure convergence guarantees to an asymptotically stable point.

Good Generative Model?

Handwritten Characters

Good Generative Model?

Handwritten Characters



Good Generative Model?

Handwritten Characters

Simulated

Real Data

Good Generative Model?

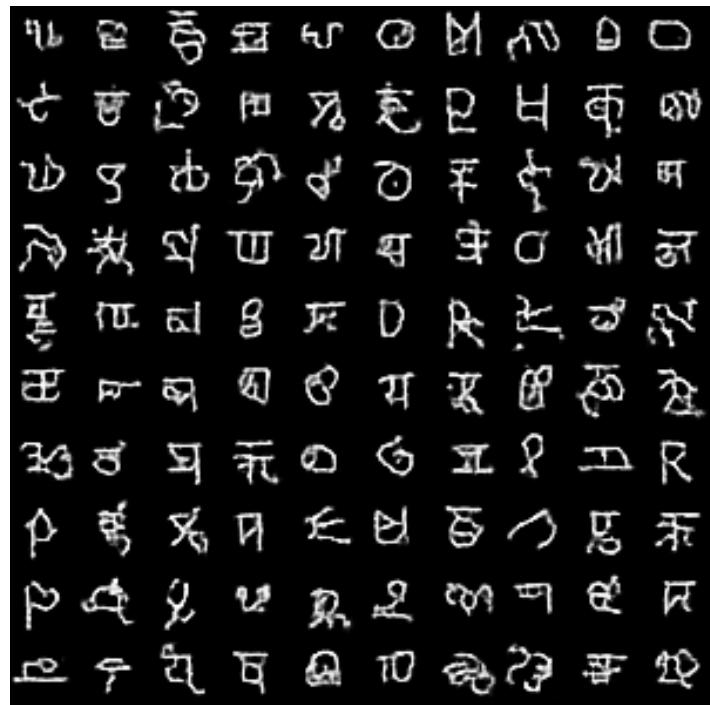
Handwritten Characters

Real Data

Simulated

Good Generative Model?

Handwritten Characters



Good Generative Model?

MNIST Handwritten Digit Dataset

1 8 3 1 5 7 1
6 6 3 3 3 1 8
4 5 8 4 4 1 9
3 7 7 9 3 7 6
1 5 3 5 0 1 2
4 2 5 1 2 4 2
3 0 5 0 7 0 9

6 2 7 4 2 1 9
1 2 5 2 0 7 5
8 1 8 4 2 6 6
0 7 9 8 6 3 2
7 5 0 5 7 9 5
1 8 7 0 6 5 0
7 5 4 8 4 4 7

Handwriting Recognition

MNIST Dataset
60,000 examples of 10 digits

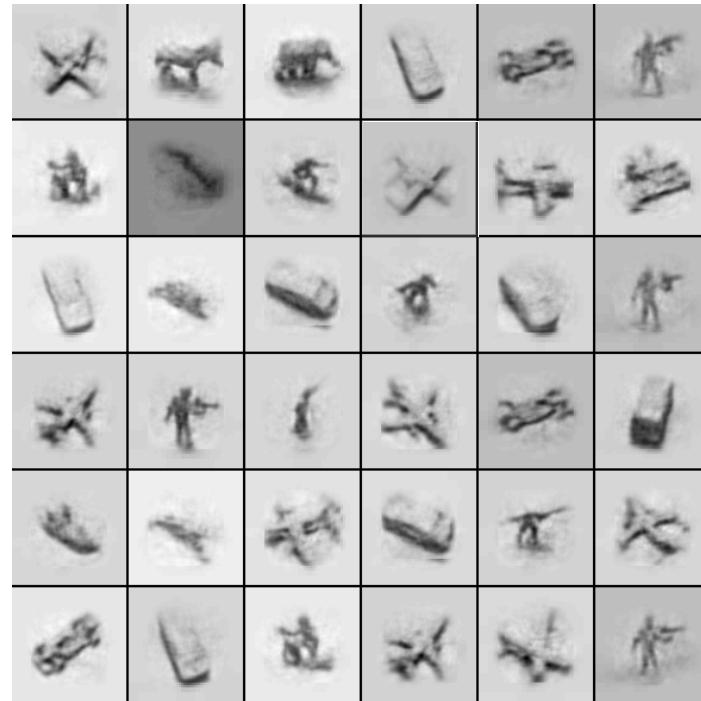
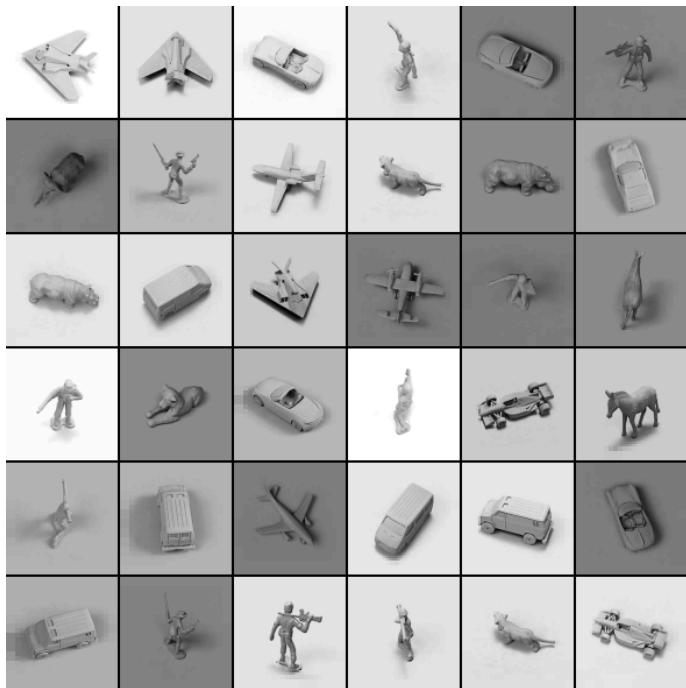
Learning Algorithm	Error
Logistic regression	12.0%
K-NN	3.09%
Neural Net (Platt 2005)	1.53%
SVM (Decoste et.al. 2002)	1.40%
Deep Autoencoder (Bengio et. al. 2007)	1.40%
Deep Belief Net (Hinton et. al. 2006)	1.20%
DBM	0.95%

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

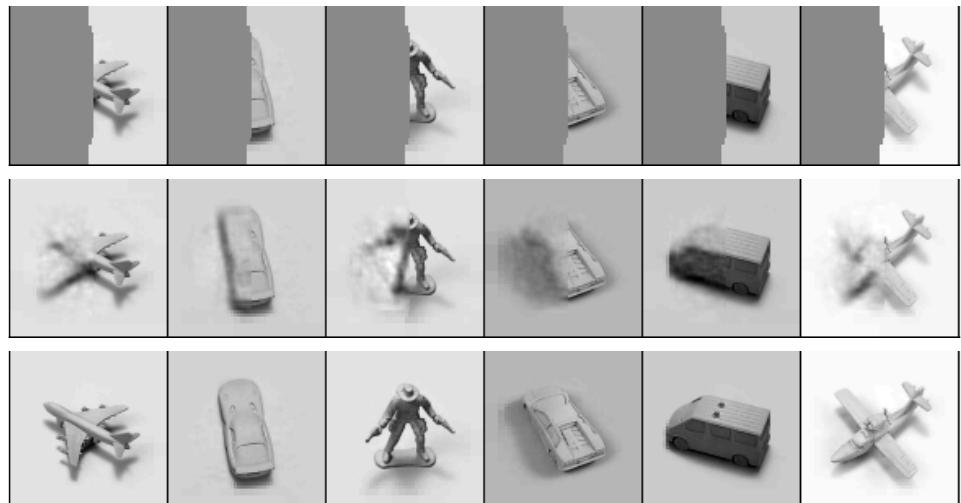


24,000 examples, 5 object categories, 5 different objects within each category, 6 lightning conditions, 9 elevations, 18 azimuths.

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

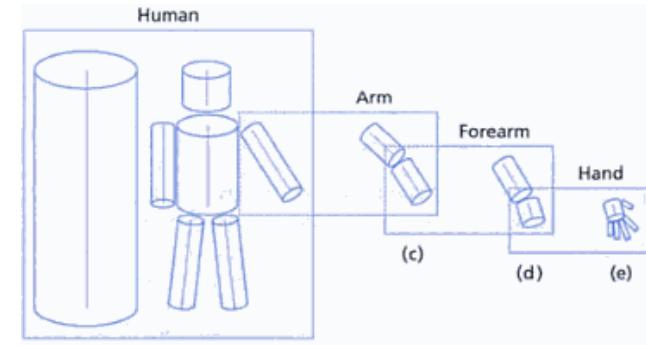


Permutation-invariant version.

Learning Hierarchical Representations

Deep Boltzmann Machines:

Learning Hierarchical Structure
in Features: edges, combination
of edges.



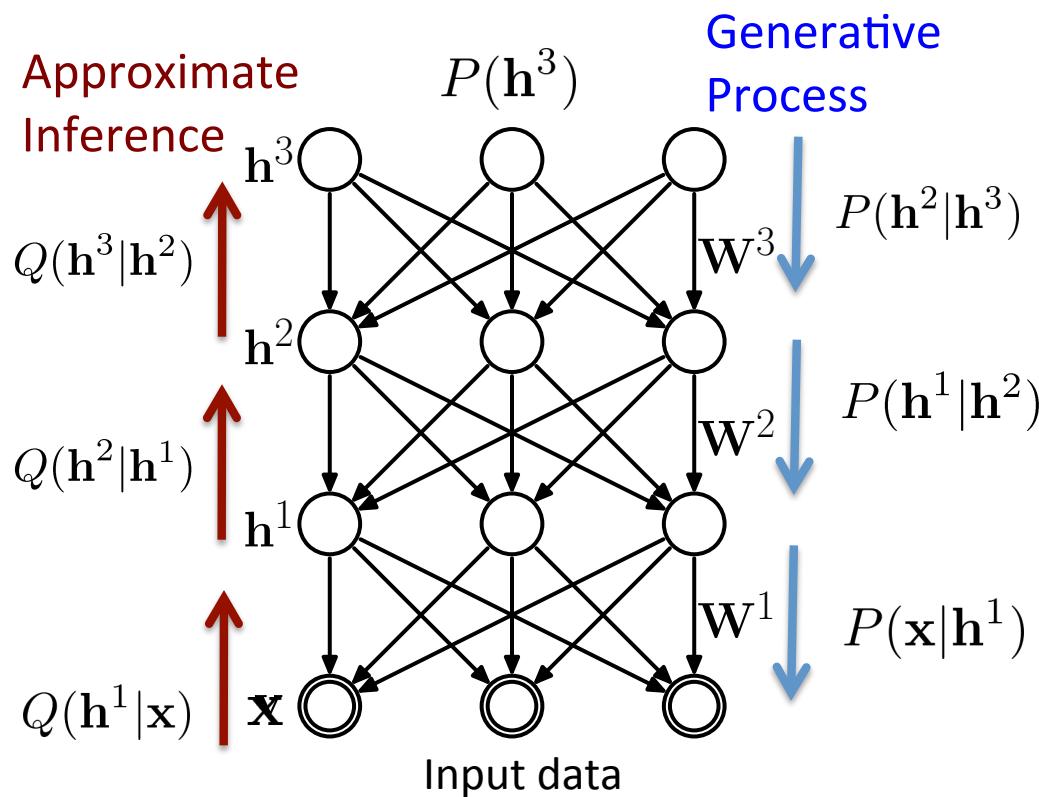
- Performs well in many application domains
- Fast Inference: fraction of a second
- Learning scales to millions of examples

Talk Roadmap

- Learning Deep Undirected Models
 - Restricted Boltzmann Machines
 - Deep Boltzmann Machines
- Learning Deep Directed Models
 - Helmholtz Machines
 - Variational & Importance Weighted Autoencoders
 - Stochastic (Hard) Attention Models
- Applications and Some Open Problems

Helmholtz Machines

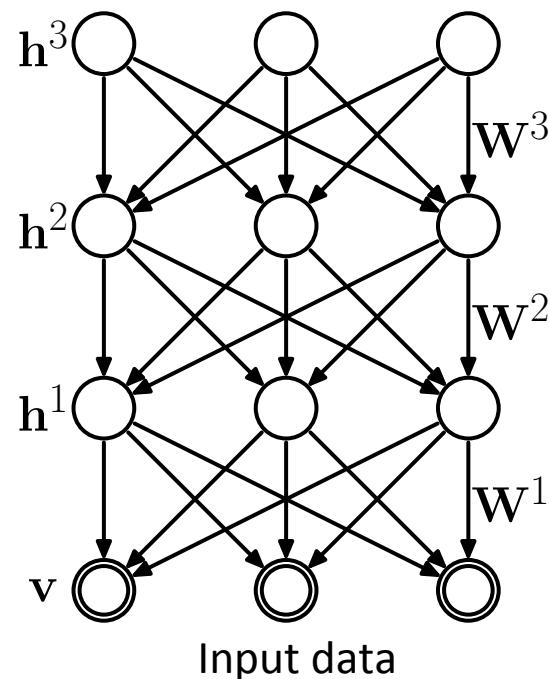
- Hinton, G. E., Dayan, P., Frey, B. J. and Neal, R., Science 1995



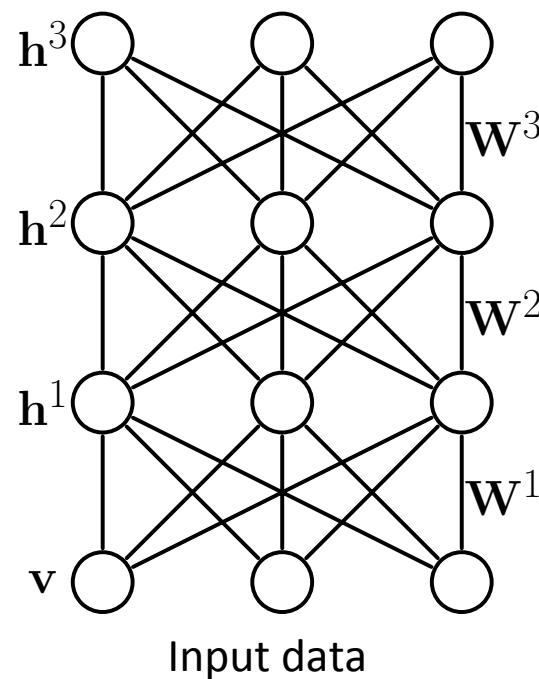
- Kingma & Welling, 2014
- Rezende, Mohamed, Daan, 2014
- Mnih & Gregor, 2014
- Bornschein & Bengio, 2015
- Tang & Salakhutdinov, 2013

Various Deep Generative Models

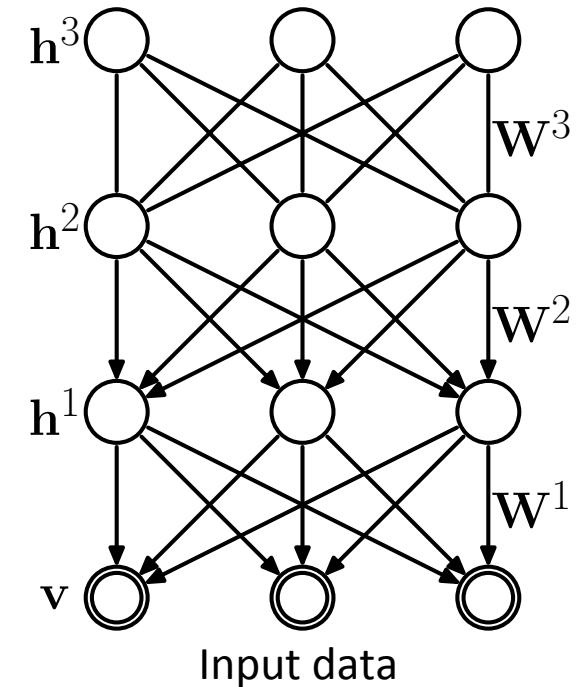
Helmholtz Machine



Deep Boltzmann Machine



Deep Belief Network

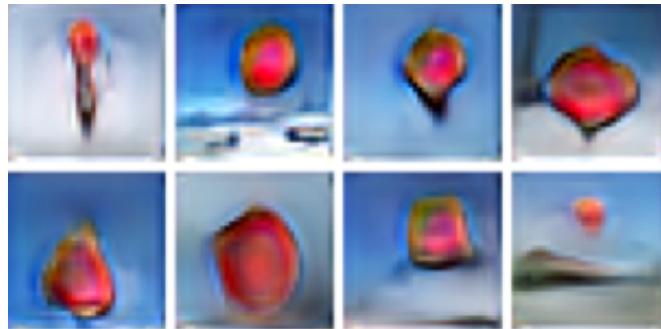


Motivating Example

(Mansimov, Parisotto, Ba, Salakhutdinov, 2015)

- Can we generate images from natural language descriptions?

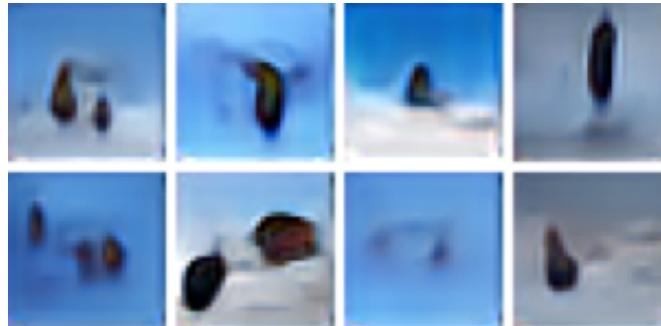
A **stop sign** is flying in blue skies



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



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

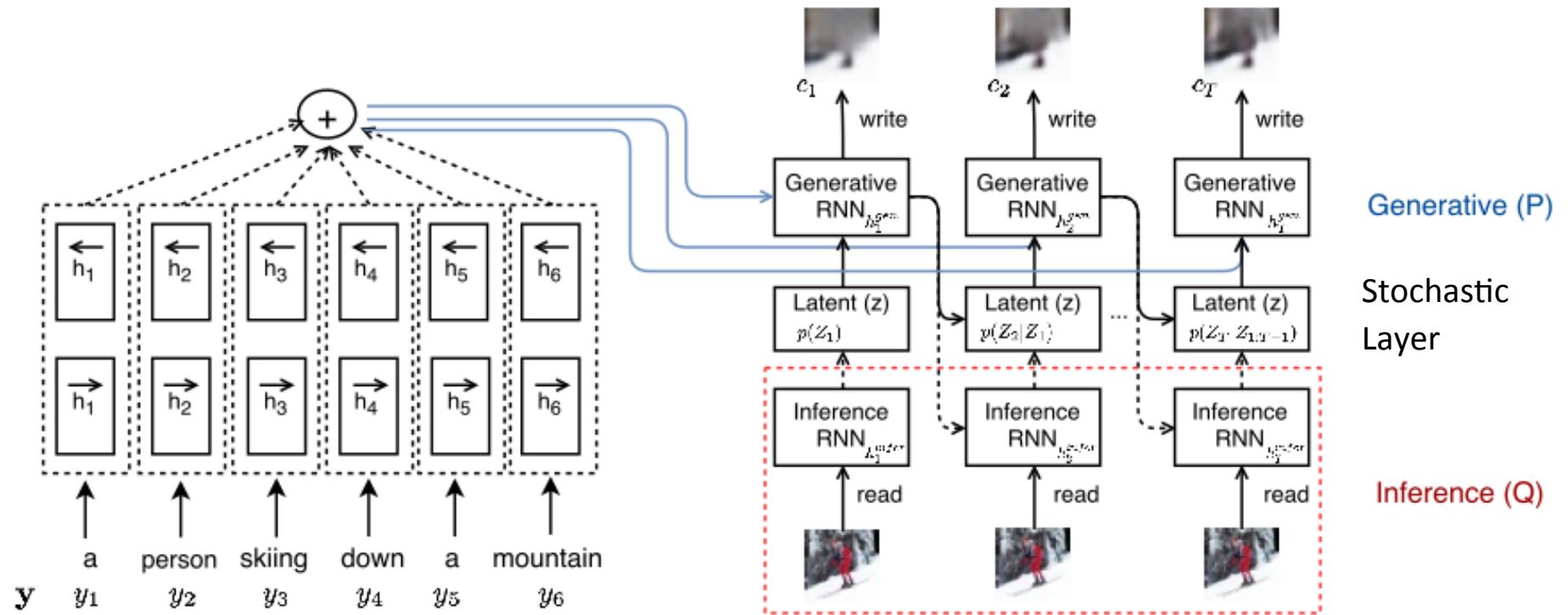


A **large commercial airplane** is flying in blue skies



Overall Model

(Mansimov, Parisotto, Ba, Salakhutdinov, 2015)



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

Flipping Colors

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



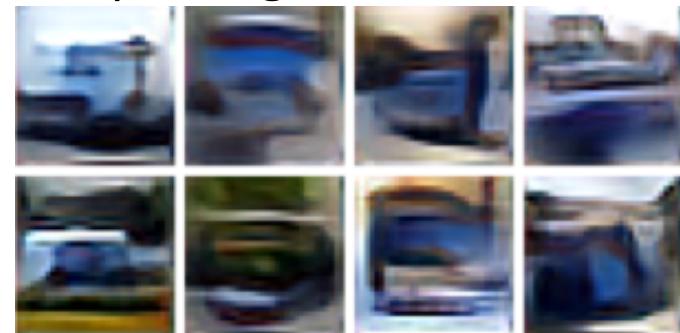
A **red school bus** parked in the parking lot



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



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



Flipping Backgrounds

A very large commercial plane flying **in clear skies**.



A very large commercial plane flying **in rainy skies**.



A herd of elephants walking across a **dry grass field**.



A herd of elephants walking across a **green grass field**.



Flipping Objects

The decadent chocolate desert is on the table.



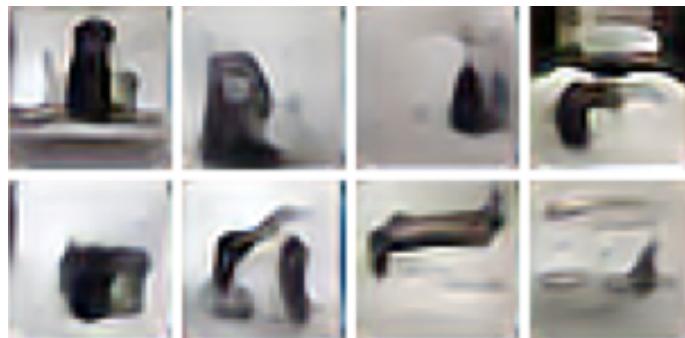
A bowl of bananas is on the table..



A vintage photo of a **cat**.



A vintage photo of a **dog**.



Qualitative Comparison

A group of people walk on a beach with surf boards

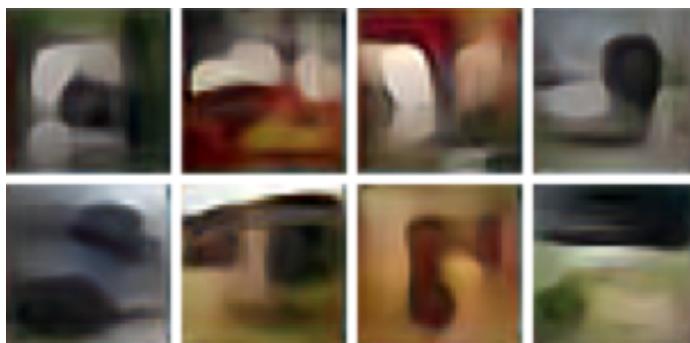
Our Model



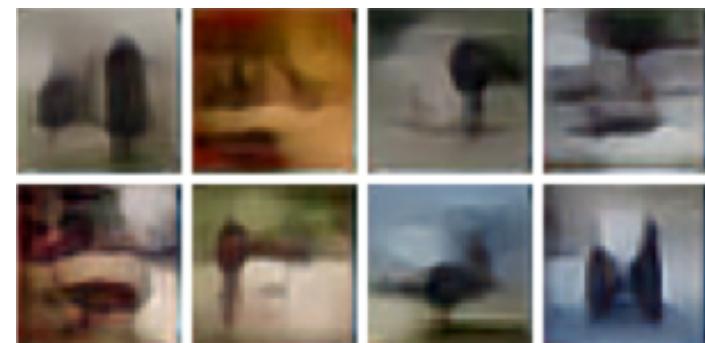
LAPGAN (Denton et. al. 2015)



Conv-Deconv VAE



Fully Connected VAE



Variational Lower-Bound

- We can estimate the variational lower-bound on the average test log-probabilities.

Model	Training	Test
Our Model	-1792,15	-1791,53
Skipthought-Draw	-1794,29	-1791,37
noAlignDraw	-1792,14	-1791,15

- At least we can see that we do not overfit to the training data, unlike many other approaches.

Novel Scene Compositions

A toilet seat sits open in the bathroom



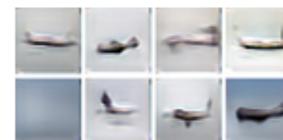
A toilet seat sits open in the grass field



Ask Google?



Bloomberg News

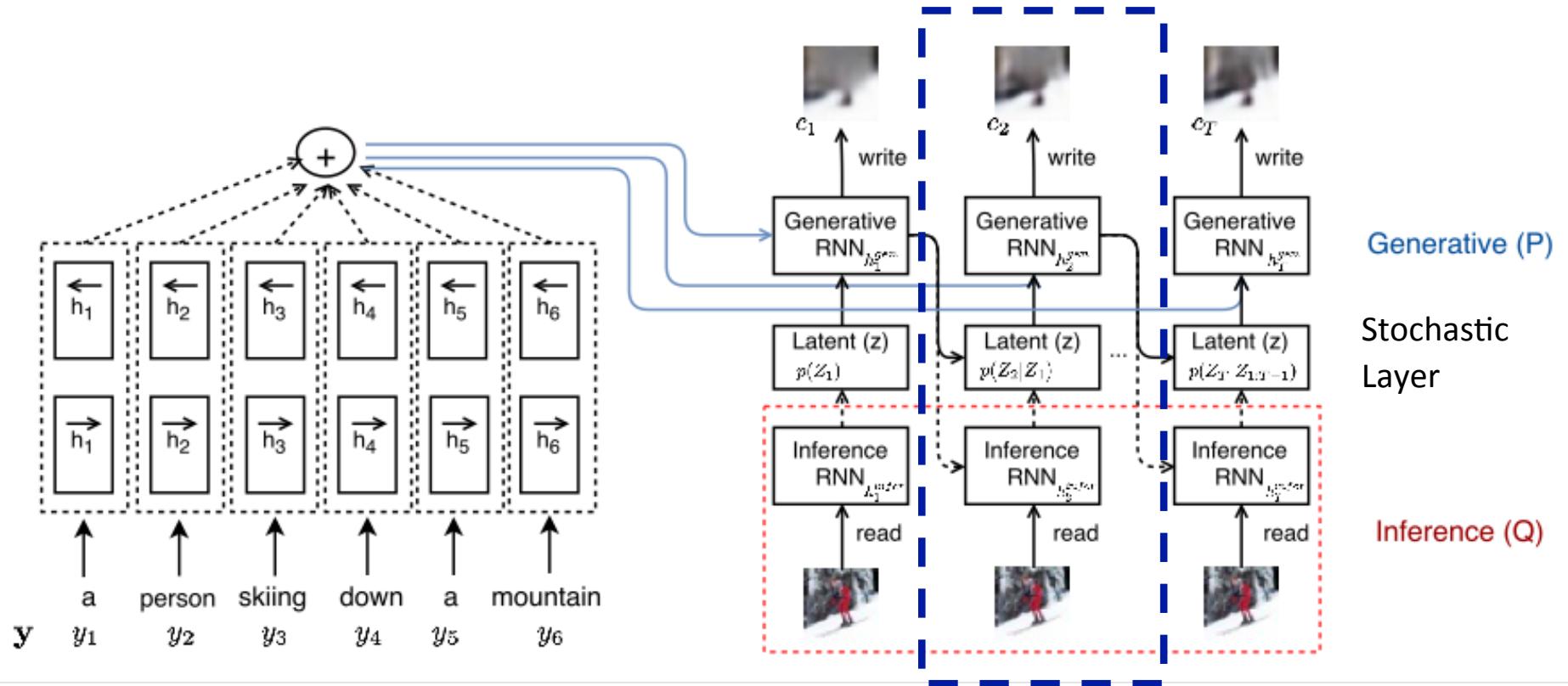


- A very large commercial plane flying in rainy skies. Source: University of Toronto

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

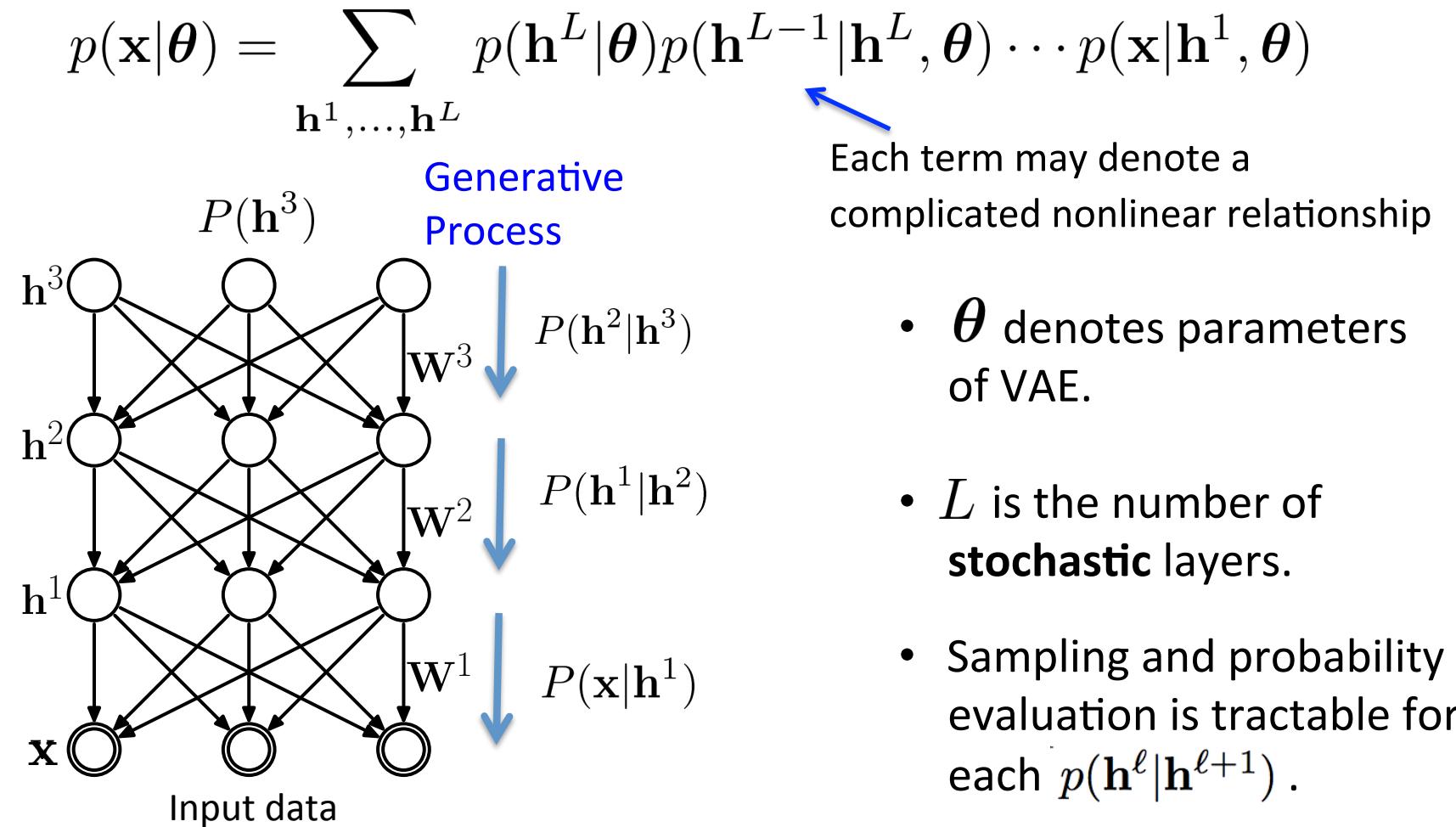
Overall Model



Focus on Variational Autoencoders (Kingma & Welling, 2014).

Variational Autoencoders (VAEs)

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



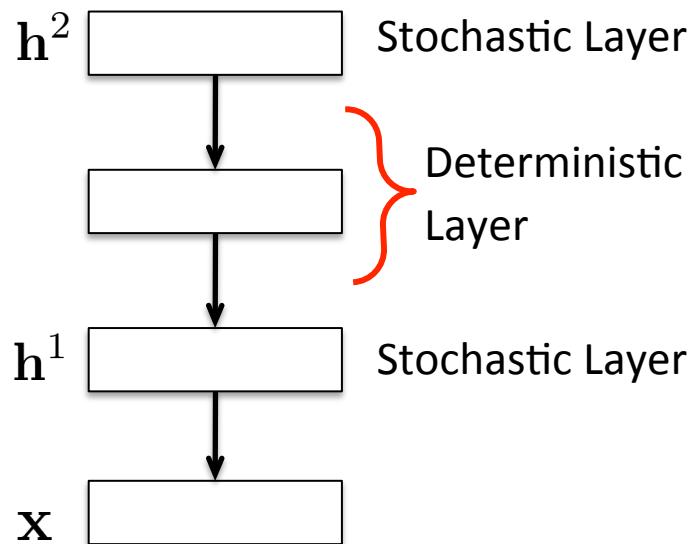
VAE: Example

- 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.

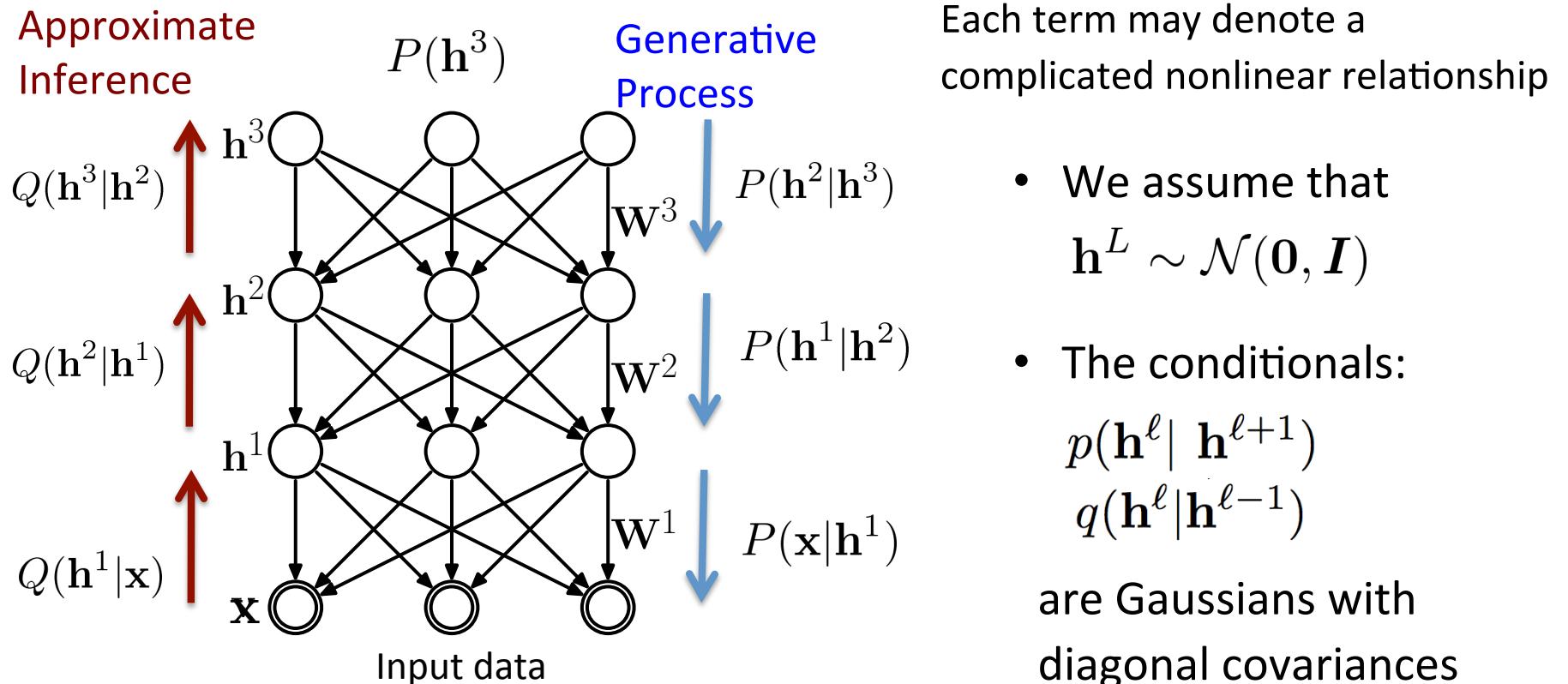


- $\boldsymbol{\theta}$ denotes parameters of VAE.
- L is the number of **stochastic** layers.
- Sampling and probability evaluation is tractable for each $p(\mathbf{h}^\ell|\mathbf{h}^{\ell+1})$.

Recognition Network

- The recognition model is defined in terms of an analogous factorization:

$$q(\mathbf{h}|\mathbf{x}, \boldsymbol{\theta}) = q(\mathbf{h}^1|\mathbf{x}, \boldsymbol{\theta})q(\mathbf{h}^2|\mathbf{h}^1, \boldsymbol{\theta}) \cdots q(\mathbf{h}^L|\mathbf{h}^{L-1}, \boldsymbol{\theta})$$



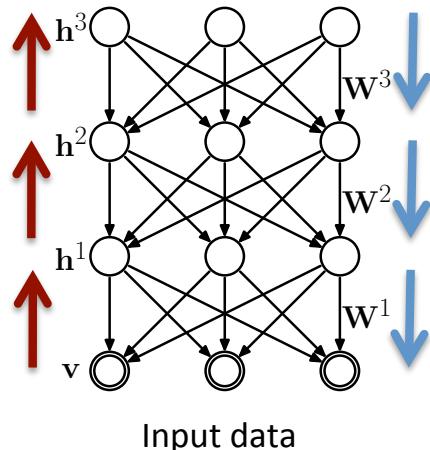
Variational Bound

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

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

$$\mathcal{L}(\mathbf{x}) = \log p(\mathbf{x}) - D_{\text{KL}}(q(\mathbf{h}|\mathbf{x}))||p(\mathbf{h}|\mathbf{x}))$$

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



- 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:

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

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

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

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

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

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

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

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:

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

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

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



Gradients can be computed by backprop



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

Computing the Gradients

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

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

- Approximate expectation by generating k samples from $\boldsymbol{\epsilon}$:

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

where we defined unnormalized importance weights:

$$w(\mathbf{x}, \mathbf{h}, \boldsymbol{\theta}) = p(\mathbf{x}, \mathbf{h}|\boldsymbol{\theta})/q(\mathbf{h}|\mathbf{x}, \boldsymbol{\theta})$$

- **VAE update**: Low variance as it uses the log-likelihood gradients with respect to the latent variables.

VAE: Assumptions

- Remember the variational bound:

$$\mathcal{L}(\mathbf{x}) = \log p(\mathbf{x}) - D_{KL}(q(\mathbf{h}|\mathbf{x}))||p(\mathbf{h}|\mathbf{x}))$$

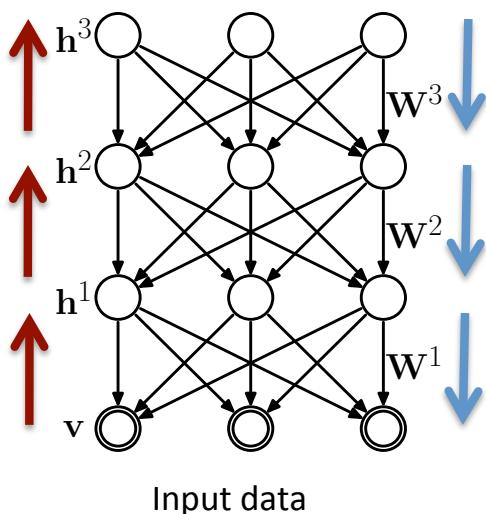
- The variational assumptions **must be approximately satisfied**.
- The posterior distribution must be approximately factorial (common practice) and predictable with a feed-forward net.
- We can relax these assumptions using a tighter lower bound on marginal log-likelihood.

Importance Weighted Autoencoders

- Consider the following k-sample importance weighting of the log-likelihood:

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

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



unnormalized
importance weights

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

- Mnih and Rezende, ICML 2016, further study these k-sample bounds with discrete latent variables.

Importance Weighted Autoencoders

- Consider the following k-sample importance weighting of the log-likelihood:

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

- 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:

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

Computing the Gradients

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

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

where we define normalized importance weights:

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

IWAEs vs. VAEs

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

$$\nabla_{\theta} \mathcal{L}_k(\mathbf{x}) \approx \sum_{i=1}^k \tilde{w}_i \left[\nabla_{\theta} \log w(\mathbf{x}, \mathbf{h}(\epsilon_i, \mathbf{x}, \theta), \theta) \right]$$

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

$$\nabla_{\theta} \mathcal{L}(\mathbf{x}) \approx \frac{1}{k} \sum_{i=1}^k \left[\nabla_{\theta} \log w(\mathbf{x}, \mathbf{h}(\epsilon_i, \mathbf{x}, \theta), \theta) \right]$$

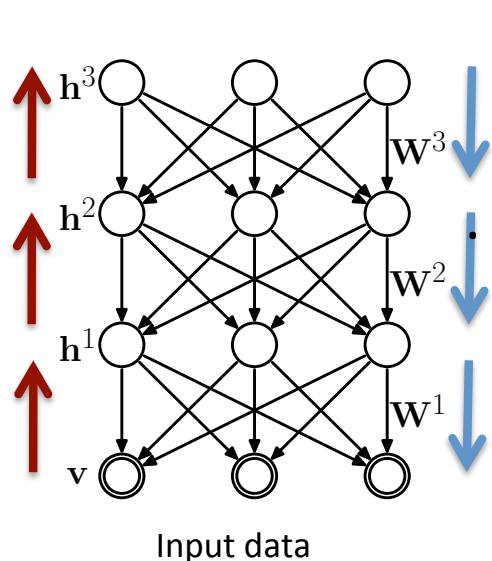
IWAE: Intuition

- The gradient of the log weights decomposes:

$$\nabla_{\theta} \log w(\mathbf{x}, \mathbf{h}(\epsilon_i, \mathbf{x}, \theta), \theta)$$

$$= \nabla_{\theta} \log p(\mathbf{x}, \mathbf{h}(\epsilon_i, \mathbf{x}, \theta) | \theta) - \log q(\mathbf{h}(\epsilon_i, \mathbf{x}, \theta) | \mathbf{x}, \theta)$$

Deterministic
decoder Deterministic
Encoder



First term:

- Decoder: encourages the generative model to assign high probability to each $\mathbf{h}^l | \mathbf{h}^{l+1}$.
- Encoder: encourages the recognition net to adjust its latent states h so that the generative network makes better predictions.

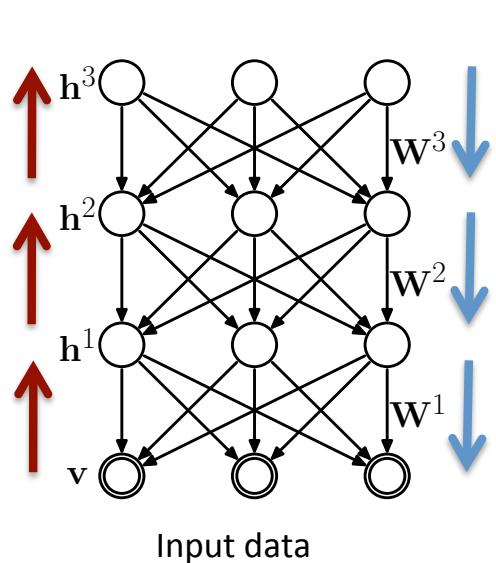
IWAE: Intuition

- The gradient of the log weights decomposes:

$$\nabla_{\theta} \log w(\mathbf{x}, \mathbf{h}(\epsilon_i, \mathbf{x}, \theta), \theta)$$

$$= \nabla_{\theta} \log p(\mathbf{x}, \mathbf{h}(\epsilon_i, \mathbf{x}, \theta) | \theta) - \log q(\mathbf{h}(\epsilon_i, \mathbf{x}, \theta) | \mathbf{x}, \theta)$$

Deterministic
decoder Deterministic
Encoder



Second term:

- Encoder: encourages the recognition network to have a spread-out distribution over predictions.

Computation with IWAEs

- **Dominant cost:** Requires forward and backward pass for each sample:

$$\nabla_{\theta} \log p(\mathbf{x}, \mathbf{h}(\epsilon_i, \mathbf{x}, \theta) | \theta)$$

- In practice: the samples are processed in parallel by replicating each training example k times within a mini-batch.
- Only forward pass is needed to compute importance weights.

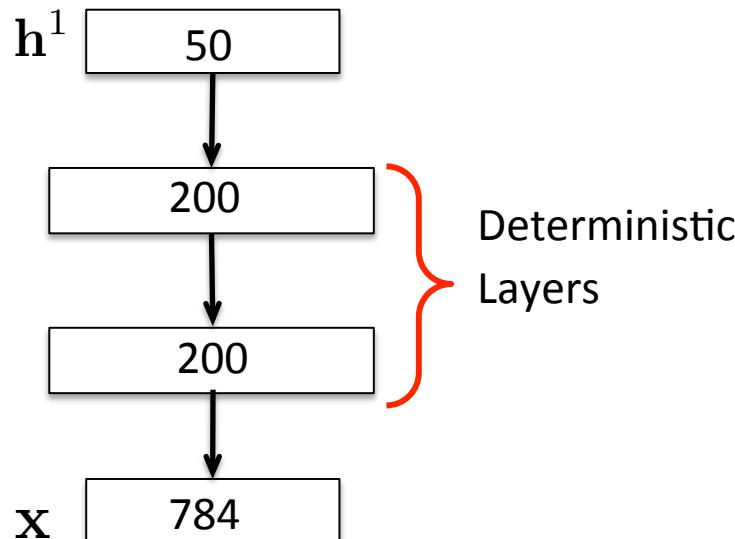
$$\nabla_{\theta} \mathcal{L}_k(\mathbf{x}) \approx \sum_{i=1}^k \tilde{w}_i \nabla_{\theta} \log w(\mathbf{x}, \mathbf{h}(\epsilon_i, \mathbf{x}, \theta), \theta)$$

- The sum can be approximated by sampling $i \sim \tilde{w}_i$.
- This requires k forward passes and 1 backward pass.

Two Architectures

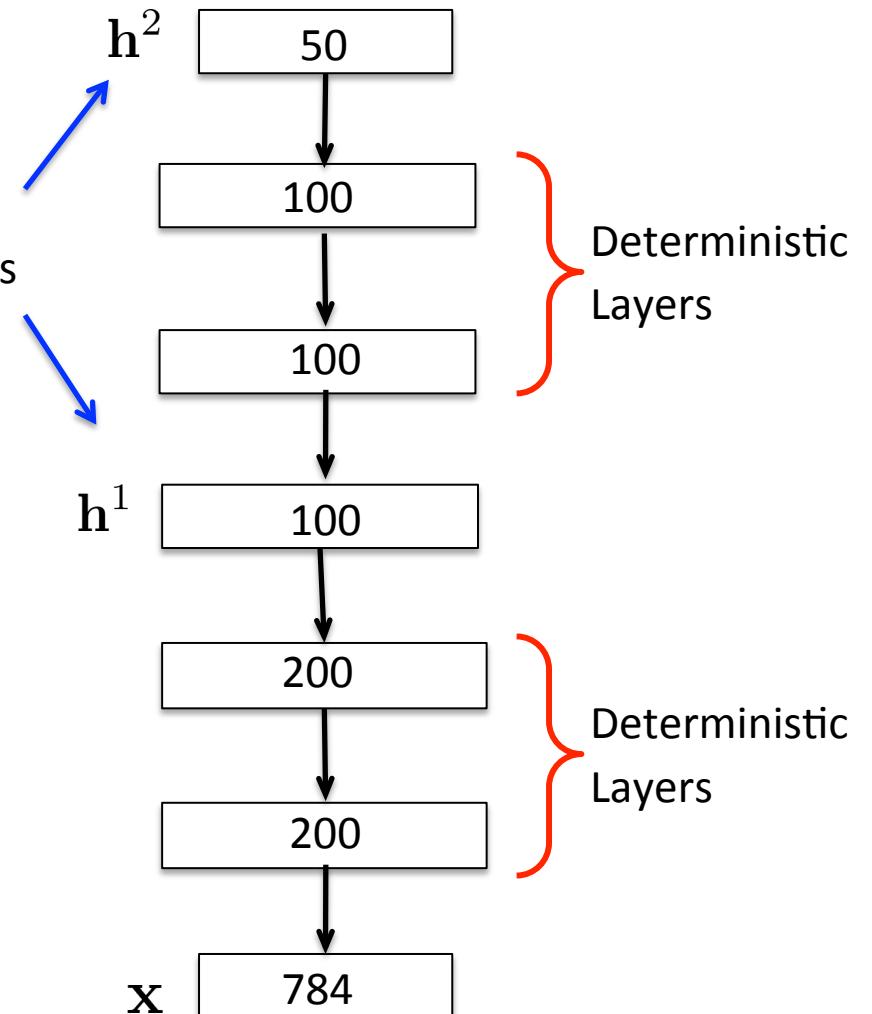
- For the MNIST experiments, we considered two architectures:

1-stochastic layer



Stochastic Layers

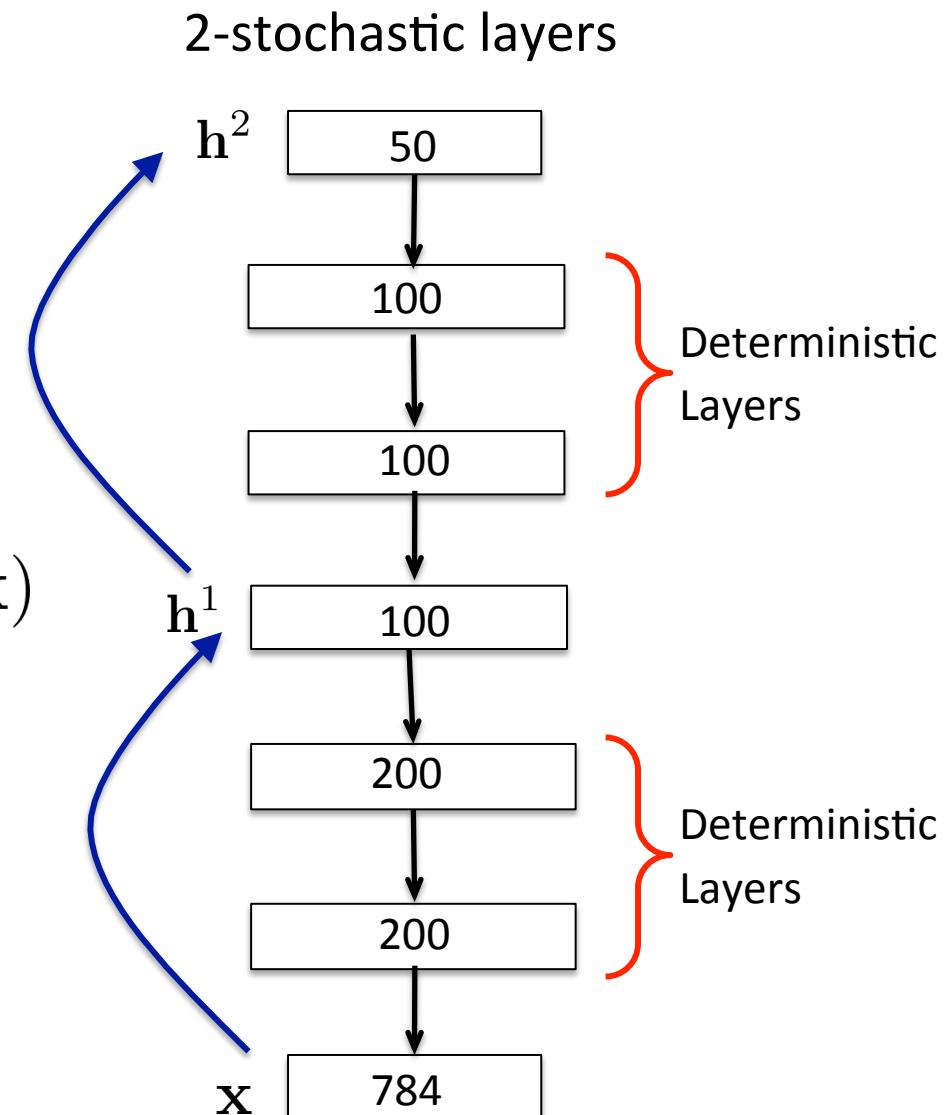
2-stochastic layers



Key Observation

- Recognition network has a stochastic layer.
- The approximate marginal posterior can be multimodal:

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



MNIST Results

MNIST

# stoch. layers	k	VAE		IWAE	
		NLL	active units	NLL	active units
1	1	86.76	19	86.76	19
	5	86.47	20	85.54	22
	50	86.35	20	84.78	25

MNIST Results

MNIST

# stoch. layers	k	VAE		IWAE	
		NLL	active units	NLL	active units
1	1	86.76	19	86.76	19
	5	86.47	20	85.54	22
	50	86.35	20	84.78	25
2	1	85.33	16+5	85.33	16+5
	5	85.01	17+5	83.89	21+5
	50	84.78	17+5	82.90	26+7

IWAEs vs. VAEs

First stage		
trained as	NLL	active units
VAE	86.76	19
IWAE, $k = 50$	84.78	25

IWAEs vs. VAEs

First stage

trained as	NLL	active units
VAE	86.76	19
IWAE, $k = 50$	84.78	25

Second stage

trained as	NLL	active units
IWAE, $k = 50$	84.88	22
VAE	86.02	23

Talk Roadmap

- Learning Deep Undirected Models
 - Restricted Boltzmann Machines
 - Deep Boltzmann Machines
- Learning Deep Directed Models
 - Helmholtz Machines
 - Variational & Importance Weighted Autoencoders
 - Stochastic (Hard) Attention Models
- Applications and Some Open Problems

Caption Generation



a car is parked in
the middle of nowhere .



a wooden table and chairs
arranged in a room .



a ferry boat on a marina
with a group of people .

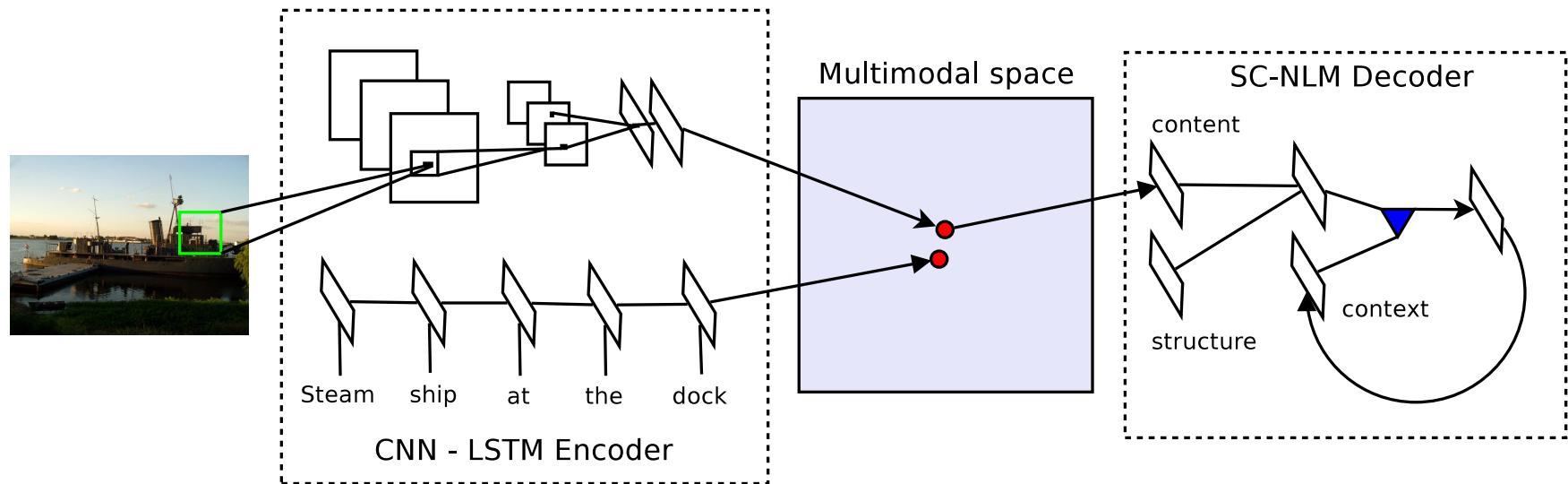


there is a cat sitting on a shelf .



a little boy with a bunch
of friends on the street .

Encode-Decode Framework



- Encoder: CNN and Recurrent Neural Net for a joint image-sentence embedding.
- Decoder: A neural language model for generating a sequence of words.

(Kiros, Salakhutdinov, Zemel, TACL 2015)

Caption Generation



the two birds are trying
to be seen in the water .
(can't count)



a giraffe is standing next
to a fence in a field .
(hallucination)



a parked car while
driving down the road .
(contradiction)

Caption Generation



the two birds are trying
to be seen in the water .
(can't count)



a giraffe is standing next
to a fence in a field .
(hallucination)



a parked car while
driving down the road .
(contradiction)



the handlebars are trying
to ride a bike rack .
(nonsensical)



a woman and a bottle of wine
in a garden .
(gender)

Caption Generation with Visual Attention

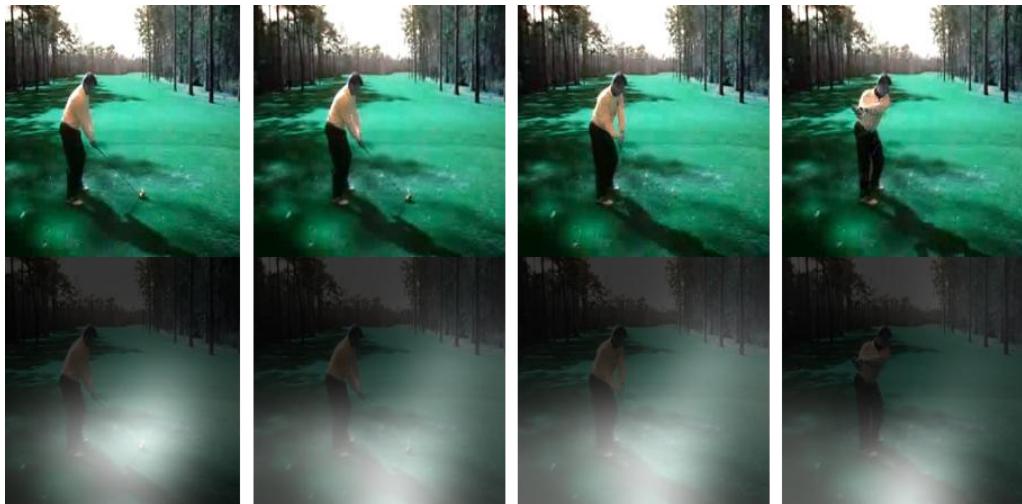


A man riding a horse
in a field.

(Xu et.al., ICML 2015)

Visual Attention

- Consider performing action recognition in a video:



- Instead of processing each frame, we can process only a small piece of each frame.
- Degree of interpretability: examine what signals the algorithm is using by seeing where it is looking.

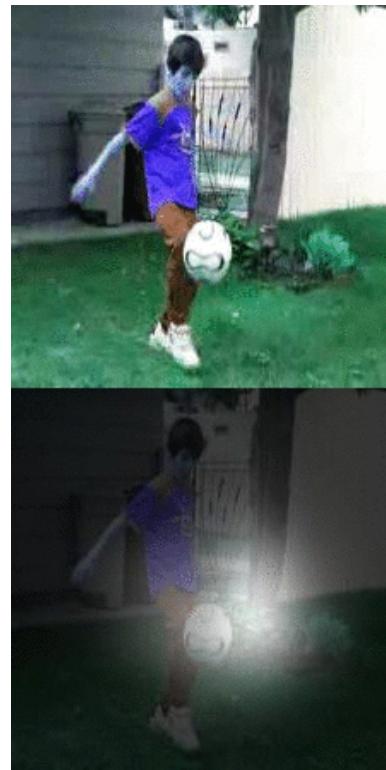
(Sharma, Kiros, Salakhutdinov, 2015)

Improving Action Recognition

Cycling



Soccer juggling



Horse back riding

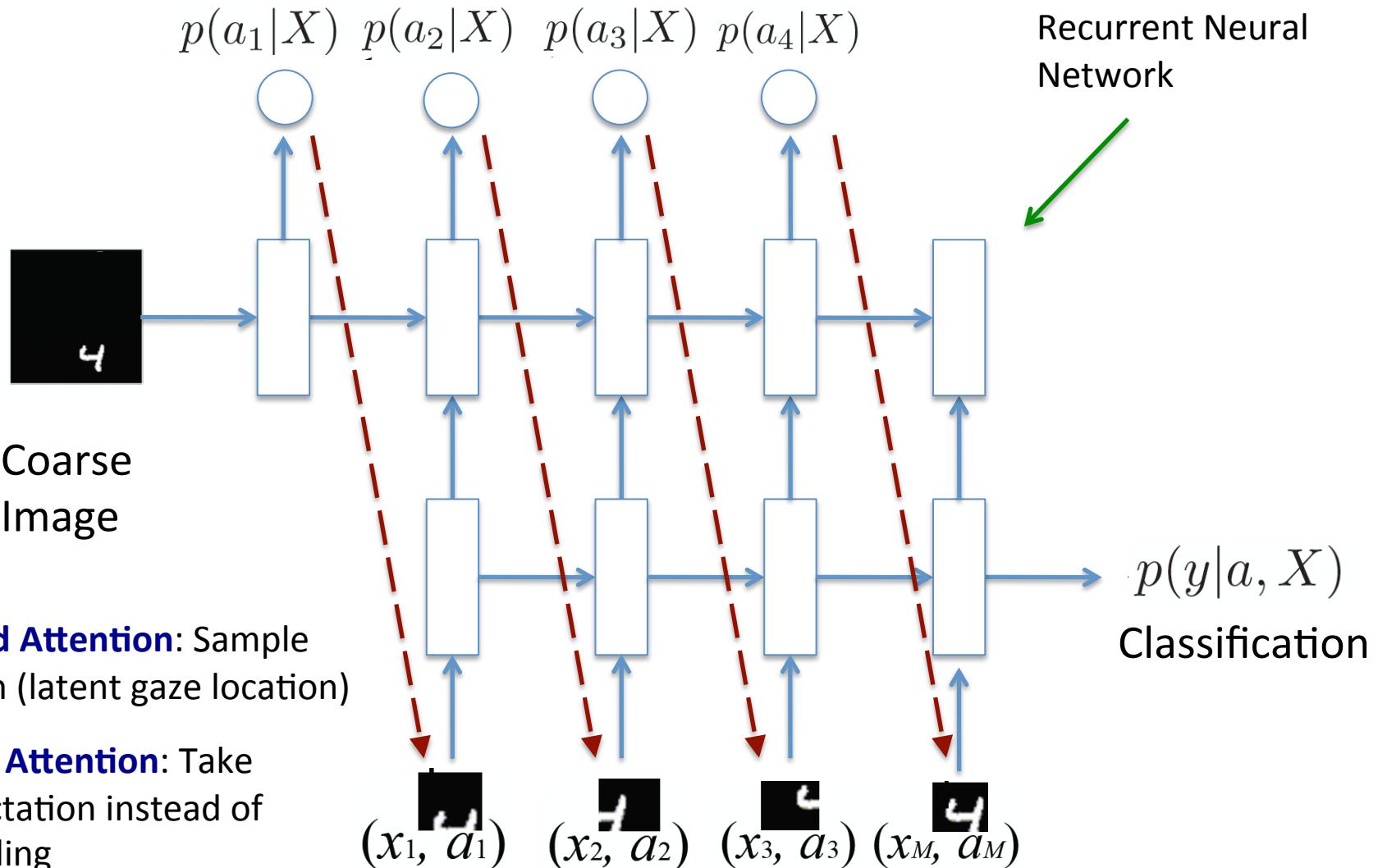


(Sharma, Kiros, Salakhutdinov, 2015)

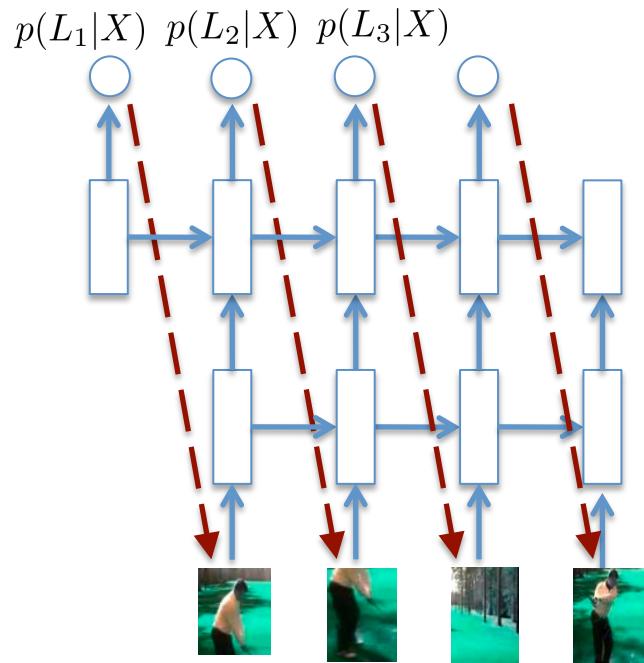
Recurrent Attention Model

Sample action:

$$\tilde{a}_1 \sim p(a_1|X) \quad \tilde{a}_2 \sim p(a_2|X)$$



Recurrent Attention Model



- Bahdanau et.al., ICLR 2015
- Mnih et.al., NIPS 2014
- Ba et.al., NIPS 2015, ICLR 2015
- Yao et.al., ICCV 2015
- Xu et. al., ICML 2015
- Larochelle and Hinton, 2010

- **Hard Attention:** Sample action (latent gaze location)
- **Soft Attention:** Take expectation instead of sampling

Model Definition

- We aim to maximize the probability of correct class by marginalizing over the actions (or latent gaze locations):

$$\mathcal{LL} = \log p(y|X, W) = \log \sum_a p(a|X, W)p(y|a, X, W).$$

where

- W is the set of parameters of the recurrent network.
- a is a set of actions (latent gaze locations, scale).
- X : is the input (e.g. image, video frame).

For clarity of presentation, I will sometimes omit conditioning on W or X . It should be obvious from the context.

Variational Learning

- A number of approaches use variational lower bound:

$$\mathcal{LL} = \log \sum_a p(a|X, W) p(y|a, X, W) \geq$$

$$\sum_a q(a|y, X) \log p(y, a|X, W) + \mathcal{H}[q] = \mathcal{F}.$$

- Here $q(a|y, X)$ is some approximation to posterior over the gaze locations.
- In the case where q is the prior, $q(a|y, X) = p(a|X, W)$, the variational bound becomes:

$$\mathcal{F} = \sum_a p(a|X, W) \log p(y|a, X, W).$$

Ba et.al., ICLR 2015
Mnih et.al., NIPS 2014

Variational Learning

$$\mathcal{F} = \sum_a p(a|X, W) \log p(y|a, X, W).$$

- Derivatives w.r.t model parameters:

$$\frac{\partial \mathcal{F}}{\partial W} = \sum_a p(a|X, W) \left[\frac{\partial \log p(y|a, X, W)}{\partial W} + \underbrace{\log p(y|a, X, W) \frac{\partial \log p(a|X, W)}{\partial W}}_{\text{Very bad term as it is unbounded.}} \right].$$

Very bad term as it is unbounded.
Introduces high variance in the estimator.

- Need to introduce heuristics (e.g. replacing this term with a 0/1 discrete indicator function, which leads to REINFORCE algorithm of Williams, 1992).

Ba et.al., ICLR 2015
Mnih et.al., NIPS 2014

Variational Learning

$$\mathcal{F} = \sum_a p(a|X, W) \log p(y|a, X, W).$$

- Derivatives w.r.t model parameters:

$$\frac{\partial \mathcal{F}}{\partial W} = \sum_a p(a|X, W) \left[\frac{\partial \log p(y|a, X, W)}{\partial W} + \log p(y|a, X, W) \frac{\partial \log p(a|X, W)}{\partial W} \right].$$

- The stochastic estimator of the gradient is given by:

$$\frac{\partial \mathcal{F}}{\partial W} \approx \frac{1}{M} \sum_{m=1}^M \left[\frac{\partial \log p(y|\tilde{a}^m, X, W)}{\partial W} + \log p(y|\tilde{a}^m, X, W) \frac{\partial \log p(\tilde{a}^m|X, W)}{\partial W} \right].$$

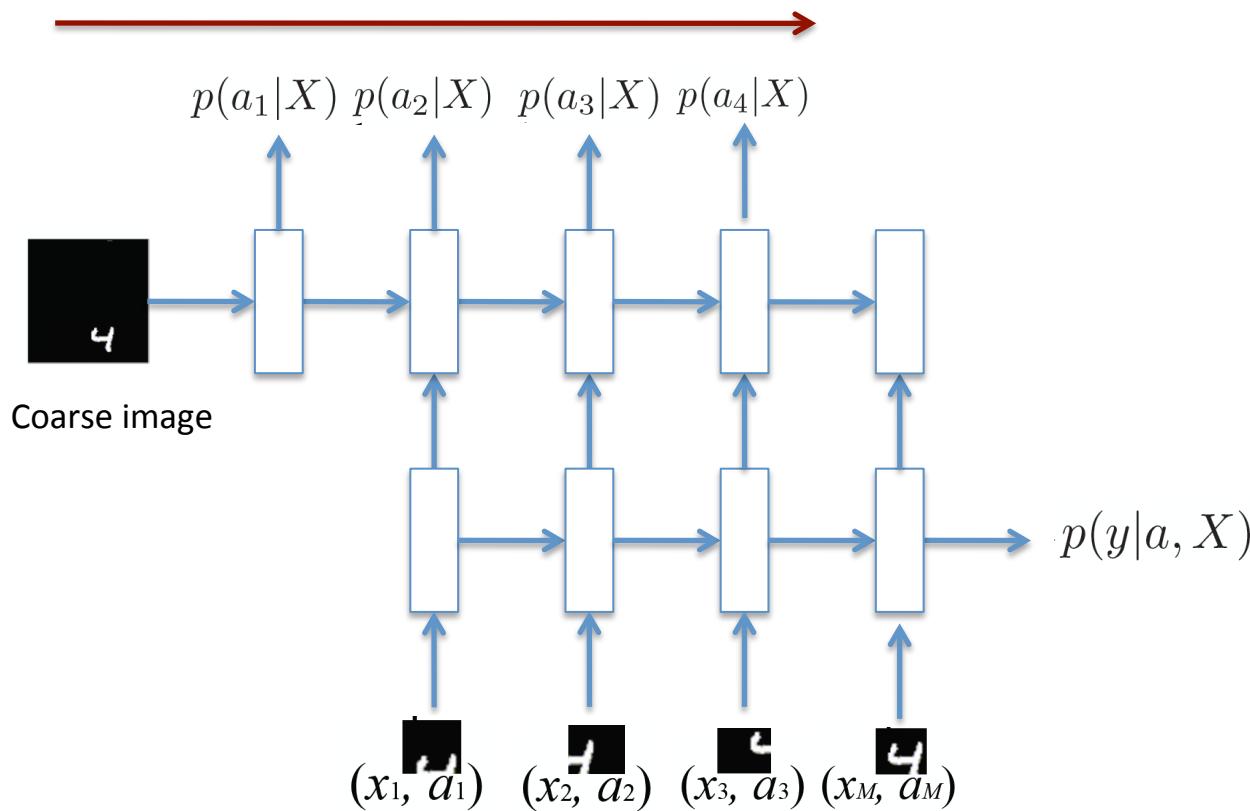
where we draw M samples from the prior: $\tilde{a}^m \sim p(a|X, W)$.

Sampling from the Prior

- Generate M samples from the prior $\tilde{a}^m \sim p(a|X, W)$.

$$\frac{\partial \mathcal{F}}{\partial W} \approx \frac{1}{M} \sum_{m=1}^M \left[\frac{\partial \log p(y|\tilde{a}^m, X, W)}{\partial W} + \log p(y|\tilde{a}^m, X, W) \frac{\partial \log p(\tilde{a}^m|X, W)}{\partial W} \right].$$

Run the network forward:



Key Observation

- We can maximize the **marginal class log-probability** directly without adhering to the variational lower bound:

$$\mathcal{LL} = \log p(y|X, W) = \log \sum_a p(a|X, W)p(y|a, X, W).$$

$$\frac{\partial \mathcal{LL}}{\partial W} = \frac{1}{Z} \sum_a p(a|X, W)p(y|a, X, W) \left[\frac{\partial \log p(y|a, X, W)}{\partial W} + \frac{\partial \log p(a|X, W)}{\partial W} \right]$$

where

Posterior: $p(a|y, X, W)$

$$Z = \sum_a p(a|X, W)p(y|a, X, W)$$

- We can use importance sampling to estimate required expectations.

(Ba, Grosse, Salakhutdinov, Frey, NIPS 2015)

Maximizing Marginal Likelihood

- Need to estimate:

$$\frac{\partial \mathcal{L}\mathcal{L}}{\partial W} = \frac{1}{Z} \sum_a p(a|X, W) p(y|a, X, W) \left[\frac{\partial \log p(y|a, X, W)}{\partial W} + \frac{\partial \log p(a|X, W)}{\partial W} \right],$$

- Let $q(a|y, X)$ be some approximation to the posterior:

$$q(a|y, X) \approx p(a|y, X, W).$$

- Using **Importance Sampling**, we obtain:

$$\tilde{w}^m = \frac{p(\tilde{a}^m|X, W) p(y|\tilde{a}^m, X, W)}{q(\tilde{a}^m|y, X)}, \quad \tilde{a}^m \sim q(a|y, X).$$

- The **stochastic estimator** of the gradient is given by:

$$\frac{\partial \mathcal{L}\mathcal{L}}{\partial W} \approx \frac{1}{\tilde{Z}} \sum_{m=1}^M \tilde{w}^m \left[\frac{\partial \log p(y|\tilde{a}^m, X, W)}{\partial W} + \frac{\partial \log p(\tilde{a}^m|X, W)}{\partial W} \right],$$

where $\tilde{Z} = \sum_m \tilde{w}^m$.

(Ba, Grosse, Salakhutdinov, Frey, NIPS 2015)

Comparing the Two Estimators

- Variational bound vs. Marginal likelihood:

$$\frac{\partial \mathcal{F}}{\partial W} \approx \frac{1}{M} \sum_{m=1}^M \left[\frac{\partial \log p(y|\tilde{a}^m, X, W)}{\partial W} + \log p(y|\tilde{a}^m, X, W) \frac{\partial \log p(\tilde{a}^m|X, W)}{\partial W} \right]$$
$$\frac{\partial \mathcal{LL}}{\partial W} \approx \frac{1}{\tilde{Z}} \sum_{m=1}^M \tilde{w}^m \left[\frac{\partial \log p(y|\tilde{a}^m, X, W)}{\partial W} + \frac{\partial \log p(\tilde{a}^m|X, W)}{\partial W} \right]$$

Very bad term, as it is unbounded

- The performance gain from importance sampling heavily relies on an appropriate choice of the proposal proposal distribution q!

When approximate posterior q is equal to the prior, this approach is equivalent to Tang and Salakhutdinov for learning generative networks (NIPS 2013). It is also similar to the Reweighted Wake-Sleep of Bornschein and Bengio (ICLR 2015).

Another Key Observation

- Using **finite number of samples M**, our importance sampling estimator can be viewed as the gradient ascent on the following objective:

$$\mathbb{E} \left[\log \frac{1}{M} \sum_{m=1}^M \tilde{w}^m \right]$$

- Using Jensen's inequality we obtain:

$$\mathbb{E} \left[\log \frac{1}{M} \sum_{m=1}^M \tilde{w}^m \right] \leq \log \mathbb{E} \left[\frac{1}{M} \sum_{m=1}^M \tilde{w}^m \right] = \log \mathbb{E} [\tilde{w}^m] = \mathcal{LL}$$

- Hence in expectation, we are optimizing a lower bound on the marginal likelihood (although the variance can be high).
- The bound becomes tighter as we increase M.

Another Key Observation

- Using **finite number of samples M**, our importance sampling estimator can be viewed as the gradient ascent on the following objective:

$$\mathbb{E} \left[\log \frac{1}{M} \sum_{m=1}^M \tilde{w}^m \right]$$

- Using Jensen's inequality we obtain:

$$\mathbb{E} \left[\log \frac{1}{M} \sum_{m=1}^M \tilde{w}^m \right] \leq \log \mathbb{E} \left[\frac{1}{M} \sum_{m=1}^M \tilde{w}^m \right] = \log \mathbb{E} [\tilde{w}^m] = \mathcal{LL}$$

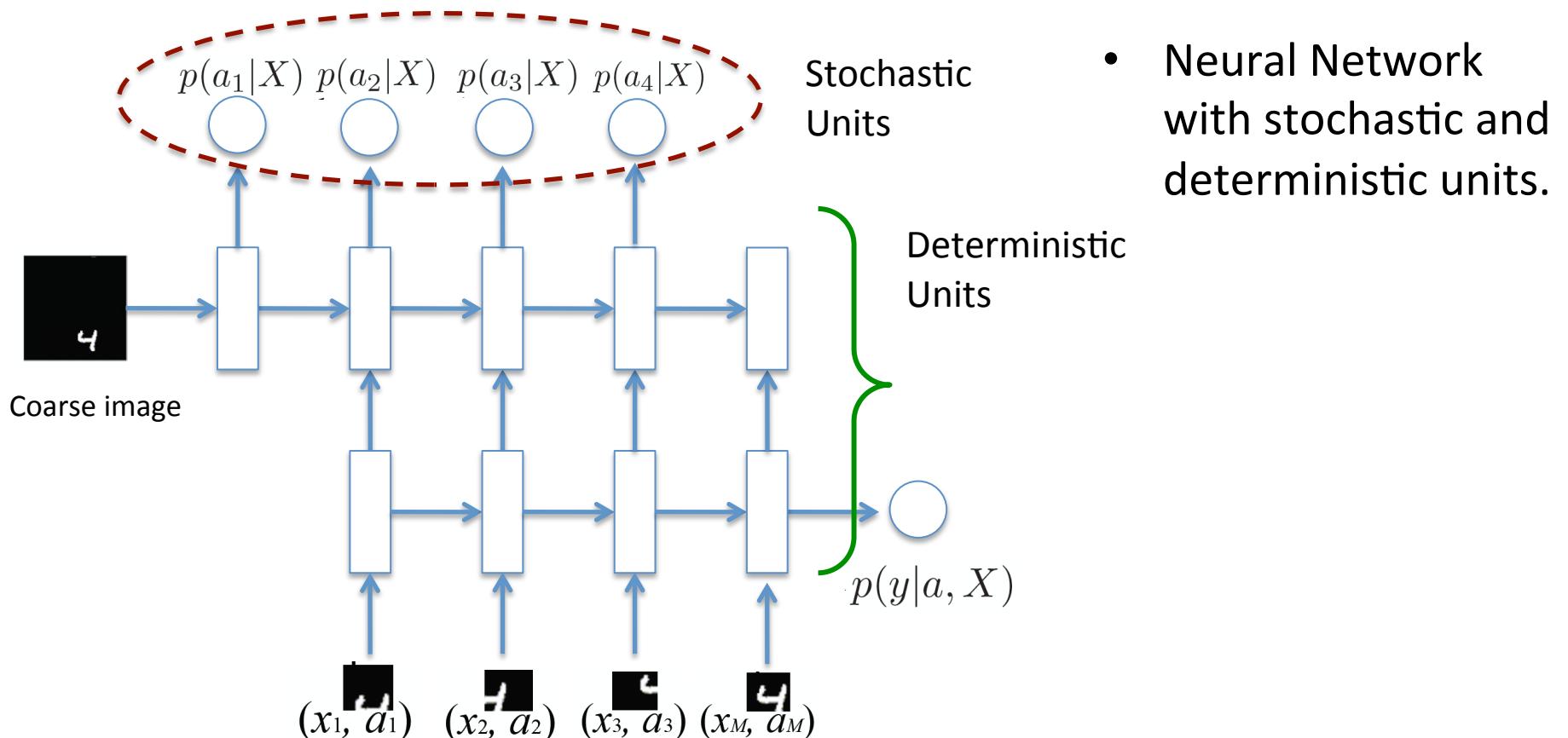
- However, our proposed bound is at least as accurate as the variational bound:

$$\mathcal{F} = \mathbb{E} [\log \tilde{w}^m] = \mathbb{E} \left[\frac{1}{M} \sum_{m=1}^M \log \tilde{w}^m \right] \leq \mathbb{E} \left[\log \frac{1}{M} \sum_{m=1}^M \tilde{w}^m \right]$$

Relationship To Helmholtz Machines

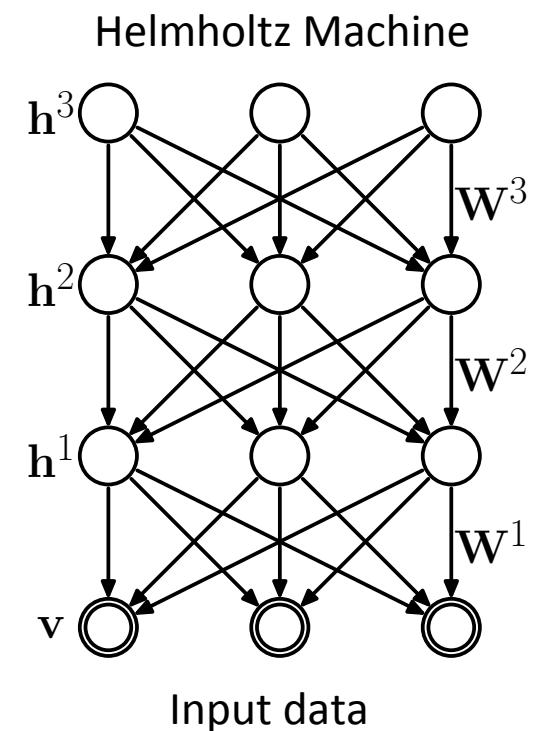
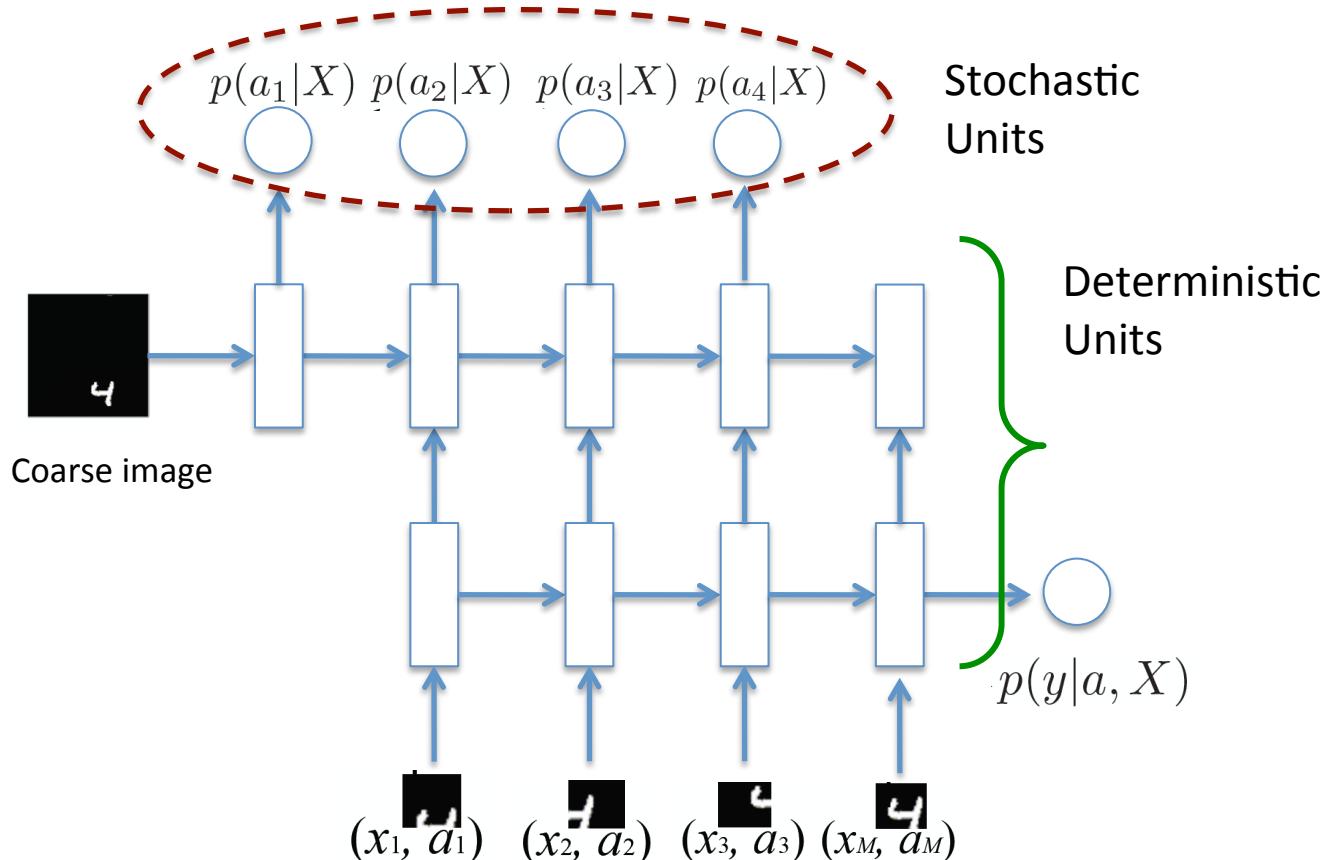
- **Goal:** Maximize the probability of correct class (sequence of words) by marginalizing over the actions (or latent gaze locations):

$$\mathcal{LL} = \log p(y|X, W) = \log \sum_a p(a|X, W)p(y|a, X, W).$$



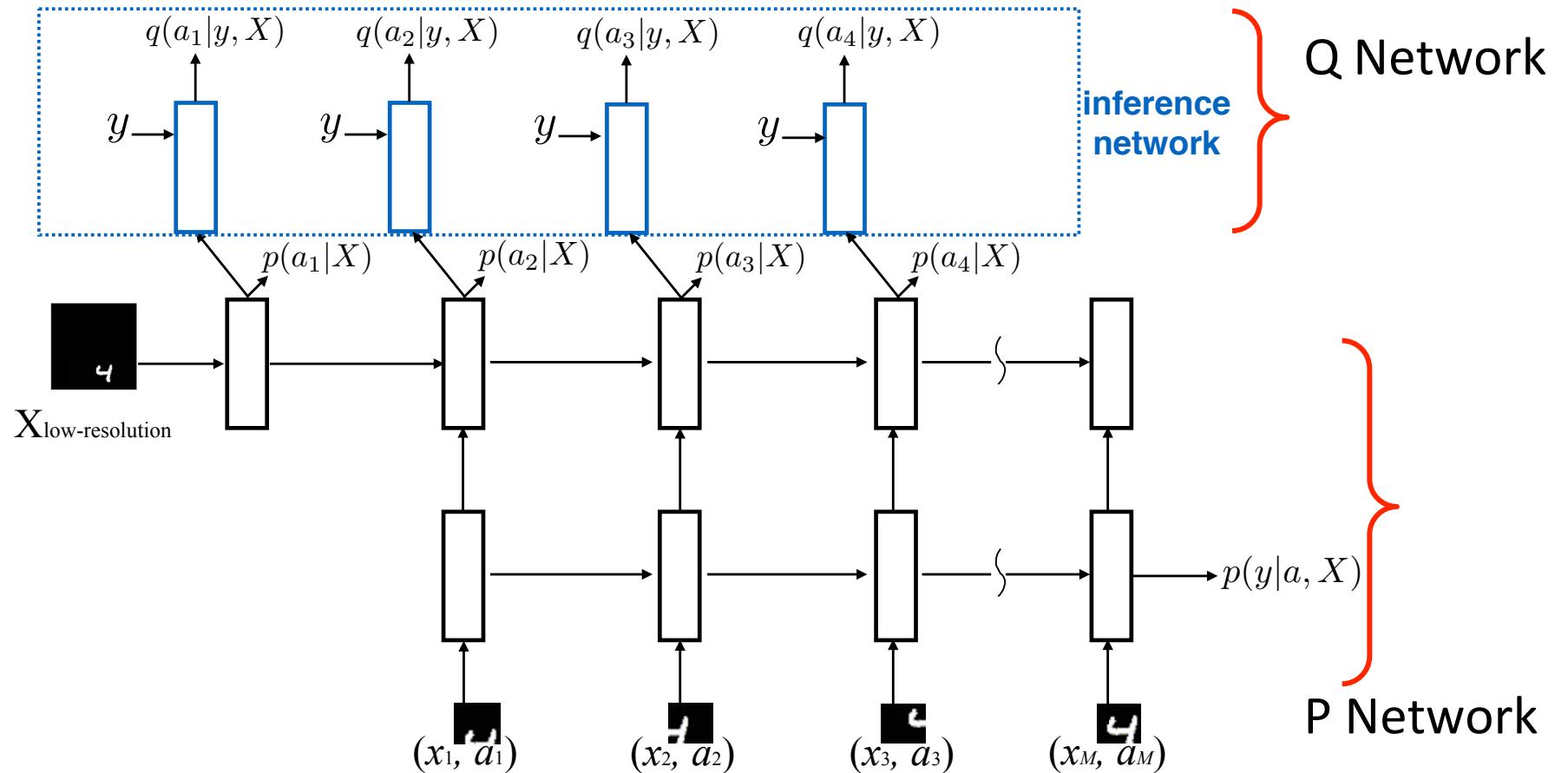
Relationship To Helmholtz Machines

- View this model as a conditional Helmholtz Machine with stochastic and deterministic units.
- Can use Wake-Sleep, Re-weighted Wake Sleep, variational autoencoders, and their variants to learn good attention policy!



The Wake-Sleep Recurrent Attention Model

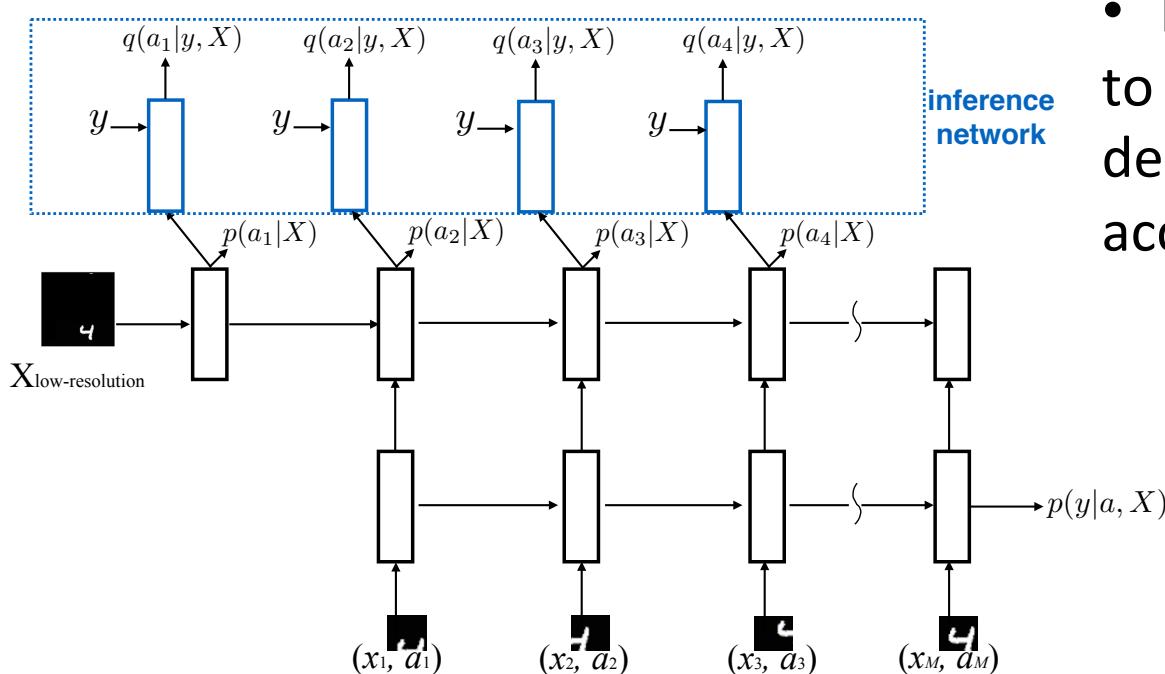
- We can learn both: generative model P and recognition model Q.



Training Inference Network

- We train an inference network to predict glimpses, **given the observations as well as the class label**, where the network should look to correctly predict that class.

$$q(a|y, X, \theta) = \prod_{n=1}^N q(a_n | y, X, \theta, a_{1:n-1}).$$



- This distribution is analogous to the prior, except that each decision also takes into account the class label \mathbf{y} .

Training Inference Network

- To train q network we optimize:

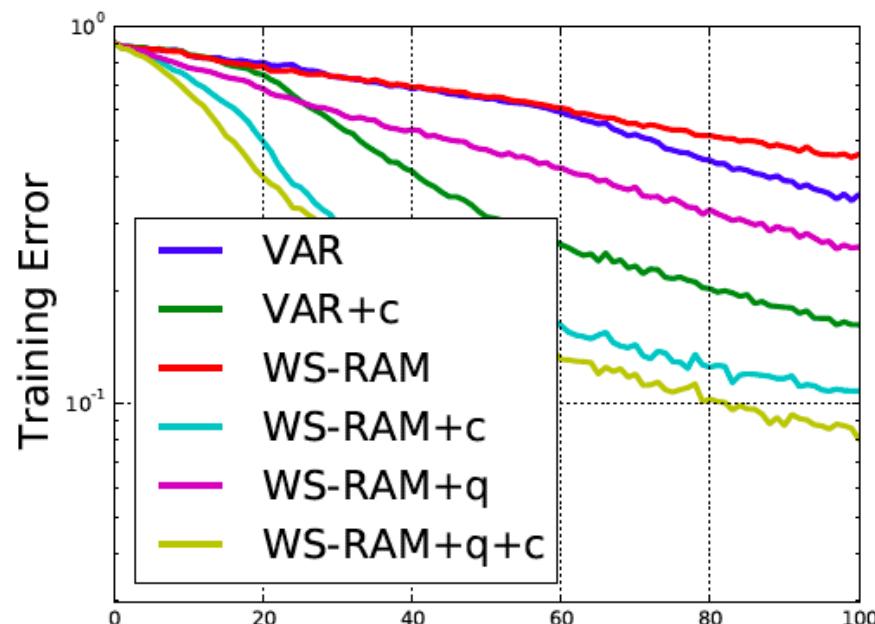
$$\frac{\partial D_{KL}(p||q)}{\partial \theta} = \mathbb{E}_{p(a|y, X, W)} \left[\frac{\partial \log q(a|y, X, \theta)}{\partial \theta} \right]$$

- Using importance sampling, we get the following stochastic estimate of the gradient:

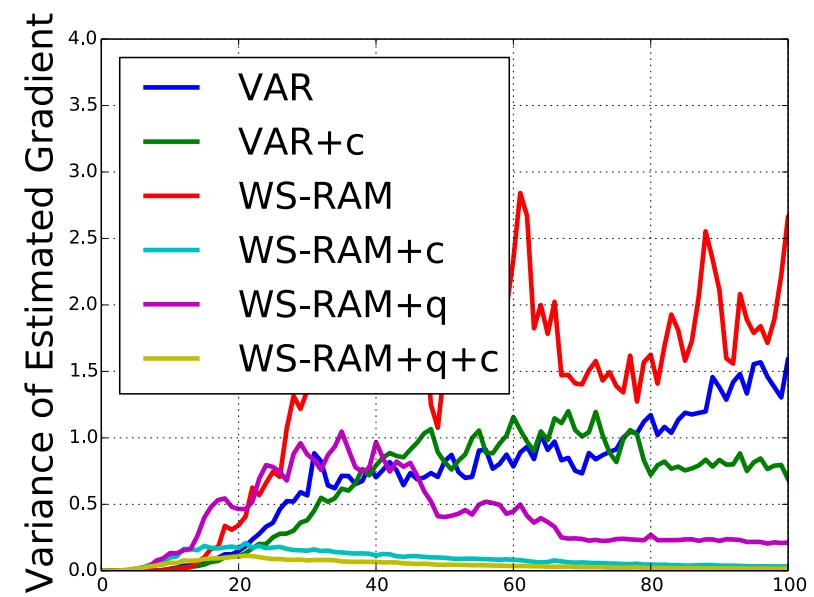
$$\frac{\partial D_{KL}(p||q)}{\partial \theta} \approx \frac{1}{Z} \sum_{m=1}^M \tilde{w}^m \frac{\partial \log q(\tilde{a}^m|y, X, \theta)}{\partial \theta}$$

- In fact we can reuse the importance weights computed for the attention model update.

MNIST Example



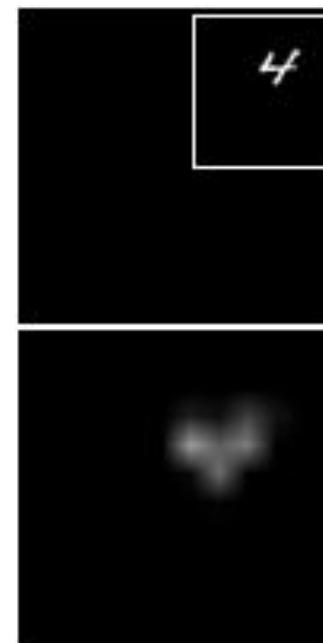
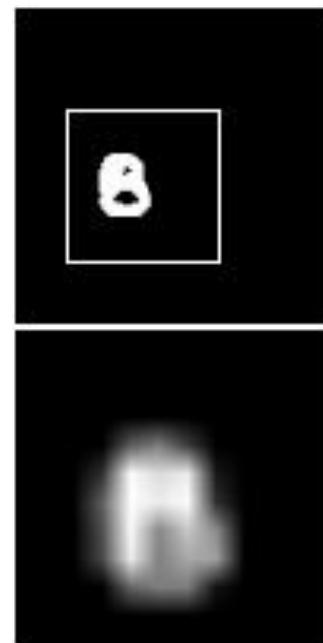
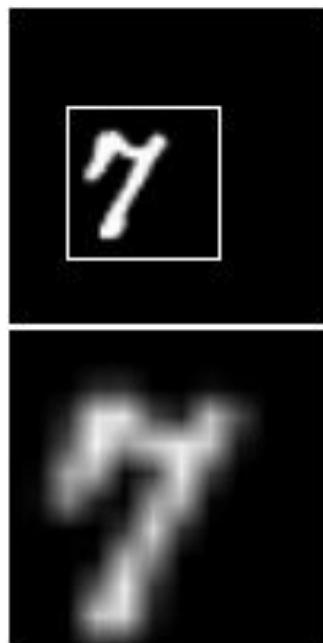
Training error as a number of updates.



Variance of Estimated Gradients.

MNIST Attention Demo

- Actions contain:
 - Location: 2-d Gaussian latent variable
 - Scale: 3-way softmax over 3 different scales



Hard vs. Soft Attention

- Soft attention models:
 - **Computationally expensive.** They have to examine every image location. Hard to scale to large video datasets.
 - **Deterministic.** They can be trained by backprop.
- Hard attention models:
 - **Computationally more efficient.** They need to process only small part of each image frame.
 - **Stochastic.** Require some form of sampling, because they must make discrete choices.
- Research is taking place on both fronts!

Talk Roadmap

- Learning Deep Undirected Models
 - Restricted Boltzmann Machines
 - Deep Boltzmann Machines
- Learning Deep Directed Models
 - Helmholtz Machines
 - Variational & Importance Weighted Autoencoders
 - Stochastic (Hard) Attention Models
- Applications and Some Open Problems

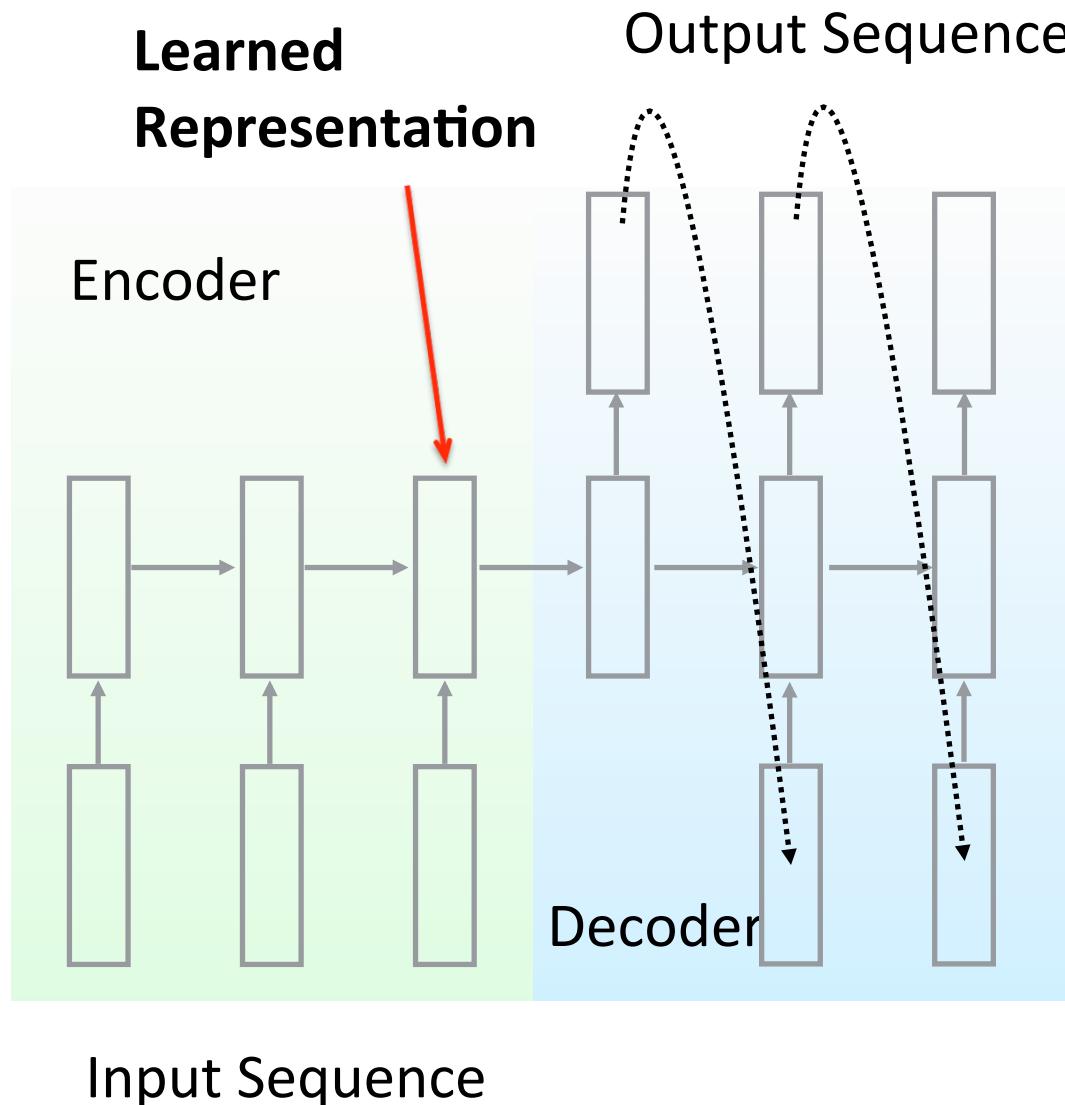
(Some) Open Problems

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

(Some) Open Problems

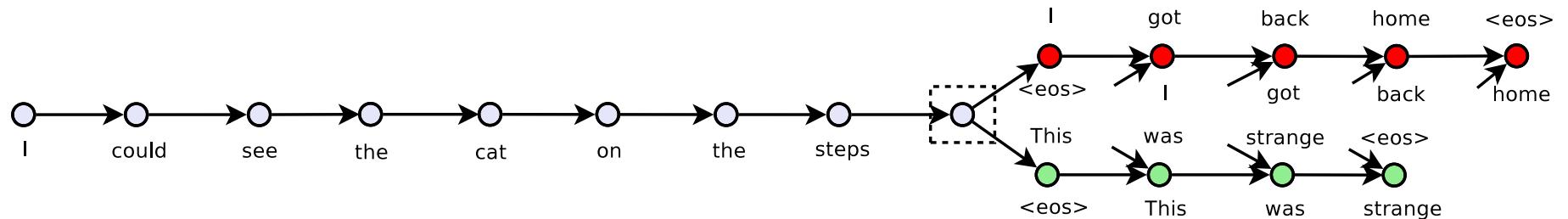
- Unsupervised Learning / Transfer Learning / One-Shot Learning
- Reasoning, Attention, and Memory
- Natural Language Understanding
- Deep Reinforcement Learning

Sequence to Sequence Learning



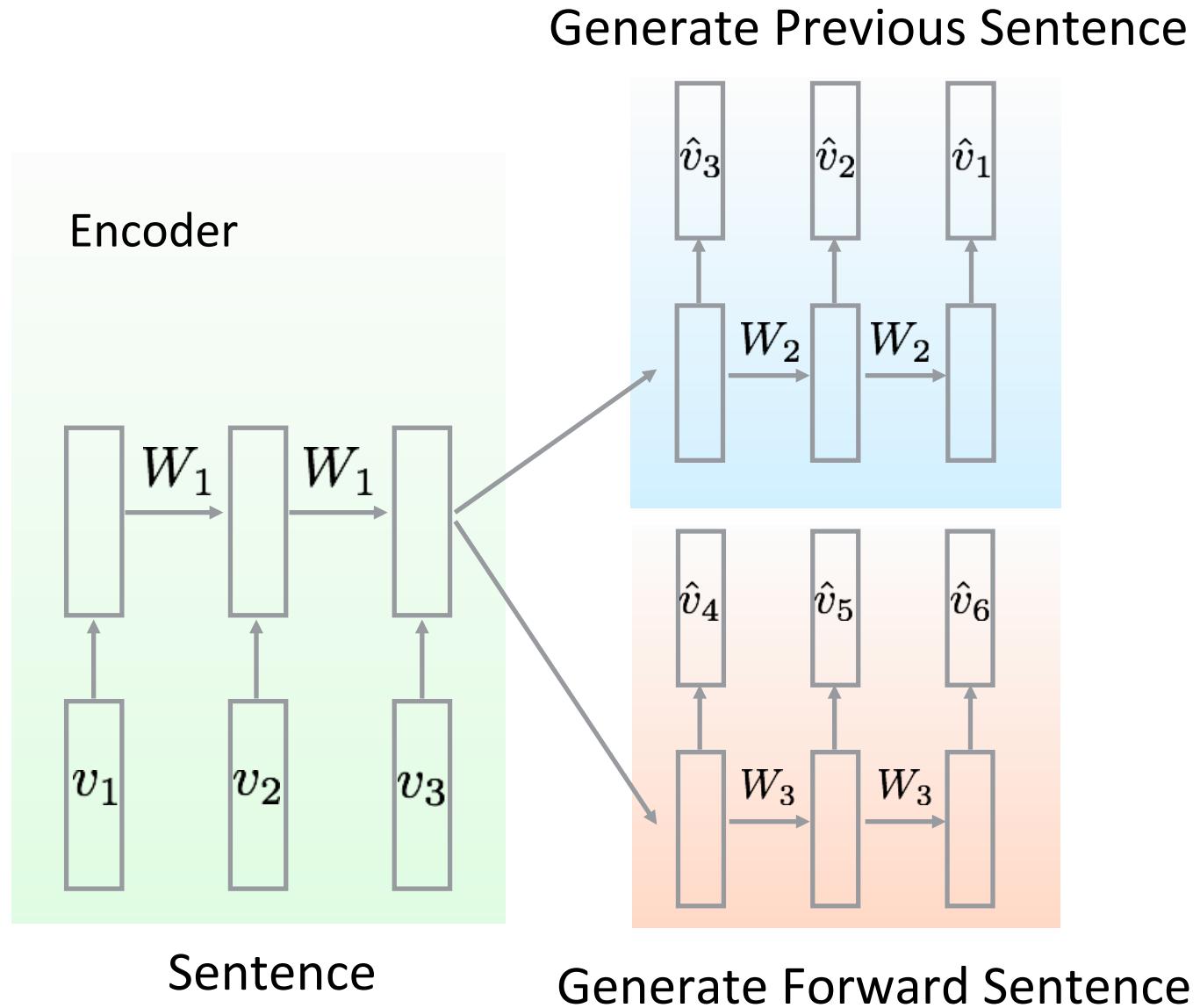
- RNN Encoder-Decoders for Machine Translation (Sutskever et al. 2014; Cho et al. 2014; Kalchbrenner et al. 2013, Srivastava et.al., 2015)

Skip-Thought Model



- Given a tuple (s_{i-1}, s_i, s_{i+1}) of contiguous sentences:
 - the sentence s_i is encoded using LSTM.
 - the sentence s_i attempts to reconstruct the previous sentence and next sentence s_{i+1} .
- The input is the sentence triplet:
 - I got back home.
 - I could see the cat on the steps.
 - This was strange.

Skip-Thought Model



Learning Objective

- **Objective:** The sum of the log-probabilities for the next and previous sentences conditioned on the encoder representation:

$$\sum_t \log P(w_{i+1}^t | w_{i+1}^{<t}, \mathbf{h}_i) + \sum_t \log P(w_{i-1}^t | w_{i-1}^{<t}, \mathbf{h}_i)$$

representation of
encoder 

Forward sentence Previous sentence

- **Data:** Book-11K corpus:

# of books	# of sentences	# of words	# of unique words
11,038	74,004,228	984,846,357	1,316,420

Semantic Relatedness

	Method	<i>r</i>	ρ	MSE
SemEval 2014 sub- missions	Illinois-LH [18]	0.7993	0.7538	0.3692
	UNAL-NLP [19]	0.8070	0.7489	0.3550
	Meaning Factory [20]	0.8268	0.7721	0.3224
	ECNU [21]	0.8414	—	—
Results reported by Tai et.al.	Mean vectors [22]	0.7577	0.6738	0.4557
	DT-RNN [23]	0.7923	0.7319	0.3822
	SDT-RNN [23]	0.7900	0.7304	0.3848
	LSTM [22]	0.8528	0.7911	0.2831
	Bidirectional LSTM [22]	0.8567	0.7966	0.2736
	Dependency Tree-LSTM [22]	0.8676	0.8083	0.2532
Ours	uni-skip	0.8477	0.7780	0.2872
	bi-skip	0.8405	0.7696	0.2995
	combine-skip	0.8584	0.7916	0.2687
	combine-skip+COCO	0.8655	0.7995	0.2561

- Outperform all previous systems from the SemEval 2014 competition.

Semantic Relatedness Recurrent Neural Network

- How similar the two sentences are on the scale 1 to 5?

Ground Truth 5.0

Prediction 4.9

A man is driving a car.

A car is being driven by a man.

Ground Truth 2.9

Prediction 3.5

A little girl is looking at a woman in costume.

A little girl in costume looks like a woman.

Ground Truth 2.6

Prediction 4.4

A person is performing tricks on a motorcycle

The performer is tricking a person on a motorcycle

Neural Story Telling



**Sample from the Generative Model
(recurrent neural network):**

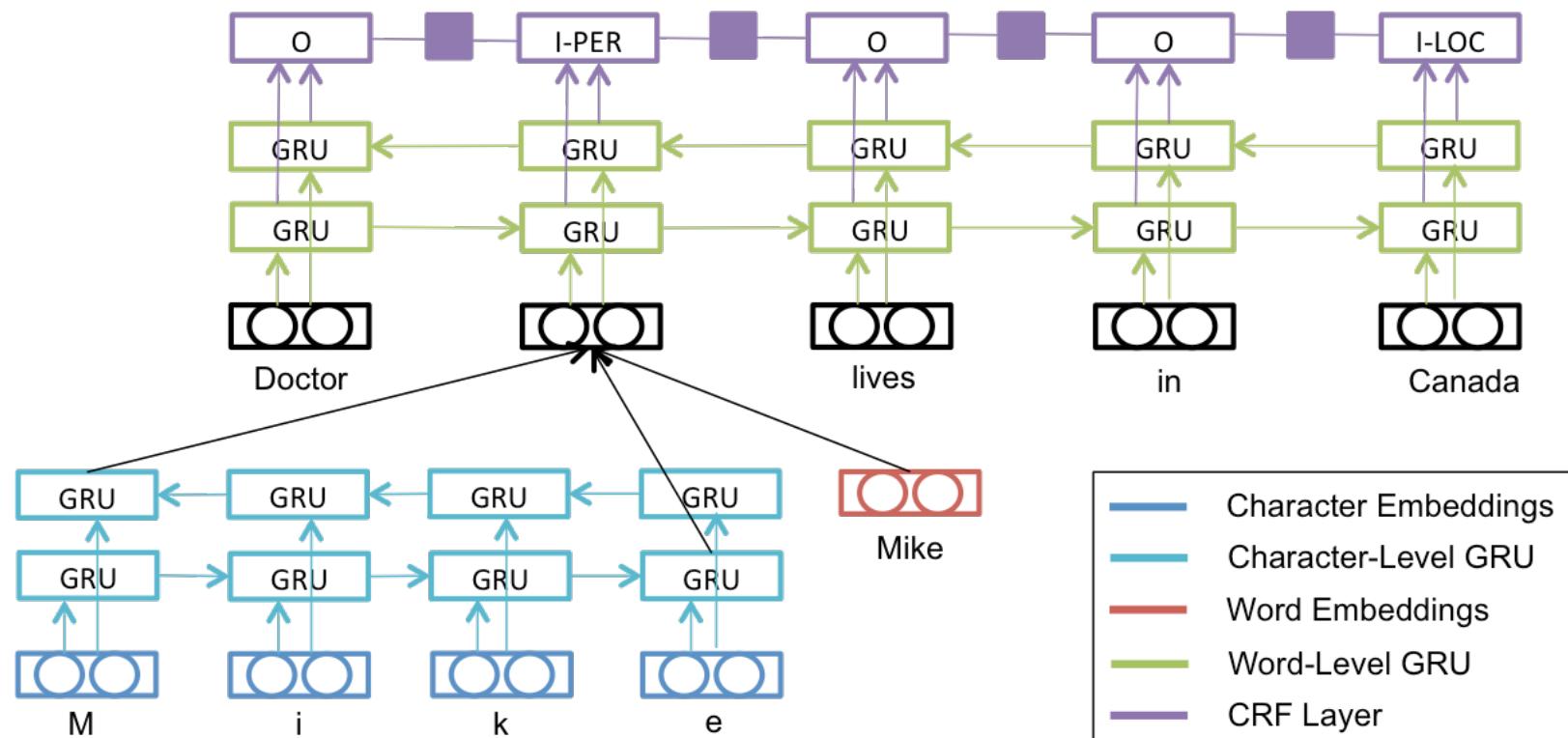
We were barely able to catch the breeze at the beach, and it felt as if someone stepped out of my mind.

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

(Kiros et.al., NIPS 2015)

Hierarchical RNNs

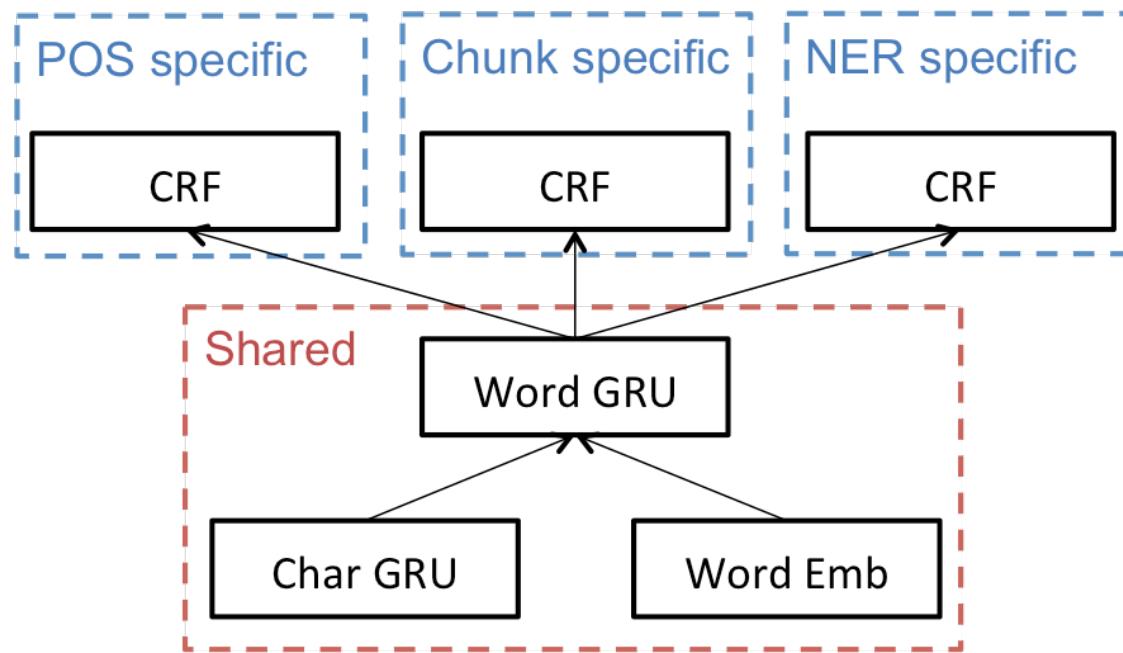
- Hierarchical Bidirectional RNN with CRF using both word-level and character- level RNNs.



(Yang, Salakhutdinov, Cohen, 2016,
Lample et al., NAACL 2016)

Hierarchical RNNs

- Hierarchical Bidirectional RNN with CRF using both word-level and character- level RNNs.



- State-of-the art performance in multiple languages on several tasks including POS tagging, chunking, and NER.

(Yang, Salakhutdinov, Cohen, 2016)

(Some) Open Problems

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

One-Shot Learning

a এ অ গ
ু অ প গ
স ম হ র
শ দ ক স

(Lake, Salakhutdinov, Tenenbaum, Science, 2015)

One-Shot Learning

“zarc”

a	କୁ	ବ୍ରା	ଗୁଣ
ଲୁ	ଅ	ଧୁ	ଶୁଣ
ମ	ମୁ	ହୁ	ତୁଳ
ଫୁ	ଅୁ	କୁଳ	ଦୁଇ

One-Shot Learning

“zarc”

a	କେ	ଅ	ଏ
ଲୁ	ଅ	ପୁ	କୁ
ମ	ମୁ	ହେ	ବ୍ର
ଫୁ	ଆ	କୁଣ୍ଡା	ସୁନ୍ଦା

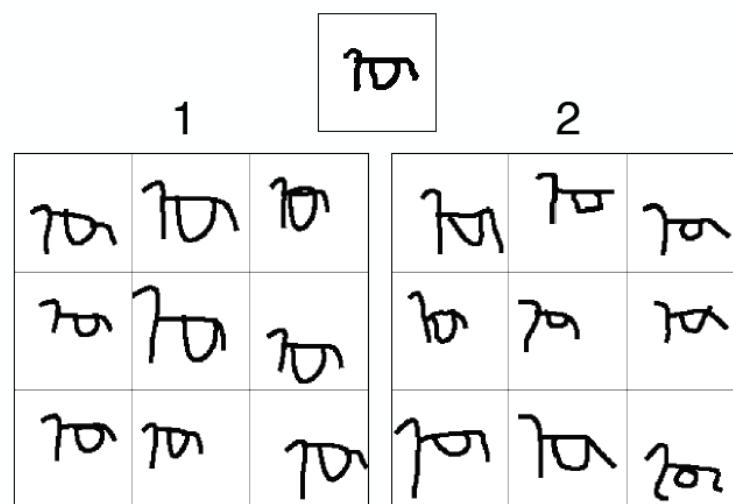
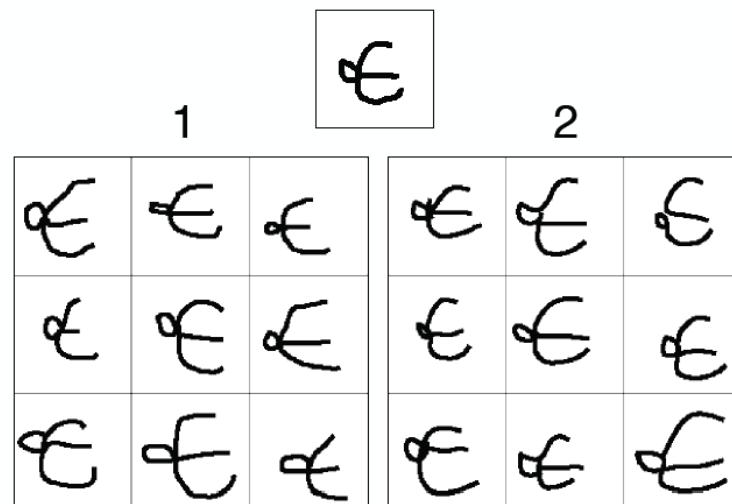
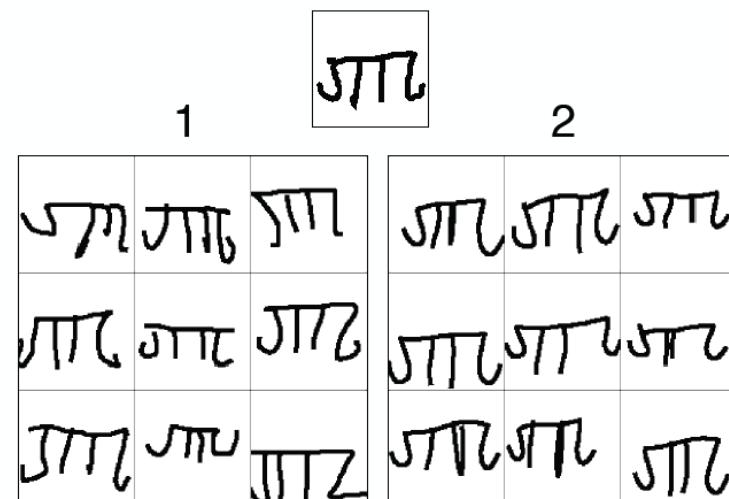
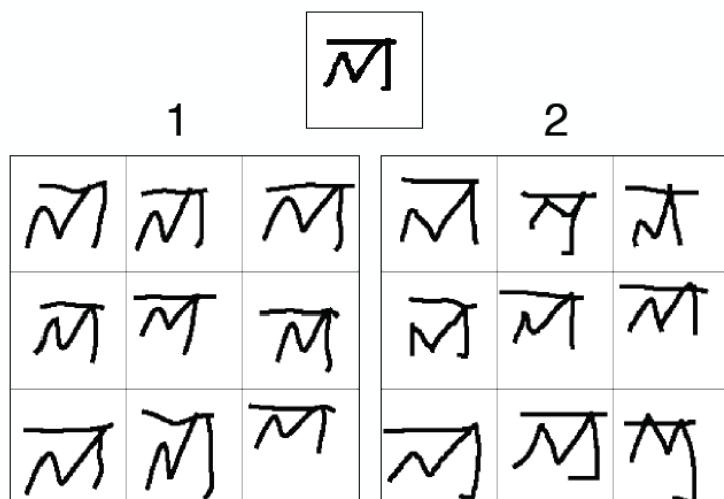
“segway”



How can we learn a novel concept – a high dimensional statistical object – from few examples.

(Lake, Salakhutdinov, Tenenbaum, Science, 2015)

One-Shot Learning: Humans vs. Machines



(Lake, Salakhutdinov, Tenenbaum, Science, 2015)

Atari Games

- Agent observes raw pixel input.

- Goal: maximize the score.

- Can a single network play many games at once?

- Can the network learn new games faster by leveraging knowledge about the previous games it learnt.

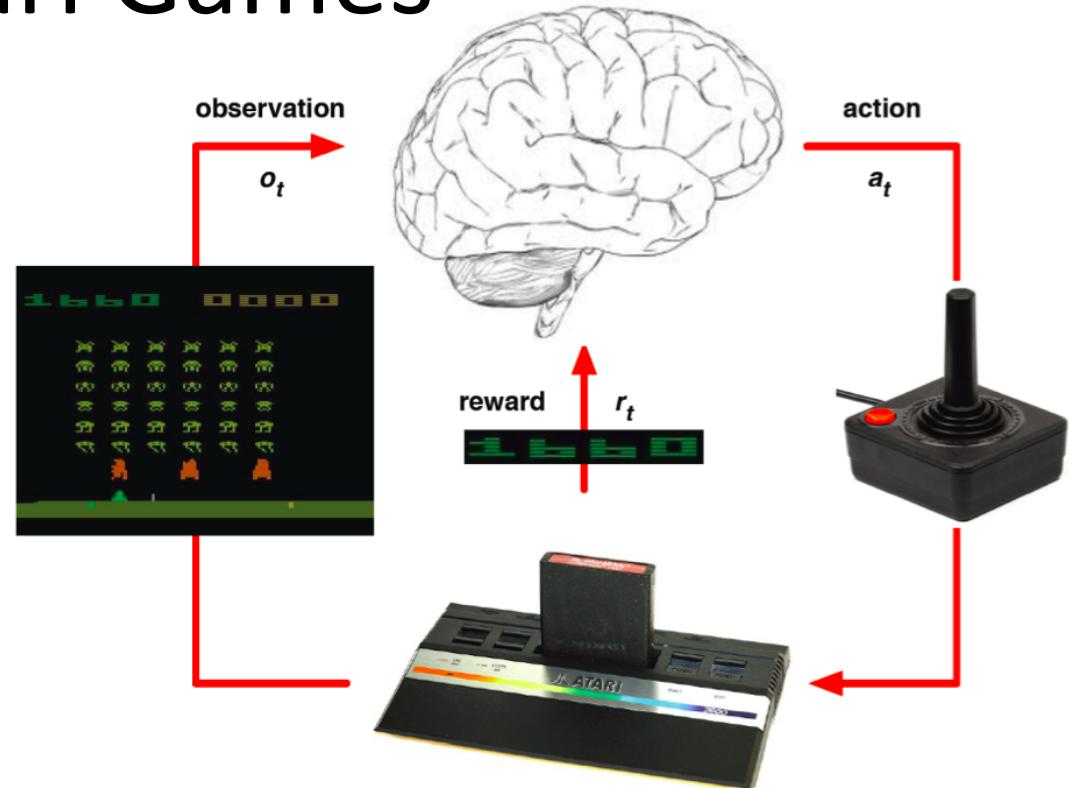


Figure credit: Nando de Freitas

Mnih et.al., 2014, Rusu et. al., 2015, Wang et.al., 2015

Actor-Mimic Net in Action

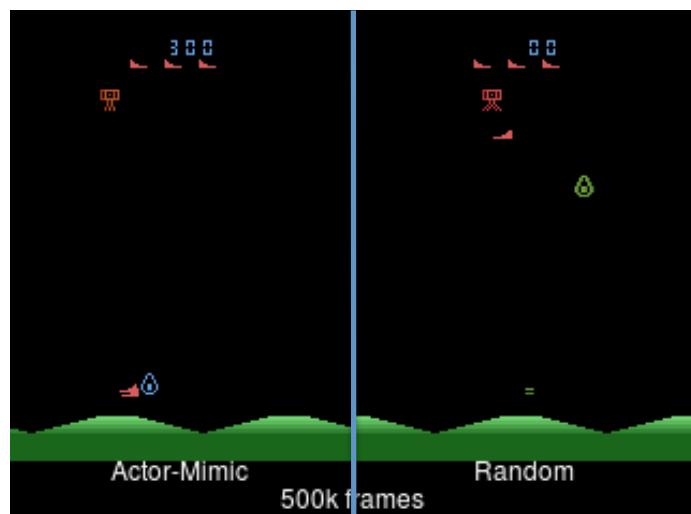
- The multitask network can match expert performance on 8 games (we are extending this to more games).



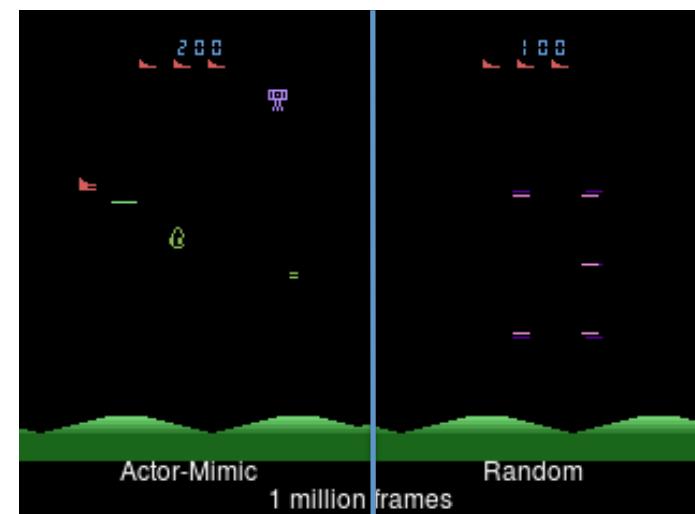
(Parisotto, Ba, Salakhutdinov, ICLR 2016)

Transfer Learning

Star Gunner: Performance after learning on 13 other games:



500K frames



1M frames

(Parisotto, Ba, Salakhutdinov, ICLR 2016)

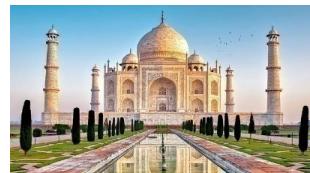
Summary

- Efficient learning algorithms for Deep Learning Models

Text & image retrieval /
Object recognition

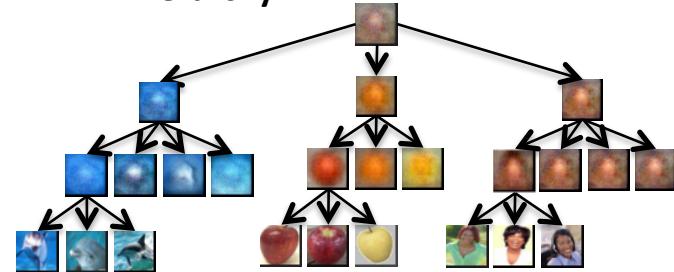


Image Tagging

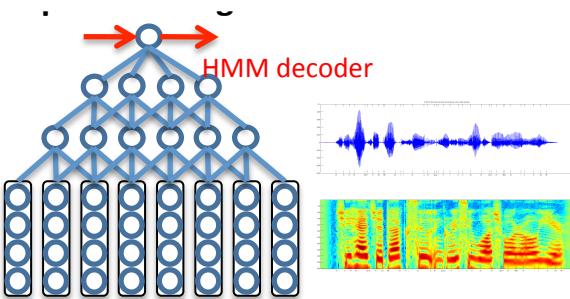


mosque, tower,
building, cathedral,
dome, castle

Learning a Category
Hierarchy



Object Detection



Multimodal Data



- 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

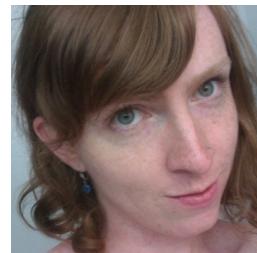
PhD Students



Nitish Srivastava



Charlie Tang



Jamie Kiros



Yura Burda



Roger Grosse



Zhilim Yang



Bhuwan Dhingra



Jimmy Ba

Master Students



Shikhar Sharma



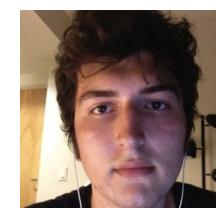
Yukun Zhu



Emilio Parisotto



Tony Wu



Elman
Mansimov