Autoencoders

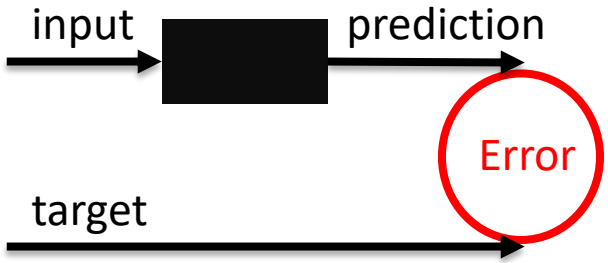
Minlie Huang
aihuang@tsinghua.edu.cn
Dept. of Computer Science and Technology
Tsinghua University

Outline

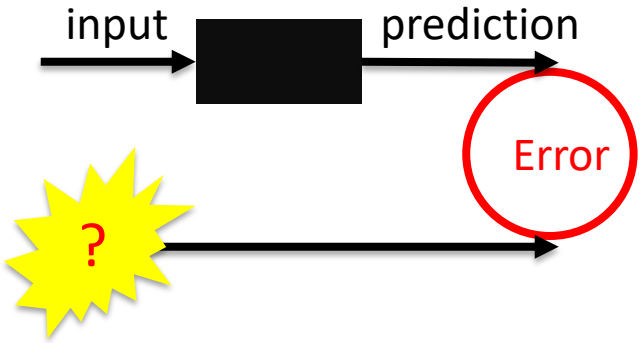
- Motivation
- Autoencoder
- Recursive auto-encoder
- Variational auto-encoder
- Other variants

Motivation

- Networks learn features and then do prediction
  - Supervised learning:

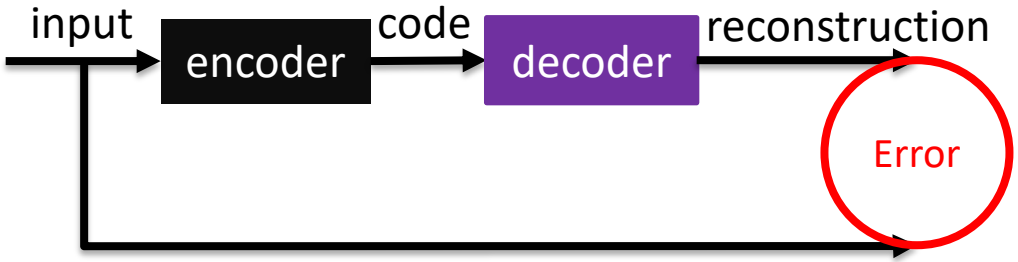


  - Unsupervised learning:

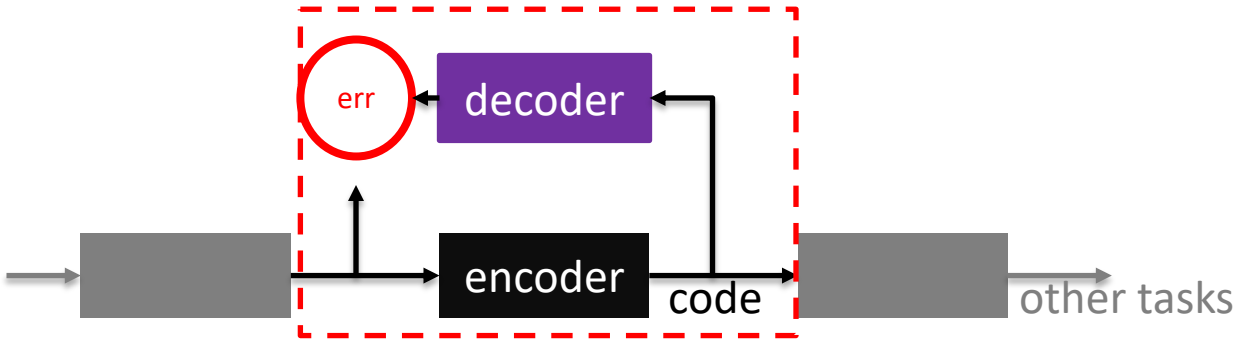


Motivation

- Autoencoder
  - Input as target



  - data-code(representation/feature)-reconstructed data



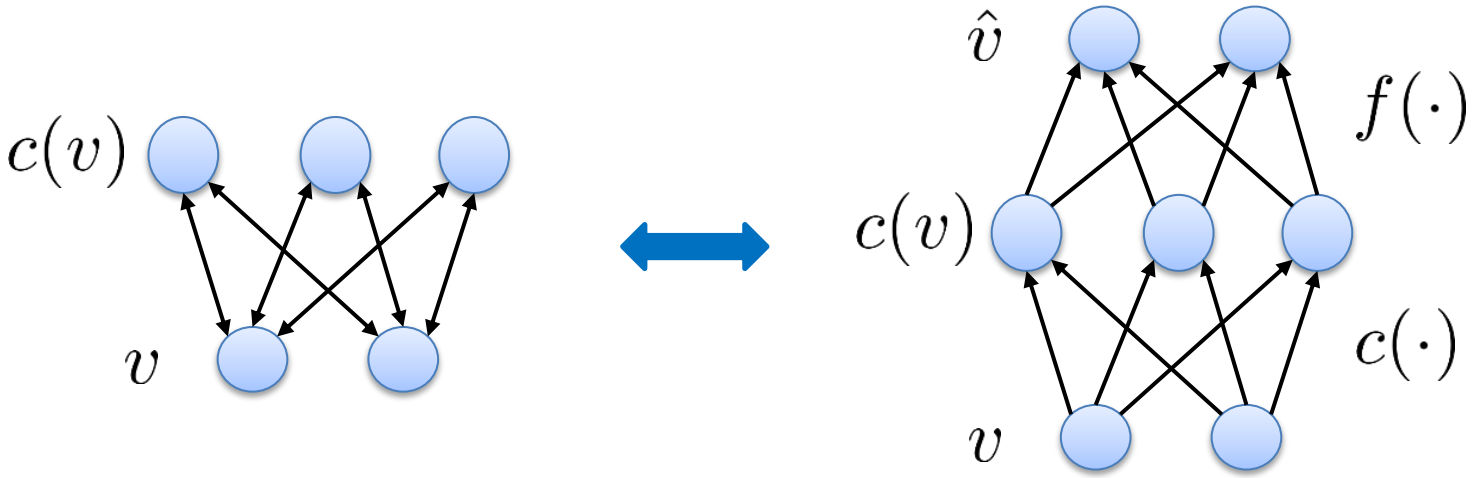
Motivation

- Applications
  - Representation Learning
  - Dimension Reduction
  - Data Denoising
  - Data Generation
  - Pre-training for other models
  - ……

Some Variants

- Not all attributes(dimensions of code) are meaningful to single input
  - *Sparse auto-encoder*
- To discover more robust features
  - *Denoising auto-encoder*
- To adapt to more data different from training set
  - *Variational auto-encoder*

Autoencoder



- Encode the input *v* into some representation *c(v)* so that the input can be reconstructed from that representation
  - Encoding function *c(v)*
  - Decoding function *f(c(v))*

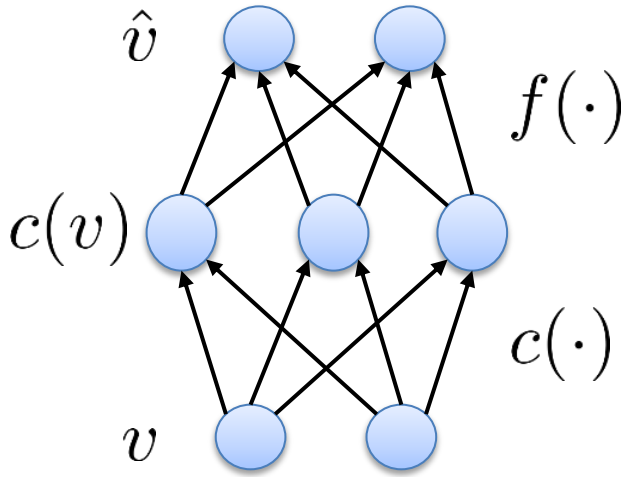
Encoding and decoding functions

- Nonlinear function

$$c(v) = sigmoid(W_1 v + \theta)$$
$$f(c) = sigmoid(W_2 c + \eta)$$

It can be constrained $W_1 = W_2^T$ or not

- If *c* and *f* are binary, then the functions can be used as probabilities



Tied weights

Loss function

- Minimize the reconstruction error or the negative data log-likelihood

$$RE = -\langle \ln P(v|c(v)) \rangle$$

$\langle \cdot \rangle$: average over samples

  - Gaussian probability (*v* is real)

$$P(v|c(v)) \propto \exp\left(\frac{-\|v - f(c(v))\|^2}{2\sigma^2}\right)$$

then

$$RE = \langle \|v - f(c(v))\|^2 \rangle$$

  - Binomial probability (*v* is binary)

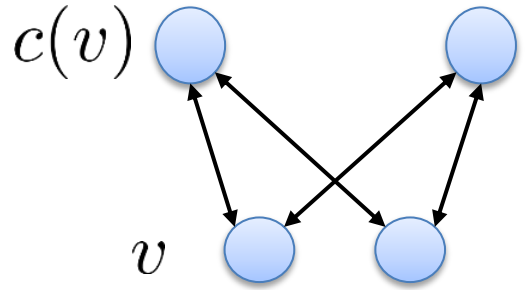
$$P(v|c(v)) \propto \Pi_i f_i(c(v))^{v_i} (1 - f_i(c(v)))^{1-v_i}$$

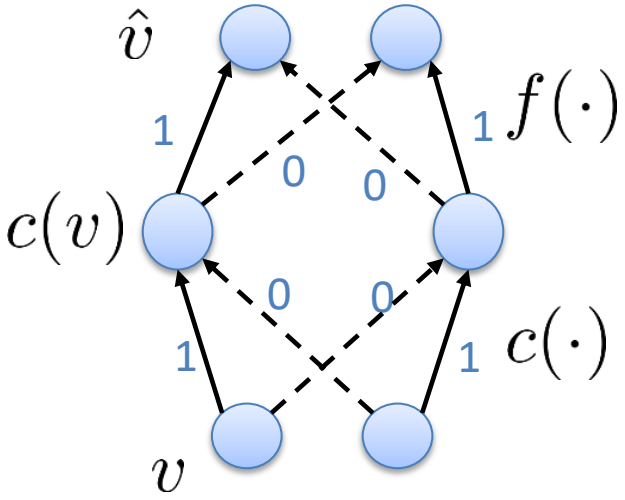
then

$$RE = -\langle \textstyle\sum_i (v_i \ln f_i(c(v)) + (1 - v_i) \ln(1 - f_i(c(v)))) \rangle$$

A trivial solution

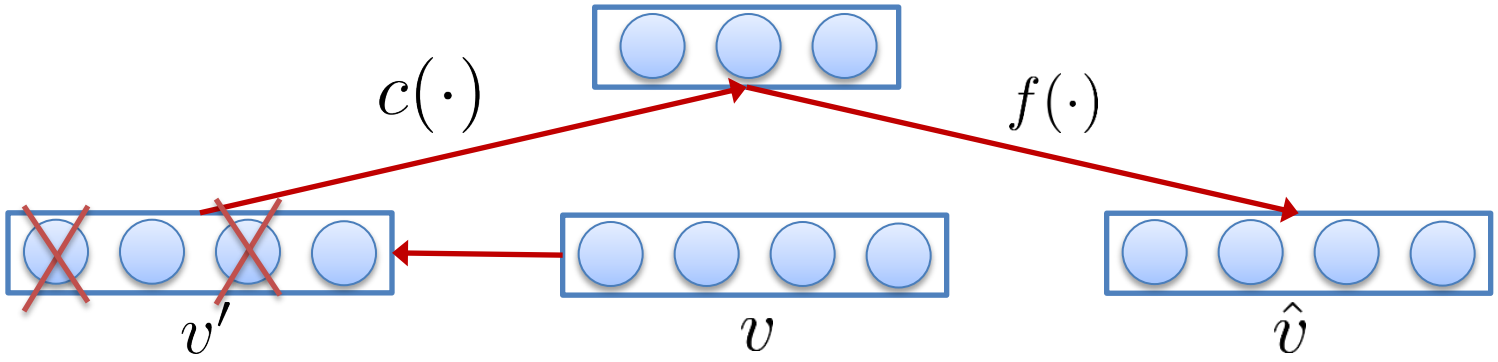
- In the following case
  - Binary units
  - The number of hidden units is equal to the number of visible units
- There is a trivial solution $W_1 = W_2 = I, \eta = \theta = -0.5$





v	$W_1 v$	$c(v)$	$W_2 c$	$f(c)$
0	-0.5	0	-0.5	0
1	0.5	1	0.5	1

Denoising auto-encoder



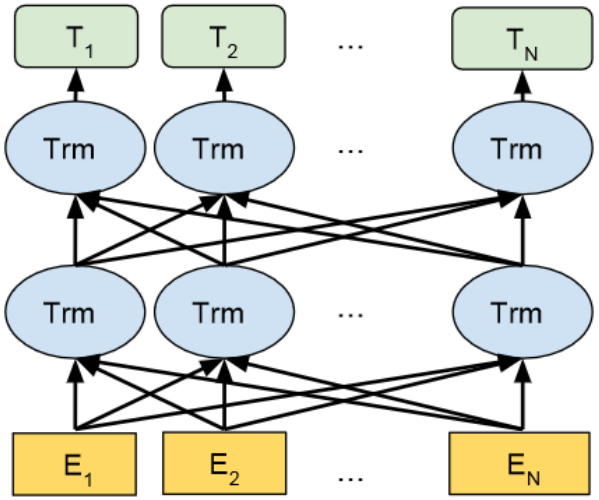
- **Corrupt** v to v' by randomly setting some elements of v to zero.
- Use v' as input and try to reconstruct v.
  - Ideally, $\hat{v}$ is the clean version of v

Vincent, et al., ICML, 2008

Denoising auto-encoder

- BERT
  - Mask-predicted language model
  - Learn features from context

Output: my dog is hairy



Input: my dog is [MASK]

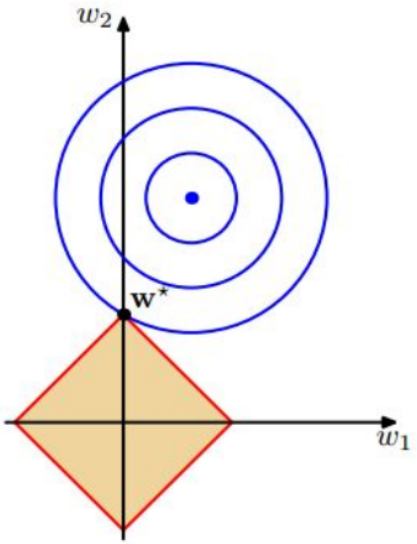
Rules of adding noise:

12% replace the word with [MASK]
My dog is hairy -> my dog is [MASK]

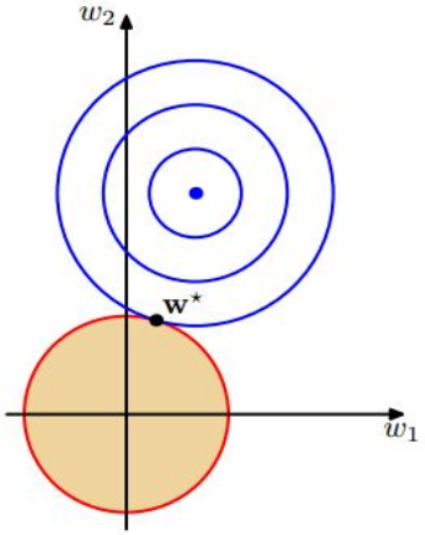
1.5% Replace the word with a random word
My dog is hairy -> my dog is apple

Sparse auto-encoder

- Add regularizer over hidden units
  - L1 / L2 regularizer
    - L1 regularizer usually makes representation more sparse



$$L = \lambda \sum_{i=1}^{n} |c_i|$$



$$L = \lambda \sum_{i=1}^{n} c_i^2$$

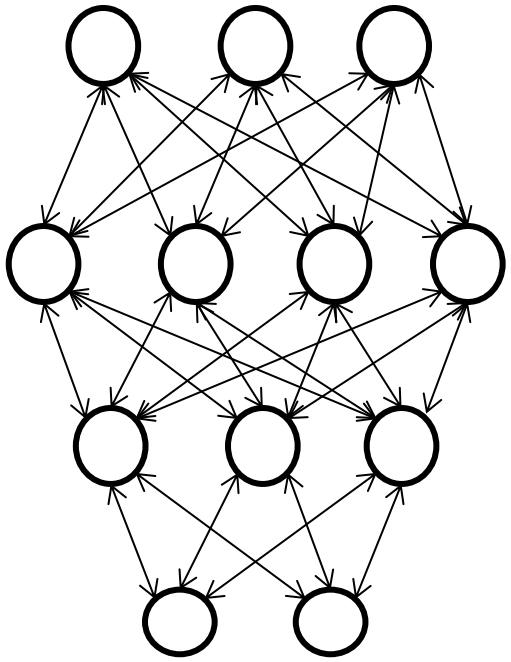
Sparse auto-encoder

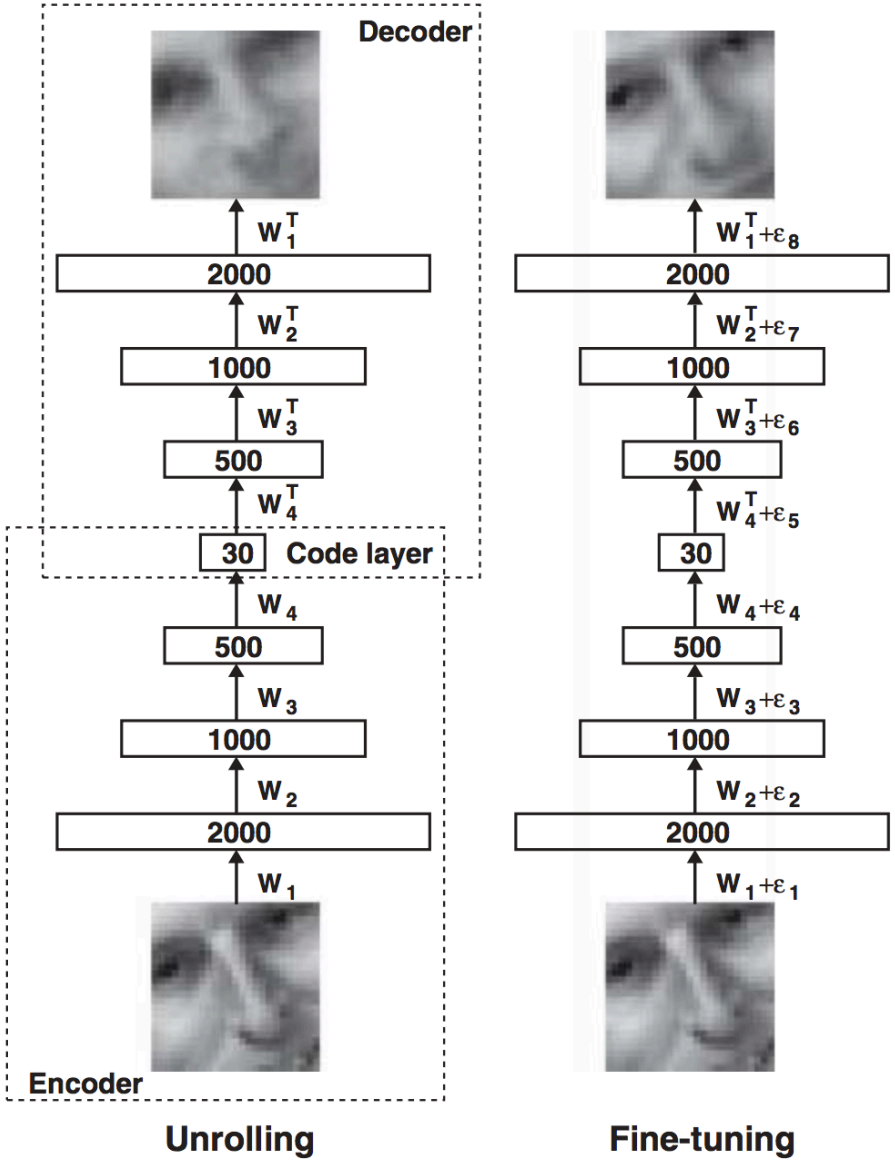
- Add regularizer over hidden units
  - A sparsity regularizer
    - $L = \sum_{i=1}^{n} KL(\rho || \overline{c_i})$

      $= \sum_{i=1}^{n} \left[\rho \log \left(\frac{\rho}{\overline{c_i}} \right) + (1 - \rho) \log \left(\frac{1 - \rho}{\overline{c_i}} \right) \right]$
    - ρ is the target sparsity (hyperparameter)
    - $\overline{c_i}$ is the average activation (after sigmoid) of hidden units i across the batch

Deep Auto-encoder

- Stack auto-encoders on top of each other
- Train layers one by one
- Fine tune with BP
- Sparsity or other regularizations can be used

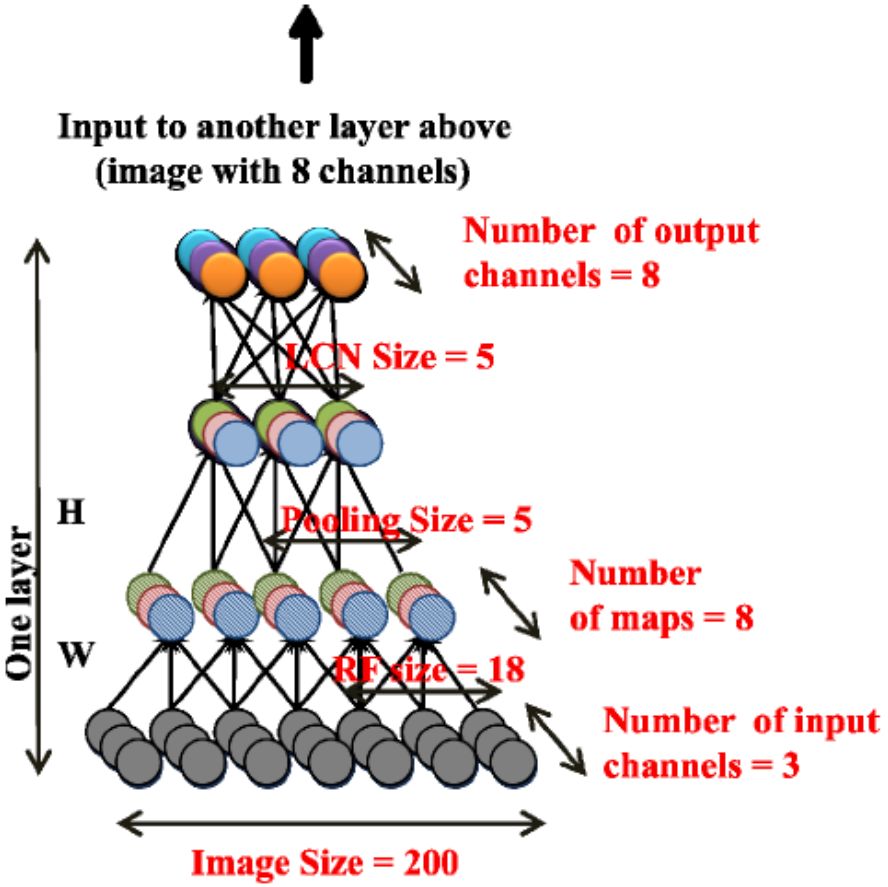




Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks[J]. science, 2006, 313(5786): 504-507.MLA

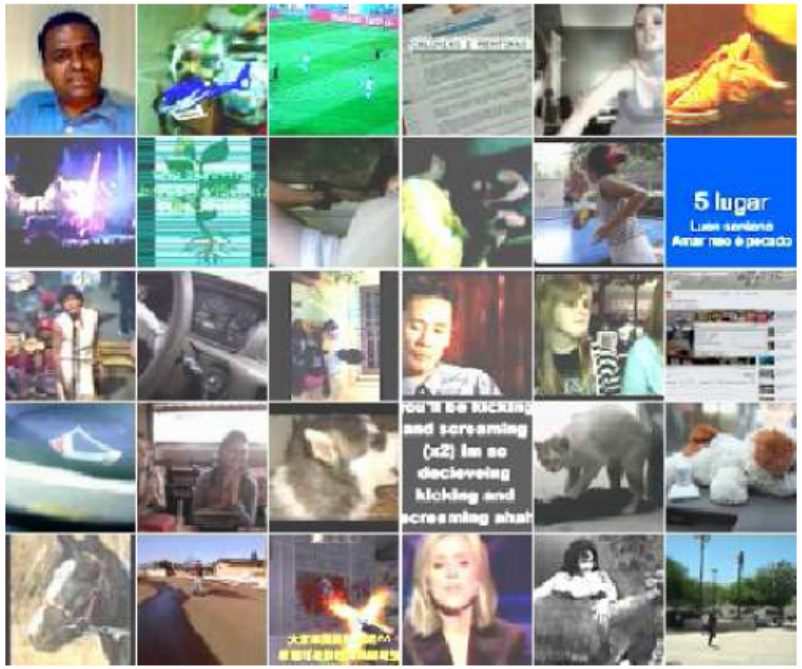
A Successful Application

- Features
  - **Local receptive field**
  - **L2 pooling**
  - **Local contrast normalization**
- The overall network replicate this architecture 3 times
- Over 1 billion parameters
- Three days on a cluster with 1,000 machines (16,000 cores).



Le Q V. Building high-level features using large scale unsupervised learning[C]//2013 IEEE international conference on acoustics, speech and signal processing. IEEE, 2013: 8595-8598.

10 million 200*200 training images



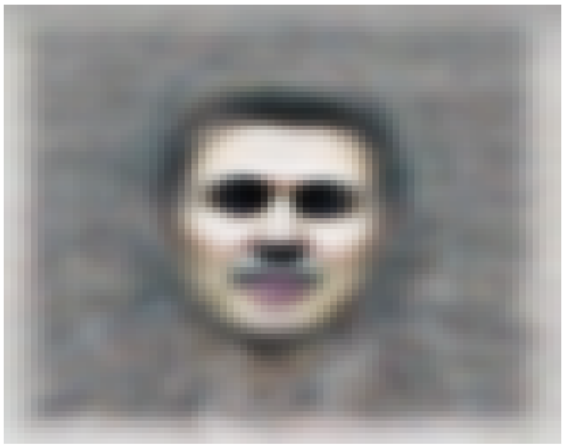
Trained on Youtube images

Tested on a mixture of Labeled Faces in The Wild and ImageNet

Optimal Stimulus

- “face neuron”

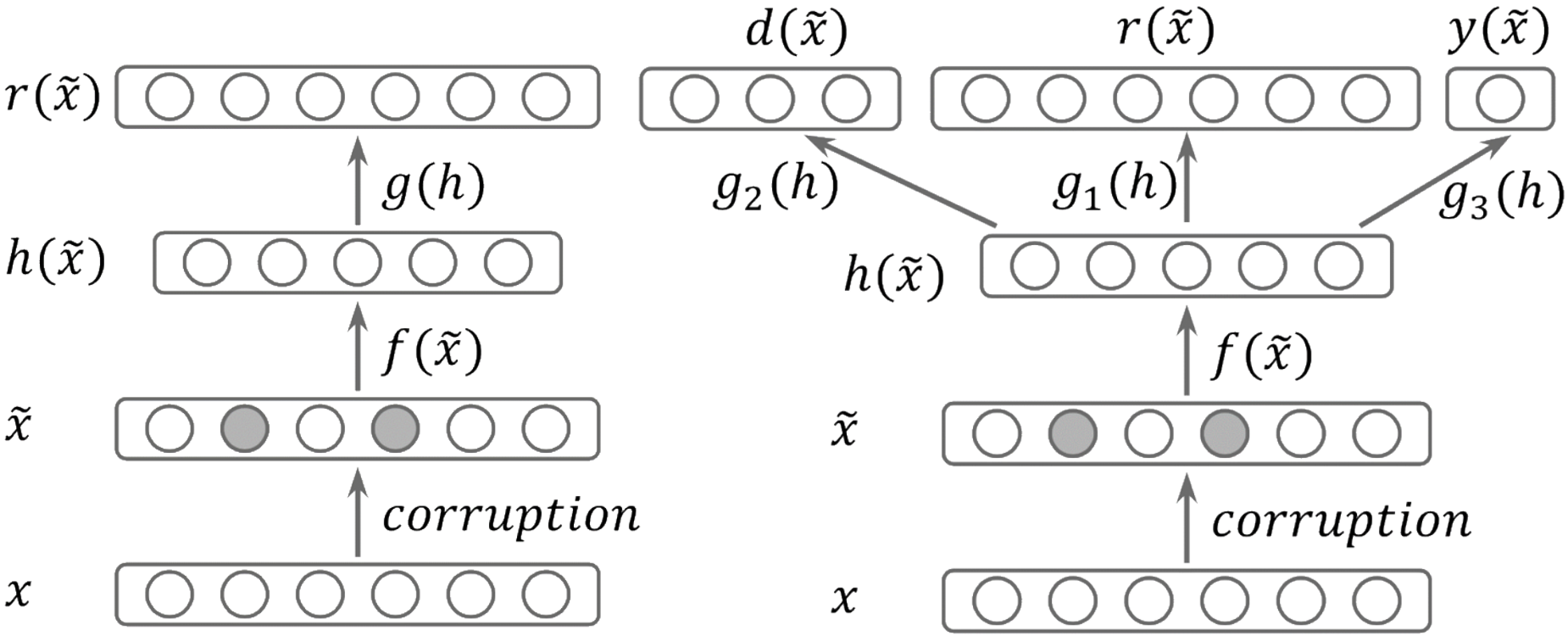
Top 48 stimuli of the best neuron from the test set.



The optimal stimulus according to numerical constraint optimization.

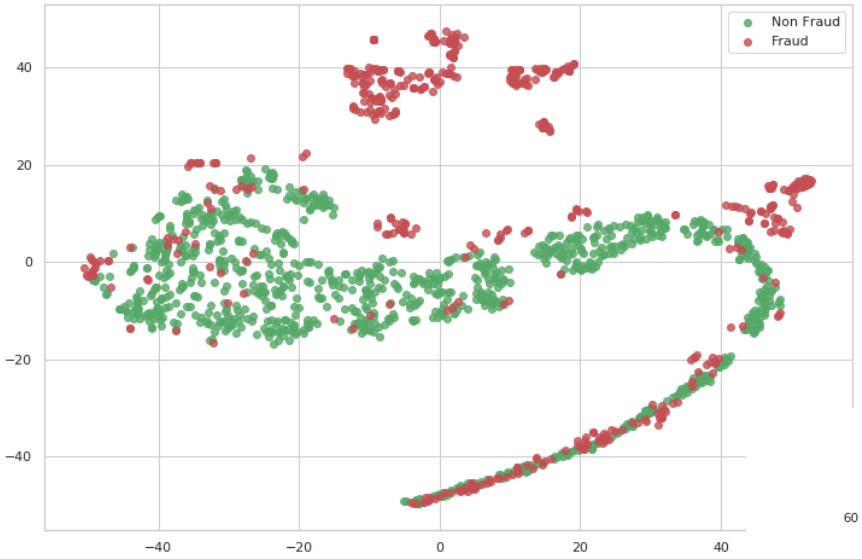
Semi-supervised Autoencoders

- Training additional supervised signals



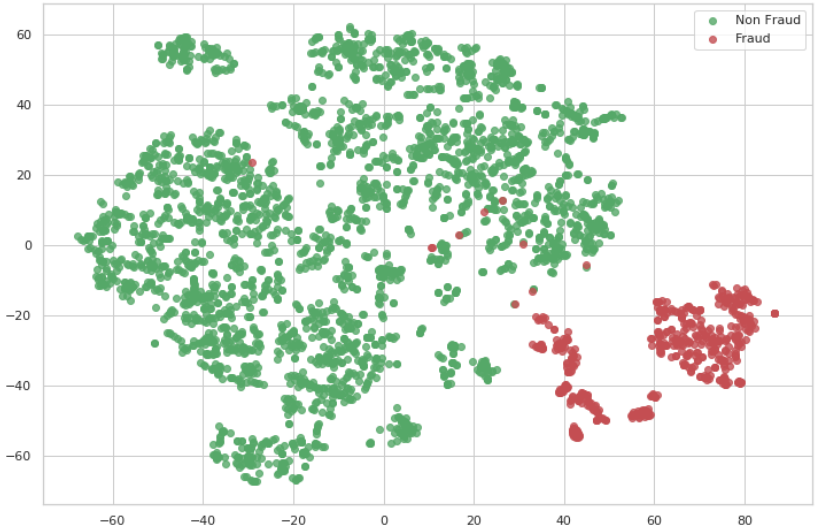
Liu, Huang et al. IJCAI 2016

Semi-supervised Autoencoders



← Autoencoders

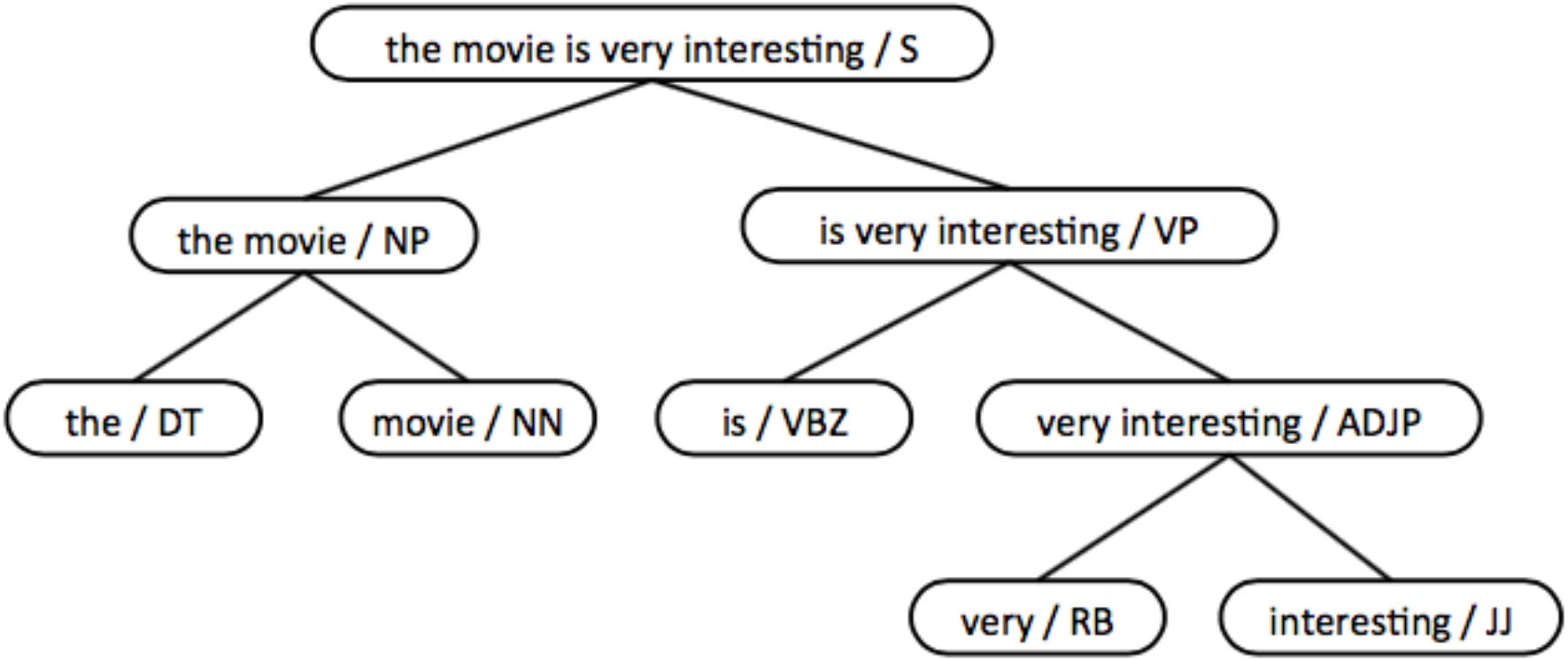
Semi-supervised Autoencoders →



Shivam Bansal. Semi Supervised Classification using AutoEncoders. Kaggle.

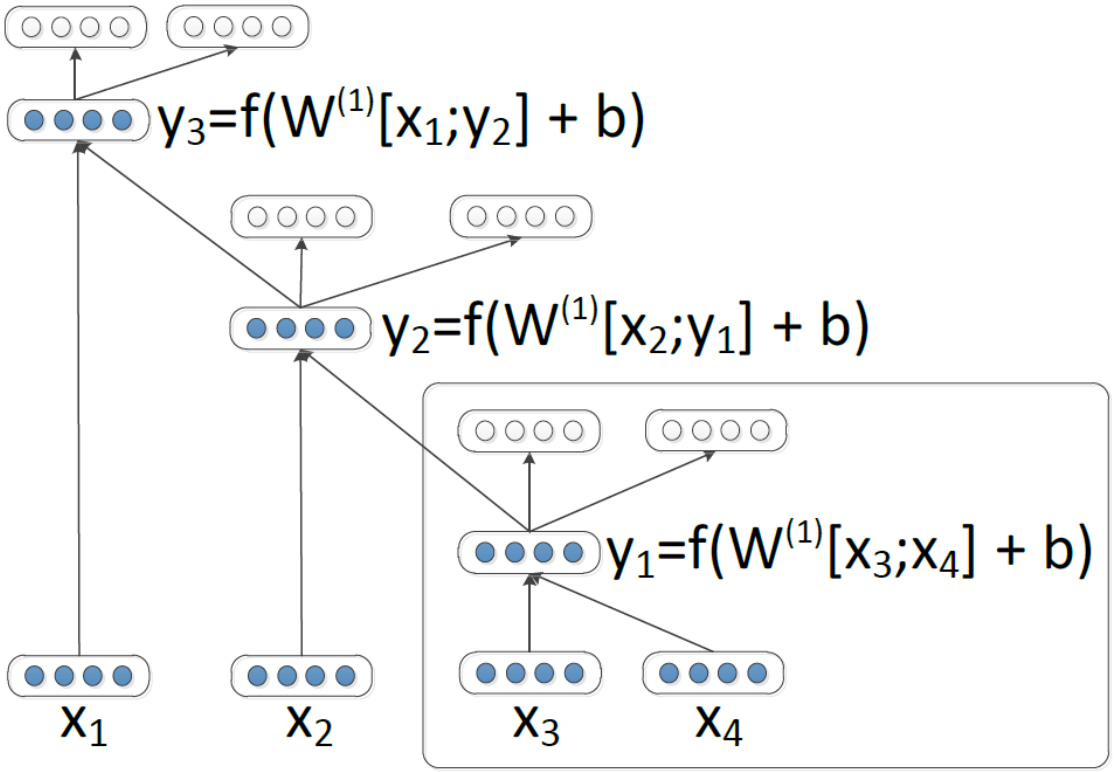
Recursive Autoencoders

- What if we have tree-structured data?



Recursive Autoencoders

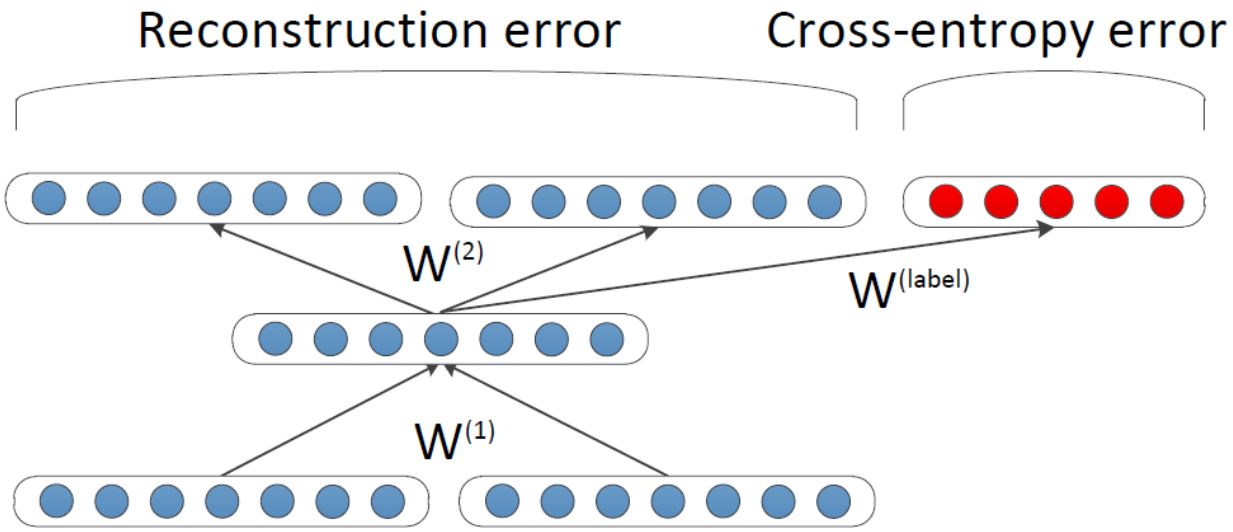
- What if we have tree-structured data?



Socher et al. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions

Recursive Autoencoders

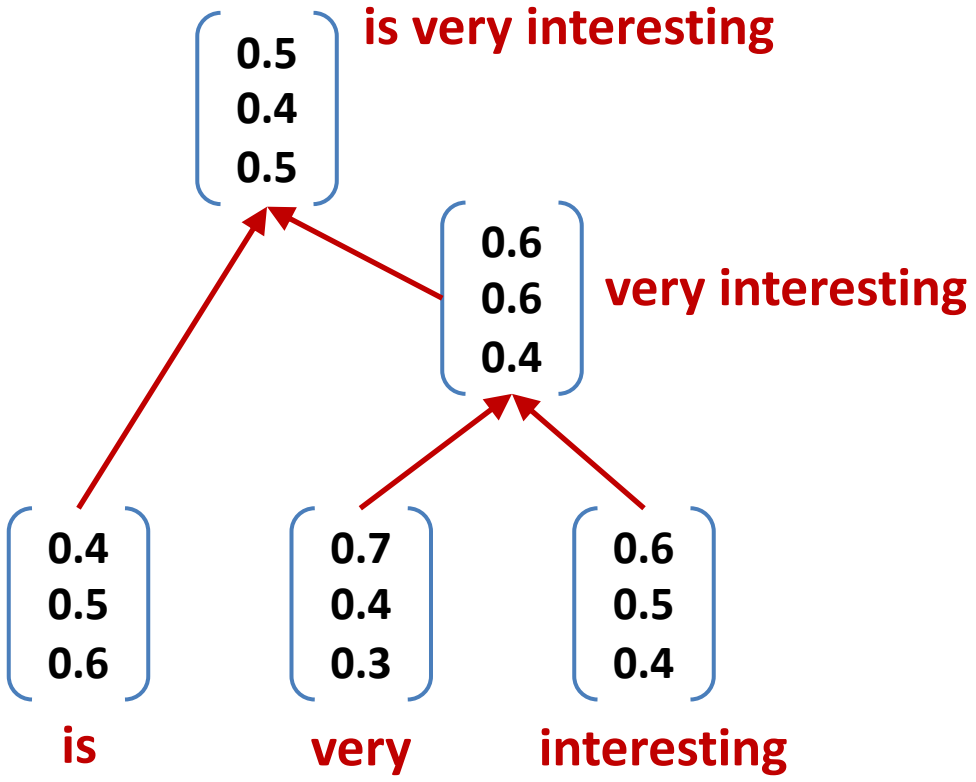
- What if we have tree-structured data?



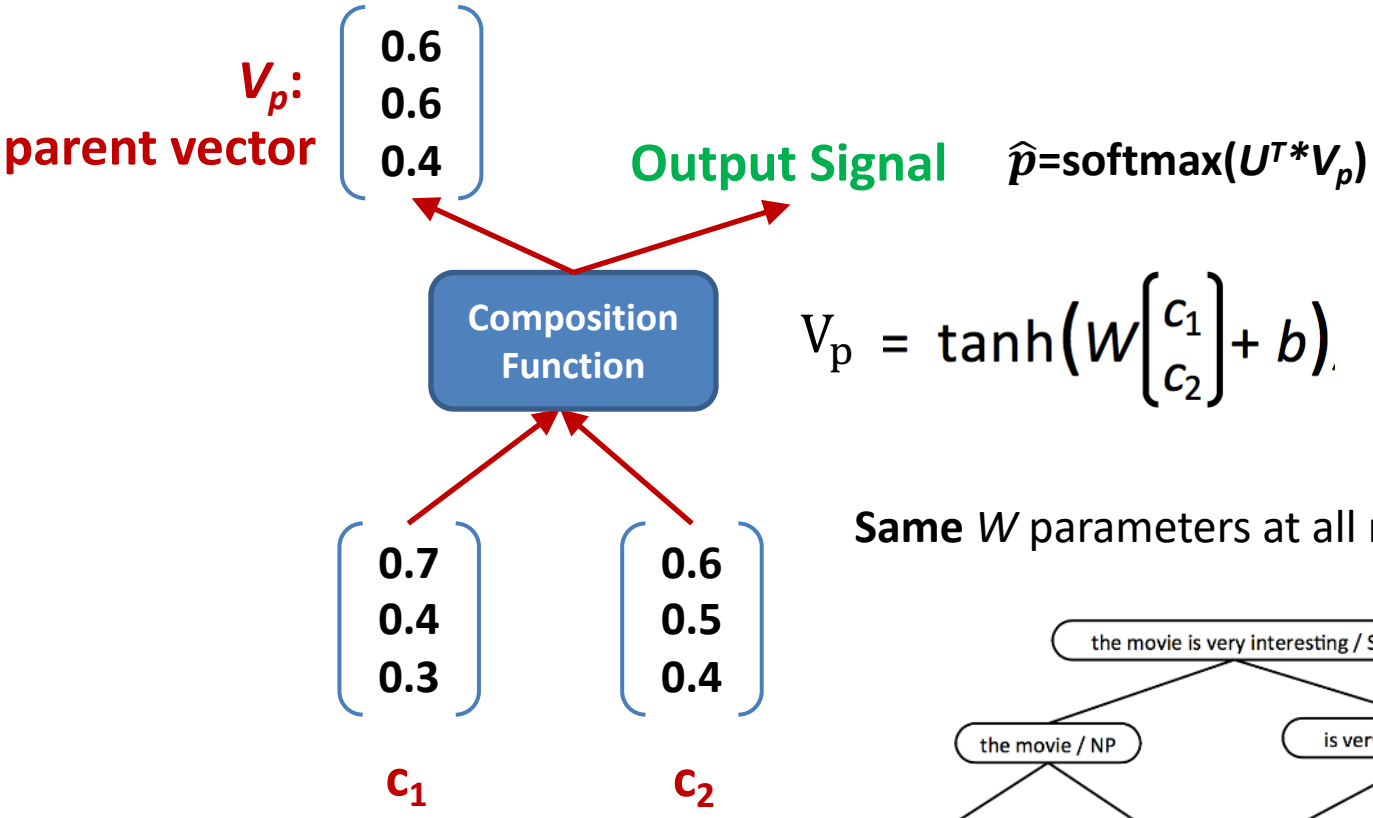
Socher et al. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions

Recursive Autoencoders

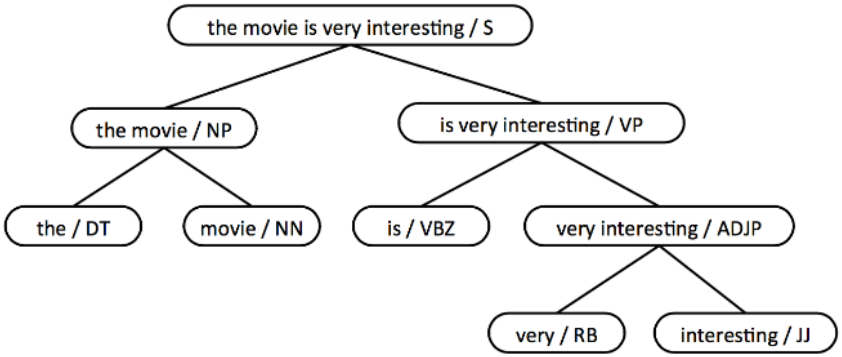
Rules of Compositionality
The meaning (vector) of a sentence is determined by
(1) **The meanings of its words**
(2) **The rules that combine them**



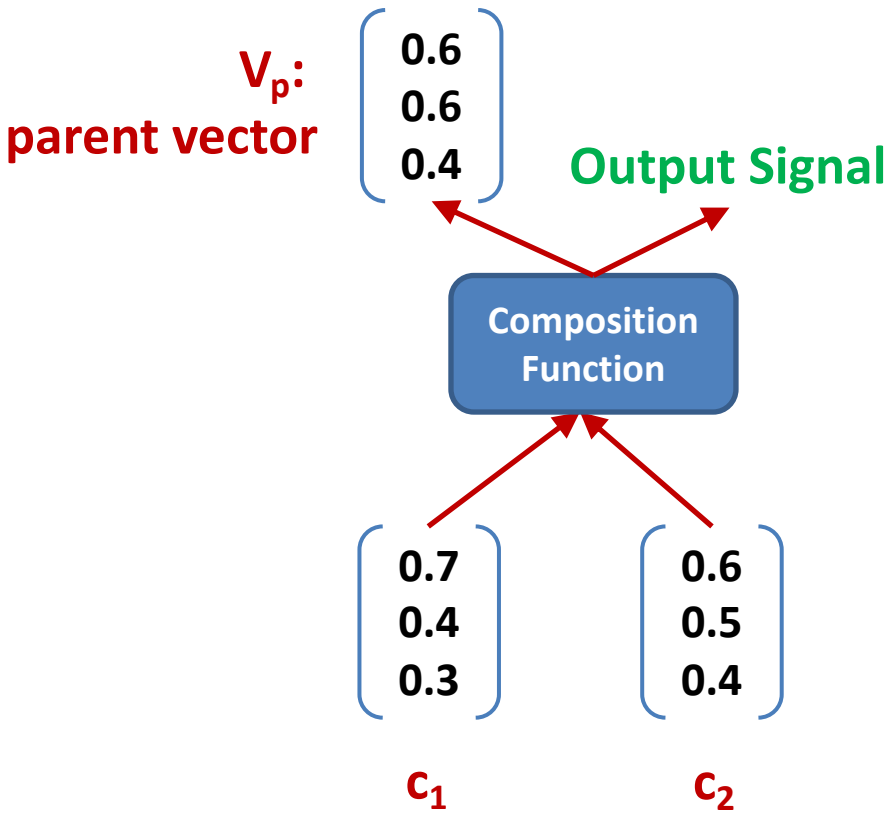
Recursive Autoencoders



Same *W* parameters at all nodes of the tree



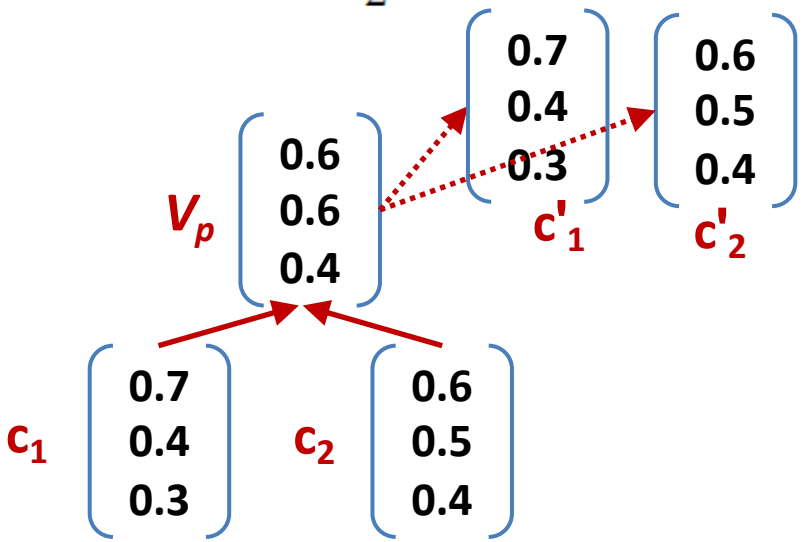
Recursive Autoencoders



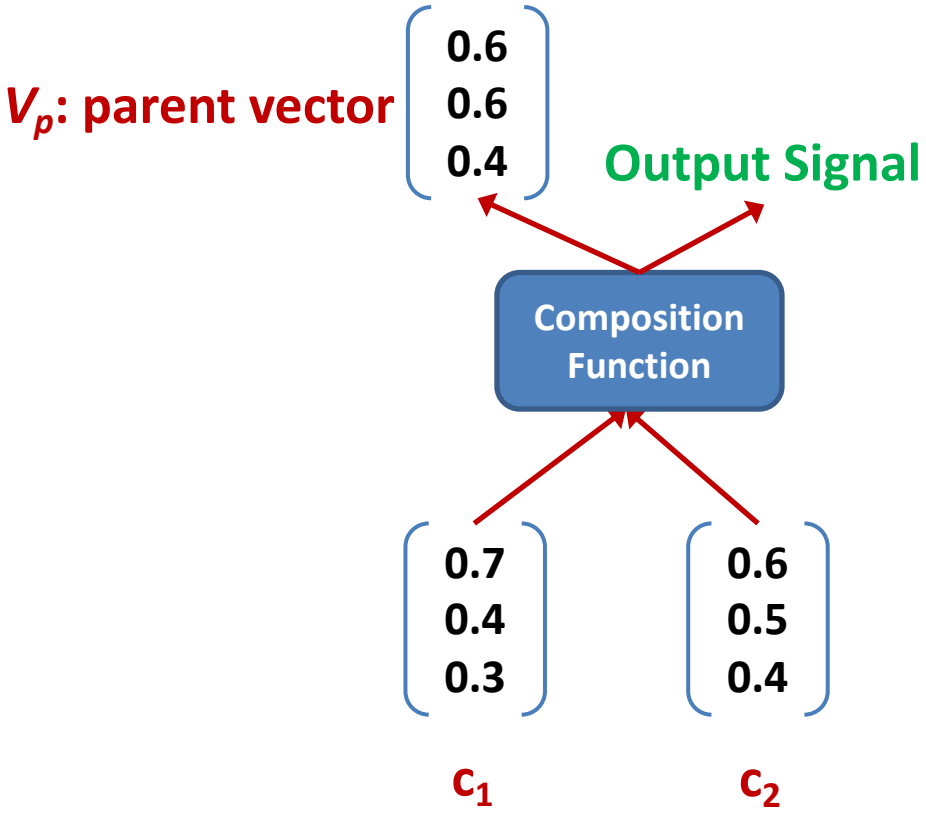
Train the model

- ✓ No supervision: minimizing reconstruction error

$$E_{rec}([c_1; c_2]) = \frac{1}{2} \left|\left|[c_1; c_2] - [c_1'; c_2']\right|\right|^2$$



Recursive Autoencoders



Train the model

✓ With supervision: minimizing cross entropy error

$$E = \sum_{k=1}^{K} -p(k) \log \hat{p}(k)$$

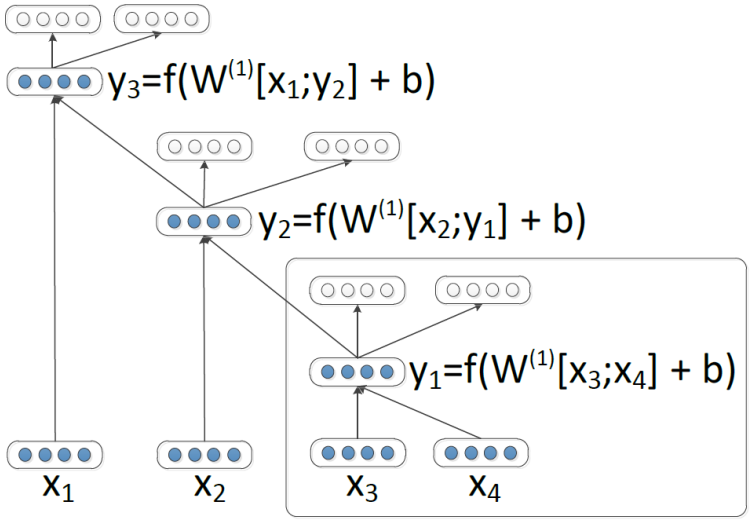
The gold distribution over class labels k

The predicted distribution based on the parent vector V_p

$$\hat{p} = \text{softmax}(U^T * V_p)$$

Recursive Autoencoder

- Given the tree structure, we can compute all the node vectors from bottom to top.
- Train by minimizing reconstruction error



$$E_{rec}([c_1; c_2]) = \frac{1}{2} \left|\left|[c_1; c_2] - [c_1'; c_2']\right|\right|^2$$

Socher et al. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions

Semi-Supervised Recursive Autoencoders

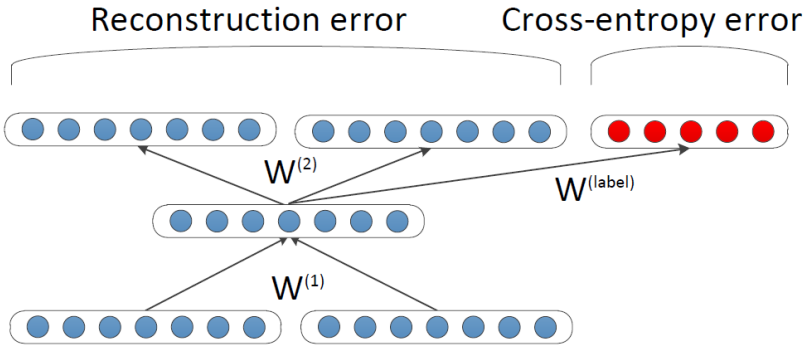
- The task label can be introduced in all nodes of the tree.

$$d(p;\theta) = \mathrm{softmax}(W^{label}p)$$

$$E_{cE}(p,t;\theta) = -\sum_{k=1}^{K} t_k \log d_k(p;\theta)$$

$$E([c_1;c_2]_s, p_s, t, \theta) =$$

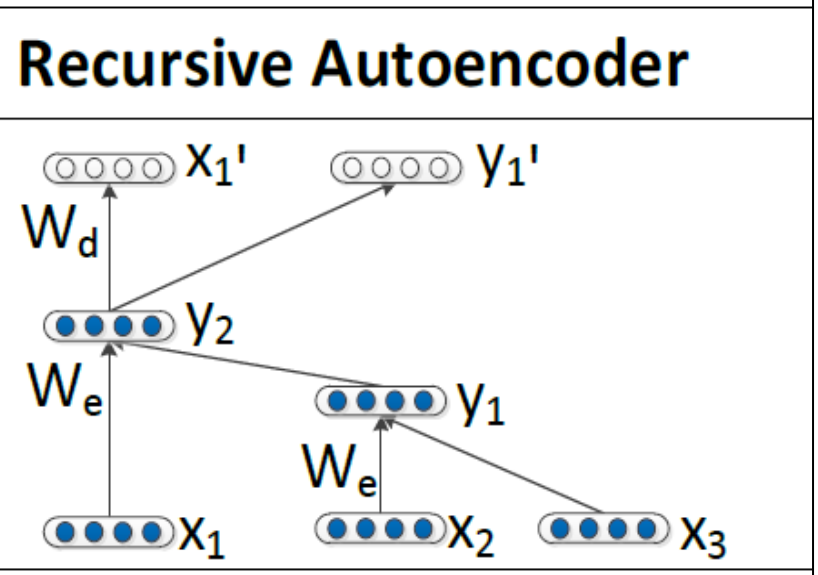
$$\alpha E_{rec}([c_1;c_2]_s;\theta) + (1-\alpha)E_{cE}(p_s,t;\theta)$$

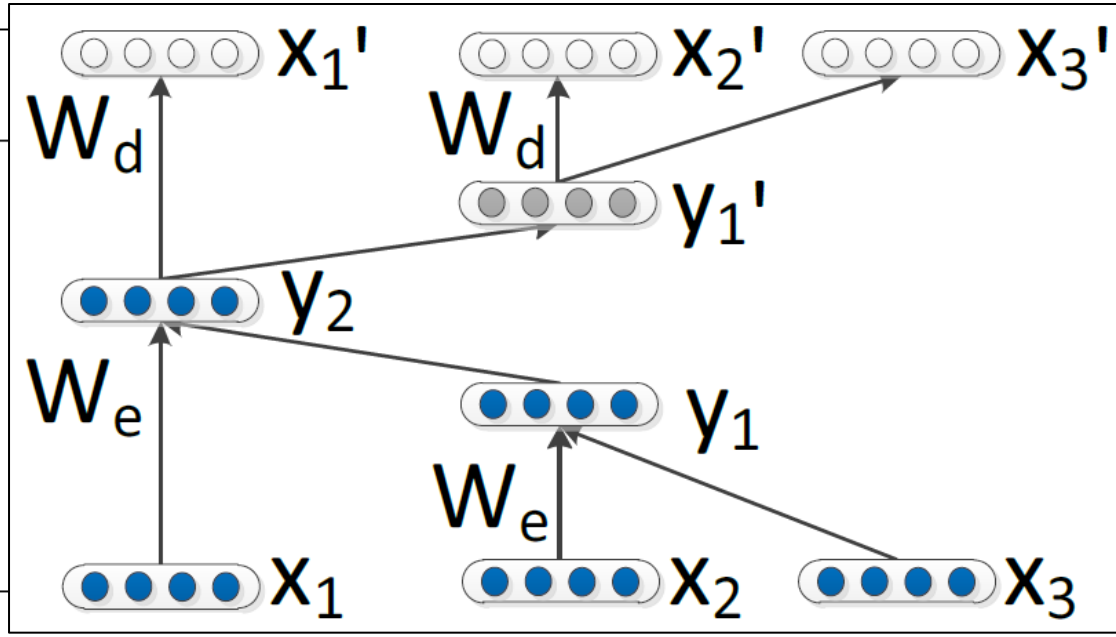


Socher et al. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions

Unfolding Recursive Autoencoders

- Unfolding the subtrees

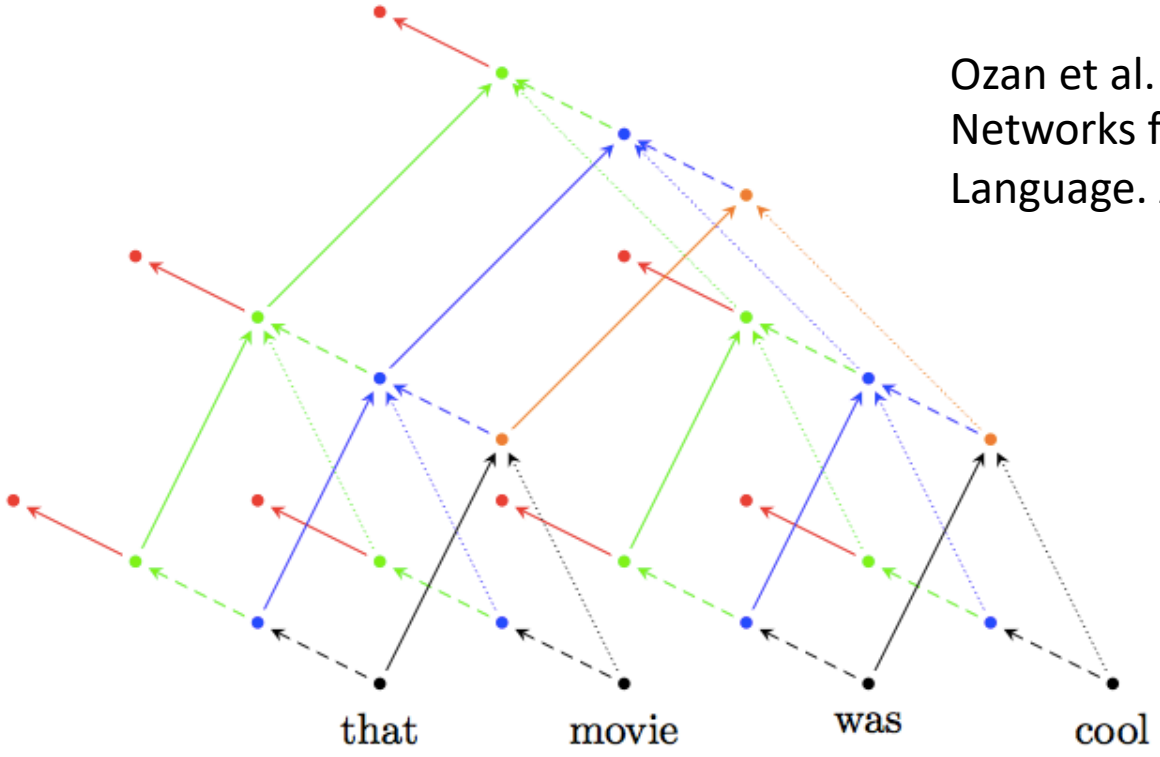




Socher et al. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection

Deep Recursive Autoencoders

- Recurrent structures on tree data

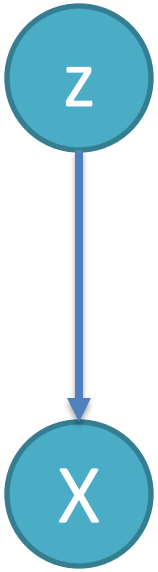


Ozan et al. Deep Recursive Neural Networks for Compositionality in Language. ACL 2014

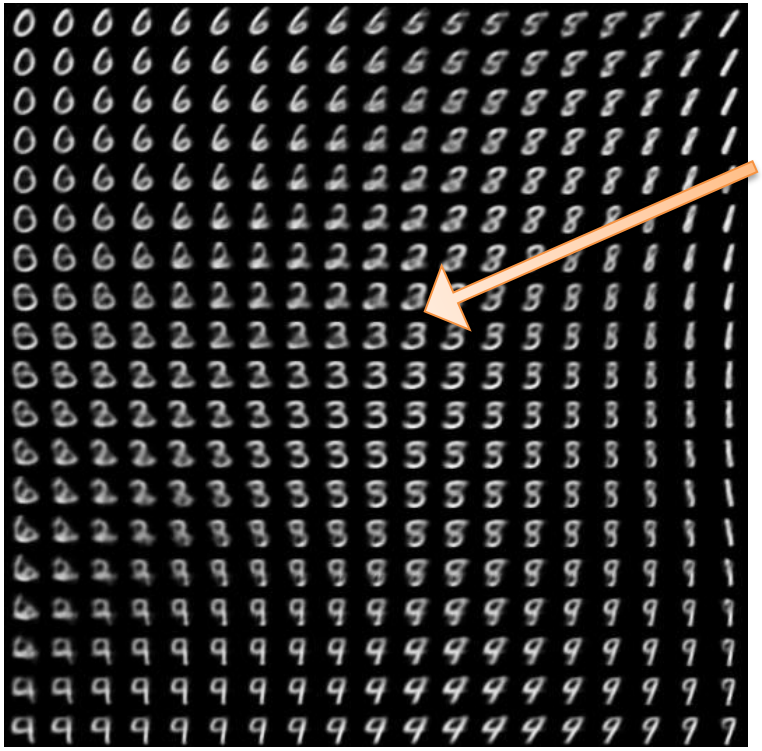
$$h_\eta^{(i)} = f(W_L^{(i)} h_{l(\eta)}^{(i)} + W_R^{(i)} h_{r(\eta)}^{(i)} + V^{(i)} h_\eta^{(i-1)} + b^{(i)})$$

Variational Autoencoder (VAE)

- Start from Probabilistic Graphical Models
  - Find a mapping from vectors to images



$z = z_0$



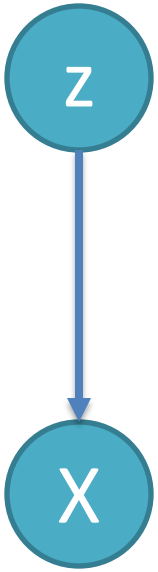
$z = (z_0 + z_1)/2$

Interpolation between two digits

$z = z_1$

Variational Autoencoder (VAE)

- Start from Probabilistic Graphical Models
  - Find a mapping from vectors to images



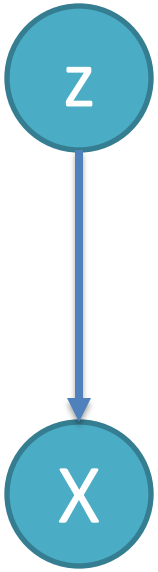
Latent Variable Assume $z \sim N(0, 1)$
standard Gaussian distribution

$x = f(z)$ How to find the mapping f?

Data Assume $x \sim P(x)$

Variational Autoencoder (VAE)

- Start from Probabilistic Graphical Models
  - Find a mapping from vectors to images



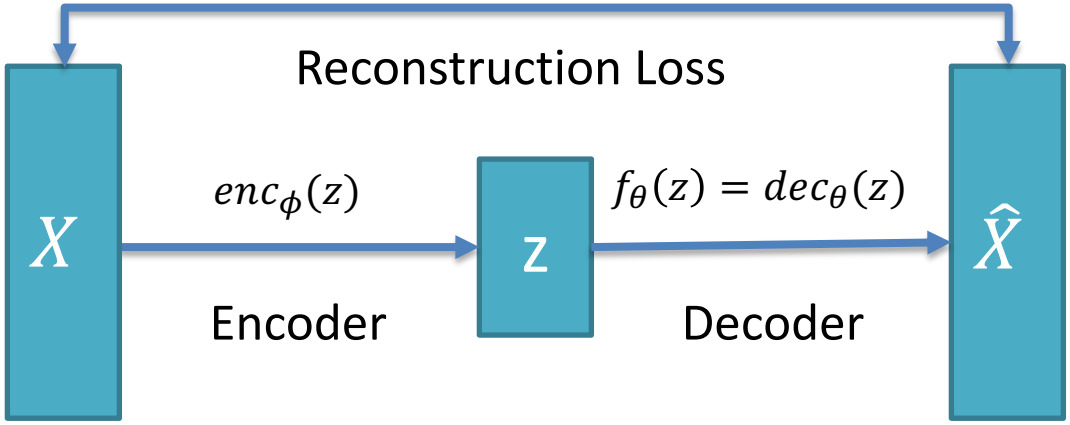
Use neural network !

$x = f_\theta(z)$, where θ is parameters

How to train the network?
We don't know (z, x) pairs

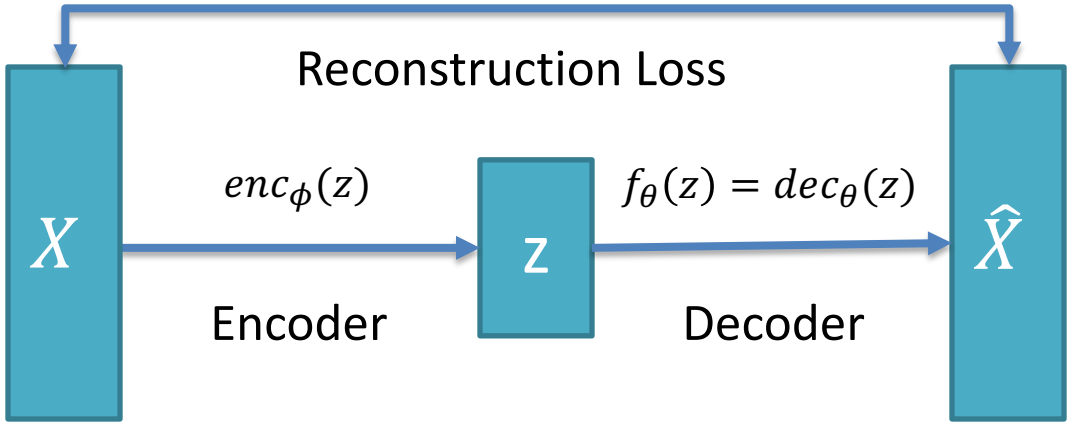
Variational Autoencoder (VAE)

- Start from Probabilistic Graphical Models
  - Find a mapping from vectors to images



But z must obey N(0, 1) !

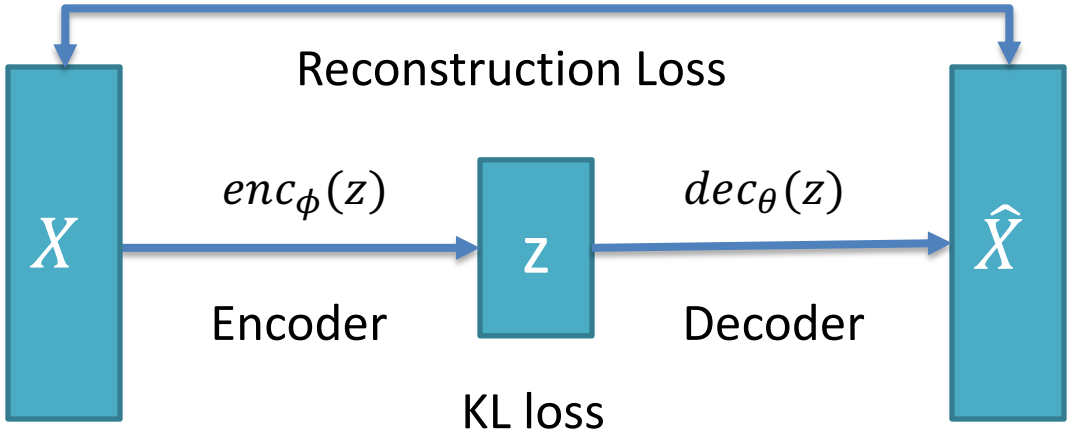
Variational Autoencoder (VAE)



- Let z become a random variable
  - Assume $z|X \sim N(\mu, \sigma)$
  - $[\mu, \sigma] = enc(X)$
  - It's better make $N(\mu, \sigma)$ close to $N(0, 1)$
  - KL loss: make two distribution closer

$$KL(N(\mu, \sigma)||N(0, 1)) = \frac{1}{2}\sum_{i=0}^{d}(\mu_i^2 + \sigma_i^2 - 2log\sigma - 1)$$

Variational Autoencoder (VAE)



– Algorithm

- Obtain X from dataset, $[\mu, \sigma] = enc_\phi(X)$
- Sample $z \sim N(\mu, \sigma)$
- $\hat{X} = dec_\theta(z)$
- $Loss = \left|\hat{X} - X\right|^2 + KL(N(\mu, \sigma)||N(0,1))$

Variational Autoencoder (VAE)

– Algorithm

- Obtain X from dataset, $[\mu, \sigma] = enc_{\phi}(X)$
- **Sample $z \sim N(\mu, \sigma)$**
- $\hat{X} = dec_{\theta}(z)$
- $Loss = \left|\left|\hat{X} - X\right|\right|^2 + KL(N(\mu, \sigma)||N(0,1))$

– But how can loss backpropagate back to μ, σ, ϕ ?

Variational Autoencoder (VAE)

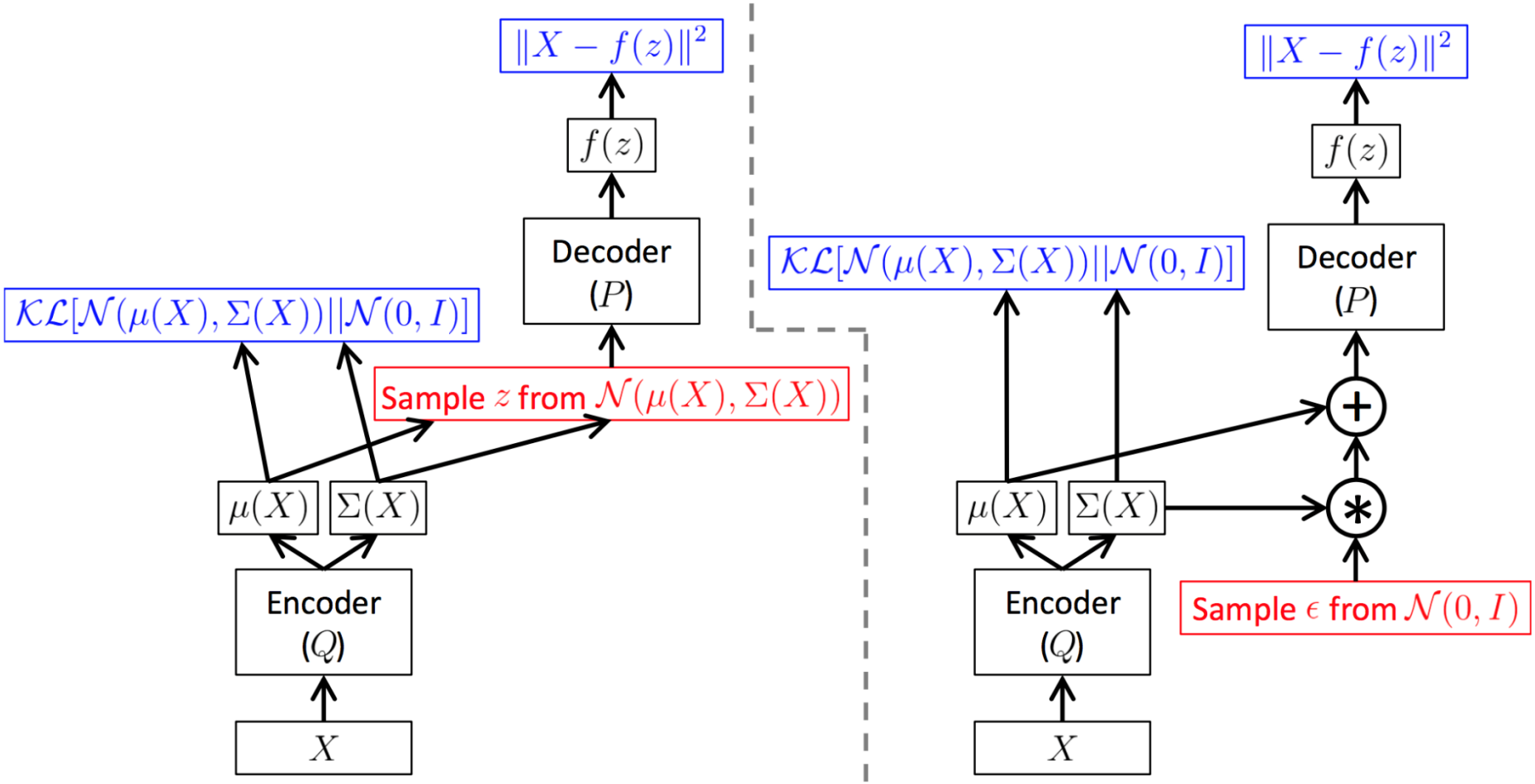
– Algorithm

- Obtain X from dataset, $[\mu, \sigma] = enc_{\phi}(X)$
- **Sample $\boldsymbol{\varepsilon \sim N(0,1)}$, $\boldsymbol{z = \sigma * \epsilon + \mu}$**
- $\hat{X} = dec_{\theta}(z)$
- $Loss = \left|\left|\hat{X} - X\right|\right|^2 + KL(N(\mu, \sigma)||N(0,1))$

$\boldsymbol{z \sim N(\mu, \sigma)}$

– This is called reparameterization trick

Variational Autoencoder (VAE)



Reparameterization skill

CARL DOERSCH. **Tutorial on Variational Autoencoders** https://arxiv.org/pdf/1606.05908.pdf

Variational Autoencoder (VAE)

- A little math about variational methods
  - Maximize P(X): The probability that the PGM generates the images

$$\log p(x) = \mathbb{E}_{q_\phi(z|x)}\left[\log p(x)\right]$$

$$= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z) + \log p(z) - \log p(z|x)\right]$$

$$= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z) + \log p(z) - \log q_\phi(z|x) + \log q_\phi(z|x) - \log p(z|x)\right]$$

$$= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z) + \log p(z) - \log q_\phi(z|x)\right] + KL(q_\phi(z|x)||p(z|x))$$

$$\geq \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z) + \log p(z) - \log q_\phi(z|x)\right]$$

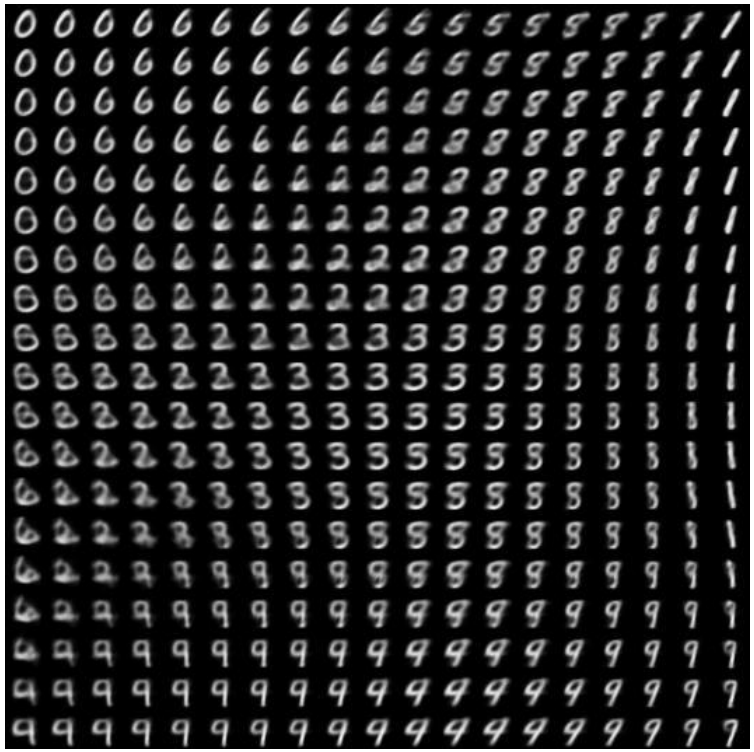
$$= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - KL(q_\phi(z|x)||p(z)) \triangleq \mathcal{L}(x;\theta,\phi)$$

Reconstruction Loss KL Loss

  - It is called ELBO (evidence lower bound)

Variational Autoencoder (VAE)

- Applications: learn a latent distribution



A man is on a ship path with the woman .
A man is on a ship path with the woman .
A man is passing on a bridge with the girl .
A man is passing on a bridge with the girl .
A man is passing on a bridge with the girl .
A man is passing on a bridge with the dogs .
A man is passing on a bridge with the dogs .

Junbo Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. Adversarially Regularized Autoencoders. In ICML, 2018

Variational Autoencoder (VAE)

- Conditional variational autoencoder
- Code c is always observable

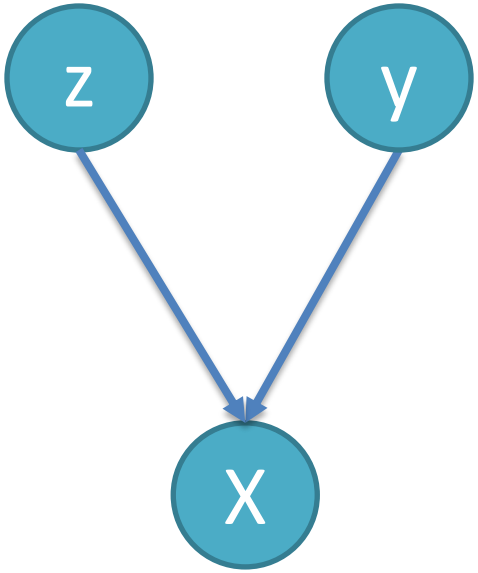
$$\mathcal{L}(x, c; \theta, \phi) = \mathbb{E}_{q_\phi(z|x,c)}\left[\log p_\theta(x|z,c)\right] - KL(q_\phi(z|x,c)||p(z|c))$$

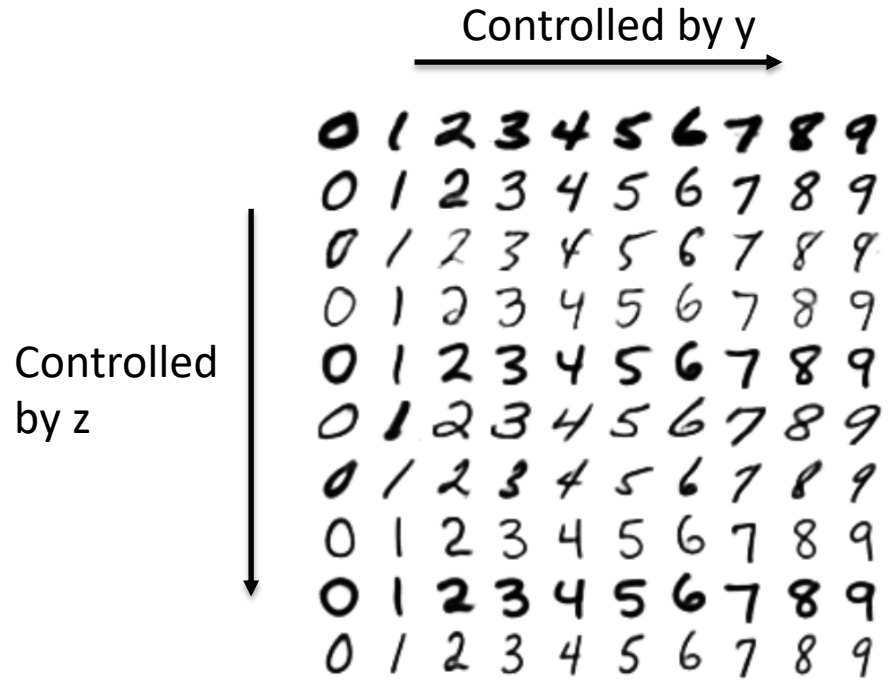
1.Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014.
2.Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *NIPS*, 2015.

Variational Autoencoder (VAE)

- Semi-supervised VAE

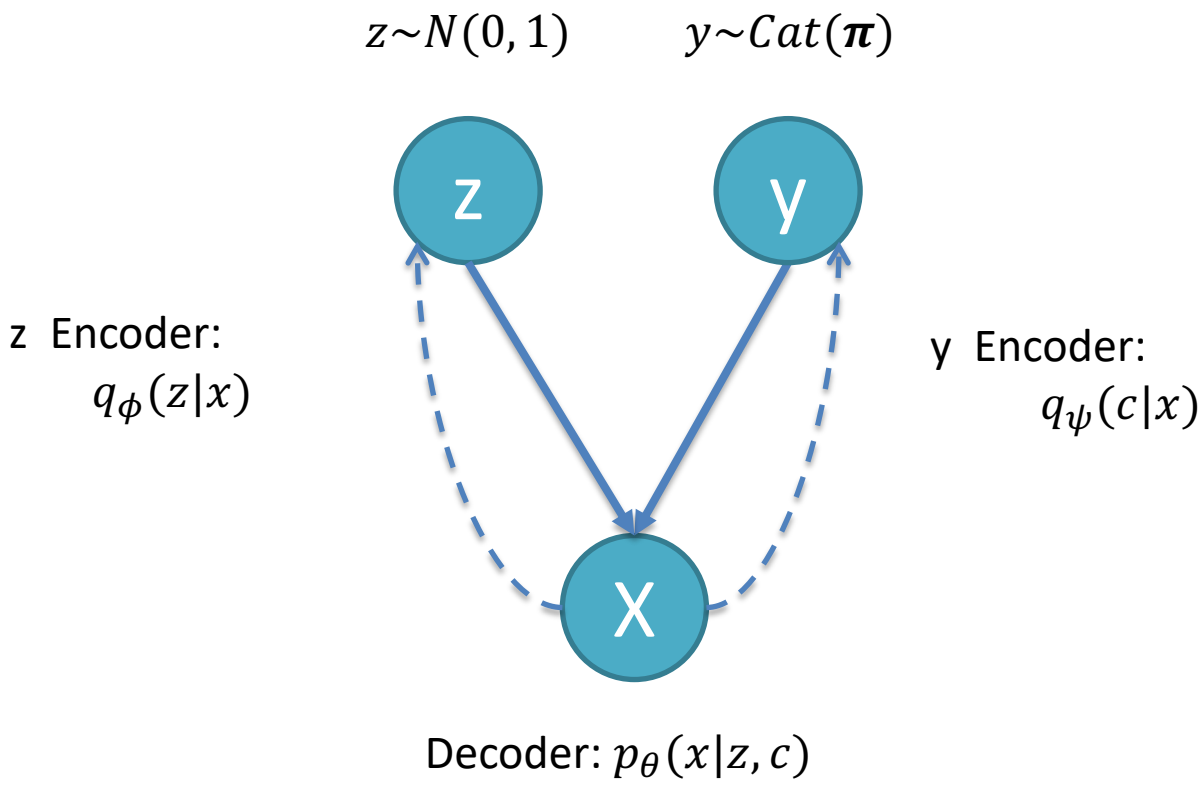
$z \sim N(0, 1)$ $y \sim Cat(\boldsymbol{\pi})$ Cat is category distribution, $P(y = i) = \pi_i$





Variational Autoencoder (VAE)

- Semi-supervised VAE



Variational Autoencoder (VAE)

- Semi-supervised autoencoder

$$\log p_\theta(\mathbf{x}, y) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},y)} \left[\log p_\theta(\mathbf{x}|y,\mathbf{z}) + \log p_\theta(y) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x},y)\right] = -\mathcal{L}(\mathbf{x},y)$$

Labeled data, y is observable

$$\begin{aligned}\log p_\theta(\mathbf{x}) &\geq \mathbb{E}_{q_\phi(y,\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|y,\mathbf{z}) + \log p_\theta(y) + \log p(\mathbf{z}) - \log q_\phi(y,\mathbf{z}|\mathbf{x})\right] \\ &= \sum\nolimits_y q_\phi(y|\mathbf{x})(-\mathcal{L}(\mathbf{x},y)) + \mathcal{H}(q_\phi(y|\mathbf{x})) = -\mathcal{U}(\mathbf{x}).\end{aligned}$$

Unlabeled data, y is not observable

$$\mathcal{J} = \sum\nolimits_{(\mathbf{x},y)\sim\widetilde{p}_l} \mathcal{L}(\mathbf{x},y) + \sum\nolimits_{\mathbf{x}\sim\widetilde{p}_u} \mathcal{U}(\mathbf{x})$$

1.Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014.
2.Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *NIPS*, 2015.

Summary

- Simple autoencoders
  - Semi-supervised version
- Recursive autoencoders for tree data
- Variants
  - Unfold recursive autoencoders
  - Variational autoencoders
  - Sparse autoencoders