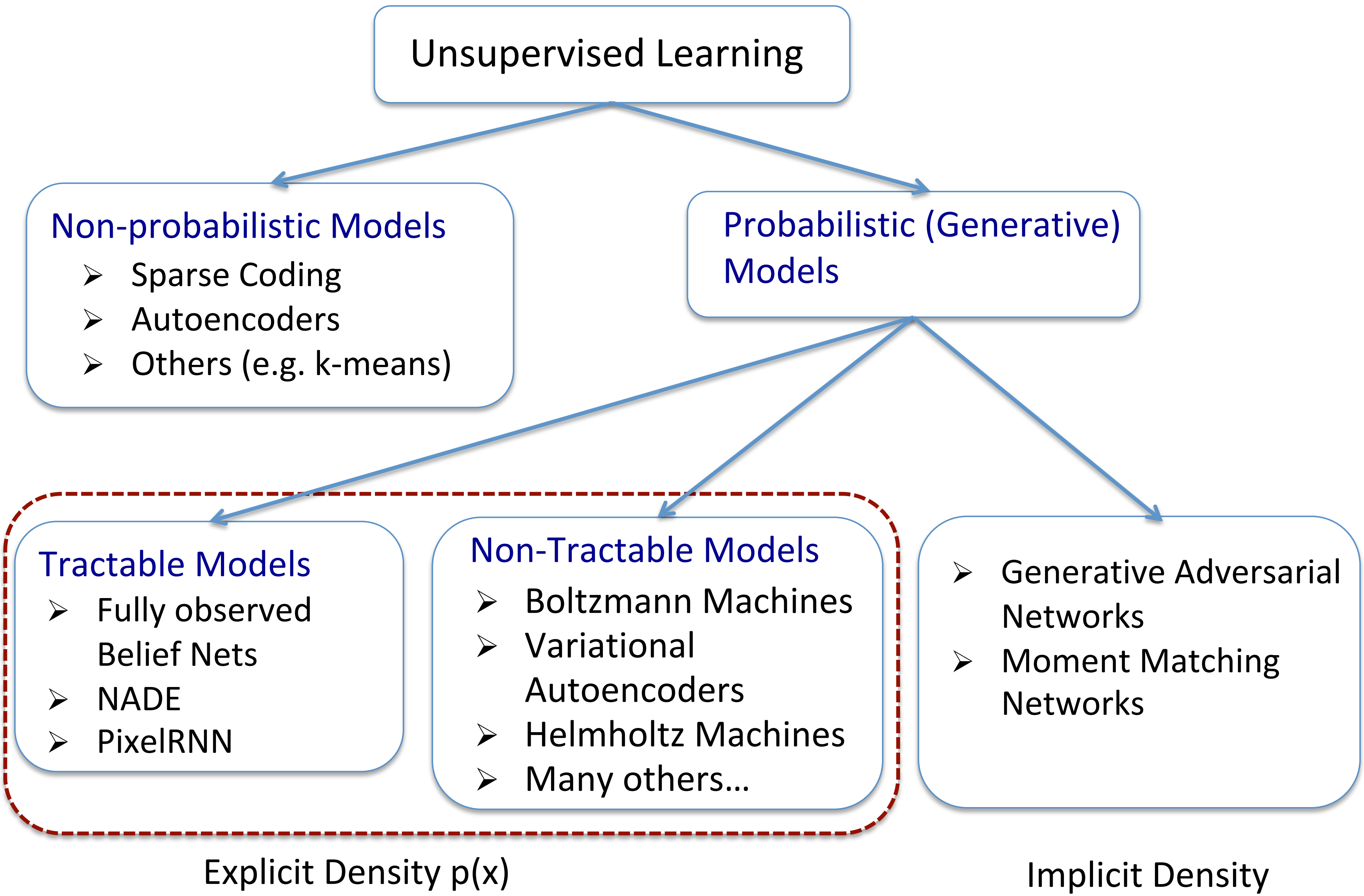
Deep Unsupervised Learning

Russ Salakhutdinov

Machine Learning Department
Carnegie Mellon University
Canadian Institute of Advanced Research

Carnegie Mellon University

MACHINE LEARNING DEPARTMENT



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

Talk Roadmap

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

Sparse Coding

- Sparse coding (Olshausen & Field, 1996). Originally developed to explain early visual processing in the brain (edge detection).

- Objective: Given a set of input data vectors $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, learn a dictionary of bases $\{\phi_1, \phi_2, ..., \phi_K\}$, such that:

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

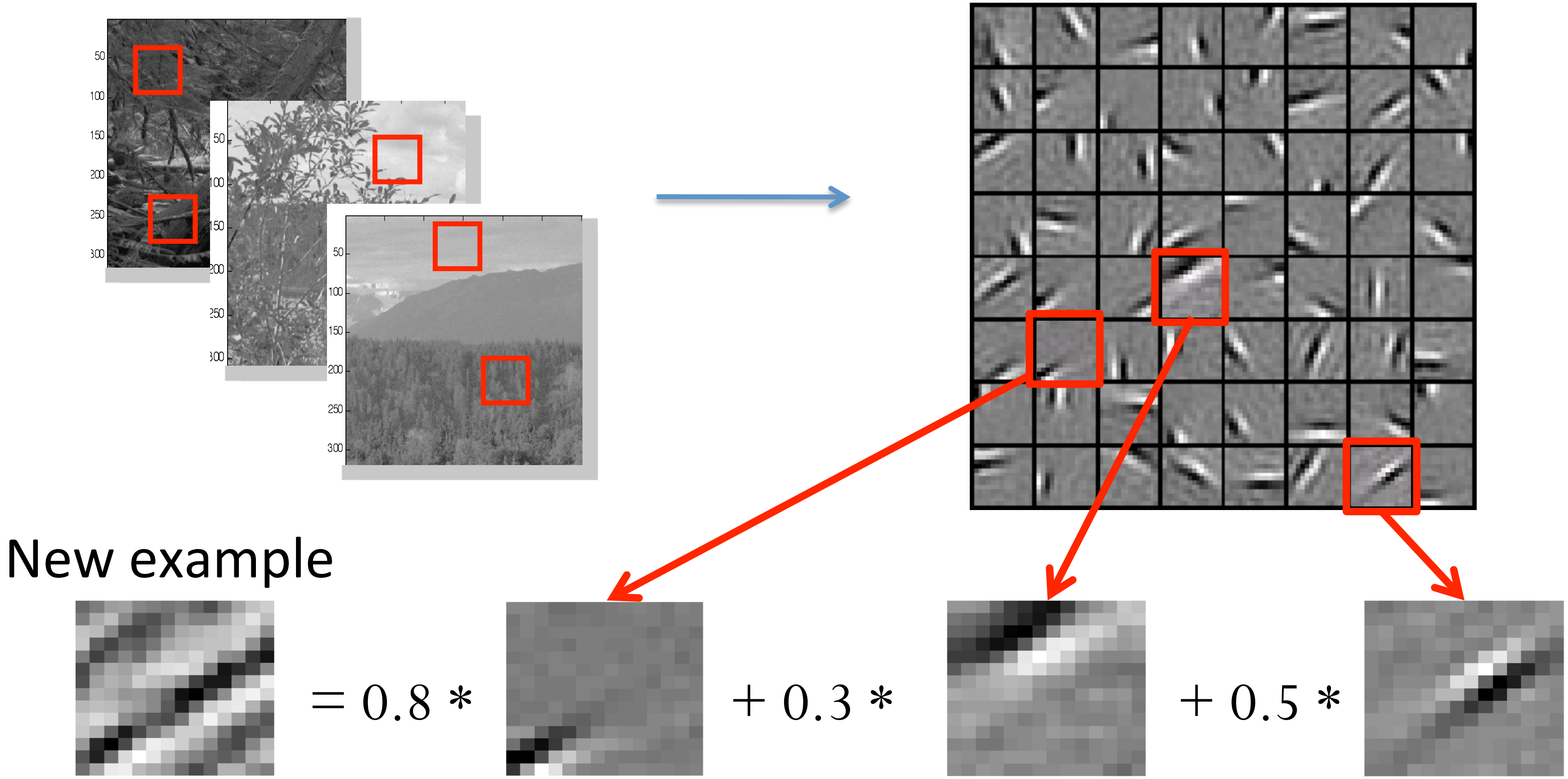
Sparse: mostly zeros

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

Sparse Coding

Natural Images

Learned bases: “Edges”



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

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

Slide Credit: Honglak Lee

Sparse Coding: Training

- Input image patches: $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N \in \mathbb{R}^D$
- Learn dictionary of bases: $\phi_1, \phi_2, ..., \phi_K \in \mathbb{R}^D$

$$\min_{\mathbf{a},\phi} \underbrace{\sum_{n=1}^{N} \left|\left| \mathbf{x}_n - \sum_{k=1}^{K} a_{nk}\phi_k \right|\right|_2^2}_{\text{Reconstruction error}} + \lambda \underbrace{\sum_{n=1}^{N}\sum_{k=1}^{K} |a_{nk}|}_{\text{Sparsity penalty}}$$

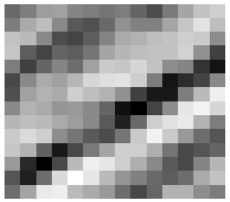
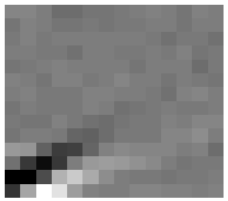
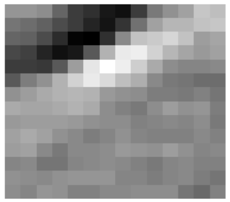
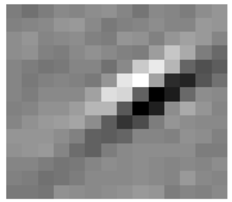
- Alternating Optimization:

  1. Fix dictionary of bases $\phi_1, \phi_2, ..., \phi_K$ and solve for activations **a** (a standard Lasso problem).
  2. Fix activations **a**, optimize the dictionary of bases (convex QP problem).

Sparse Coding: Testing Time

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

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

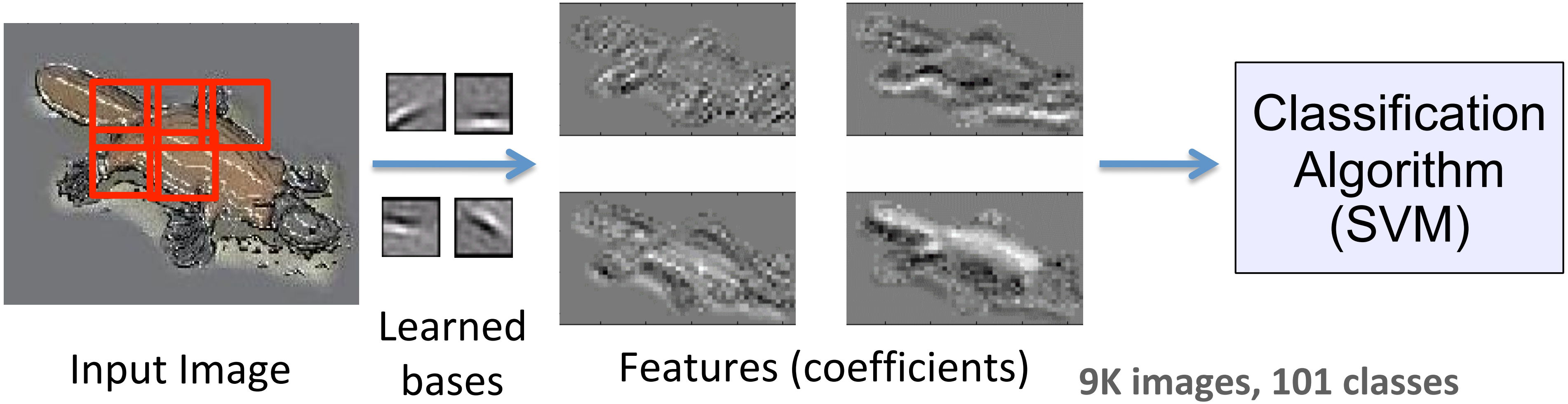
 $= 0.8 *$  $+ 0.3 *$  $+ 0.5 *$ 

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

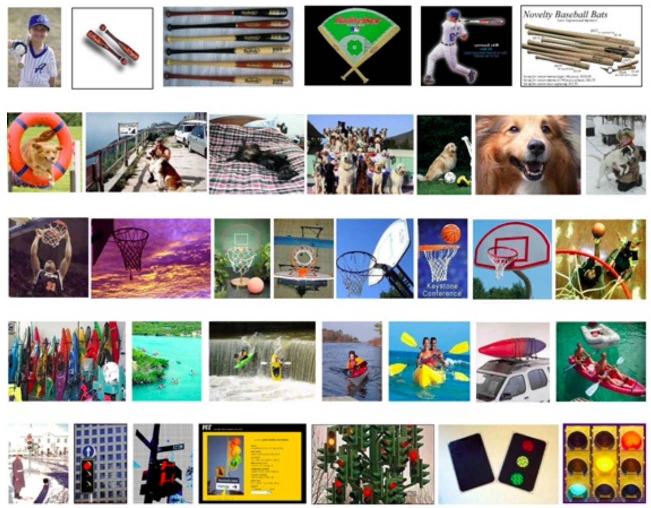
[0, 0, … **0.8**, …, **0.3**, …, **0.5**, …] = coefficients (feature representation)

Image Classification

Evaluated on Caltech101 object category dataset.



9K images, 101 classes



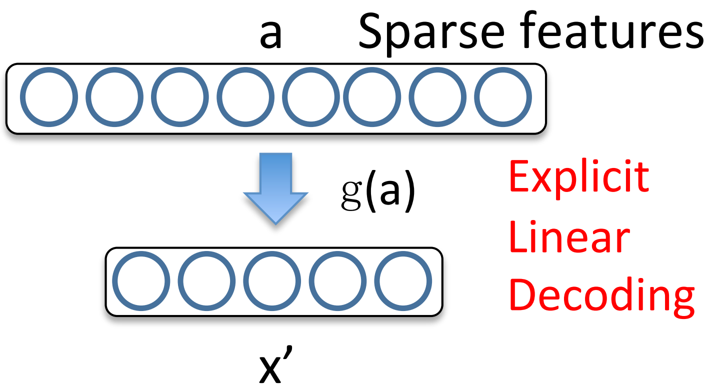
Algorithm	Accuracy
Baseline (Fei-Fei et al., 2004)	16%
PCA	37%
Sparse Coding	**47%**

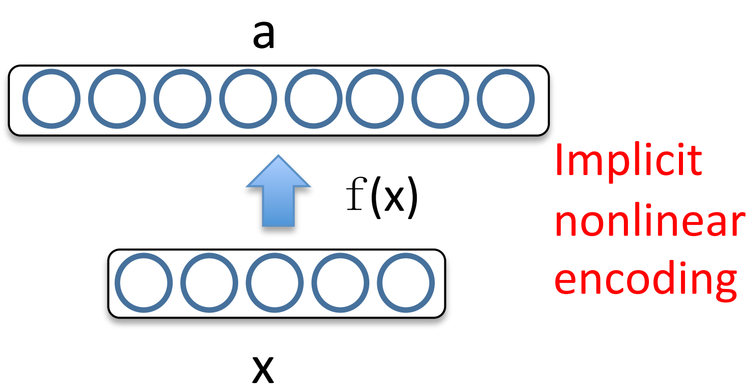
Slide Credit: Honglak Lee

Lee, Battle, Raina, Ng, 2006

Interpreting Sparse Coding

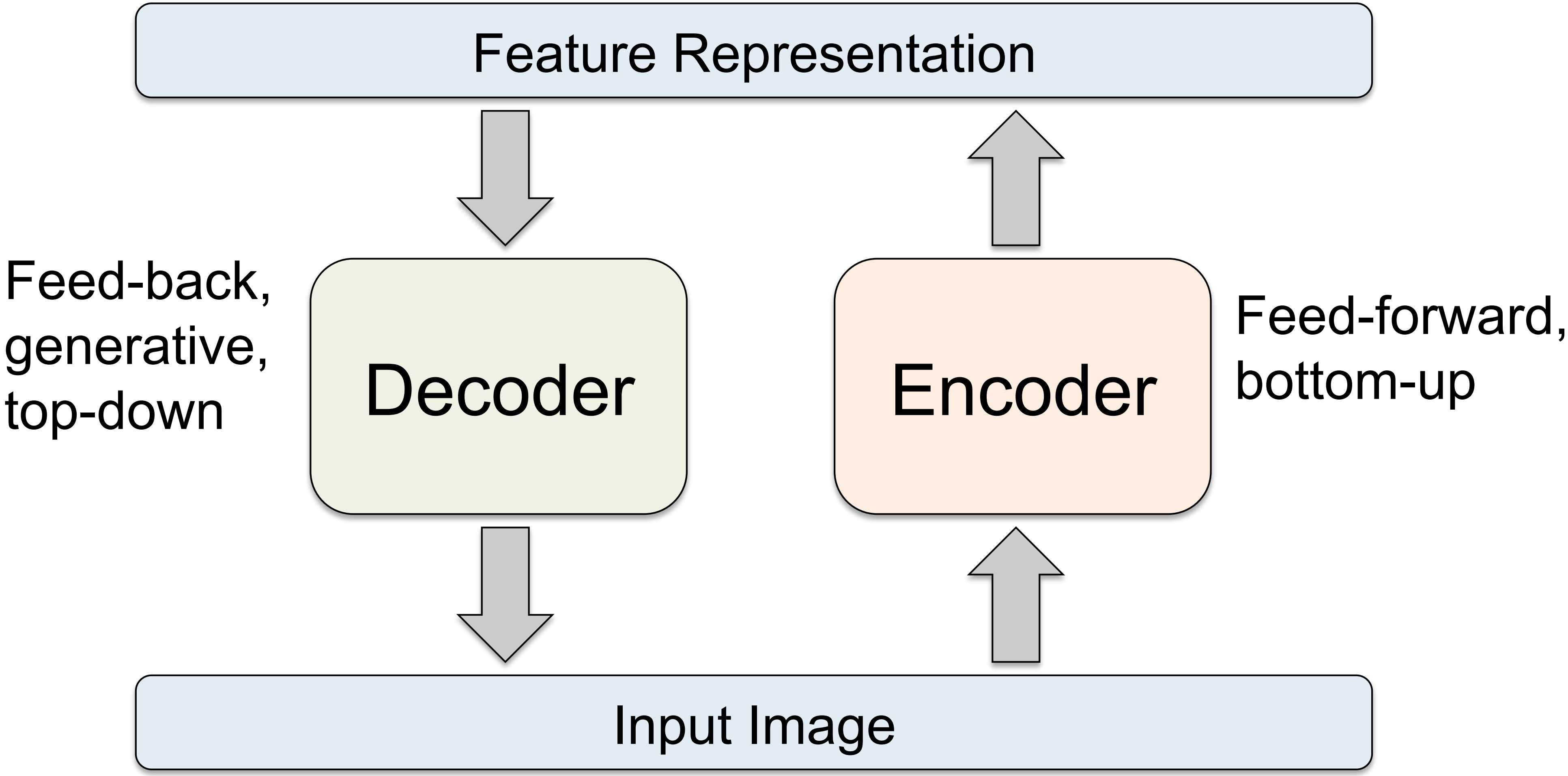
$$\min_{\mathbf{a},\phi} \sum_{n=1}^{N} \left|\left| \mathbf{x}_n - \sum_{k=1}^{K} a_{nk}\phi_k \right|\right|_2^2 + \lambda \sum_{n=1}^{N} \sum_{k=1}^{K} |a_{nk}|$$





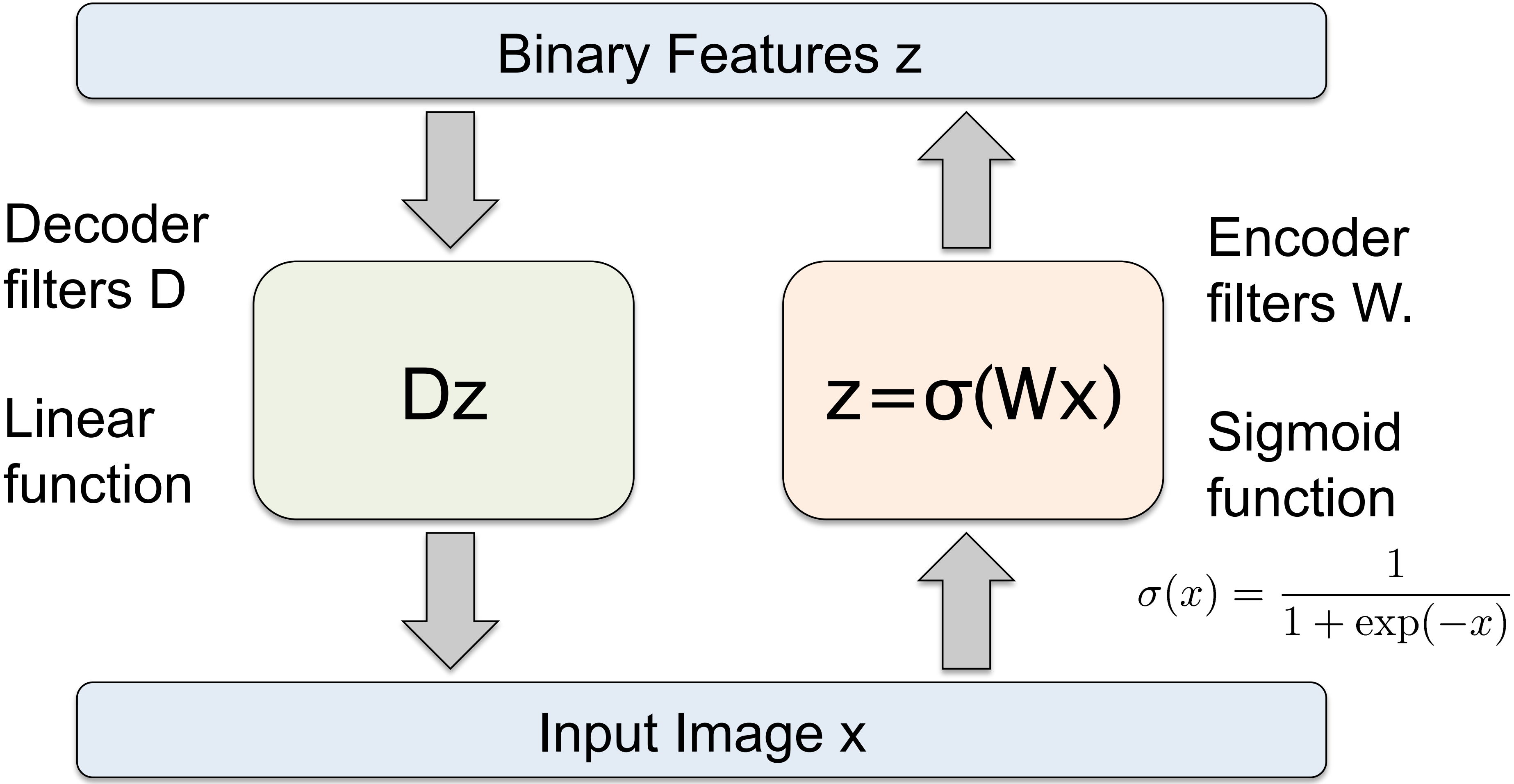
- Sparse, over-complete representation **a.**
- Encoding **a** = f(**x**) is implicit and nonlinear function of **x**.
- Reconstruction (or decoding) **x'** = g(**a**) is linear and explicit.

Autoencoder

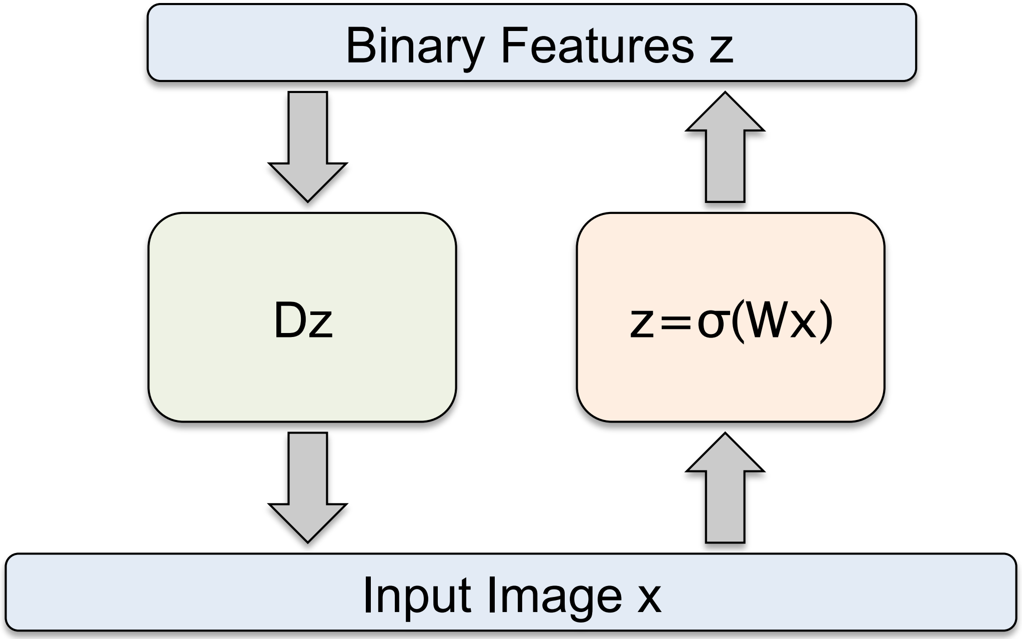


- Details of what goes insider the encoder and decoder matter!
- Need constraints to avoid learning an identity.

Autoencoder



Autoencoder



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

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

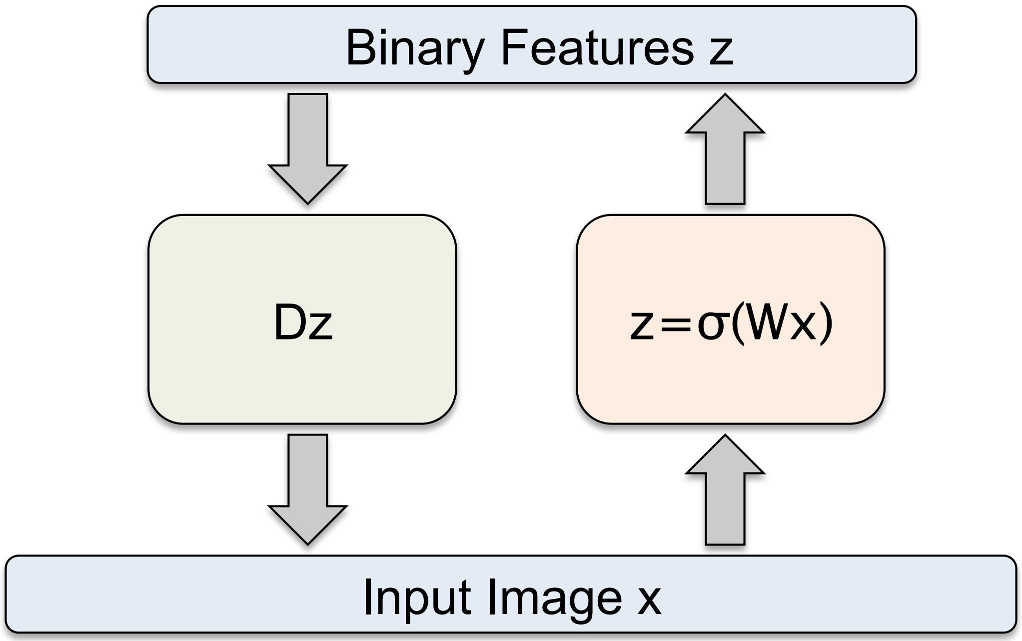
$$y_j(\mathbf{x}, W, D) = \underbrace{\sum_{k=1}^{K} D_{jk}}_{\text{Decoder}} \underbrace{\sigma\left(\sum_{i=1}^{D} W_{ki} x_i\right)}_{\text{Encoder}}, \quad j = 1, .., D.$$

Decoder

Encoder

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

Autoencoder

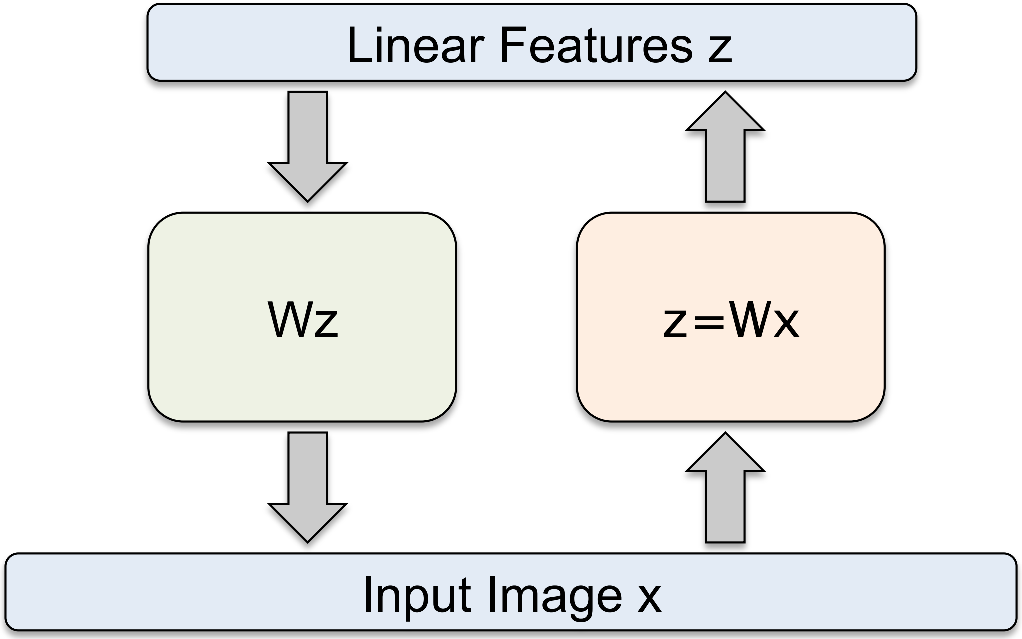


- An autoencoder with D inputs, D outputs, and K hidden units, with K<D.

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

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

Autoencoder

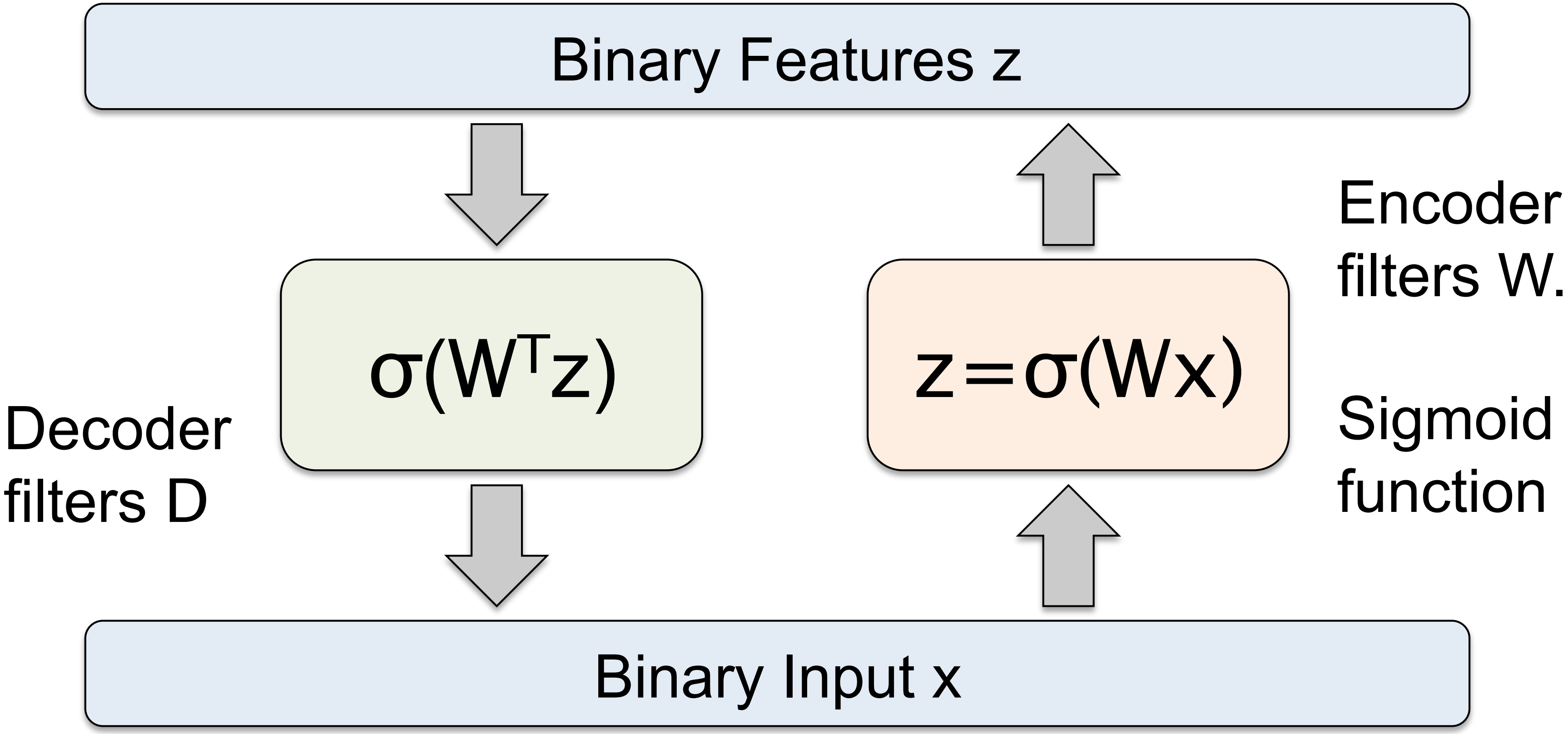


- If the hidden and output layers are linear, it will learn hidden units that are a linear function of the data and minimize the squared error.

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

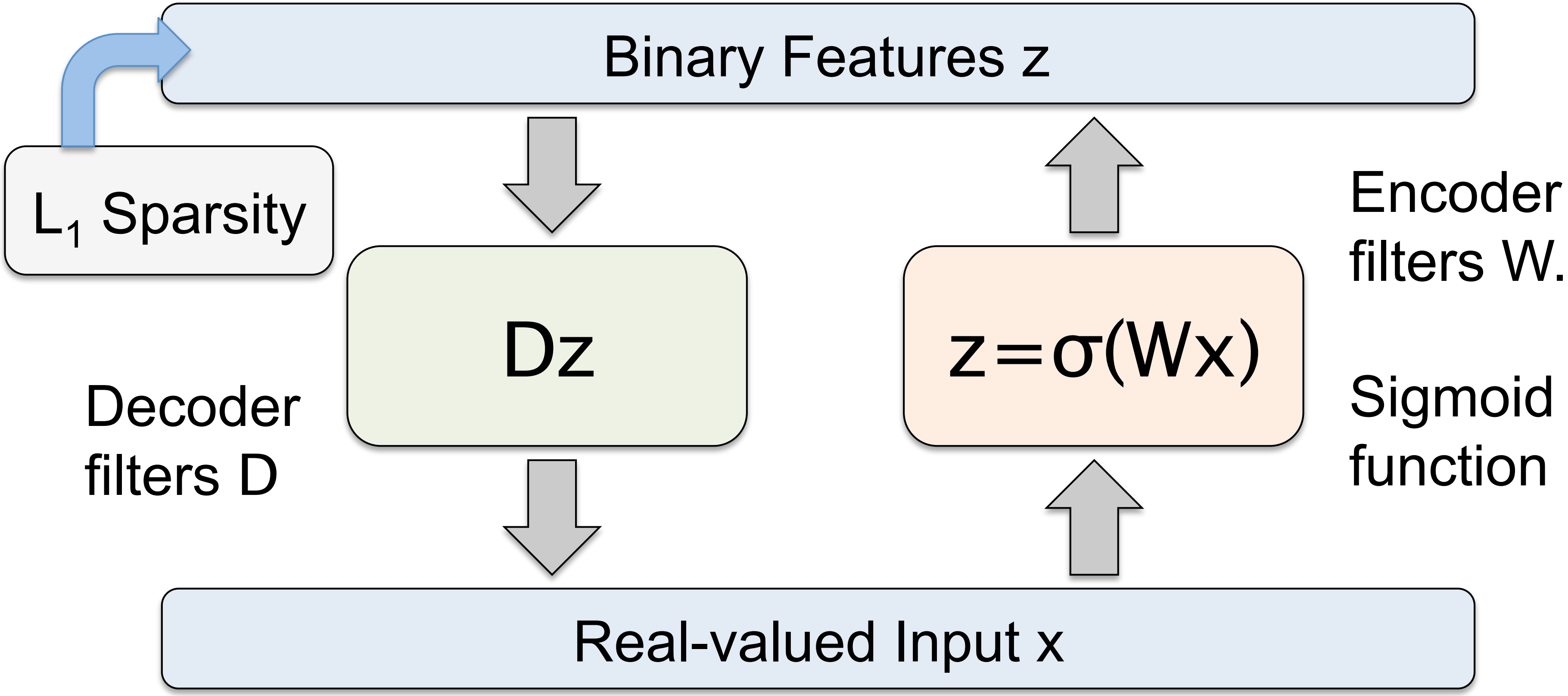
- With nonlinear hidden units, we have a nonlinear generalization of PCA.

Another Autoencoder Model



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

Predictive Sparse Decomposition

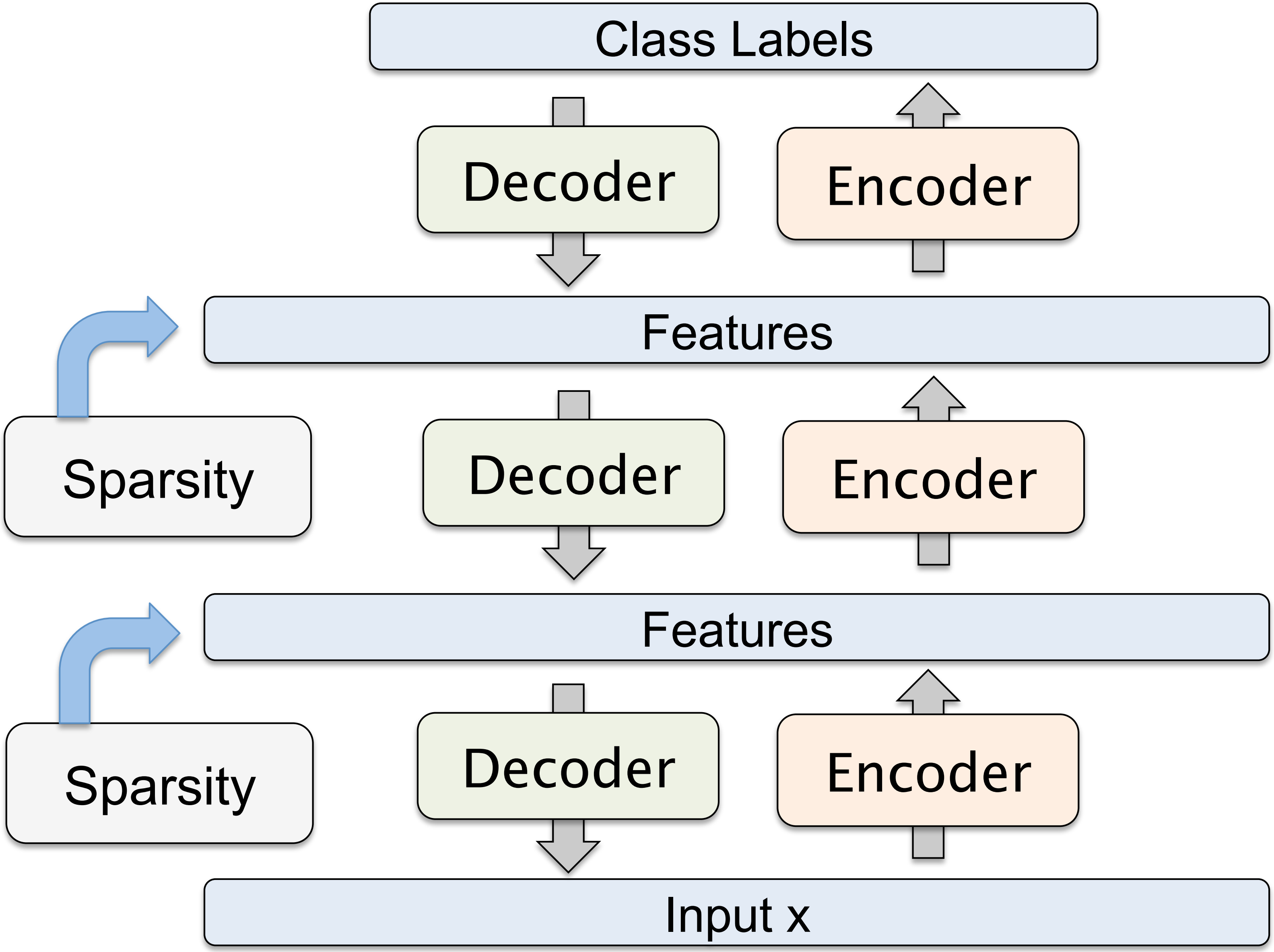


At training time

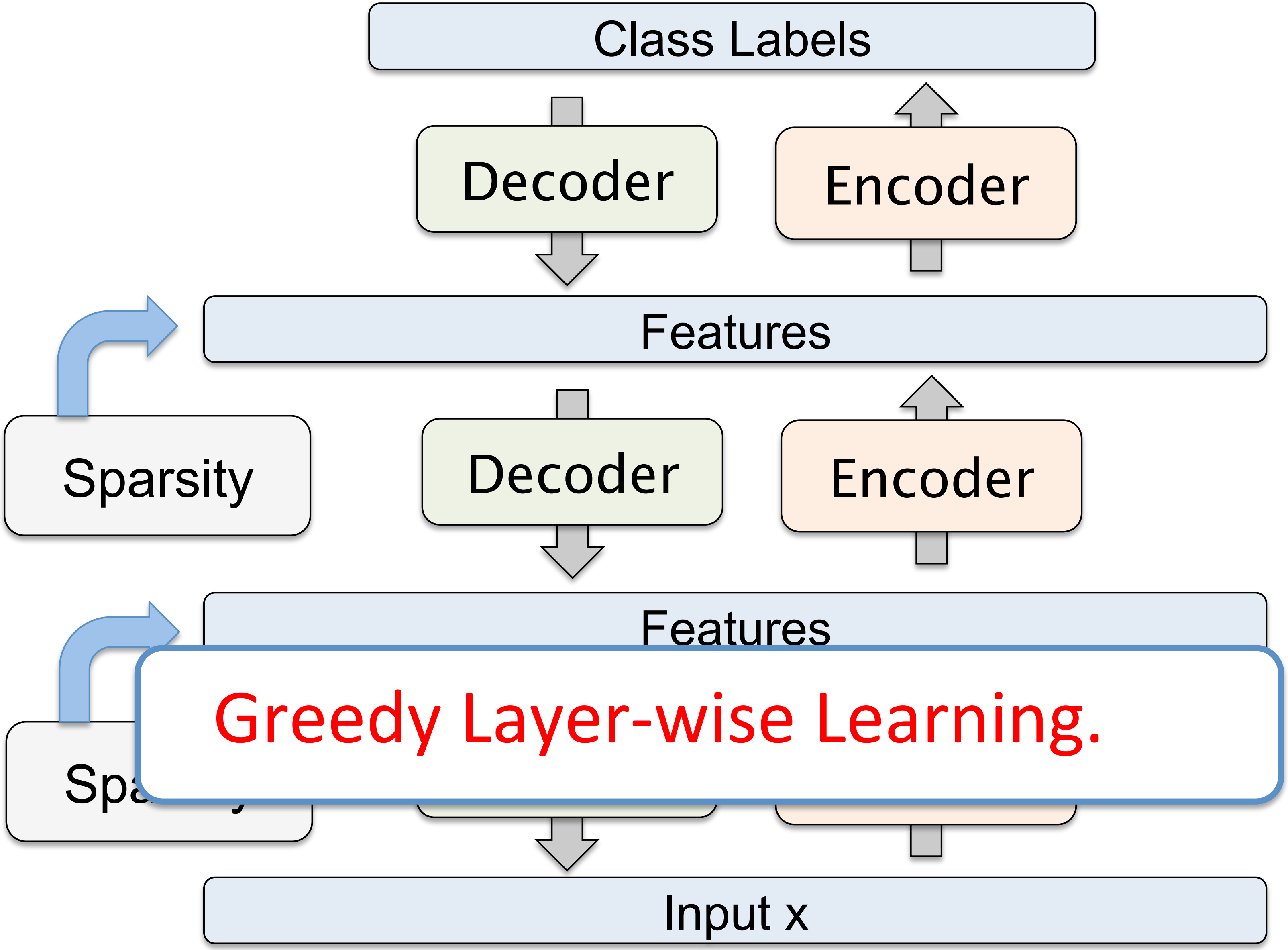
$$\min_{D,W,\mathbf{z}} \underbrace{||D\mathbf{z} - \mathbf{x}||_2^2 + \lambda|\mathbf{z}|_1}_{\text{Decoder}} + \underbrace{||\sigma(W\mathbf{x}) - \mathbf{z}||_2^2}_{\text{Encoder}}$$

Kavukcuoglu, Ranzato, Fergus, LeCun, 2009

Stacked Autoencoders



Stacked Autoencoders



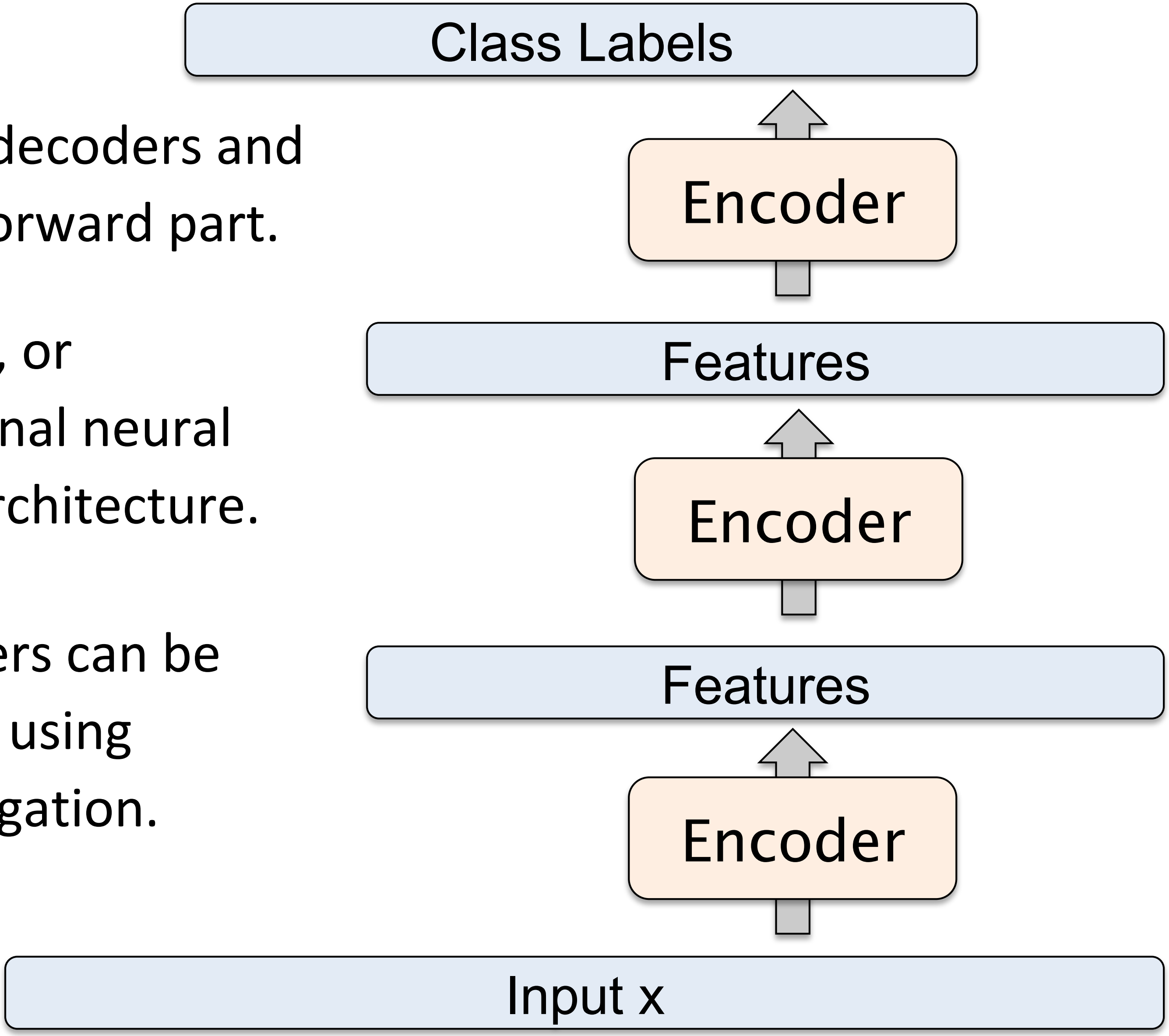
Greedy Layer-wise Learning.

Stacked Autoencoders

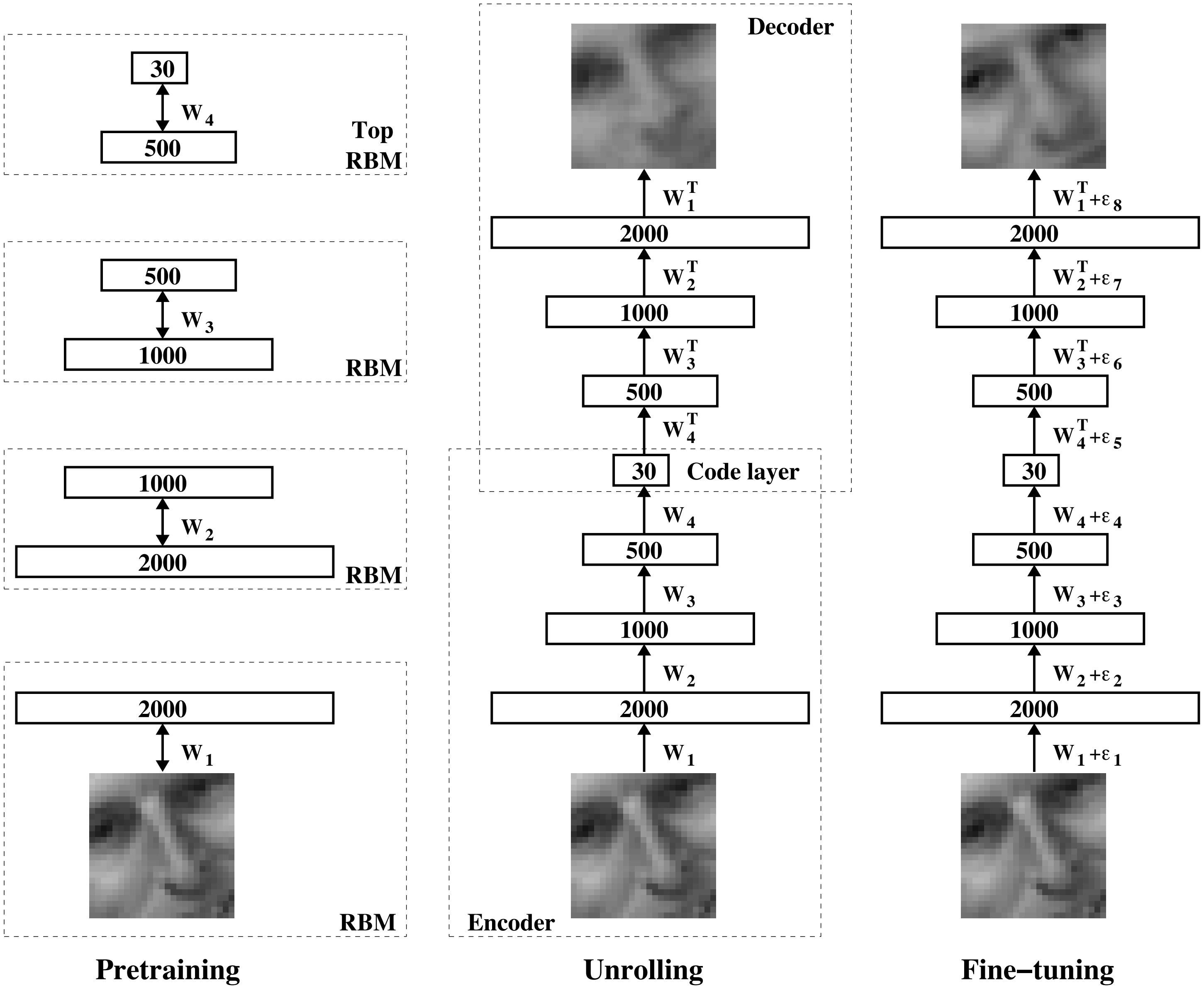
- Remove decoders and use feed-forward part.

- Standard, or convolutional neural network architecture.

- Parameters can be fine-tuned using backpropagation.



Deep Autoencoders



Deep Autoencoders

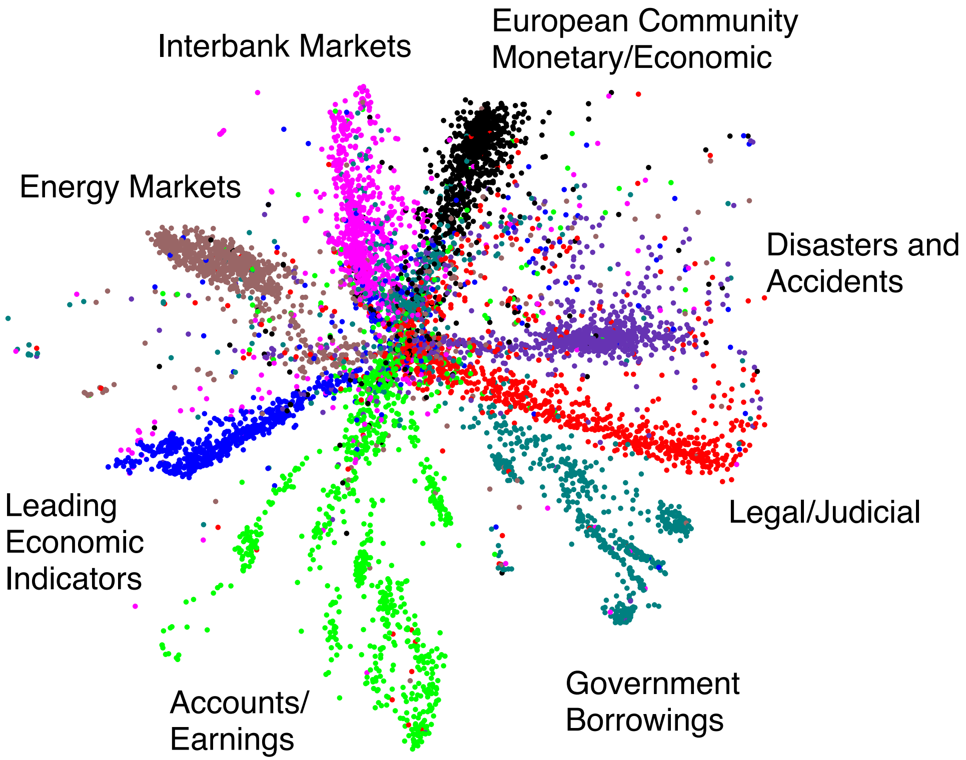
• 25x25 – 2000 – 1000 – 500 – 30 autoencoder to extract 30-D real-valued codes for Olivetti face patches.

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

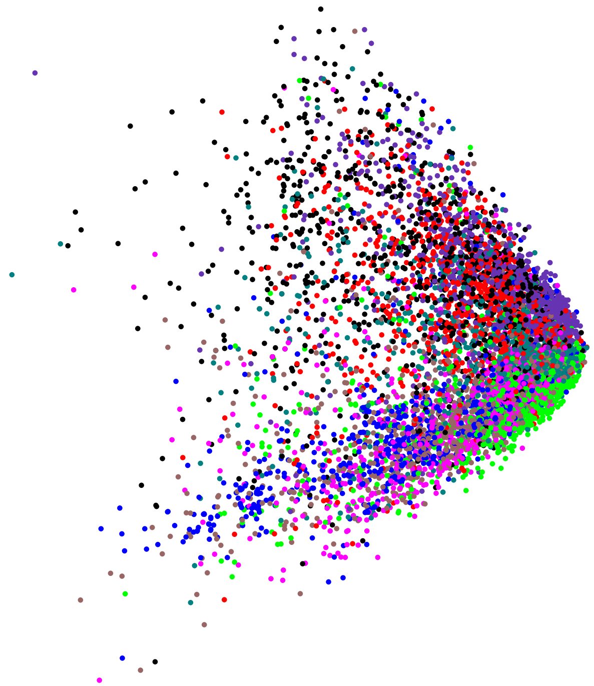
• **Middle**: Reconstructions by the 30-dimensional deep autoencoder.

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

Information Retrieval



2-D LSA space



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

• "Bag-of-words" representation: each article is represented as a vector containing the counts of the most frequently used 2000 words in the training set.

(Hinton and Salakhutdinov, Science 2006)

Talk Roadmap

- Basic Building Blocks:
  - Sparse Coding
  - Autoencoders

- Deep Generative Models
  - Restricted Boltzmann Machines
  - Deep Boltzmann Machines
  - Helmholtz Machines / Variational Autoencoders

- Generative Adversarial Networks

Fully Observed Models

- Explicitly model conditional probabilities:

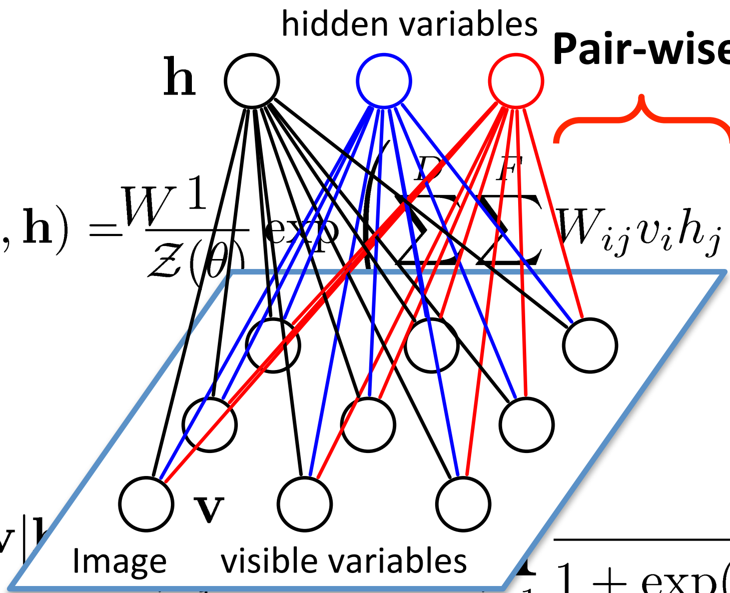
$$p_{\text{model}}(\boldsymbol{x}) = p_{\text{model}}(x_1) \prod_{i=2}^{n} p_{\text{model}}(x_i \mid x_1, \ldots, x_{i-1})$$

Each conditional can be a complicated neural network

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

Pixel CNN

Restricted Boltzmann Machines



Pair-wise **Unary**

$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\left(\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij}v_ih_j + \sum_{i=1}^{D} v_ib_i + \sum_{j=1}^{F} h_ja_j\right)$$

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

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

RBM is a Markov Random Field with:

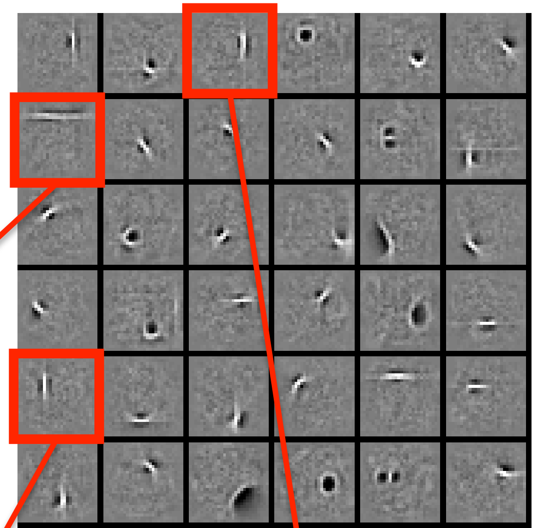
- Stochastic binary visible variables $\mathbf{v} \in \{0,1\}^D$.
- Stochastic binary hidden variables $\mathbf{h} \in \{0,1\}^F$.
- Bipartite connections.

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

Learning Features

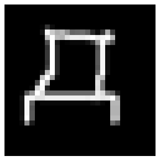
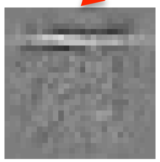
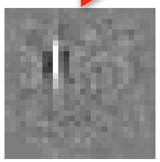
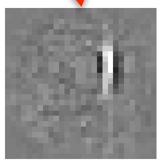
Observed Data
Subset of 25,000 characters

Learned W: "edges"
Subset of 1000 features



Sparse representations

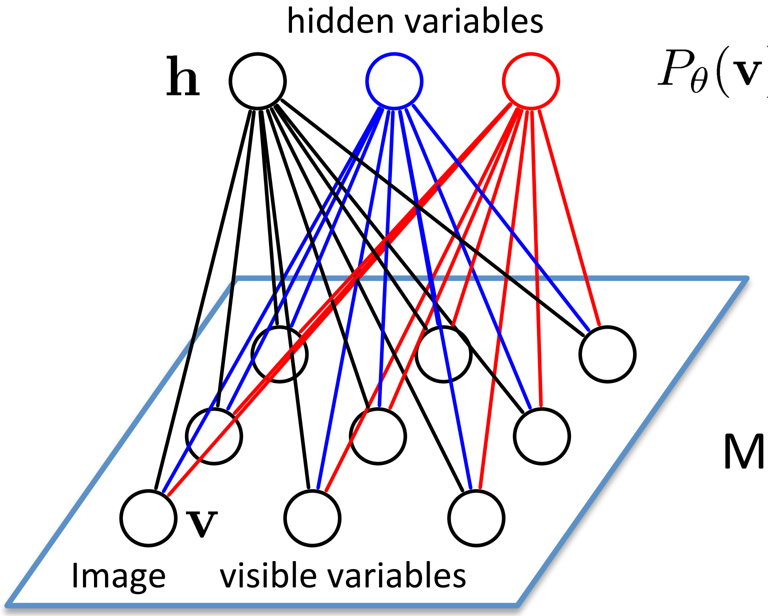
New Image: $p(h_7 = 1|v)$ $p(h_{29} = 1|v)$

 = $\sigma\Big(0.99 \times$  $+ \; 0.97 \times$  $+ \; 0.82 \times$  ....$\Big)$

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

Logistic Function: Suitable for modeling binary images

Model Learning



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

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

Maximize log-likelihood objective:

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

Derivative of the log-likelihood:

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

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

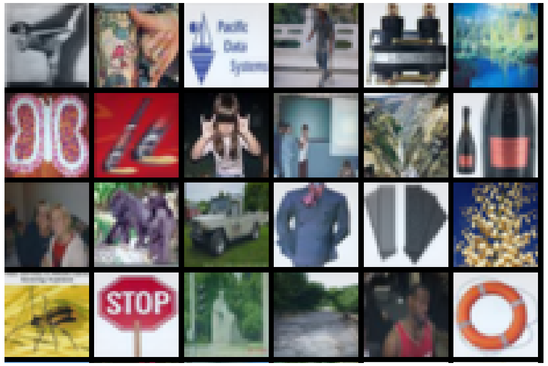
Difficult to compute: exponentially many configurations

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

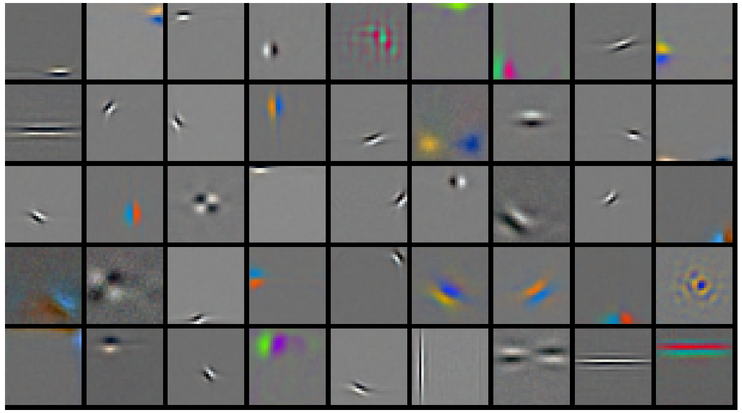
$$P_{data}(\mathbf{v}) = \frac{1}{N} \sum_{n} \delta(\mathbf{v} - \mathbf{v}^{(n)})$$

RBMs for Word Counts

4 million **unlabelled** images



Learned features (out of 10,000)



REUTERS

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

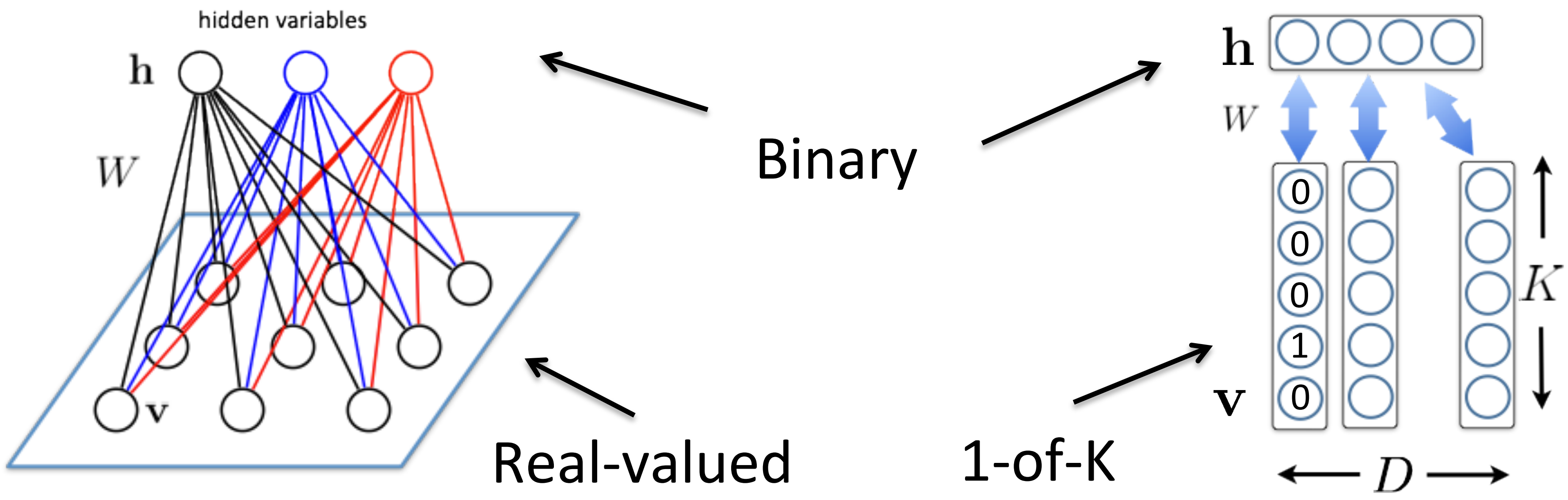
RBMs for Word Counts

One-step reconstruction from the RBM model

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

Different Data Modalities

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



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

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

Product of Experts

The joint distribution is given by:

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

Marginalizing over hidden variables:

Product of Experts

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

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

Silvio Berlusconi

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

Product of Experts

The joint distribution is given by:

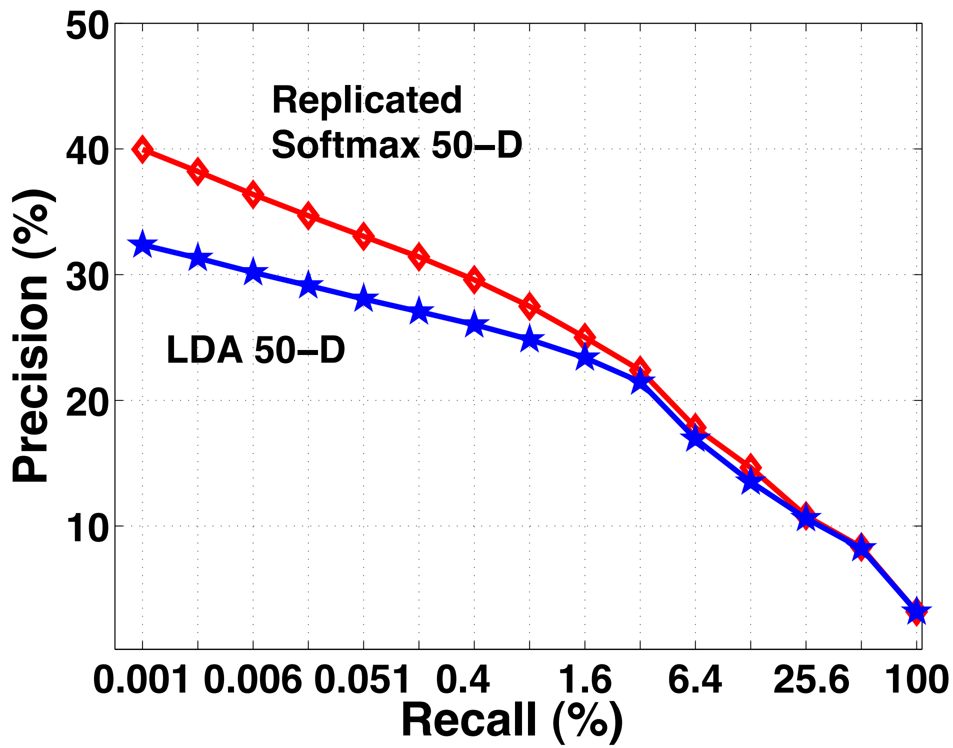
$$P_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{\mathcal{Z}(\theta)} \exp\Big(\sum_{ij} W_{ij} v_i h_j + \sum_i b_i v_i + \sum_j a_j h_j\Big)$$

Marginalizing c

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

duct of Experts

$W_{ij} v_i)\Big)$



government
authority
power
empire
federation

clint
hou
pres
bill
con

", "corruption"

bine to give very

word "Silvio

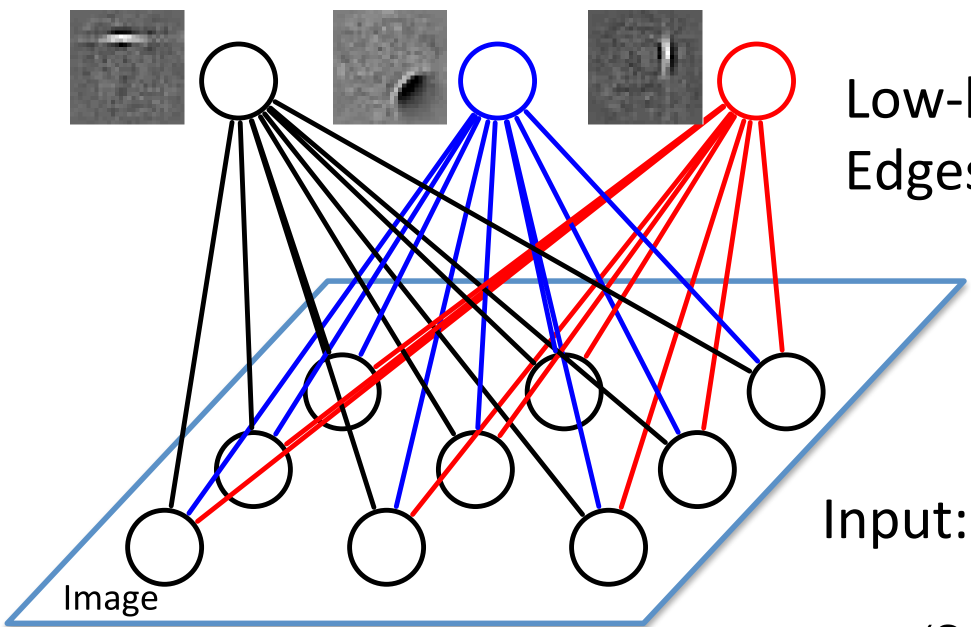
Silvio Berlusconi

Berlusconi".

Talk Roadmap

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

Deep Boltzmann Machines



Low-level features:
Edges

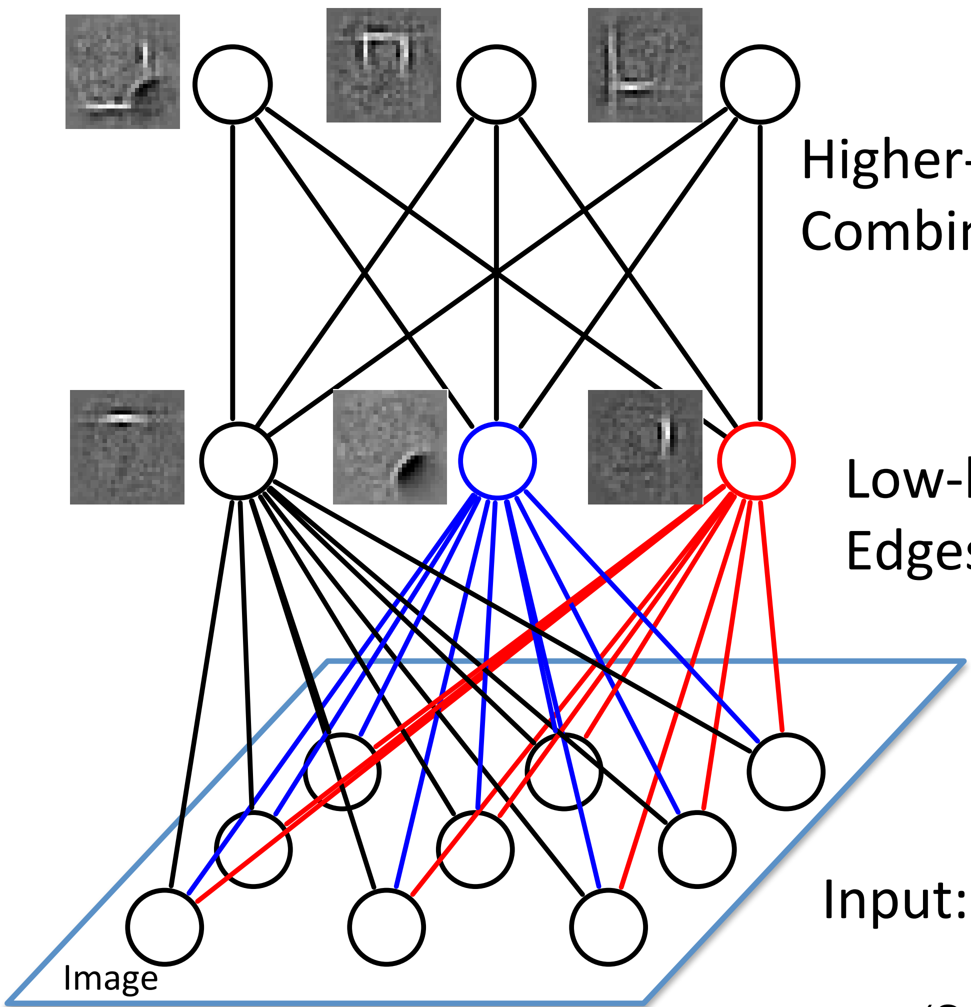
Built from **unlabeled** inputs.

Input: Pixels

(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)

Deep Boltzmann Machines

Learn simpler representations,
then compose more complex ones



Higher-level features:
Combination of edges

Low-level features:
Edges

Built from **unlabeled** inputs.

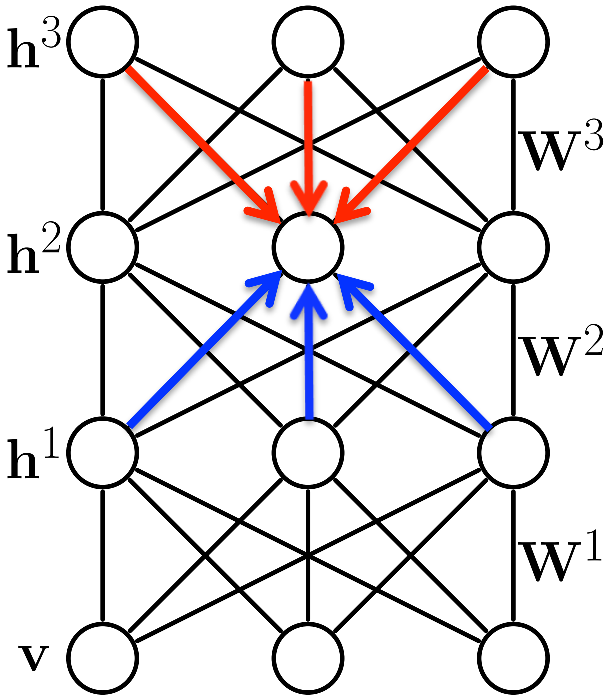
Input: Pixels

(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)

Model Formulation

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

Same as RBMs



Input

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

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

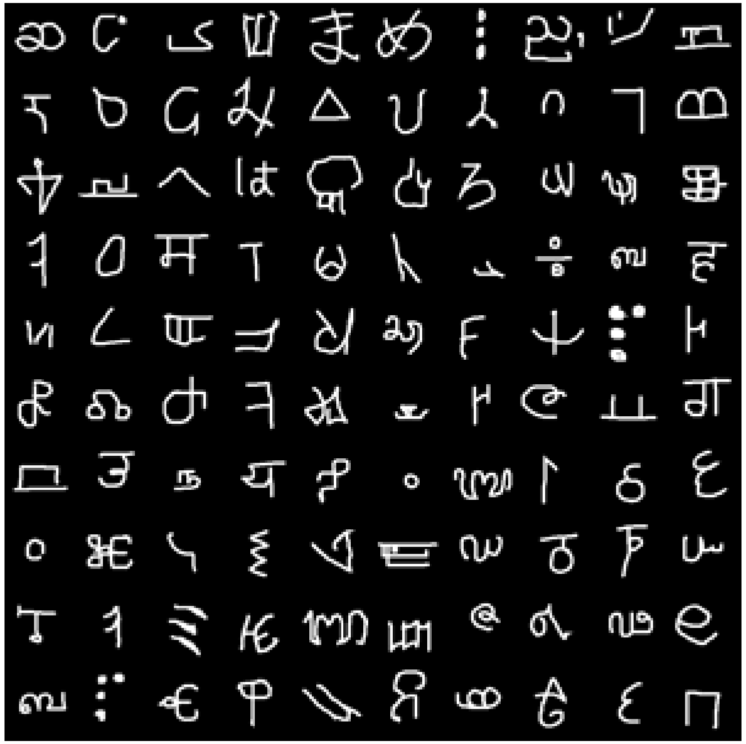
$$P(h^2_j = 1 | \mathbf{h}^1, \mathbf{h}^3) = \sigma\left(\sum_k W^3_{kj} h^3_k + \sum_m W^2_{mj} h^1_m\right)$$

Top-down   Bottom-up

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

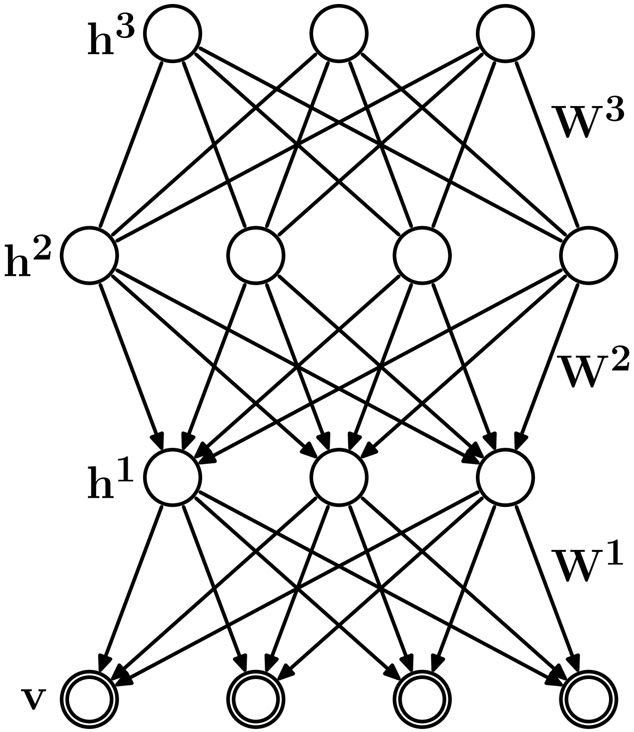
Good Generative Model?

Handwritten Characters

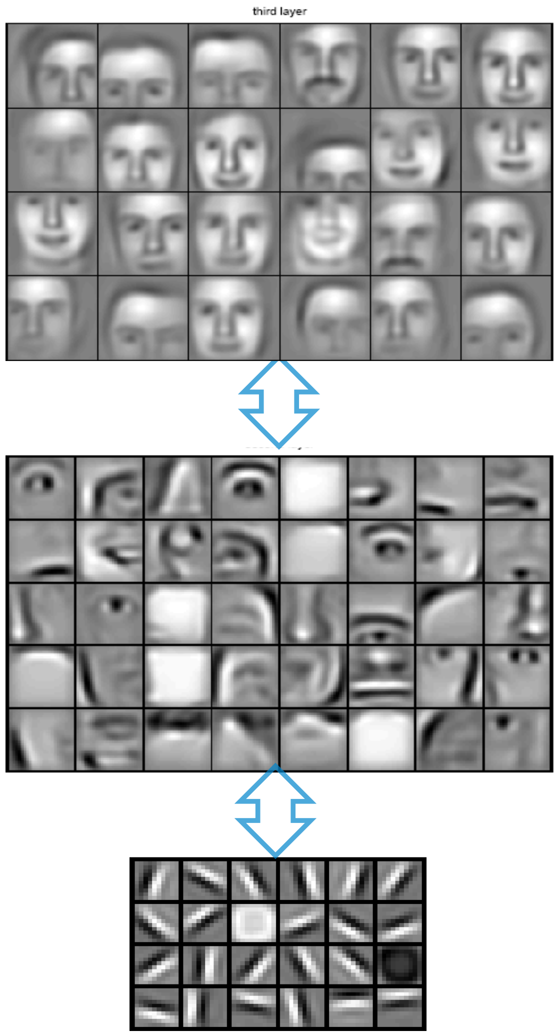


Learning Part-based Representation

Convolutional DBN



Faces



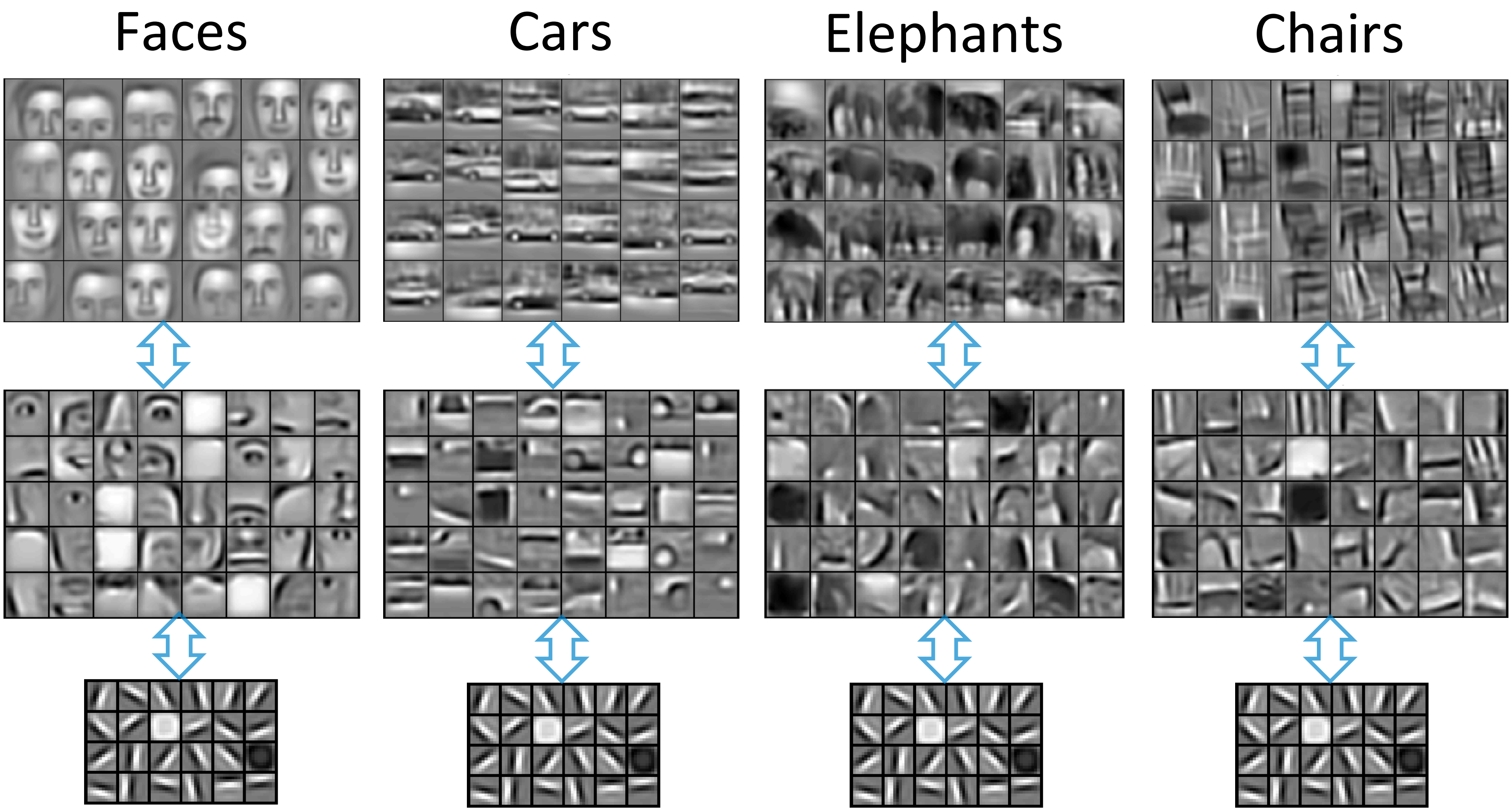
Groups of parts.

Object Parts

Trained on face images.

(Lee, Grosse, Ranganath, Ng, ICML 2009)

Learning Part-based Representation



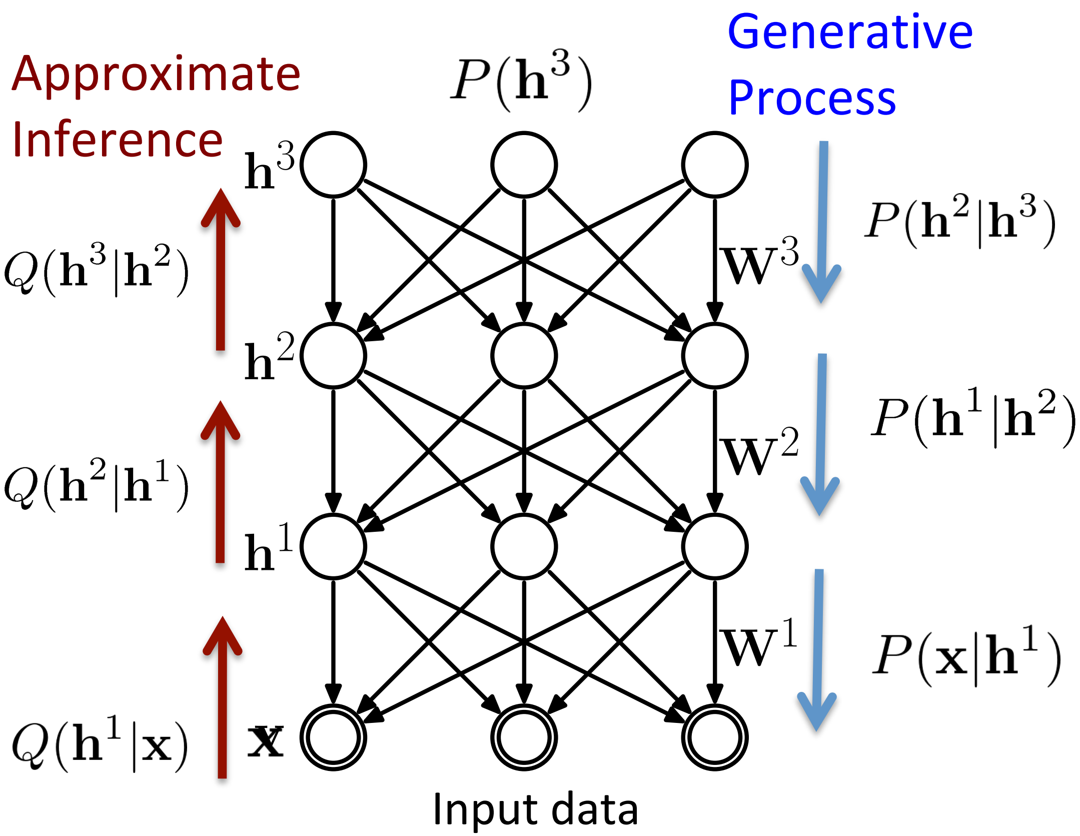
(Lee, Grosse, Ranganath, Ng, ICML 2009)

Talk Roadmap

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

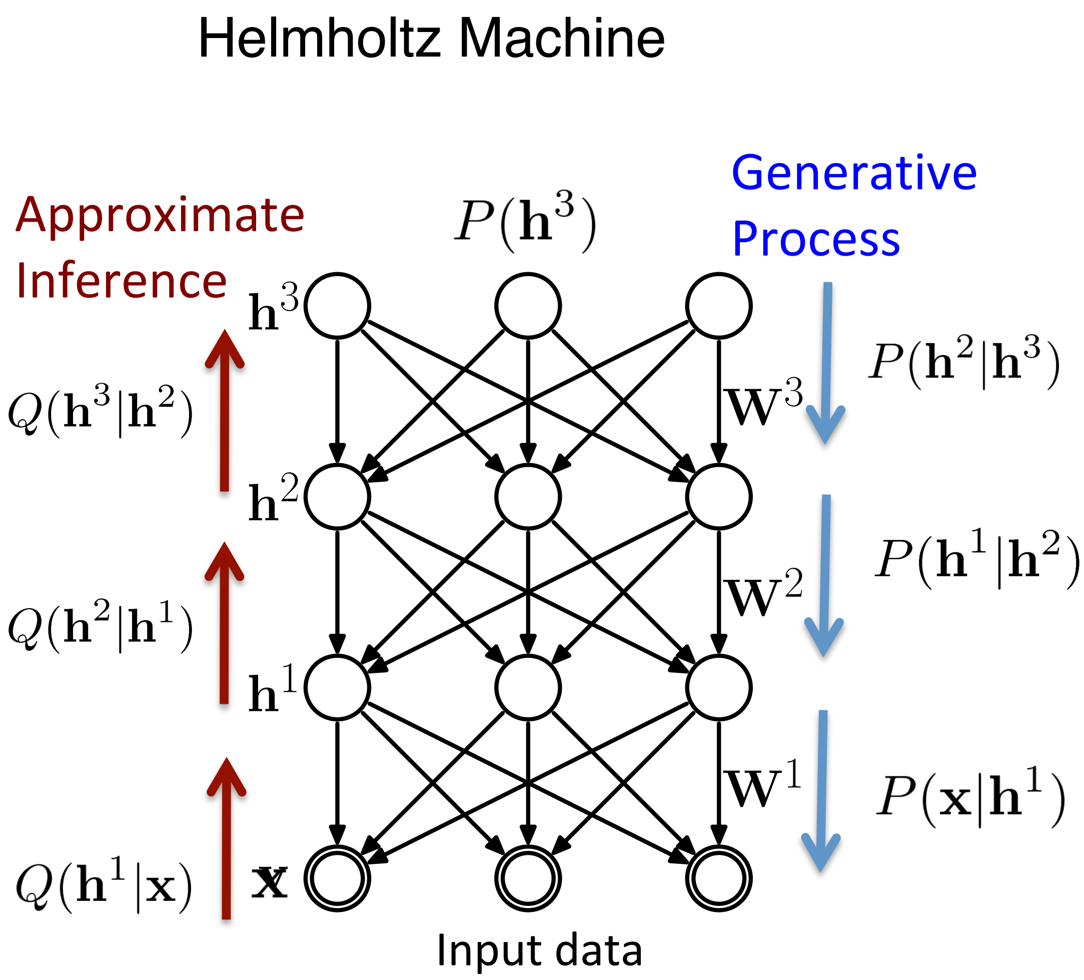
Helmholtz Machines

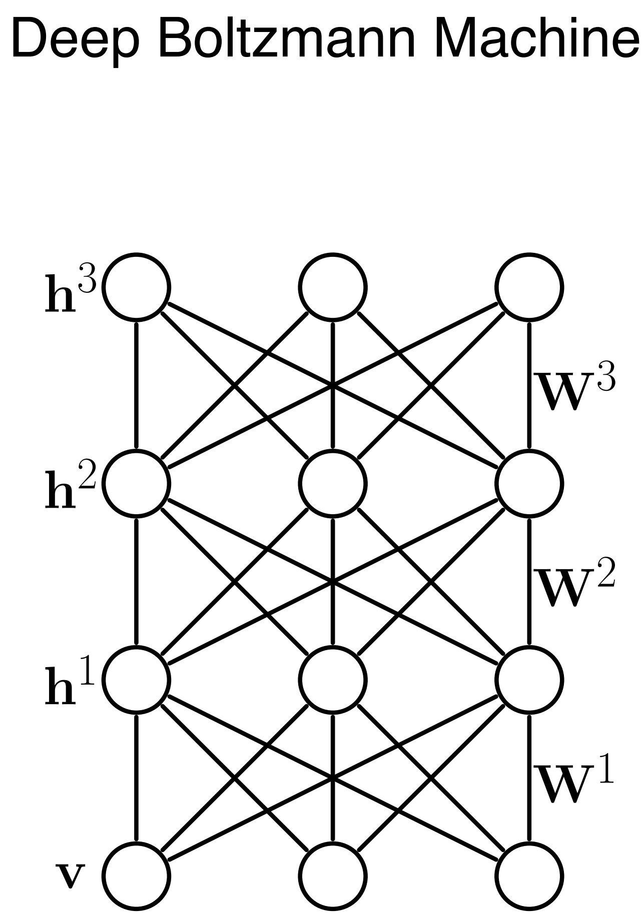
- Hinton, G. E., Dayan, P., Frey, B. J. and Neal, R., Science 1995



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

Helmholtz Machines vs. DBMs



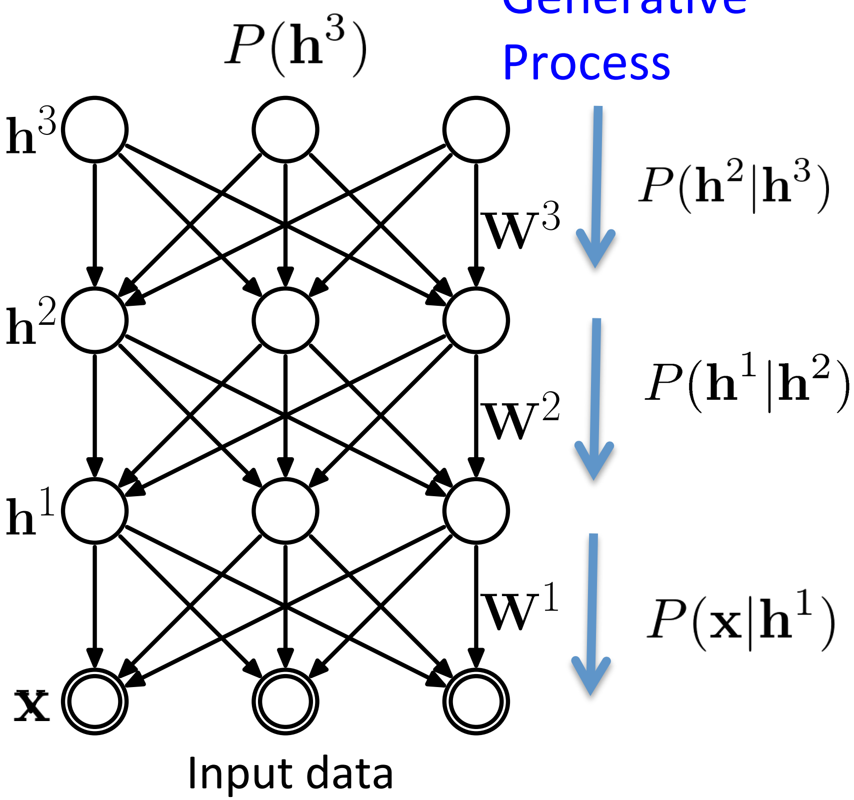


Variational Autoencoders (VAEs)

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

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

Each term may denote a complicated nonlinear relationship



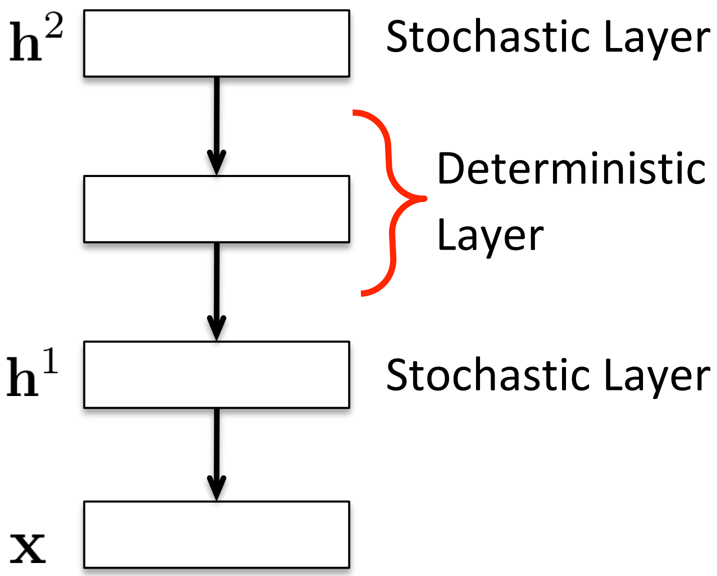
- $\boldsymbol{\theta}$ denotes parameters of VAE.
- L is the number of **stochastic** layers.
- Sampling and probability evaluation is tractable for each $p(\mathbf{h}^\ell|\mathbf{h}^{\ell+1})$.

VAE: Example

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

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

This term denotes a one-layer neural net.



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

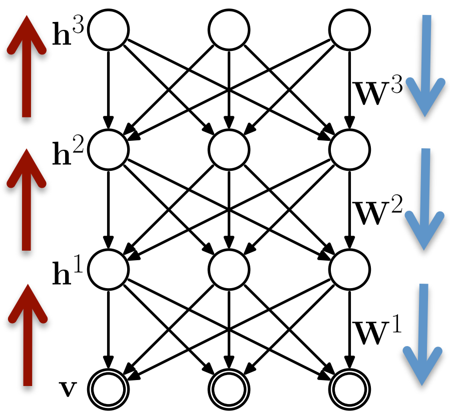
Variational Bound

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

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

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

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



Input data

- Hard to optimize the variational bound with respect to the recognition network (high-variance).

- Key idea of Kingma and Welling is to use reparameterization trick.

Reparameterization Trick

- Assume that the recognition distribution is Gaussian:

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

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

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

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

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

Reparameterization Trick

- Assume that the recognition distribution is Gaussian:

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

- Or

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

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

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

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

Deterministic Encoder

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

Computing the Gradients

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

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

Autoencoder

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

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

Gradients can be computed by backprop

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

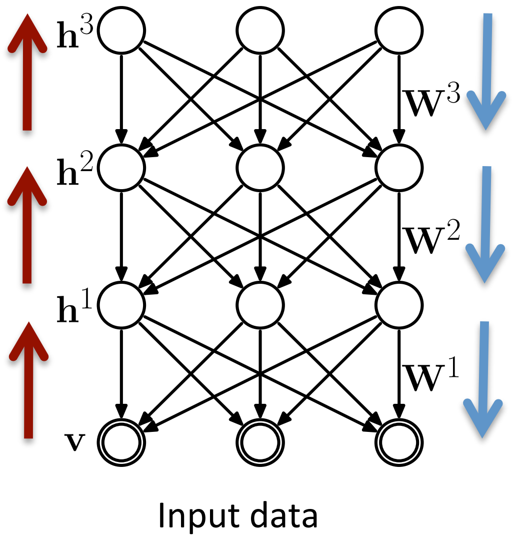
Importance Weighted Autoencoders

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

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

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

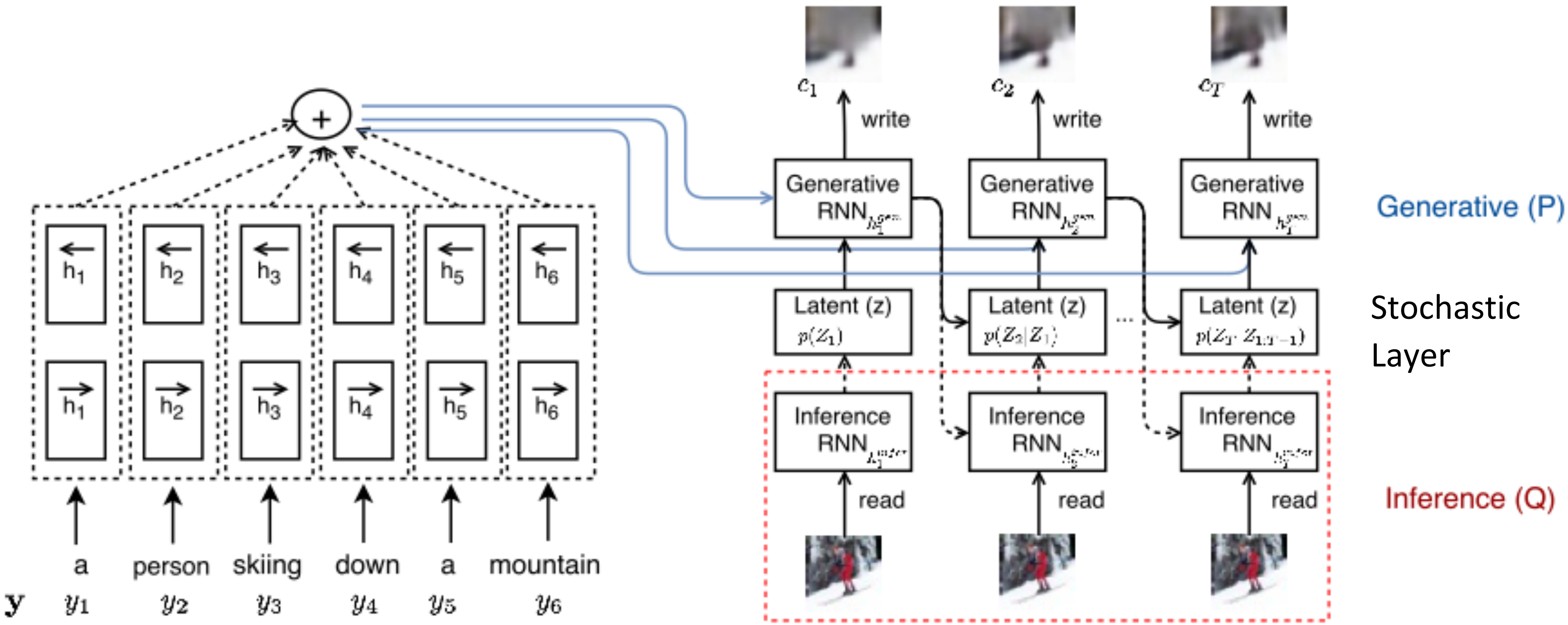
unnormalized importance weights



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

Burda, Grosse, Salakhutdinov, 2015

Generating Images from Captions



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

Gregor et. al. 2015

(Mansimov, Parisotto, Ba, Salakhutdinov, 2015)

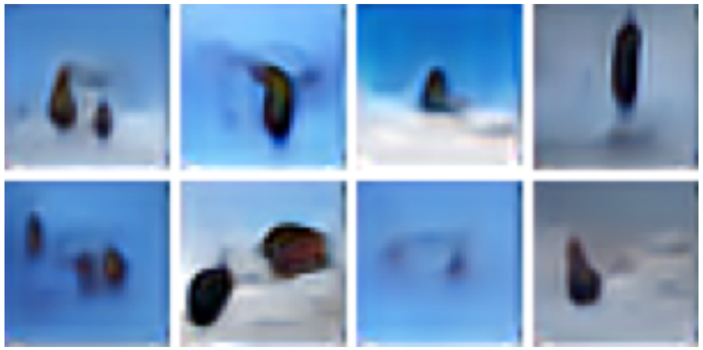
Motivating Example

- Can we generate images from natural language descriptions?

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

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

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



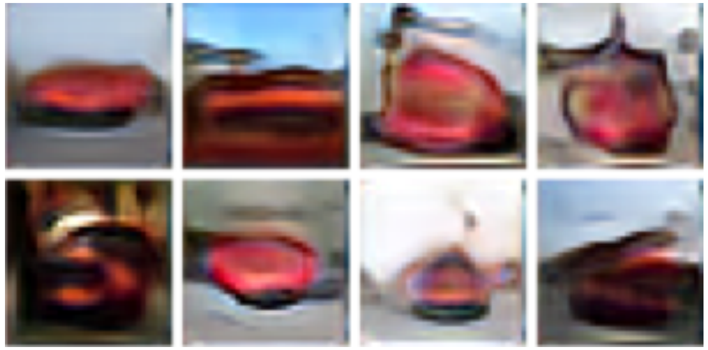
A **large commercial airplane** is flying in blue skies

(Mansimov, Parisotto, Ba, Salakhutdinov, 2015)

Flipping Colors

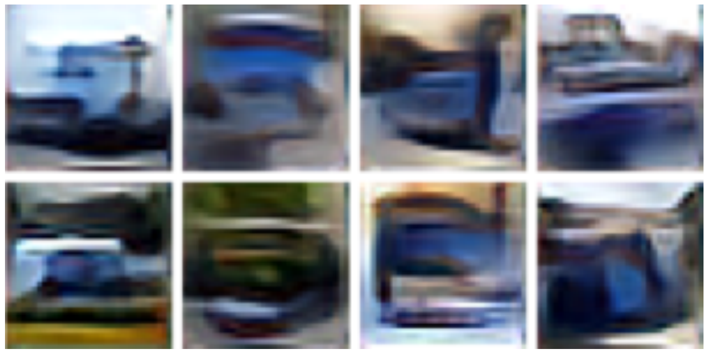
A **yellow school bus** parked in the parking lot

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



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

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



(Mansimov, Parisotto, Ba, Salakhutdinov, 2015)

Qualitative Comparison

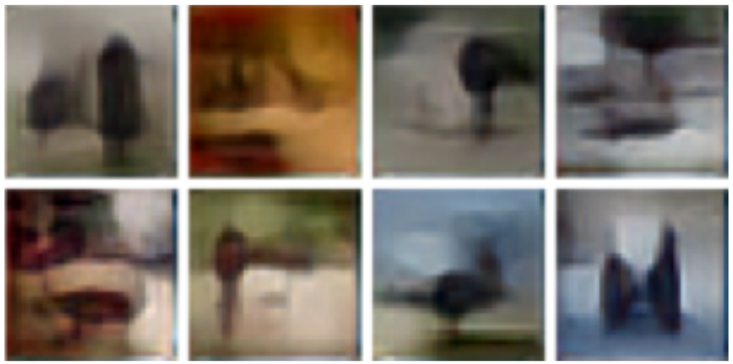
A group of people walk on a beach with surf boards

Our Model

LAPGAN (Denton et. al. 2015)

Conv-Deconv VAE

Fully Connected VAE



Novel Scene Compositions

A toilet seat sits open in the bathroom

A toilet seat sits open in the grass field

Ask Google?

Talk Roadmap

- Basic Building Blocks:
  - ➢ Sparse Coding
  - ➢ Autoencoders

- Deep Generative Models
  - ➢ Restricted Boltzmann Machines
  - ➢ Deep Boltzmann Machines
  - ➢ Helmholtz Machines / Variational Autoencoders

- Generative Adversarial Networks

Generative Adversarial Networks

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

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

- By playing a game!

Generative Adversarial Networks

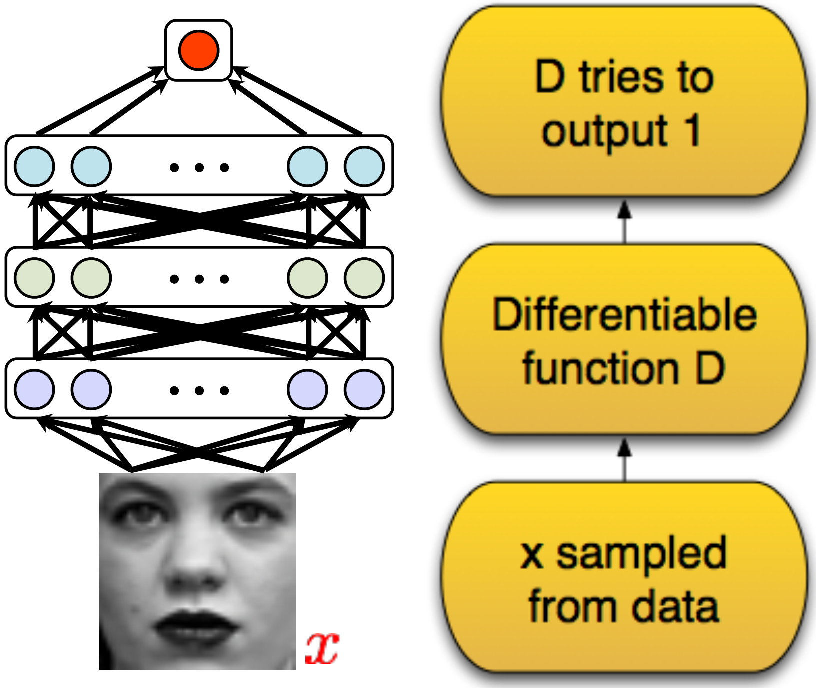
- Set up a game between two players:
  - Discriminator D
  - Generator G

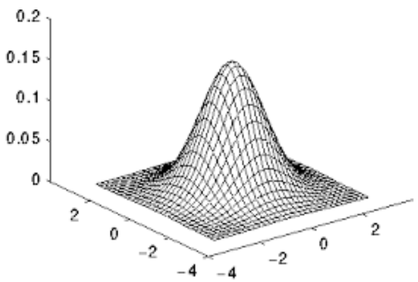
- Discriminator D tries to discriminate between:
  - A sample from the data distribution.
  - And a sample from the generator G.

- The Generator G attempts to “fool” D by generating samples that are hard for D to distinguish from the real data.

Generative Adversarial Networks” Goodfellow et al., NIPS 2014

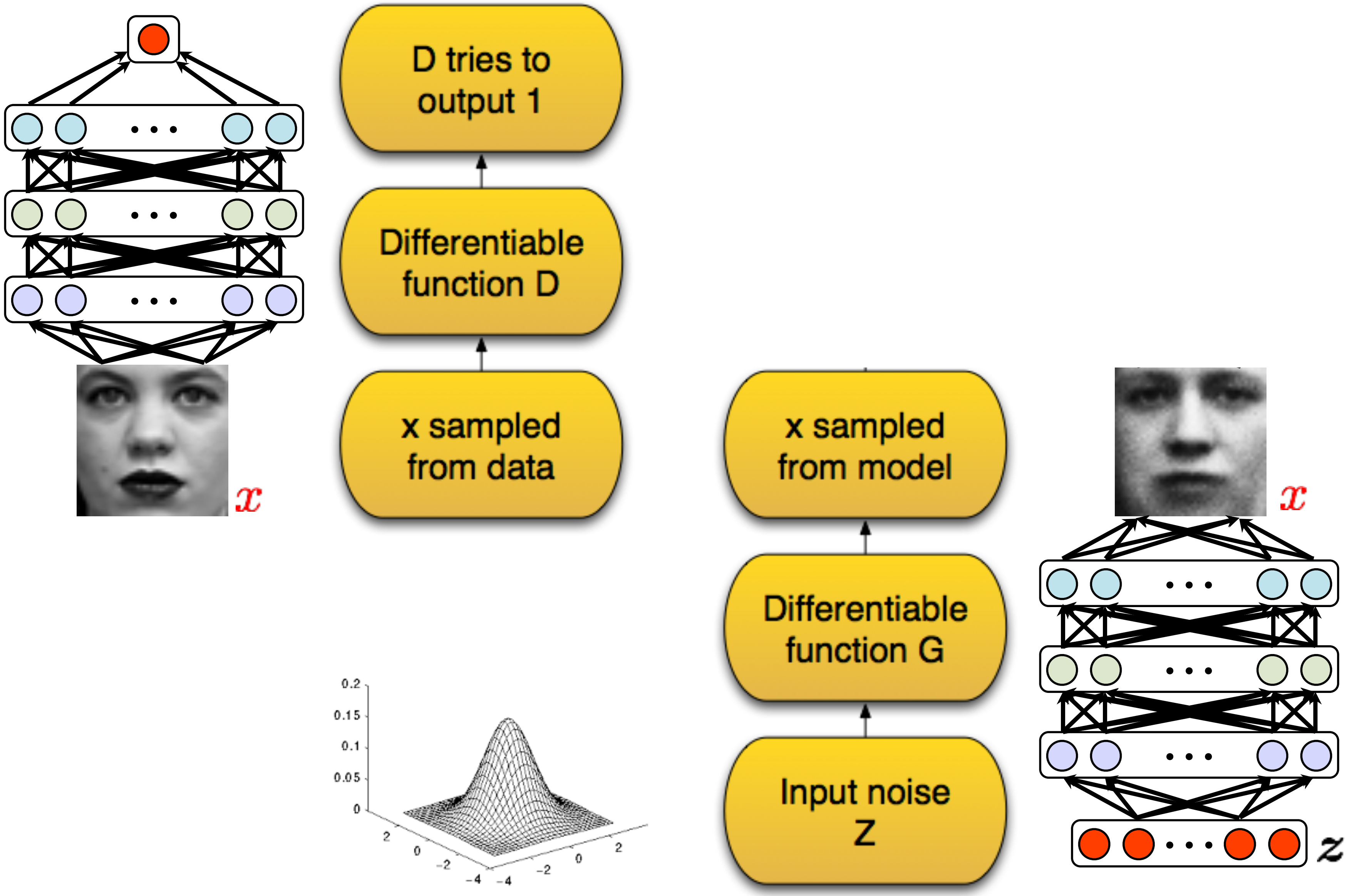
Generative Adversarial Networks





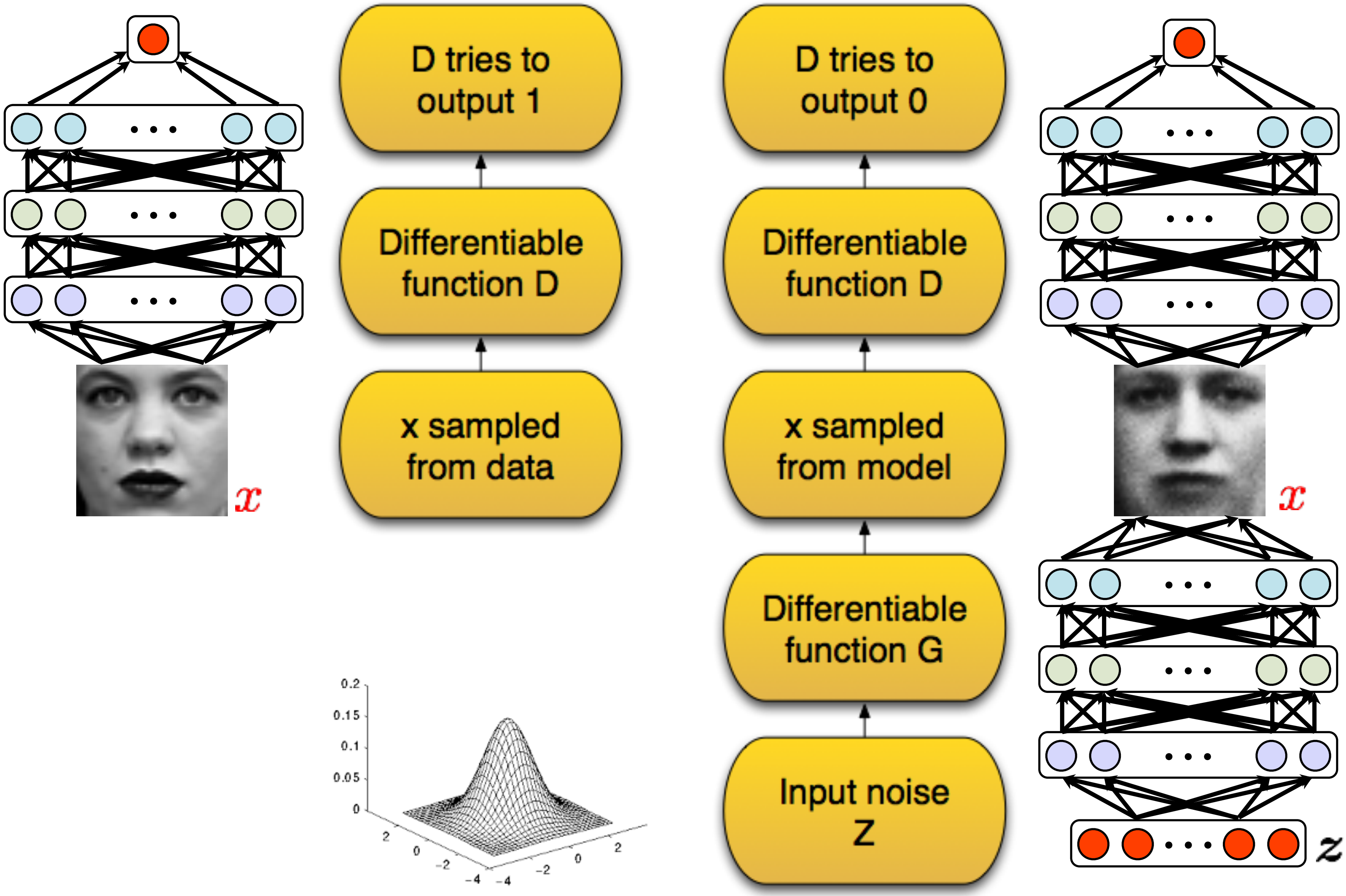
Slide Credit: Ian Goodfellow

Generative Adversarial Networks



Slide Credit: Ian Goodfellow

Generative Adversarial Networks



Slide Credit: Ian Goodfellow

Generative Adversarial Networks

- Minimax value function

Generator: generate samples that D would classify as real

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

Discriminator: Pushes up

Discriminator: Classify data as being real

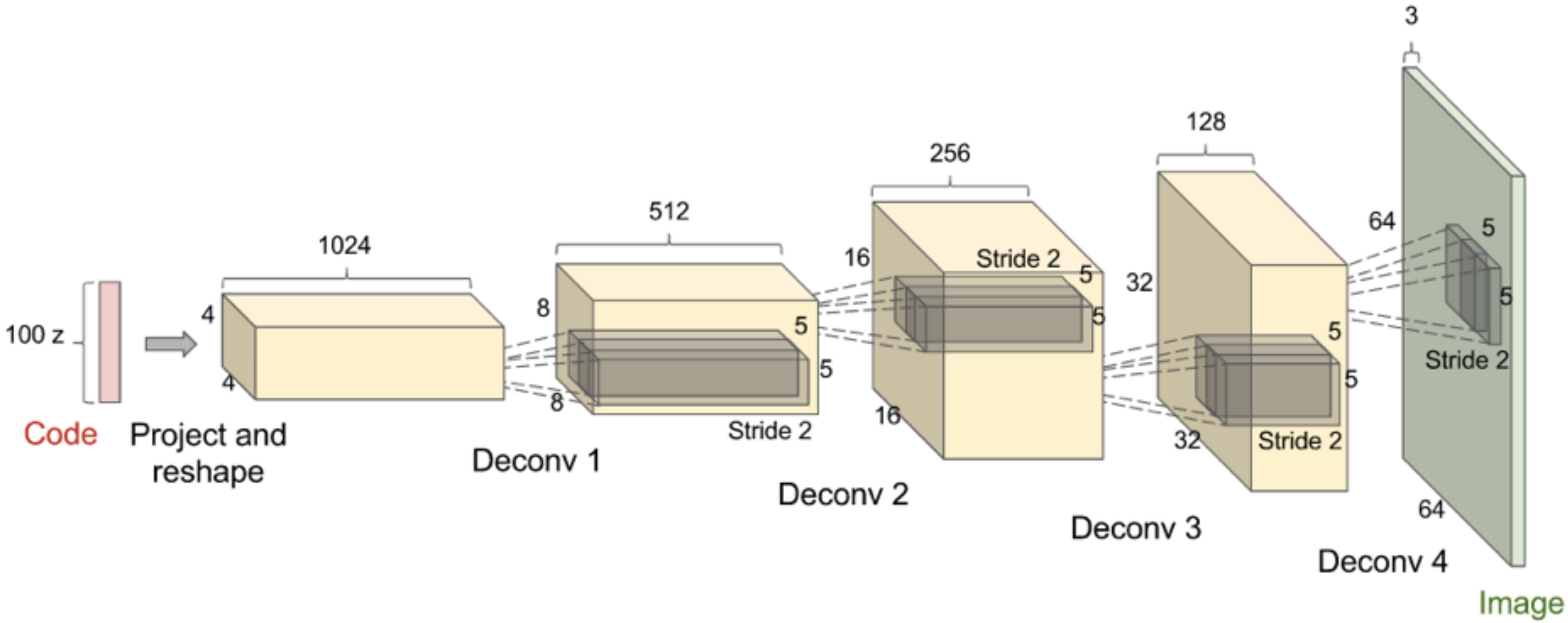
Discriminator: Classify generator samples as being fake

Generator: Pushes down

- Optimal strategy for Discriminator is:

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

DCGAN Architecture

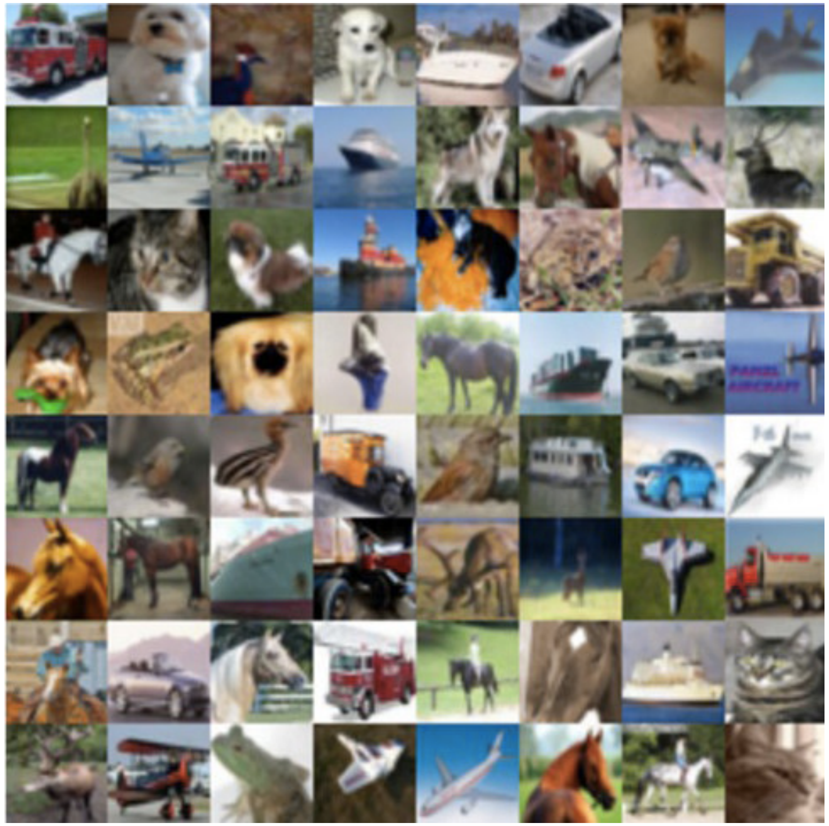


(Radford et al 2015)

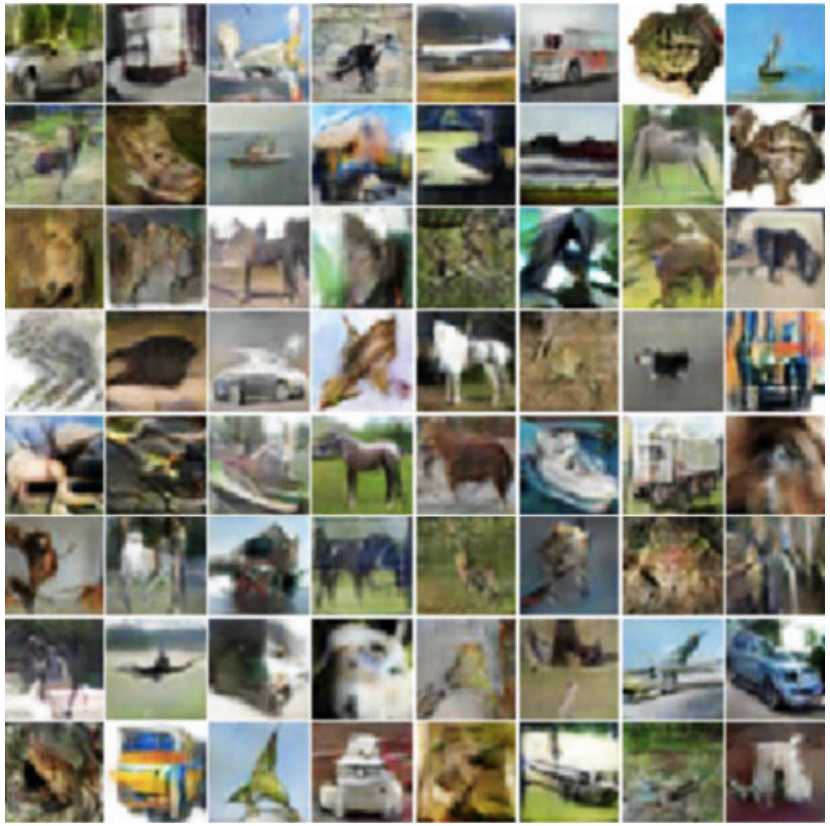
LSUN Bedrooms: Samples

(Radford et al 2015)

CIFAR



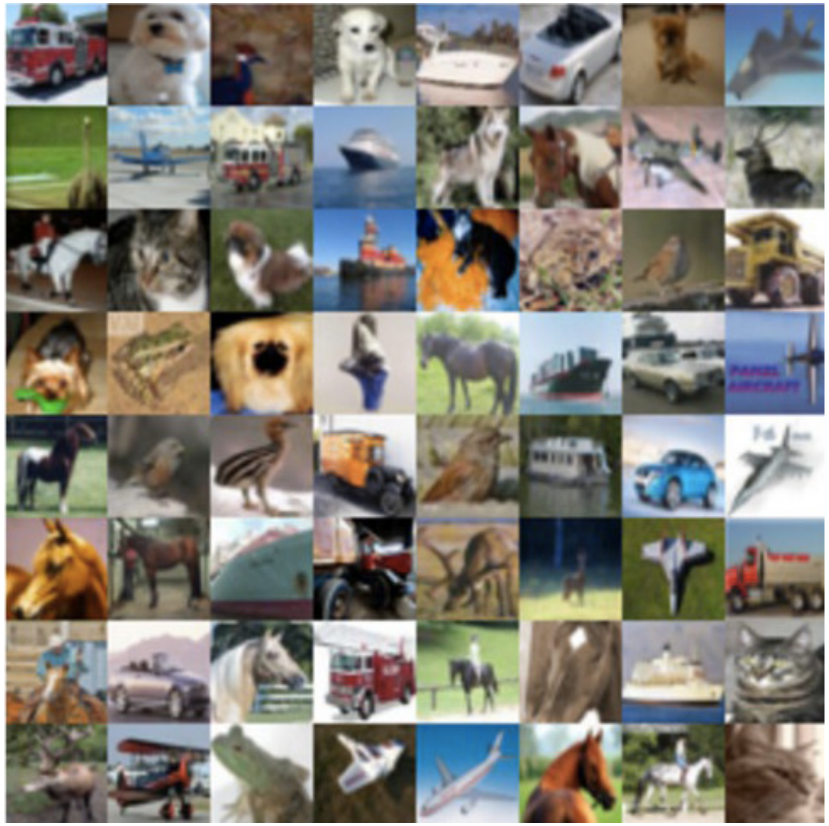
Training



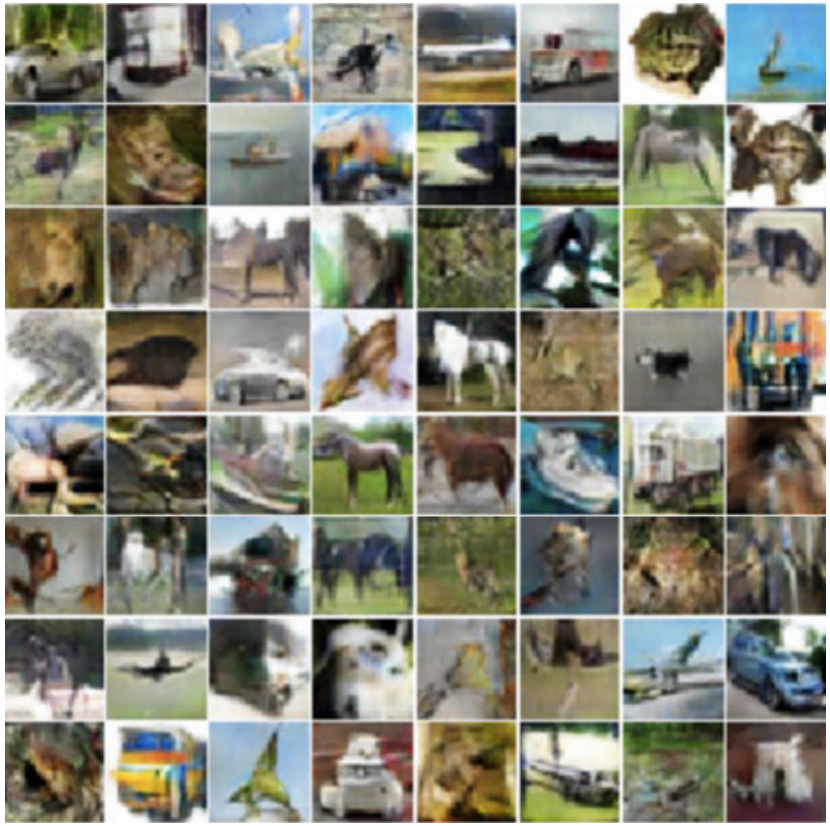
Samples

(Salimans et. al., 2016)

IMAGENET



Training



Samples

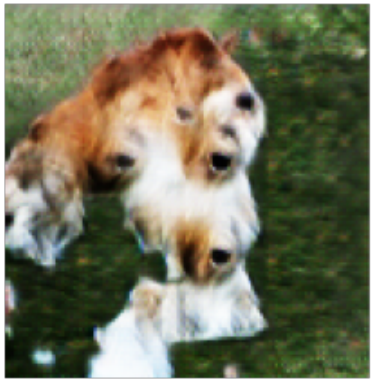
(Salimans et. al., 2016)

ImageNet: Cherry-Picked Results

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

Slide Credit: Ian Goodfellow

ImageNet: Cherry-Picked Results



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

Slide Credit: Ian Goodfellow