Learning Deep Generative Models

Russ Salakhutdinov

Machine Learning Department
Carnegie Mellon University
Canadian Institute for Advanced Research

Carnegie Mellon University

MACHINE LEARNING DEPARTMENT

Impact of Deep Learning

- Speech Recognition
- Computer Vision
- Recommender Systems
- Language Understanding
- Drug Discovery and Medical Image Analysis

Statistical Generative Models

+

Model family, loss function, optimization algorithm, etc.

Data

Learning

Prior Knowledge

Image x

A probability distribution p(x)

probability p(x)

Sampling from p(x) **generates** new images:

Statistical Generative Models

Training Data(CelebA)

Model Samples (Karras et.al., 2018)

4 years of progression on Faces

Brundage et al., 2017

Conditional Generation

- Conditional generative model P(zebra images| horse images)

- Style Transfer

Input Image

Monet

Van Gogh

Zhou el al., Cycle GAN 2017

Conditional Generation

- Conditional generative model P(zebra images| horse images)

- Failure Case

Zhou el al., Cycle GAN 2017

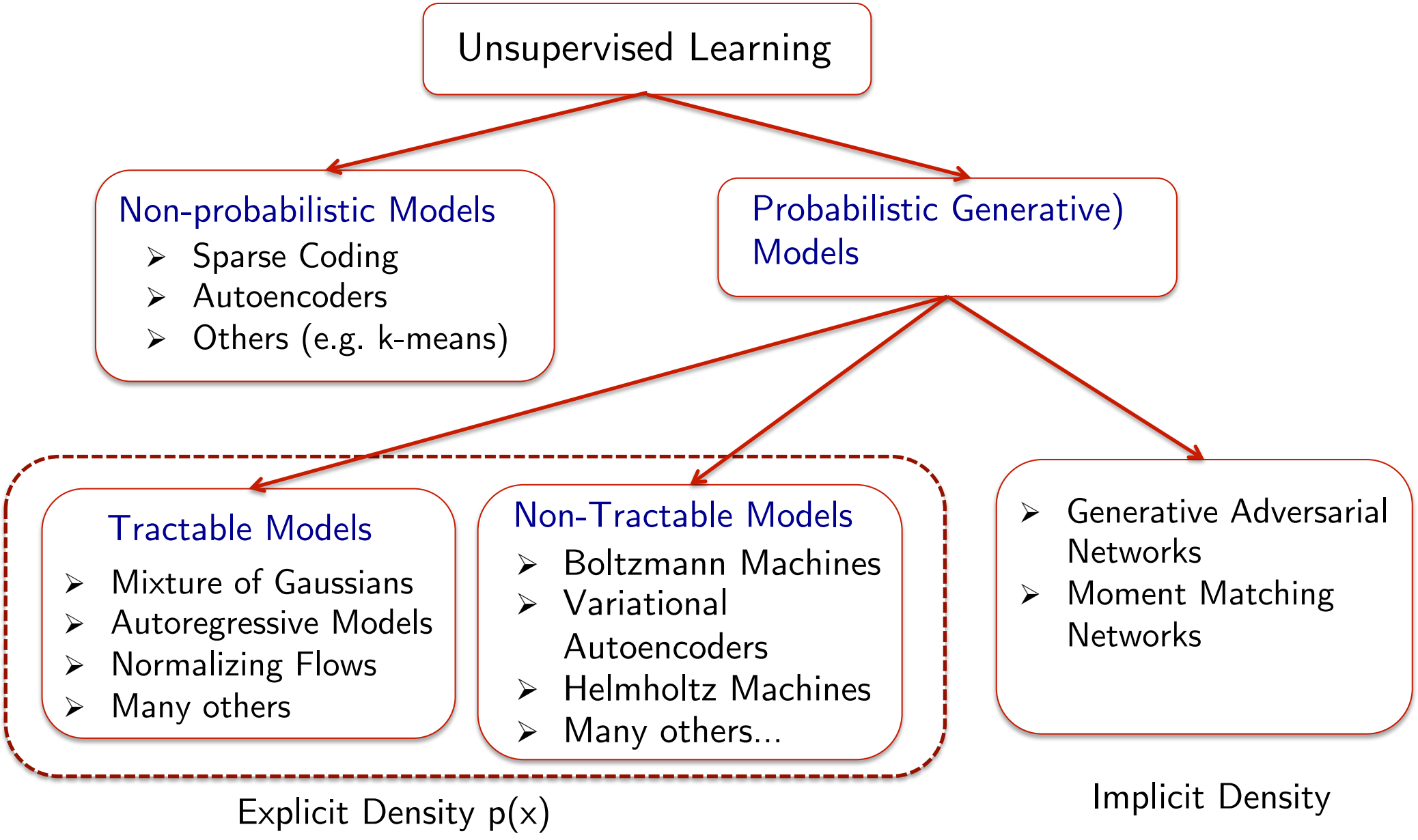
Face2Face

Source actor

Target actor

Real-time reenactment

Face2Face, Thies et al, CVPR 2016


Unsupervised Learning
Non-probabilistic Models
Sparse Coding
Autoencoders
Others (e.g. k-means)
Probabilistic Generative) Models
Tractable Models
Mixture of Gaussians
Autoregressive Models
Normalizing Flows
Many others
Non-Tractable Models
Boltzmann Machines
Variational Autoencoders
Helmholtz Machines
Many others...
Generative Adversarial Networks
Moment Matching Networks
Explicit Density p(x)
Implicit Density

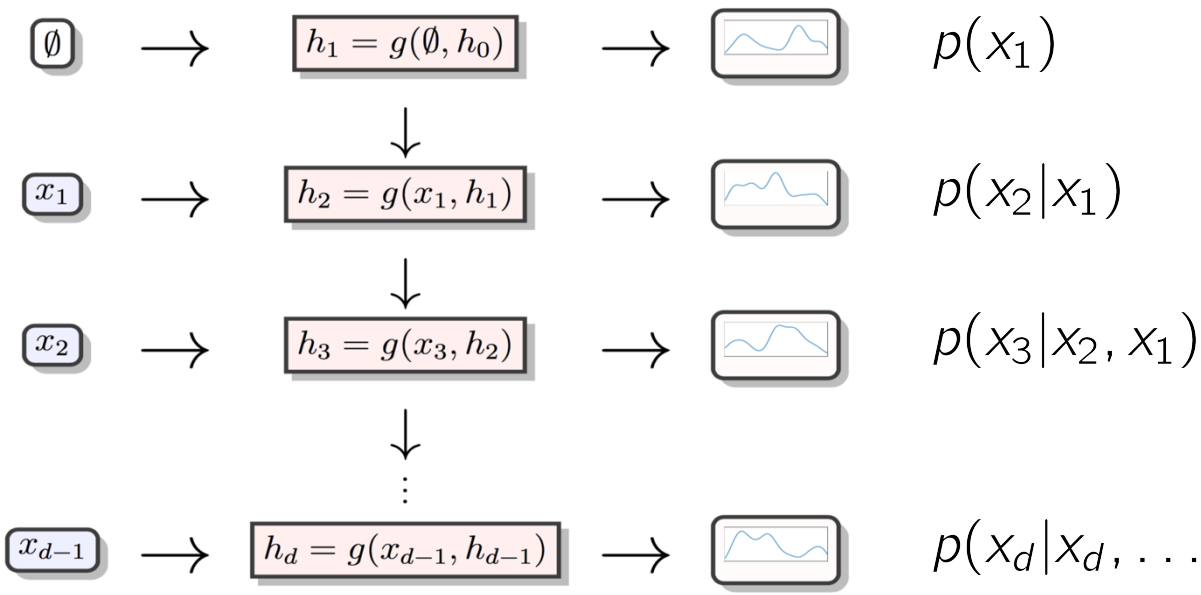
Talk Roadmap

- Fully Observed Models
- Undirected Deep Generative Models
  - Restricted Boltzmann Machines (RBMs),
  - Deep Boltzmann Machines (DBMs)
- Directed Deep Generative Models
  - Variational Autoencoders (VAEs)
  - Normalizing Flows
- Generative Adversarial Networks (GANs)

Fully Observed Models

- Density Estimation by Autoregression

$$p(x_1, \ldots, x_d) = \prod_{i=1}^{d} p(x_i | x_{i-1}, \ldots, x_1) \approx \prod_{i=1}^{d} p(x_i | g(x_{i-1}, \ldots, x_1))$$



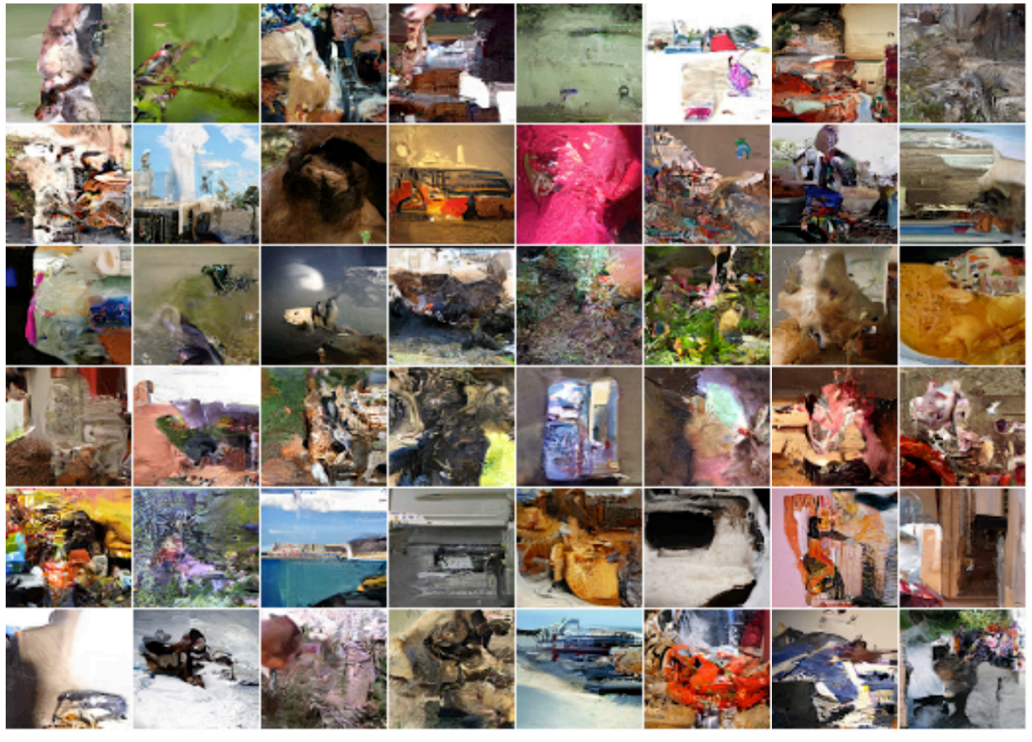
Each conditional can be a deep neural network

- Ordering of variables is crucial

NADE (Uria 2013), MADE (Germain 2017), MAF (Papamakarios 2017), PixelCNN (van den Oord, et al, 2016)

Fully Observed Models

- Density Estimation by Autoregression

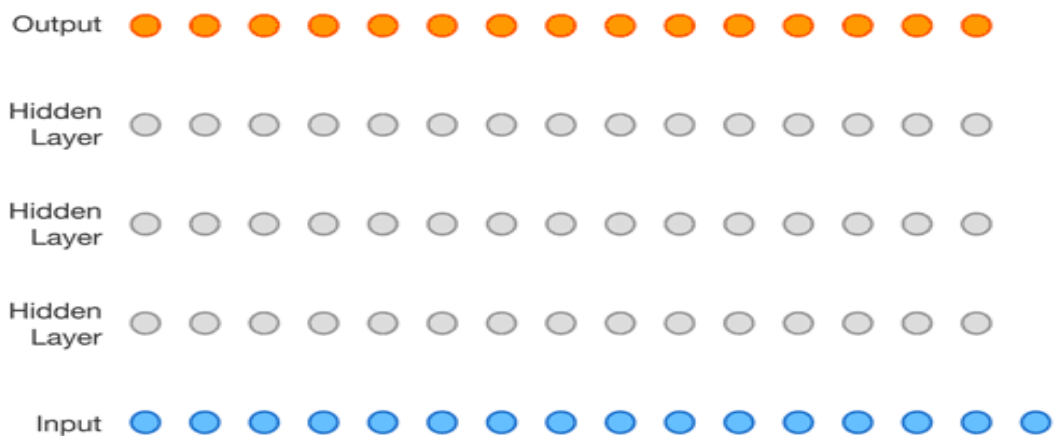


PixelCNN (van den Oord, et al, 2016)

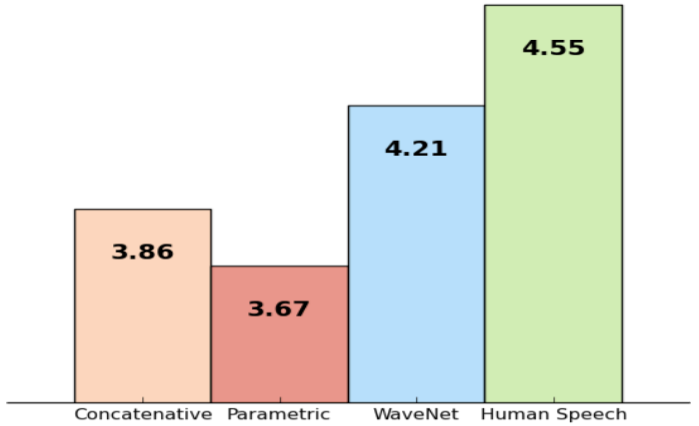
NADE (Uria 2013), MADE (Germain 2017), MAF (Papamakarios 2017), PixelCNN (van den Oord, et al, 2016)

WaveNet

- Generative Model of Speech Signals



Quality: Mean Opinion Scores

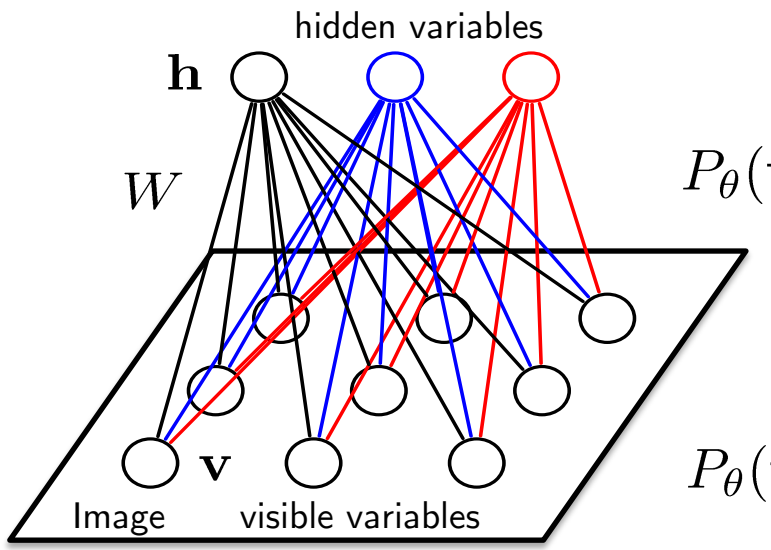


van den Oord et al, 2016

Talk Roadmap

- Fully Observed Models
- Undirected Deep Generative Models
  - Restricted Boltzmann Machines (RBMs),
  - Deep Boltzmann Machines (DBMs)
- Directed Deep Generative Models
  - Variational Autoencoders (VAEs)
  - Normalizing Flows
- Generative Adversarial Networks (GANs)

Restricted Boltzmann Machines



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

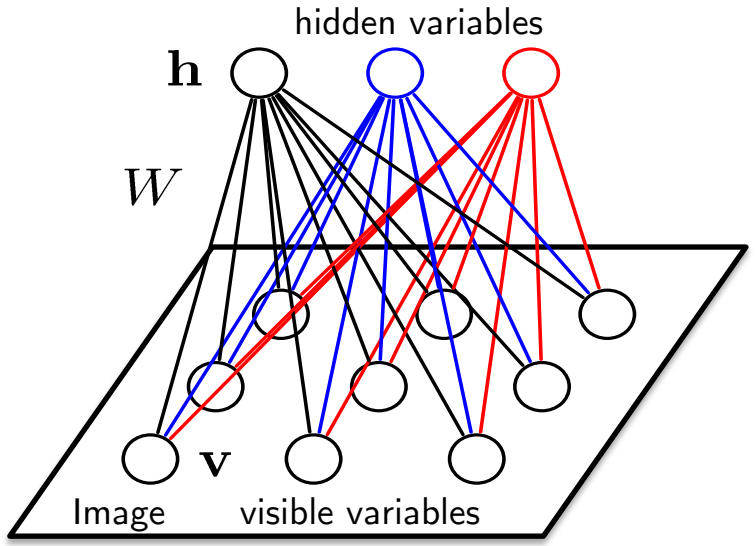
$$\theta = \{W, a, b\}$$

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

- 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.

Maximum Likelihood Learning



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

- Maximize log-likelihood objective:

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

- Derivative of the log-likelihood:

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

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

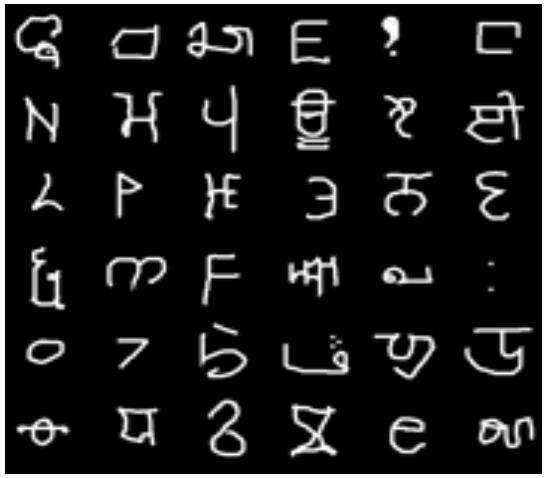
Difficult to compute: exponentially many configurations

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

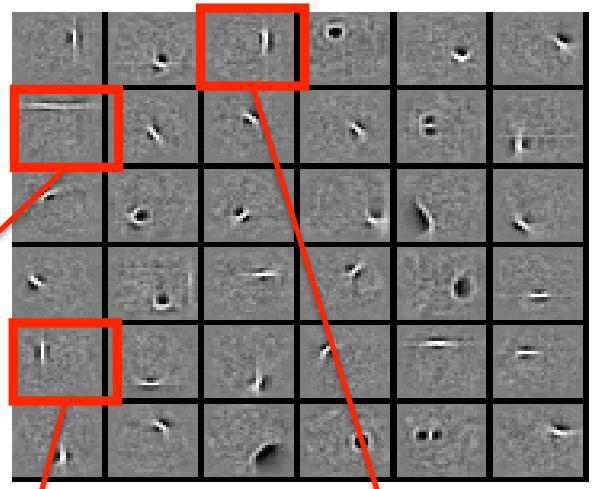
$$P_{data}(\mathbf{v}) = \frac{1}{N} \sum_{n} \delta(\mathbf{v} - \mathbf{v}^{(n)})$$

Learning Features

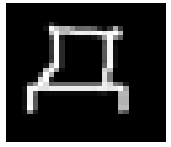
Observed Data
Subset of 25,000 characters



Learned W: "edges"
Subset of 1000 features



New Image:



$$= \sigma\Big(0.99 \times \text{[image]} + 0.97 \times \text{[image]} + 0.82 \times \text{[image]} \cdots\Big)$$

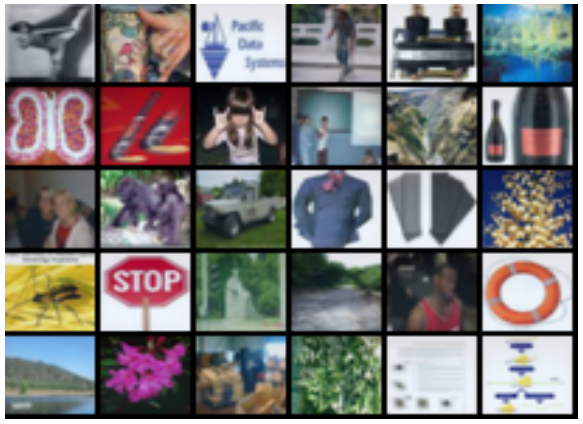
$p(h_7 = 1|v)$ $p(h_{29} = 1|v)$

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

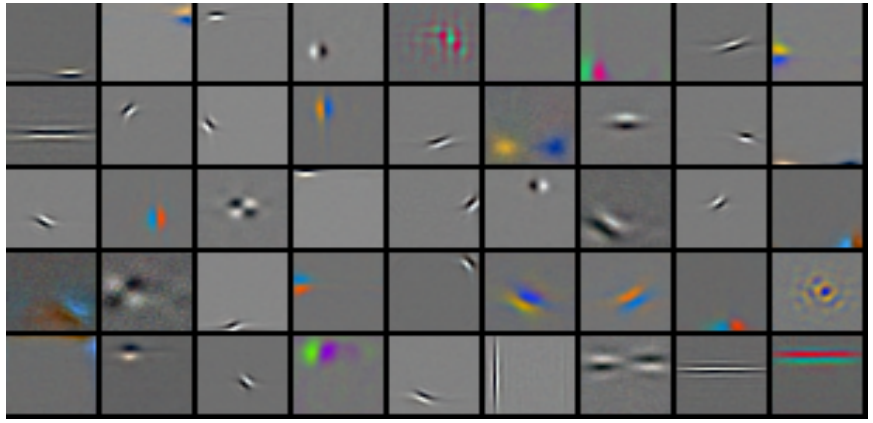
Logistic Function: Suitable for modeling binary images

RBMs for Real-valued & Count Data

4 million **unlabelled** images



Learned features (out of 10,000)



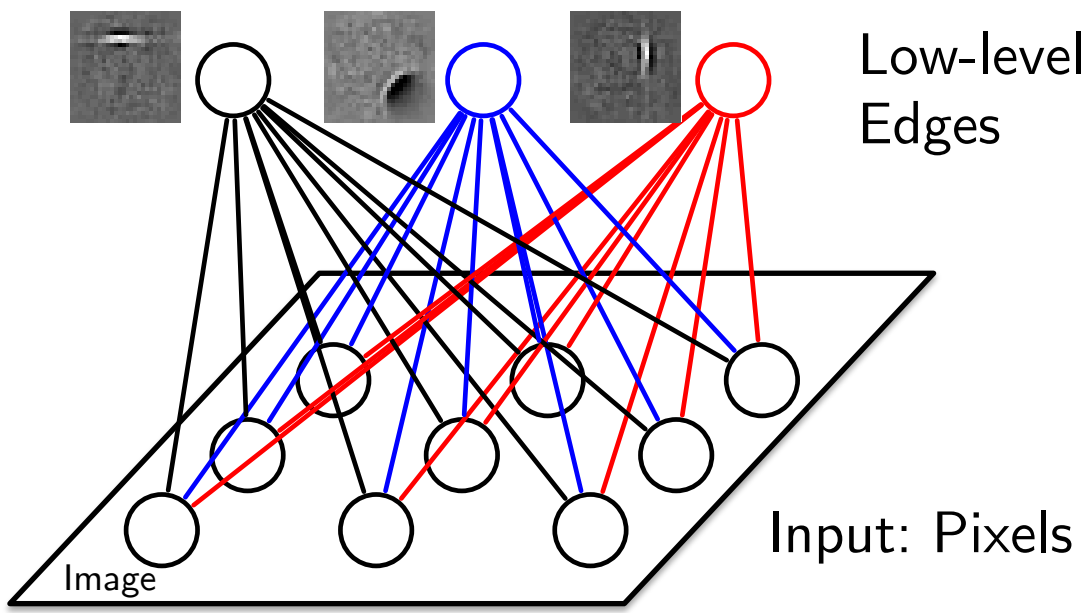
Reuters dataset:
804,414 **unlabeled**
newswire stories

Bag-of-Words

Learned features: ``topics''

russian	clinton	computer	trade	stock
russia	house	system	country	wall
moscow	president	product	import	street
yeltsin	bill	software	world	point
soviet	congress	develop	economy	dow

Deep Boltzmann Machines

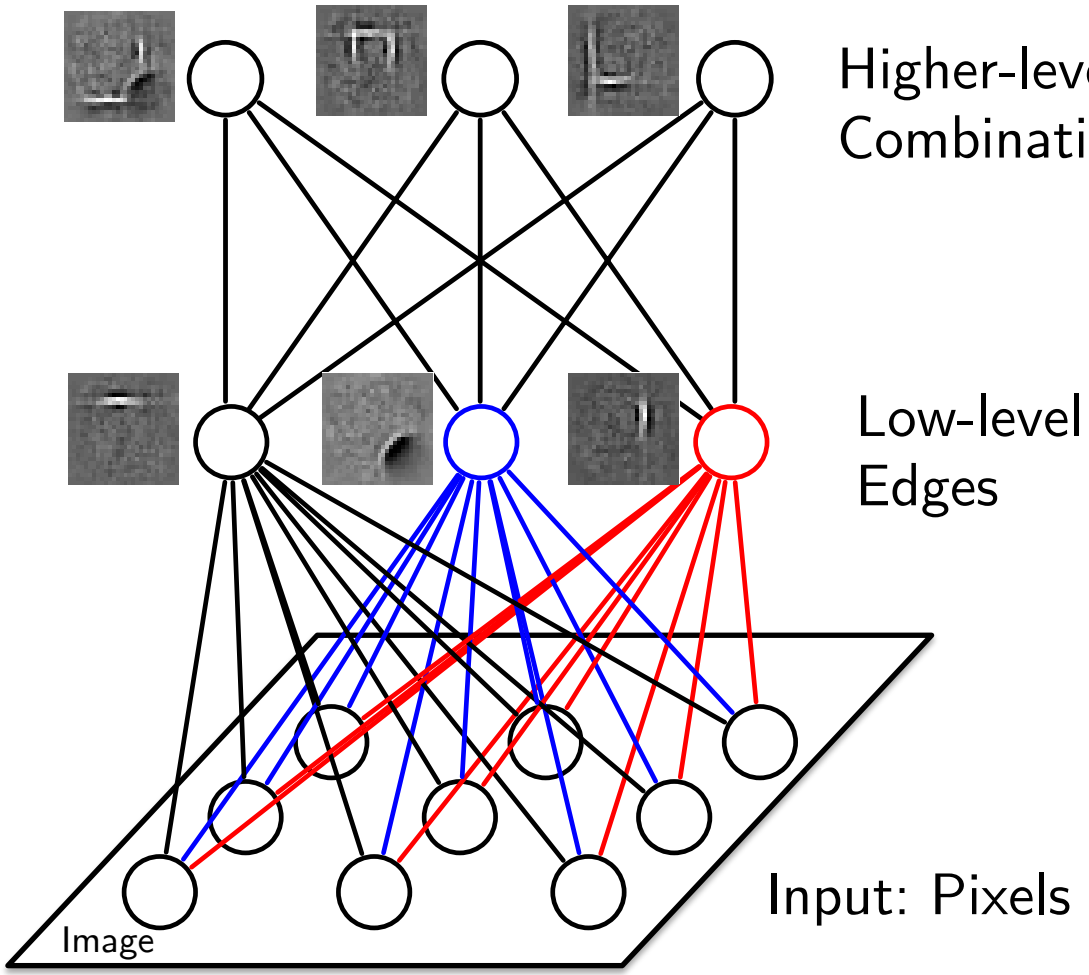


Built from **unlabeled** inputs.

(Salakhutdinov 2008, Salakhutdinov & Hinton 2009)

Deep Boltzmann Machines

Learn simpler representations,
then compose more complex ones

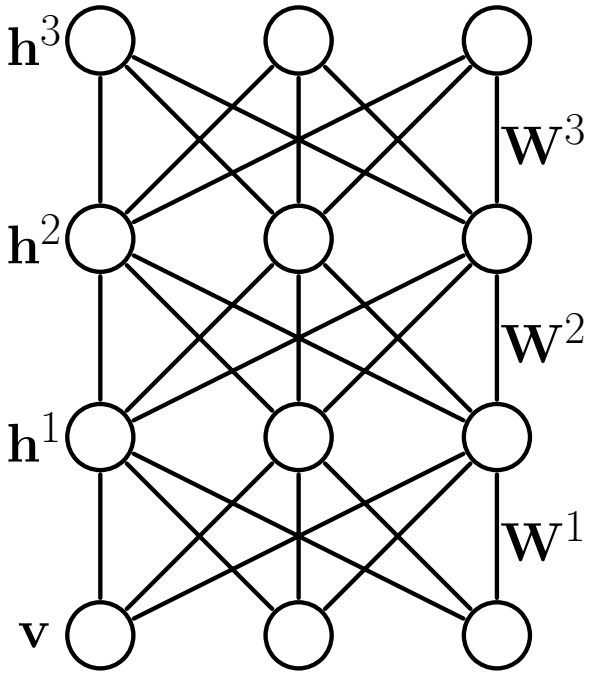


Built from **unlabeled** inputs.

(Salakhutdinov 2008, Salakhutdinov & Hinton 2009)

Model Formation

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



Same as RBMs

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

- Dependencies between hidden variables
- All connections are undirected

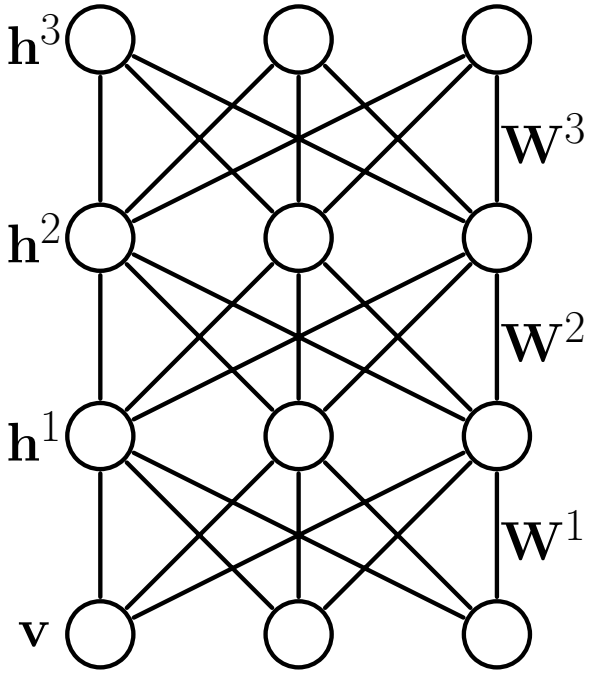
- Maximum Likelihood Learning:

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

  - Both expectations are intractable

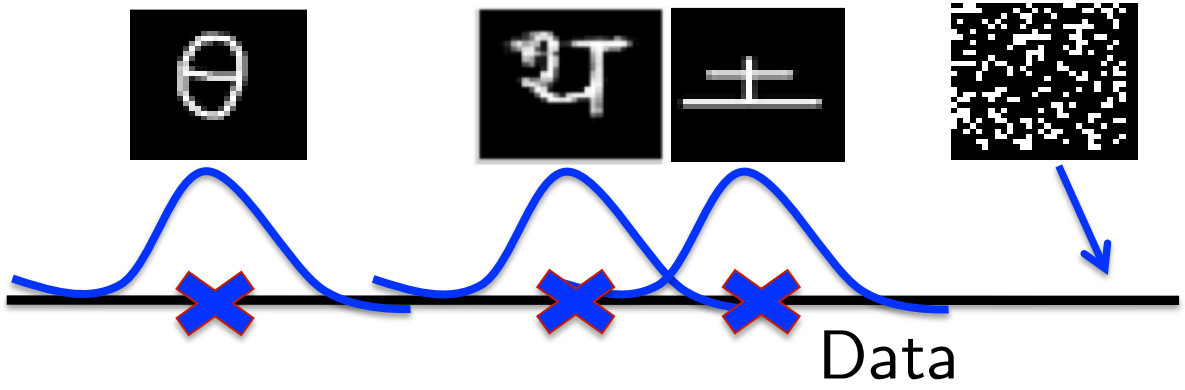
Approximate Learning

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



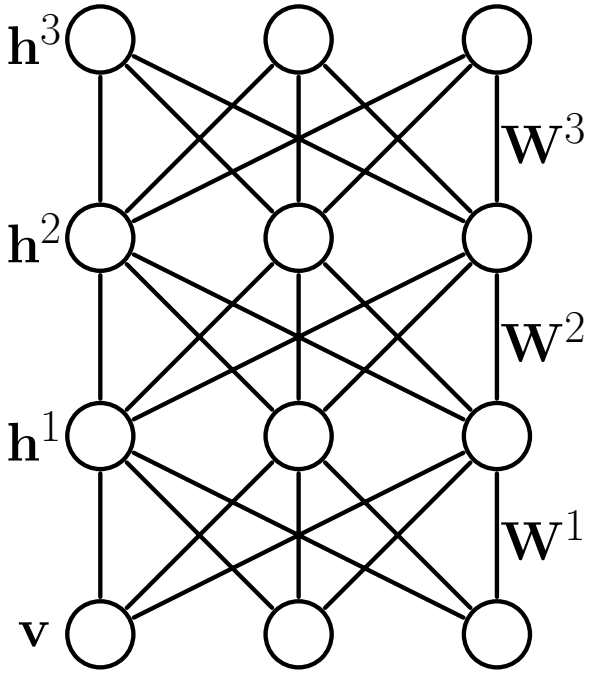
- Maximum Likelihood Learning:

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



Approximate Learning

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



- Maximum Likelihood Learning:

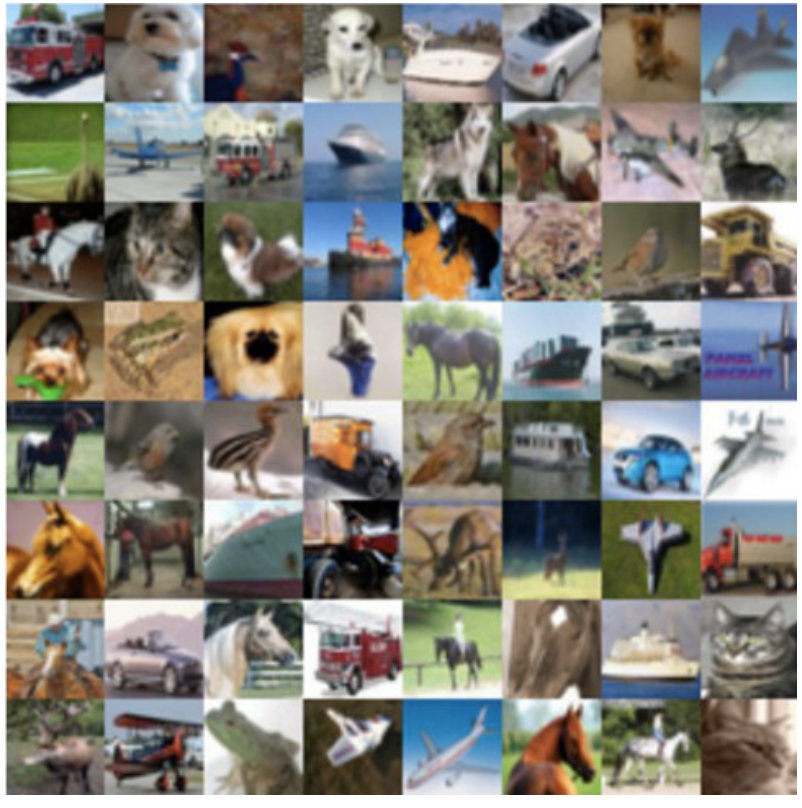
$$\frac{\partial \log P_\theta(\mathbf{v})}{\partial W^1} = \mathbb{E}_{P_{data}}[\mathbf{v}\mathbf{h}^{\mathbf{1}\top}] - \mathbb{E}_{P_\theta}[\mathbf{v}\mathbf{h}^{\mathbf{1}\top}]$$

Variational Inference

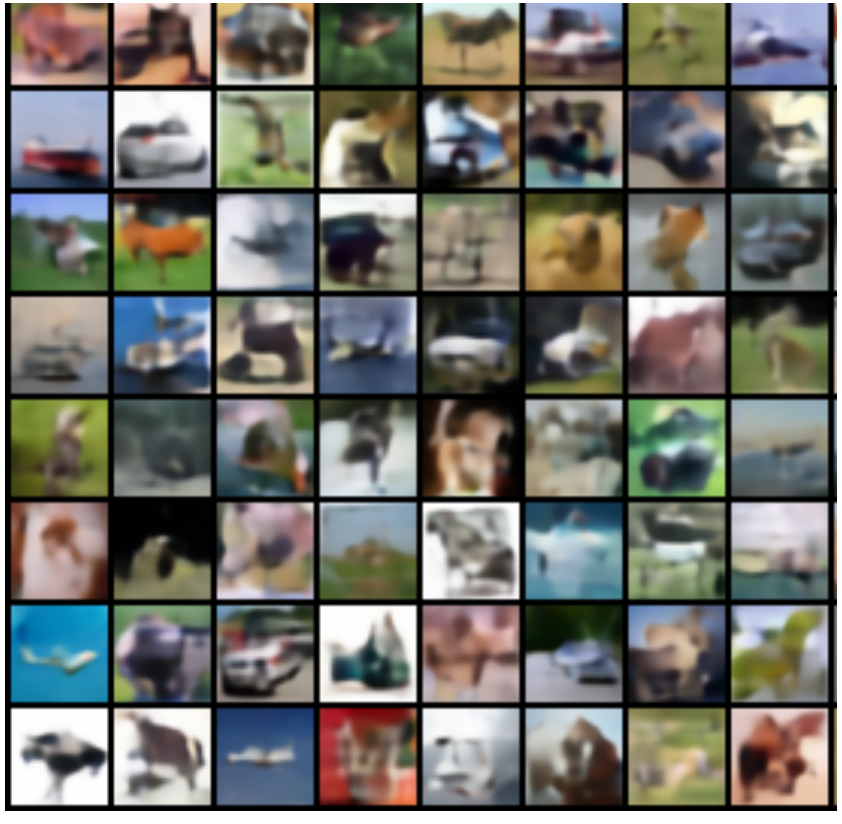
Stochastic Approximation (MCMC-based)

Good Generative Model?

- CIFAR Dataset



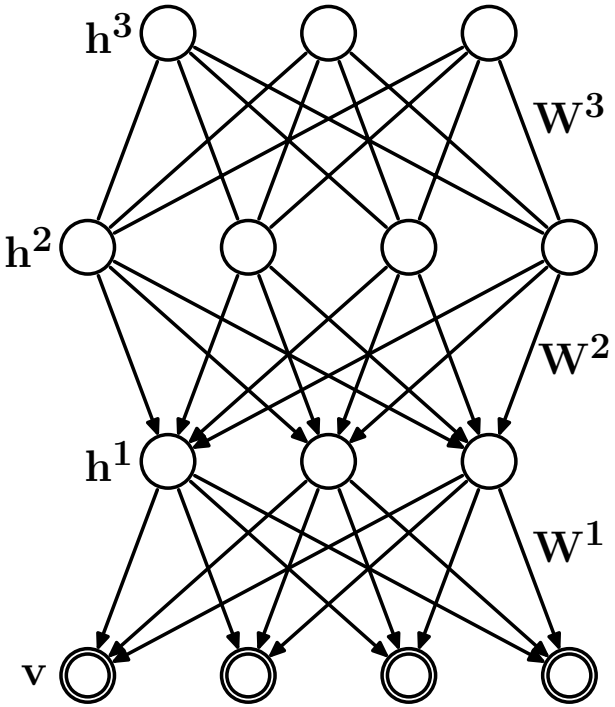
Training

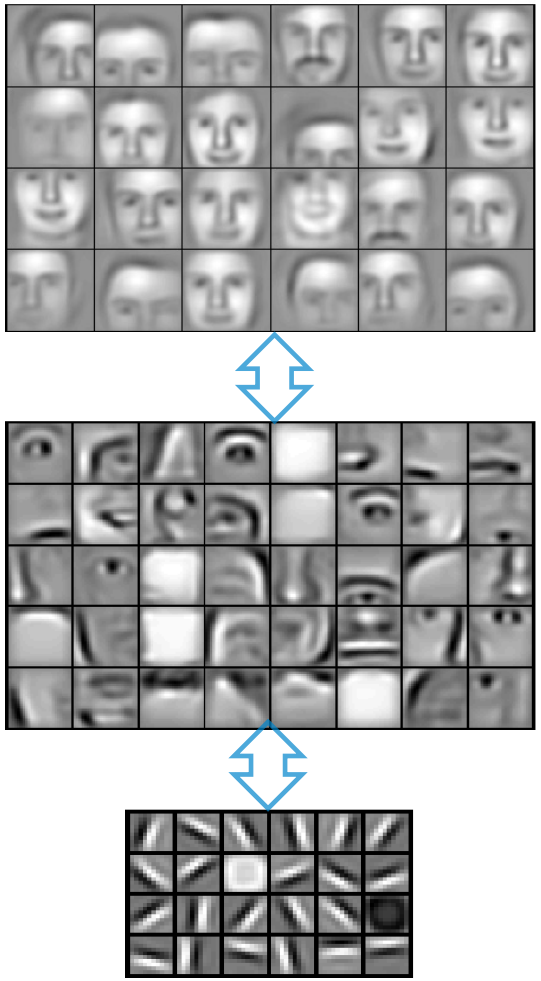


Samples

Learning Part-Based Representations

Deep Belief Network





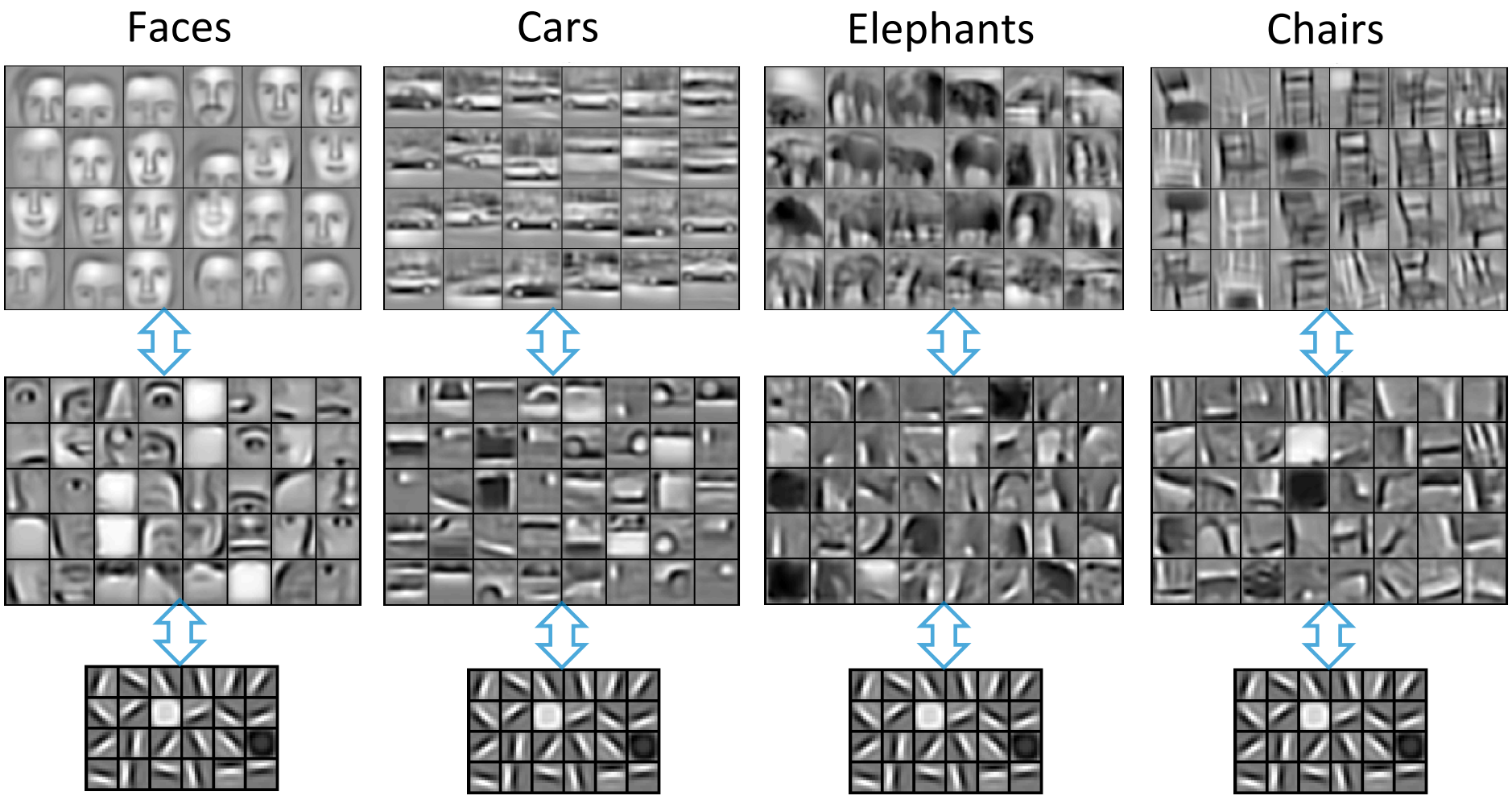
Groups of parts

Object Parts

Trained on face images.

Lee, Grosse, Ranganath, Ng, ICML 2009

Learning Part-Based Representations



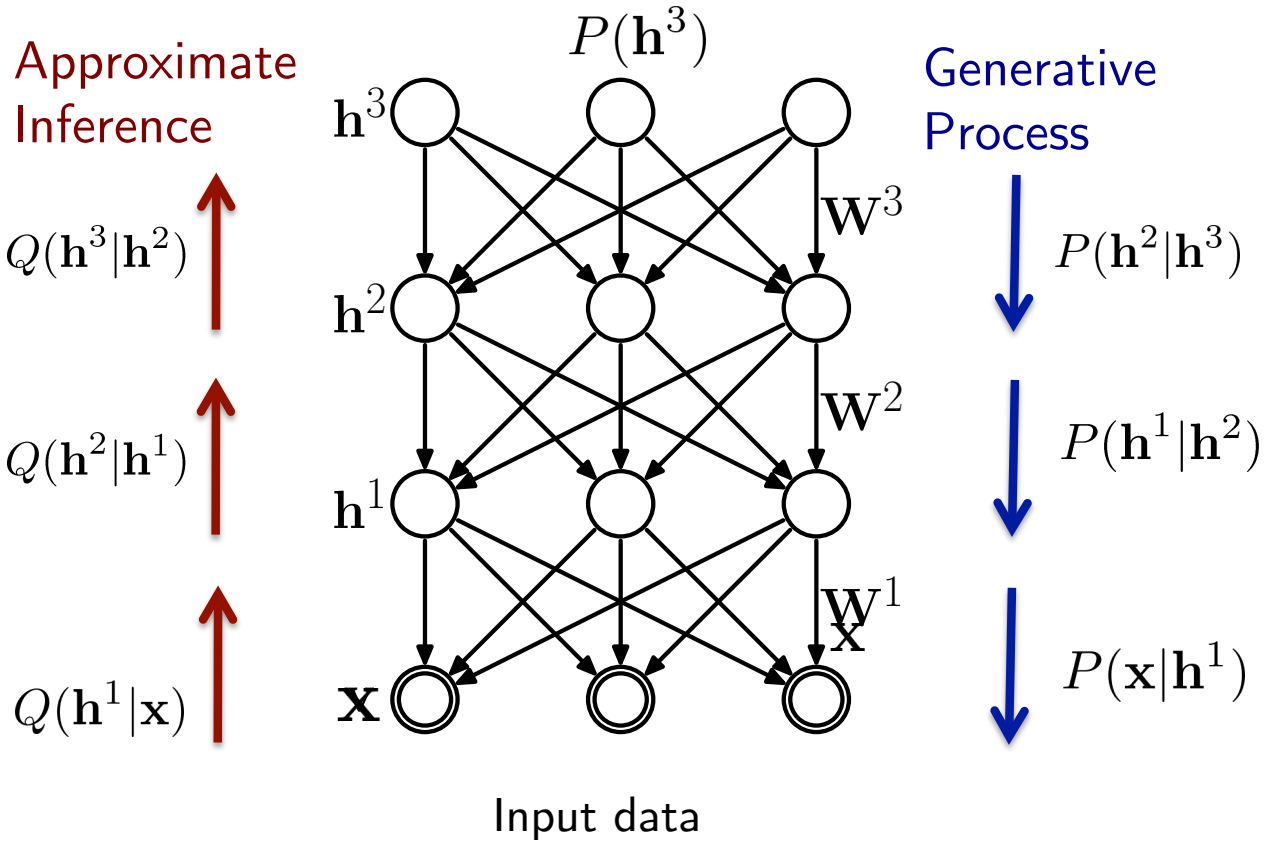
Lee, Grosse, Ranganath, Ng, ICML 2009

Talk Roadmap

- Fully Observed Models
- Undirected Deep Generative Models
  - Restricted Boltzmann Machines (RBMs),
  - Deep Boltzmann Machines (DBMs)
- Directed Deep Generative Models
  - Variational Autoencoders (VAEs)
  - Normalizing Flows
- Generative Adversarial Networks (GANs)

Helmholtz Machines

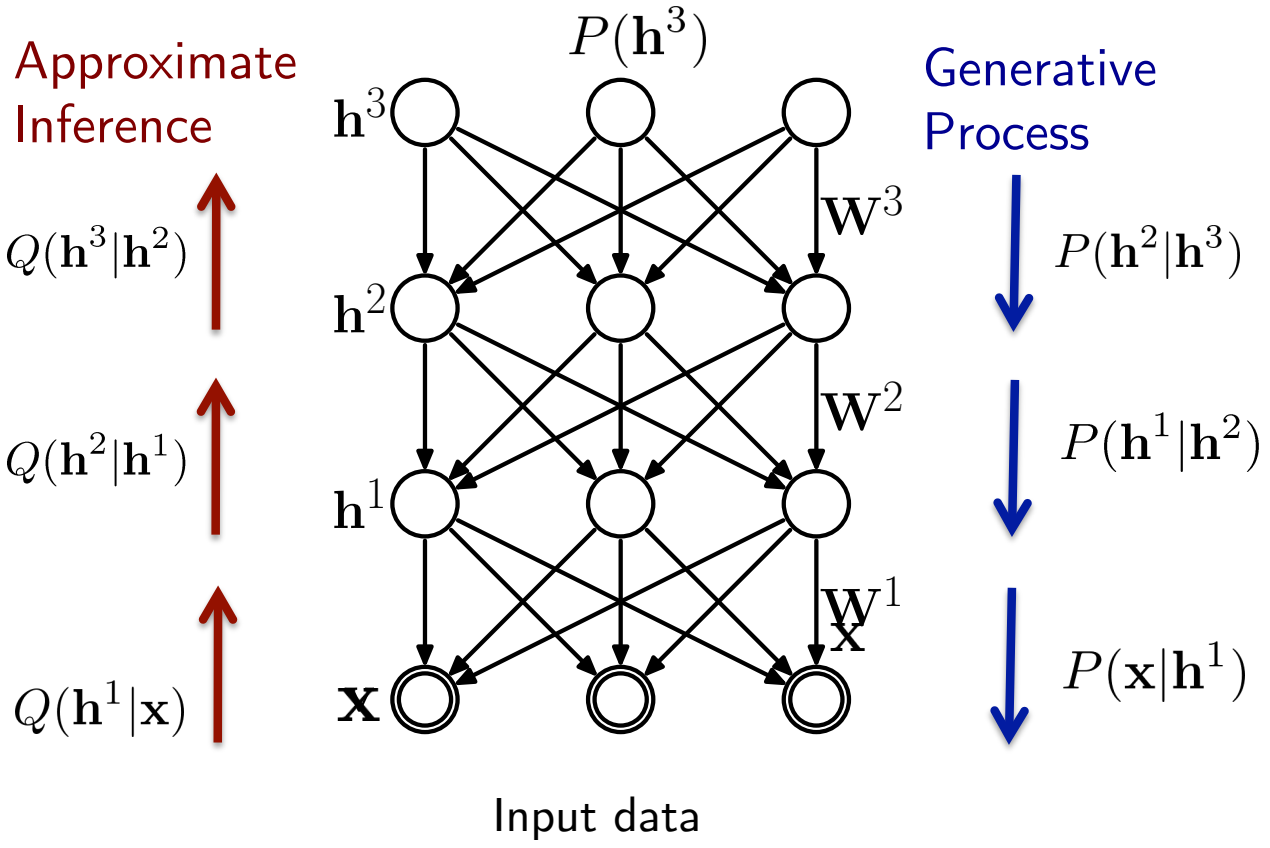
- Hinton, G. E., Dayan, P., Frey, B. J. and Neal, R., Science 1995



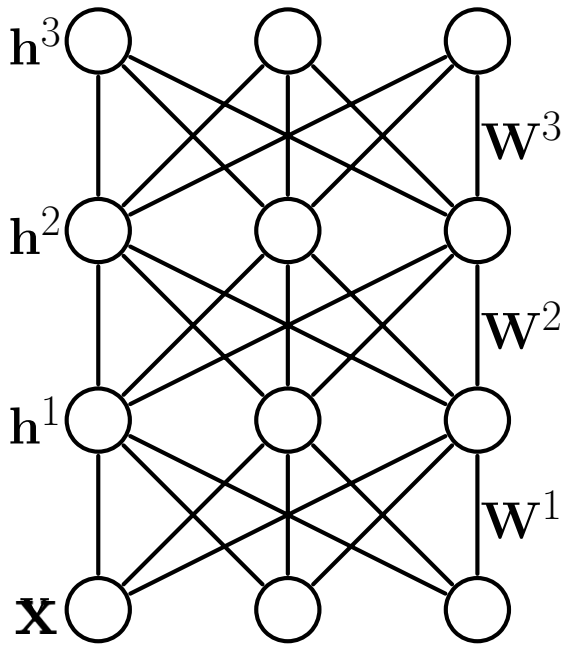
- Kingma & Welling, 2014
- Rezende, Mohamed, Daan, 2014
- Mnih & Gregor, 2014
- Bornschein & Bengio, 2015
- Tang & Salakhutdinov, 2013

Helmholtz Machines

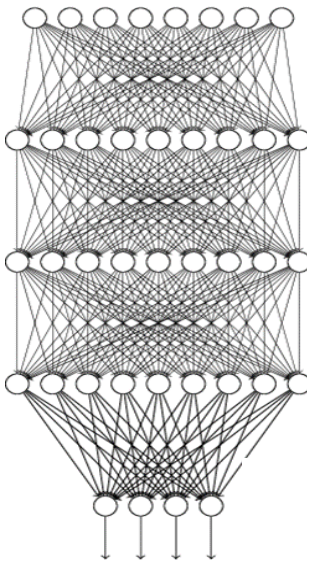
Helmholtz Machine



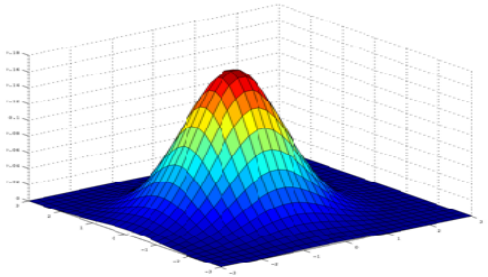
Deep Boltzmann Machine



Deep Directed Generative Models

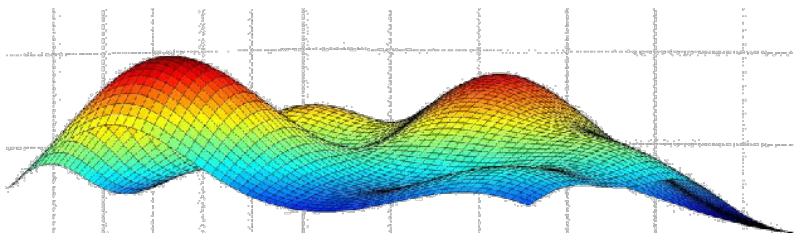


Code Z



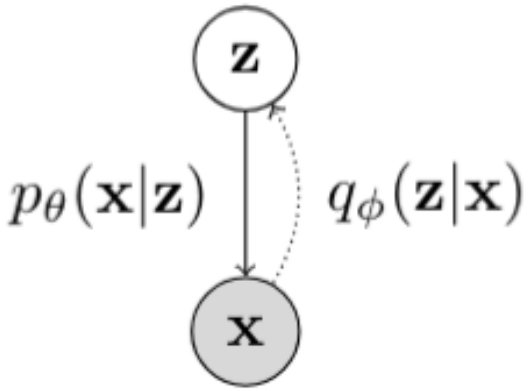
- Recognition
- Bottom-up
- Q(z|x)

- Generative
- Top-Down
- P(x|z)



D_{real}

- Latent Variable Models

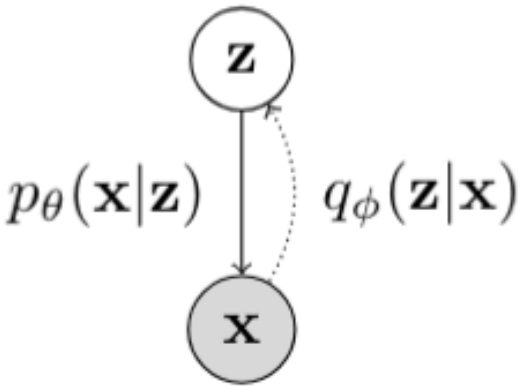


$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}, \mathbf{z}) \mathrm{d}\mathbf{z}$$

- Conditional distributions are parameterized by deep neural networks

Directed Deep Generative Models

- Directed Latent Variable Models with Inference Network



- Maximum log-likelihood objective

$$\max_\theta \sum_{\mathbf{x}\in\mathcal{D}} \log p_\theta(\mathbf{x})$$

- Marginal log-likelihood is intractable:

$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}, \mathbf{z}) \mathrm{d}\mathbf{z}$$

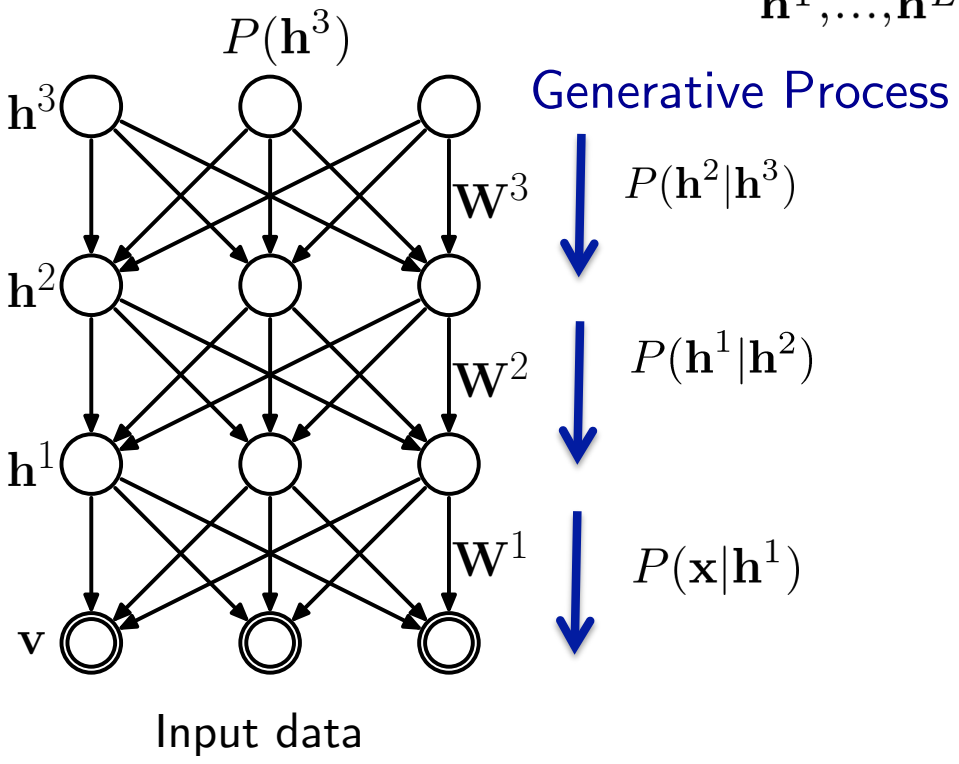
- Key idea: Approximate true posterior p(z|x) with a simple, tractable distribution q(z|x) (inference/recognition network).

Variational Autoencoders (VAEs)

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

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

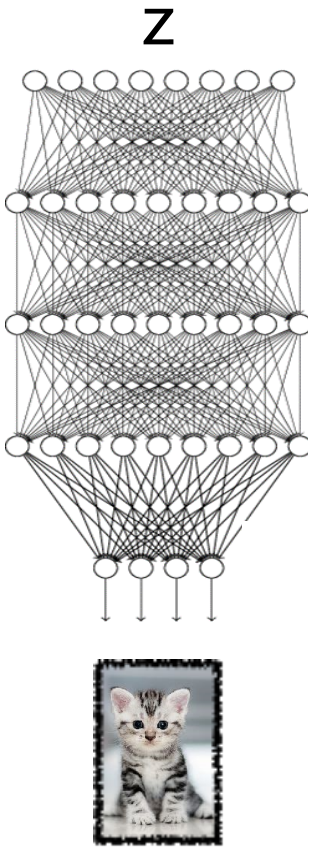
Each conditional term denotes a nonlinear relationship

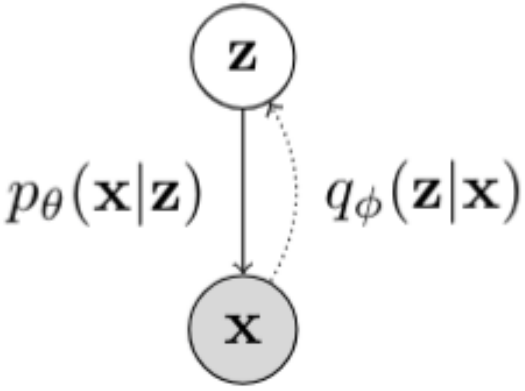


- L is the number of **stochastic** layers
- Sampling and probability evaluation is tractable for each $p(\mathbf{h}^\ell|\mathbf{h}^{\ell+1})$

Variational Autoencoders (VAEs)

- Single stochastic (Gaussian) layer, followed by many deterministic layers





$$p(\mathbf{z}) = \mathcal{N}(0, I)$$

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}\big(\mu(\mathbf{z}, \theta), \Sigma(\mathbf{z}, \theta)\big)$$

Deep neural network parameterized by θ.
(Can use different noise models)

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}\big(\mu(\mathbf{x}, \phi), \Sigma(\mathbf{x}, \phi)\big)$$

Deep neural network parameterized by ϕ.

Variational Bound

- VAE is trained to maximize the variational lower bound:

$$\begin{aligned}
\log p_\theta(\mathbf{x}) &= \log \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \log \int q_\phi(\mathbf{z}|\mathbf{x}) \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\
&= \log \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \\
&\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log \left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \\
&= \log p_\theta(\mathbf{x}) - \mathrm{KL}\big(q_\phi(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z}|\mathbf{x})\big) = \mathcal{L}(\mathbf{x})
\end{aligned}$$

Tightness Condition:

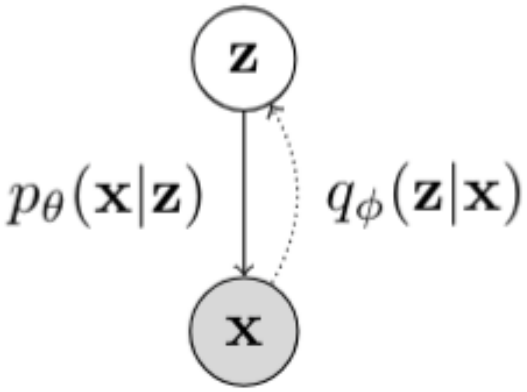
$$q_\phi(\mathbf{z}|\mathbf{x}) = p_\theta(\mathbf{z}|\mathbf{x})$$

- Trading off the data log-likelihood and the KL divergence from the true posterior
- Hard to optimize the variational bound with respect to the q recognition network (high variance)
- Key idea of Kingma and Welling is to use reparameterization trick

Reparameterization

- Assume that the recognition distribution is Gaussian:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}\big(\mu(\mathbf{x},\phi), \Sigma(\mathbf{x},\phi)\big)$$



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

$$\mathbf{z}(\epsilon, \mathbf{x}, \phi) = \Sigma(\mathbf{x},\phi)^{1/2}\epsilon + \mu(\mathbf{x},\phi), \quad \epsilon \sim \mathcal{N}(0, I)$$

- The recognition distribution can be expressed as a deterministic mapping

$$\underbrace{\mathbf{z}(\epsilon, \mathbf{x}, \phi)}$$

Deterministic Encoder

  - Distribution of ε does not depend on ϕ

Computing Gradients

- The gradients of the variational bound w.r.t the recognition (similar w.r.t the generative) parameters:

$$\begin{aligned}
\nabla_\phi \mathcal{L}(\mathbf{x}) &= \nabla_\phi \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] \\
&= \nabla_\phi \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z}(\epsilon,\mathbf{x},\phi))}{q_\phi(\mathbf{z}(\epsilon,\mathbf{x},\phi)|\mathbf{x})}\right] \\
&= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}\left[\nabla_\phi \log \frac{p_\theta(\mathbf{x},\mathbf{z}(\epsilon,\mathbf{x},\phi))}{q_\phi(\mathbf{z}(\epsilon,\mathbf{x},\phi)|\mathbf{x})}\right]
\end{aligned}$$

Autoencoder

Gradients can be computed by backprop

The mapping $\mathbf{z}$ is a deterministic neural net for fixed ϵ

VAE Assumptions

- Remember the variational bound:

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

- The variational assumptions must be approximately satisfied.

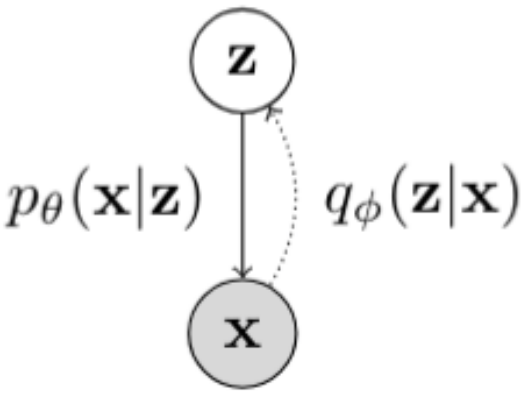
- The posterior distribution must be approximately factorial (common practice) and predictable with a feed-forward net.

- We can relax these assumptions using a tighter lower bound on marginal log-likelihood.

Importance Weighted Autoencoder

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

$$\mathcal{L}_k(\mathbf{x}) = \mathbb{E}_{\mathbf{z}_1,\mathbf{z}_2,\ldots,\mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{1}{k}\sum_{i=1}^{k}\frac{p_\theta(\mathbf{x},\mathbf{z}_i)}{q_\phi(\mathbf{z}_i|\mathbf{x})}\right]$$



$$= \mathbb{E}_{\mathbf{z}_1,\mathbf{z}_2,\ldots,\mathbf{z}_k \sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{1}{k}\sum_{i=1}^{k} w_i\right]$$

- where multiple z are sampled from the recognition network.

unnormalized importance weights

- Can improve the tightness of the bound.

Burda, Grosse, Salakhutdinov, ICLR 2016,
Mnih & Rezende, ICML 2016

Tighter Lower Bound

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

- For all k, the lower bounds satisfy:

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

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

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

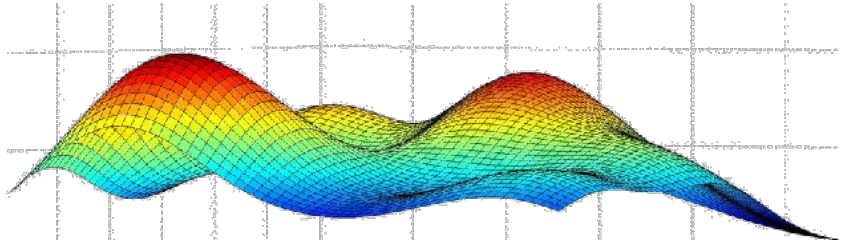
Burda, Grosse, Salakhutdinov, ICLR 2016,
Mnih & Rezende, ICML 2016

Talk Roadmap

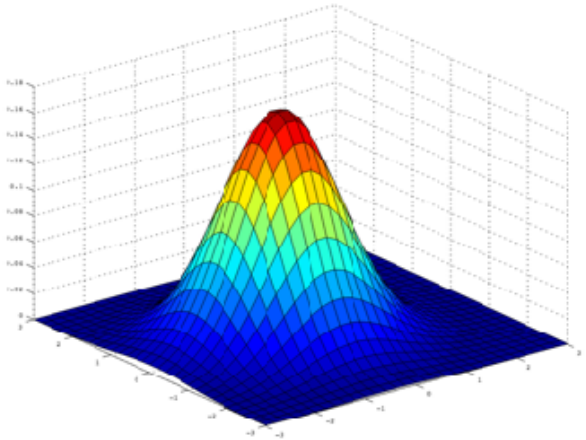
- Fully Observed Models
- Undirected Deep Generative Models
  - Restricted Boltzmann Machines (RBMs),
  - Deep Boltzmann Machines (DBMs)
- Directed Deep Generative Models
  - Variational Autoencoders (VAEs)
  - Normalizing Flows
- Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GAN)

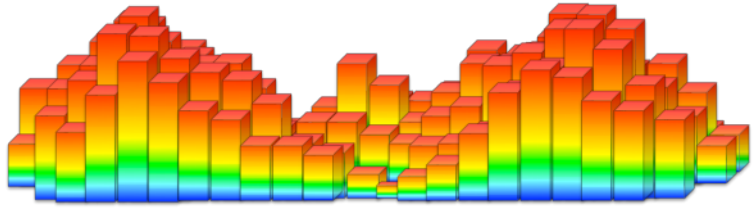
- Implicit generative model for an unknown target density $p(x)$
- Converts sample from a known noise density $p_Z(z)$ to the target $p(x)$



Unknown target density $p(x)$ of data over domain $\mathcal{X}$, e.g. $\mathbb{R}^{32\times32}$



Noise density $p_Z(z)$ over space $\mathcal{Z}$



Distribution of generated samples should follow target density $p(x)$

[Slide Credit: Manzil Zaheer]

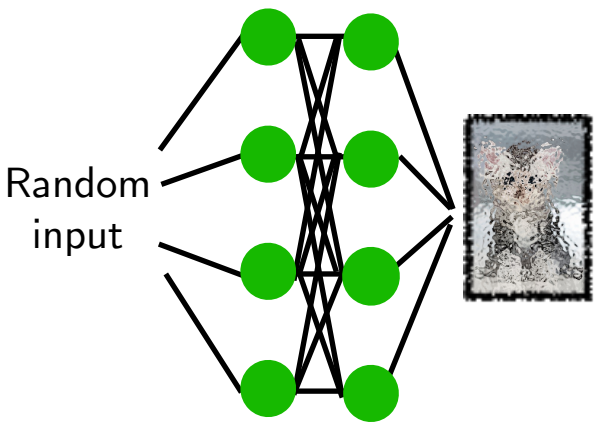
Goodfellow et al, 2014

GAN Formulation

- GAN consists of two components

Generator

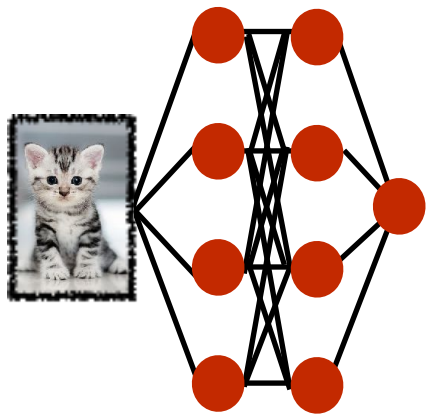
$G : \mathcal{Z} \rightarrow \mathcal{X}$



Goal: Produce samples indistinguishable from true data

Discriminator

$D : \mathcal{X} \rightarrow \mathbb{R}$



Goal: Distinguish true and generated data apart

[Slide Credit: Manzil Zaheer]

Goodfellow et al, 2014

Wasserstein GAN

- WGAN optimization

$$\min_G \max_D W(G, D) = \mathbb{E}_{x \sim p}\big[D(x)\big] - \mathbb{E}_{z \sim p_Z}\big[D(G(z))\big]$$

- Difference in expected output on real vs. generated images
  - Generator attempts to drive objective ≈ 0
- More stable optimization

Compare to training DBMs

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

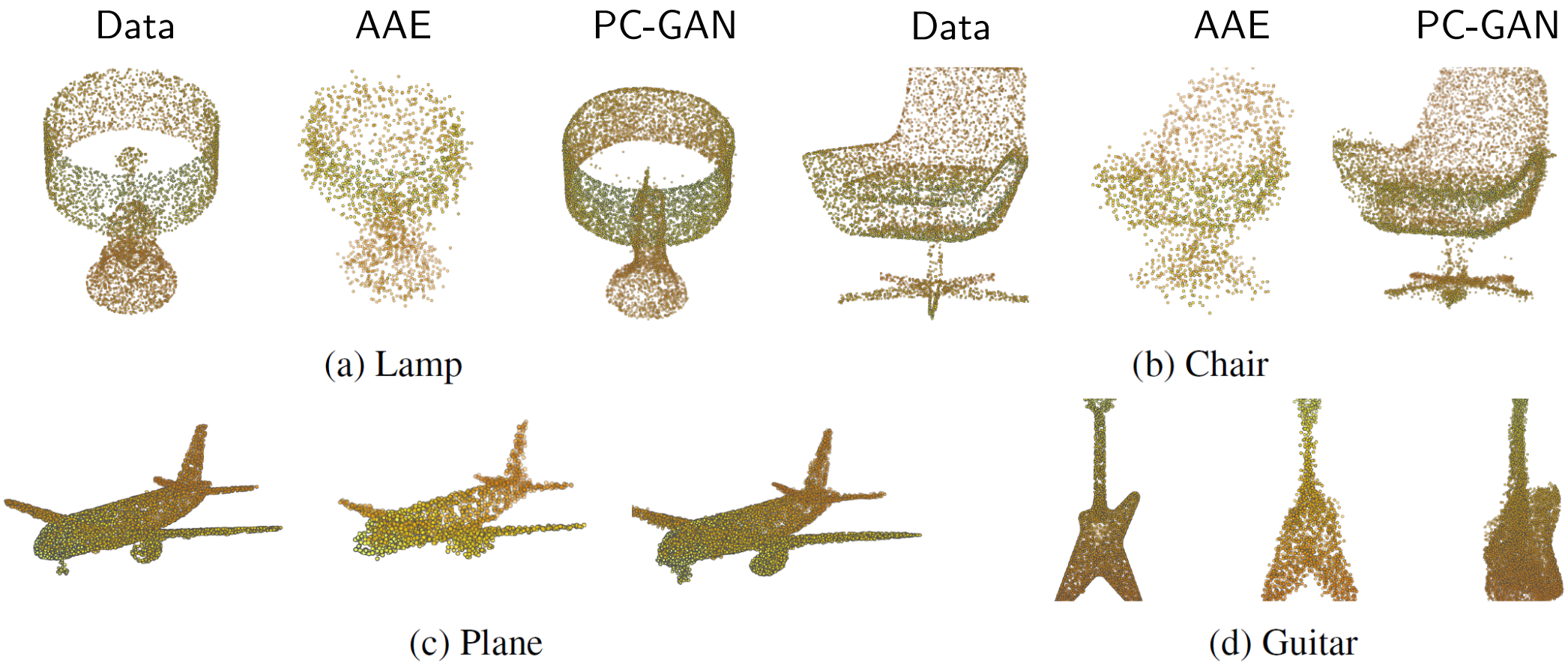
D outputs:

$D(x) = 1$ real

$D(x) = 0$ generated

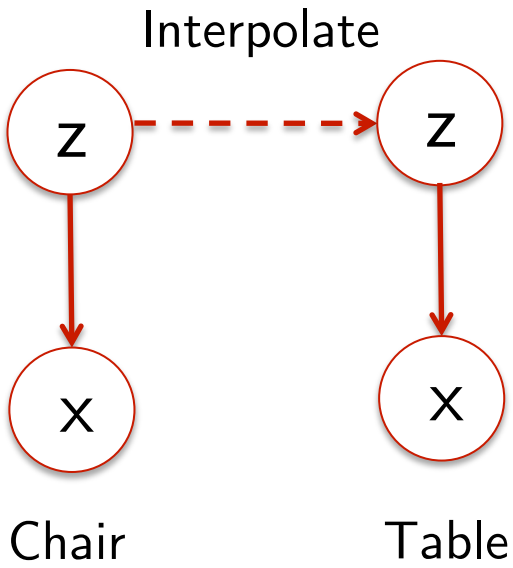
Arjovsky et al., 2017

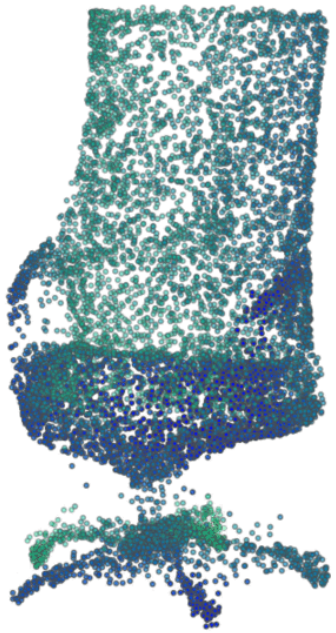
Modelling Point Cloud Data

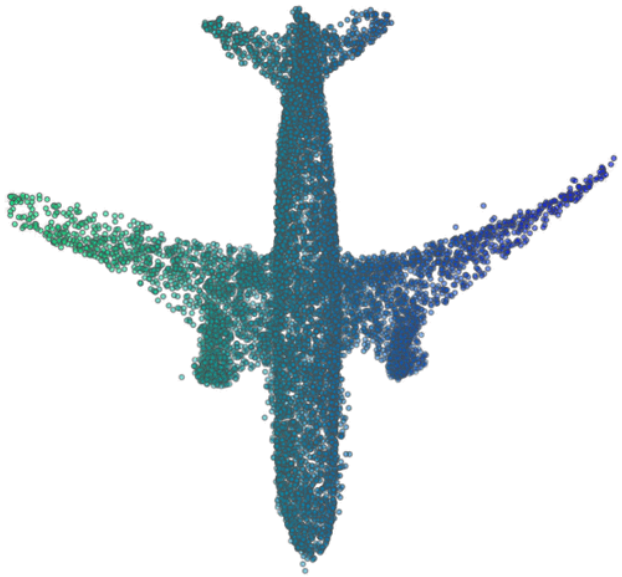


Zaheer et al. Point Cloud GAN 2018

Interpolation in Latent Space



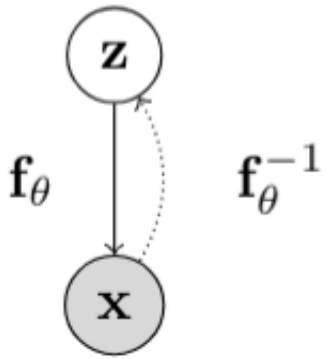




Zaheer et al. Point Cloud GAN 2018

Normalizing Flows

- Directed Latent Variable Invertible models



  - The mapping between x and z is deterministic and invertible:

$$\mathbf{x} = \mathbf{f}_\theta(\mathbf{z})$$
$$\mathbf{z} = \mathbf{f}_\theta^{-1}(\mathbf{x})$$

- Use change-of-variables to relate densities between z and x

$$p_X(\mathbf{x};\theta) = p_Z(\mathbf{z}) \left| \det \frac{\partial \mathbf{f}_\theta^{-1}(\mathbf{x})}{\partial X} \right|_{X=\mathbf{x}}$$

Grover and Ermon DGM Tutorial, NICE (Dinh et al. 2014), Real NVP (Dinh et al. 2016)

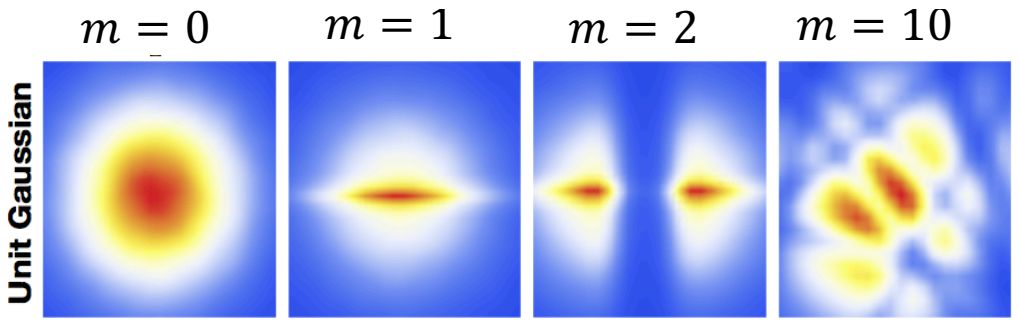
Normalizing Flows

- Invertible transformations can be composed:

$$\mathbf{x} = \mathbf{f}_\theta^M \circ \cdots \circ \mathbf{f}_\theta^1(\mathbf{z}^0); \qquad p_X(\mathbf{x};\theta) = p_{Z^0}(\mathbf{z}^0) \prod_{m=1}^{M} \left| \det \frac{\partial (\mathbf{f}_\theta^m)^{-1}}{\partial Z^m} \right|_{Z^m = \mathbf{z}^m}$$

- Planar Flows

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u} g(\mathbf{w}^\top \mathbf{z} + b)$$



Rezendre and Mohamed, 2016

Rezendre and Mohamed, 2016, Grover and Ermon DGM Tutorial

Normalizing Flows

- Maximum log-likelihood objective

$$\max_{\theta} \log p_X(\mathcal{D};\theta) = \sum_{\mathbf{x}\in\mathcal{D}} \left(\log p_Z(\mathbf{z}) - \log \left| \det \frac{\partial (\mathbf{f}_\theta)^{-1}}{\partial X} \right|_{X=\mathbf{x}} \right)$$

- Exact log-likelihood evaluation via inverse transformations
- Sampling from the model

$$\mathbf{z} \sim p_Z(\mathbf{z}), \quad \mathbf{x} = \mathbf{f}_\theta(\mathbf{z})$$

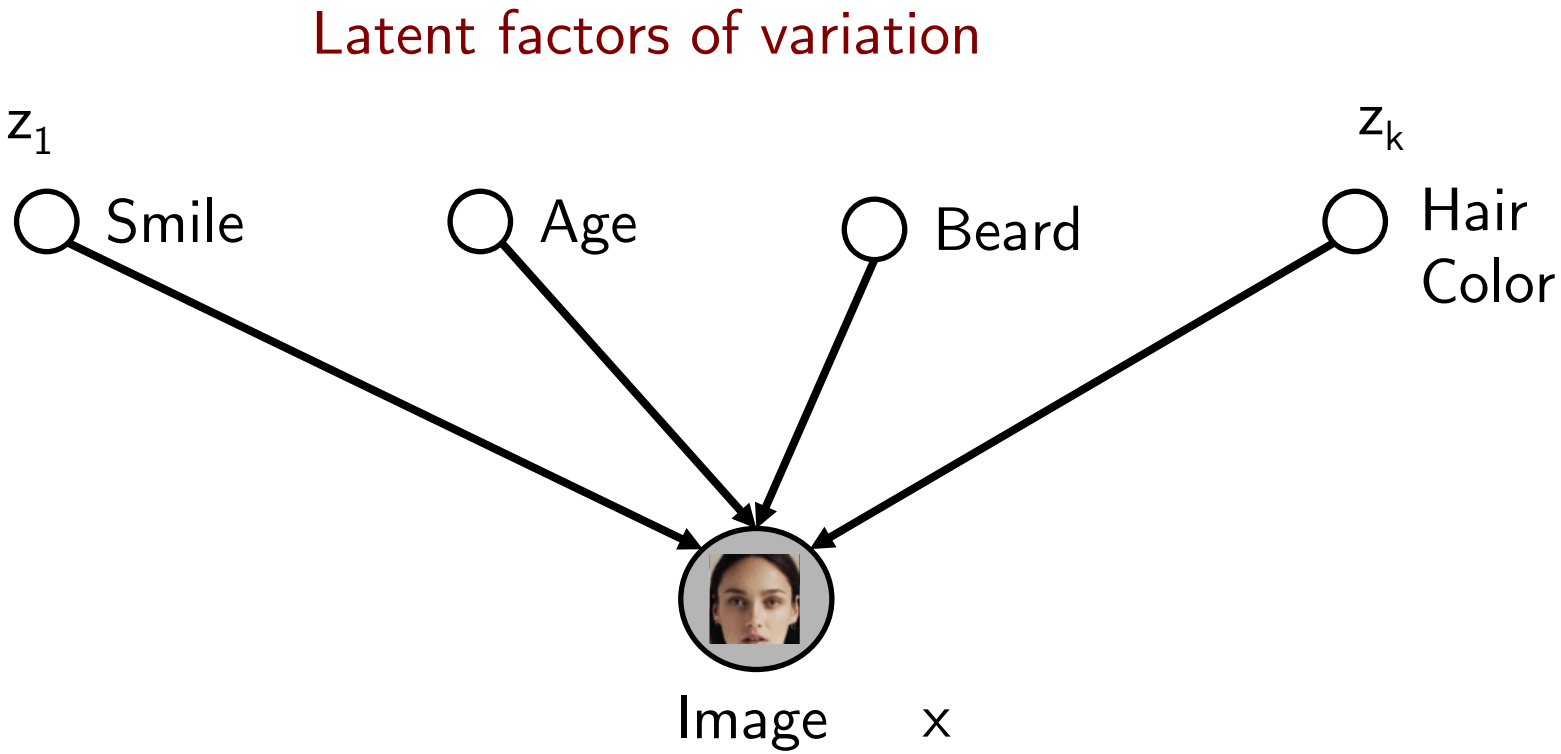
- Inference over the latent representations:

$$\mathbf{z} \quad = \quad \mathbf{f}_\theta^{-1}(\mathbf{x})$$

Rezendre and Mohamed, 2016, Grover and Ermon DGM Tutorial

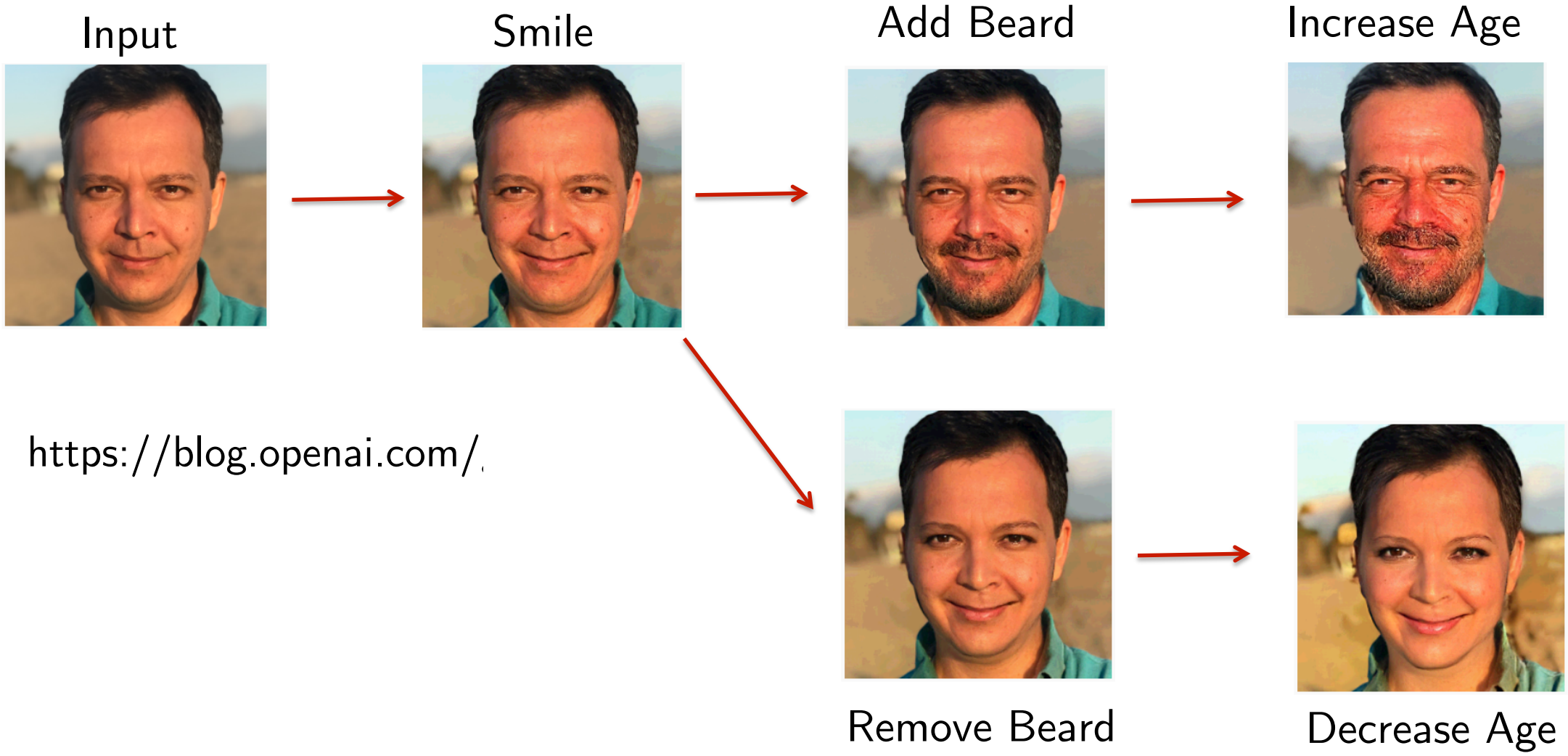
Example: GLOW

- Generative Flow with Invertible 1x1 Convolutions
  https://blog.openai.com/glow/



Kingma, Dhariwal, 2018

Example: GLOW

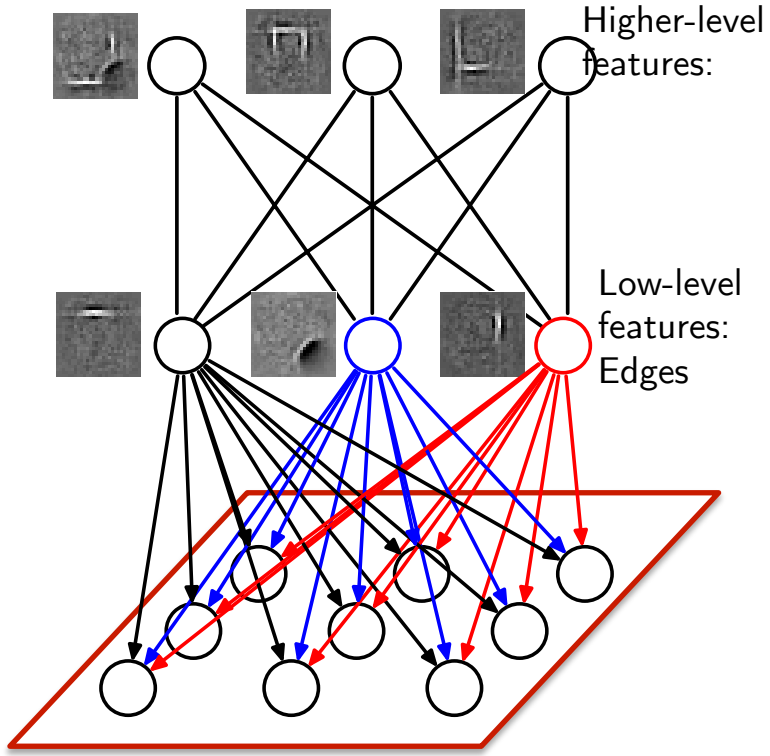


https://blog.openai.com/

Thank you

Important Breakthroughs

- Deep Belief Networks, 2006 (Unsupervised)

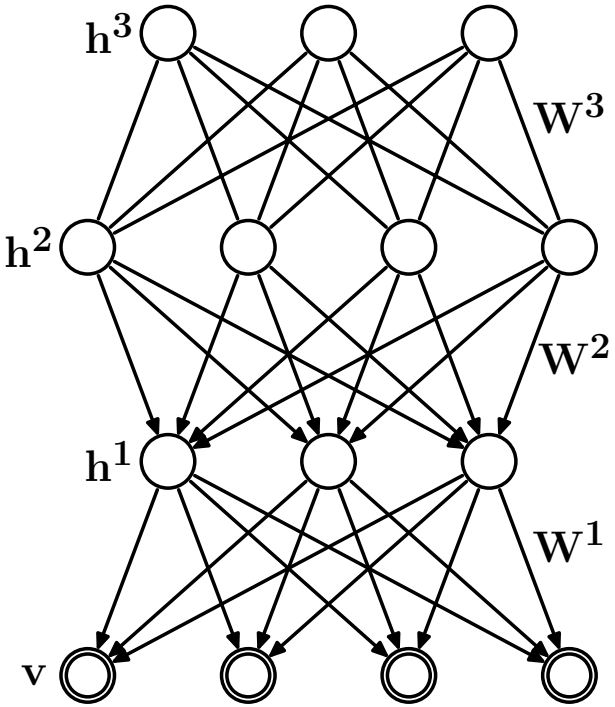


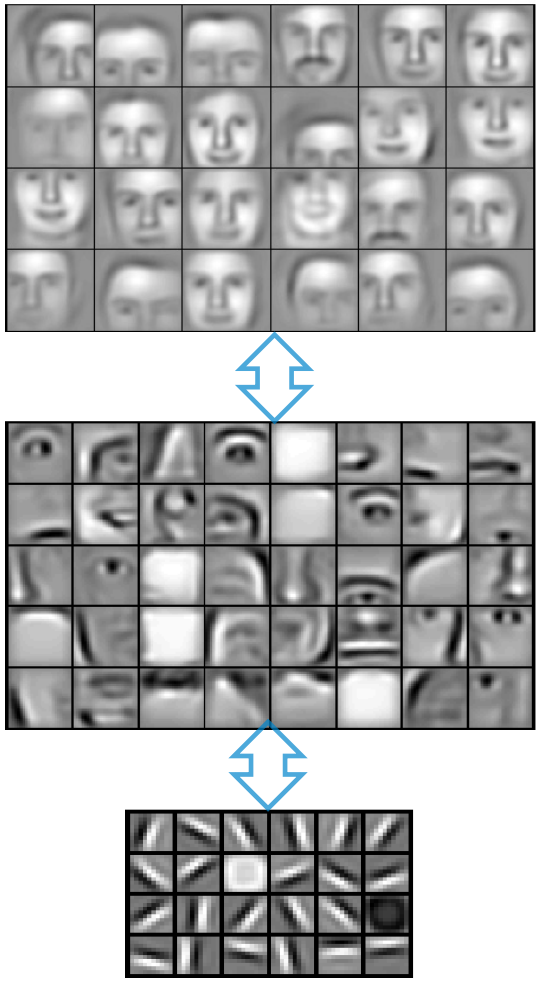
- Theoretical Breakthrough:
  - Adding additional layers improves variational lower-bound.
- Efficient Learning and Inference with multiple layers

Hinton, G. E., Osindero, S. and Teh, Y., A fast learning algorithm for deep belief nets, Neural Computation, 2006.

Learning Part-Based Representations

Deep Belief Network





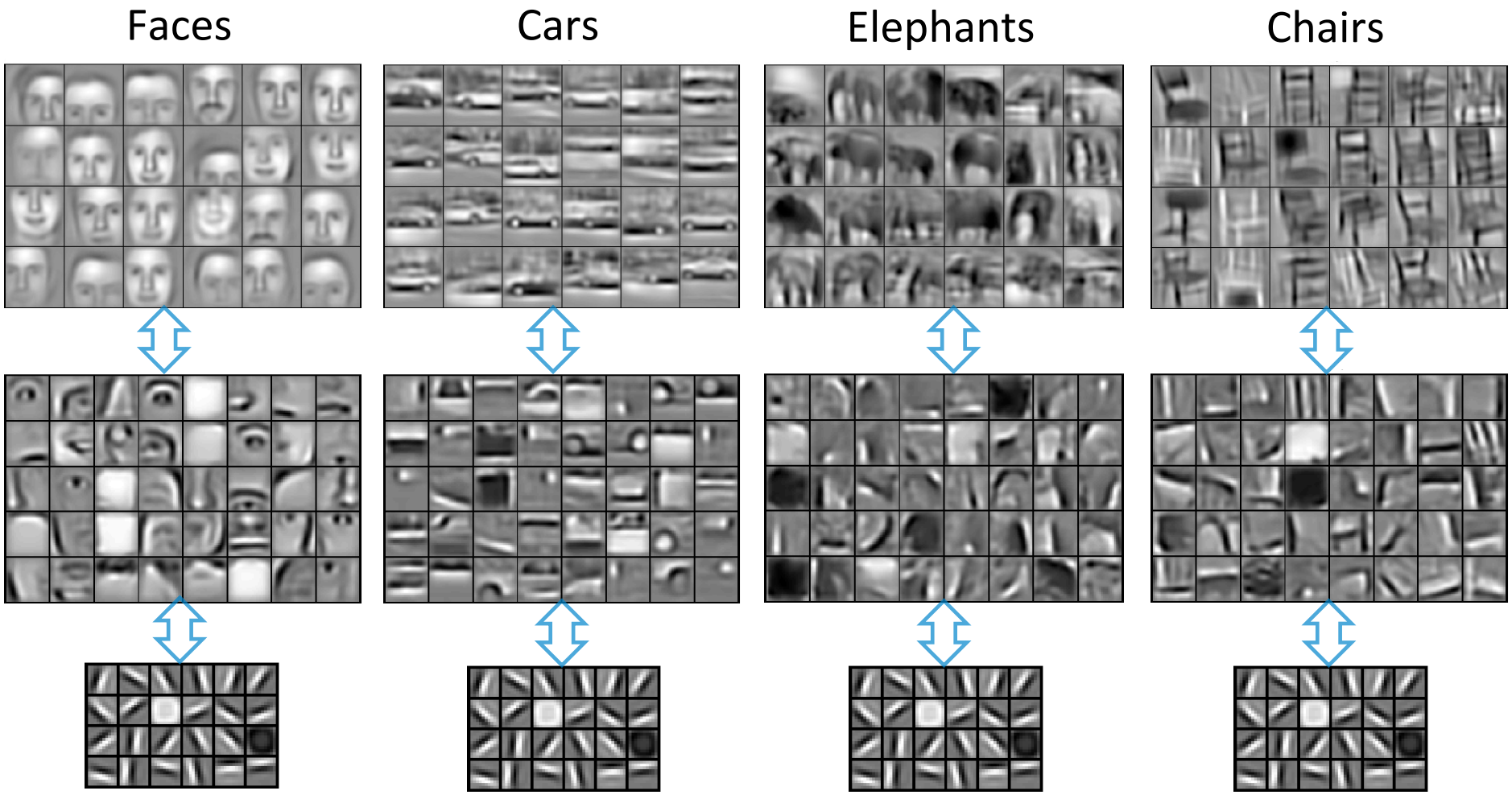
Groups of parts

Object Parts

Trained on face images.

Lee, Grosse, Ranganath, Ng, ICML 2009

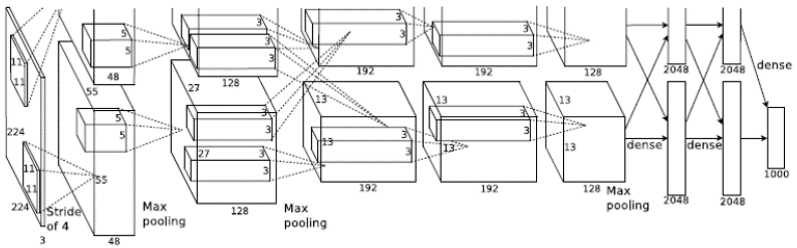
Learning Part-Based Representations



Lee, Grosse, Ranganath, Ng, ICML 2009

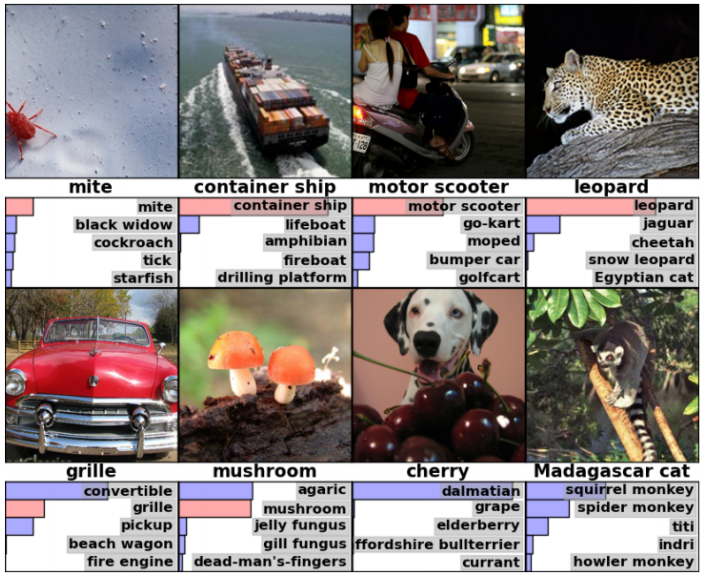
Important Breakthroughs

- Deep Convolutional Nets for Vision (Supervised)
  - Krizhevsky, A., Sutskever, I. and Hinton, G. E., ImageNet Classification with Deep Convolutional Neural Networks, NIPS, 2012.



IMAGENET

1.2 million training images
1000 classes

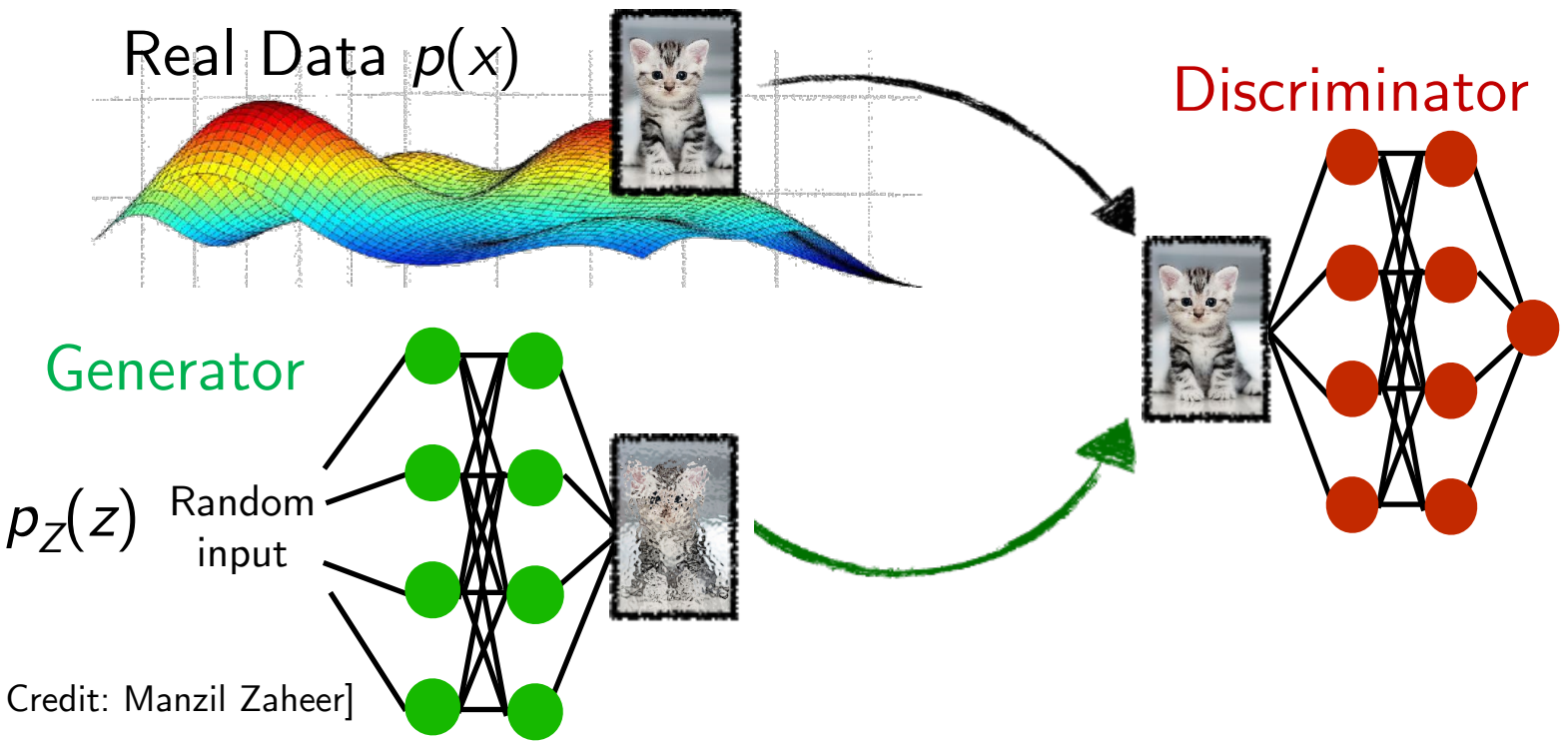


- Deep Nets for Speech (Supervised)
  - Hinton et. al. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups, IEEE Signal Processing Magazine. 2012.

GAN Formulation: Discriminator

- Discriminator's objective: Tell real and generated data apart like a classifier

$$\max_D \mathbb{E}_{x\sim p}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_Z}\left[\log\left(1 - D(G(z))\right)\right]$$



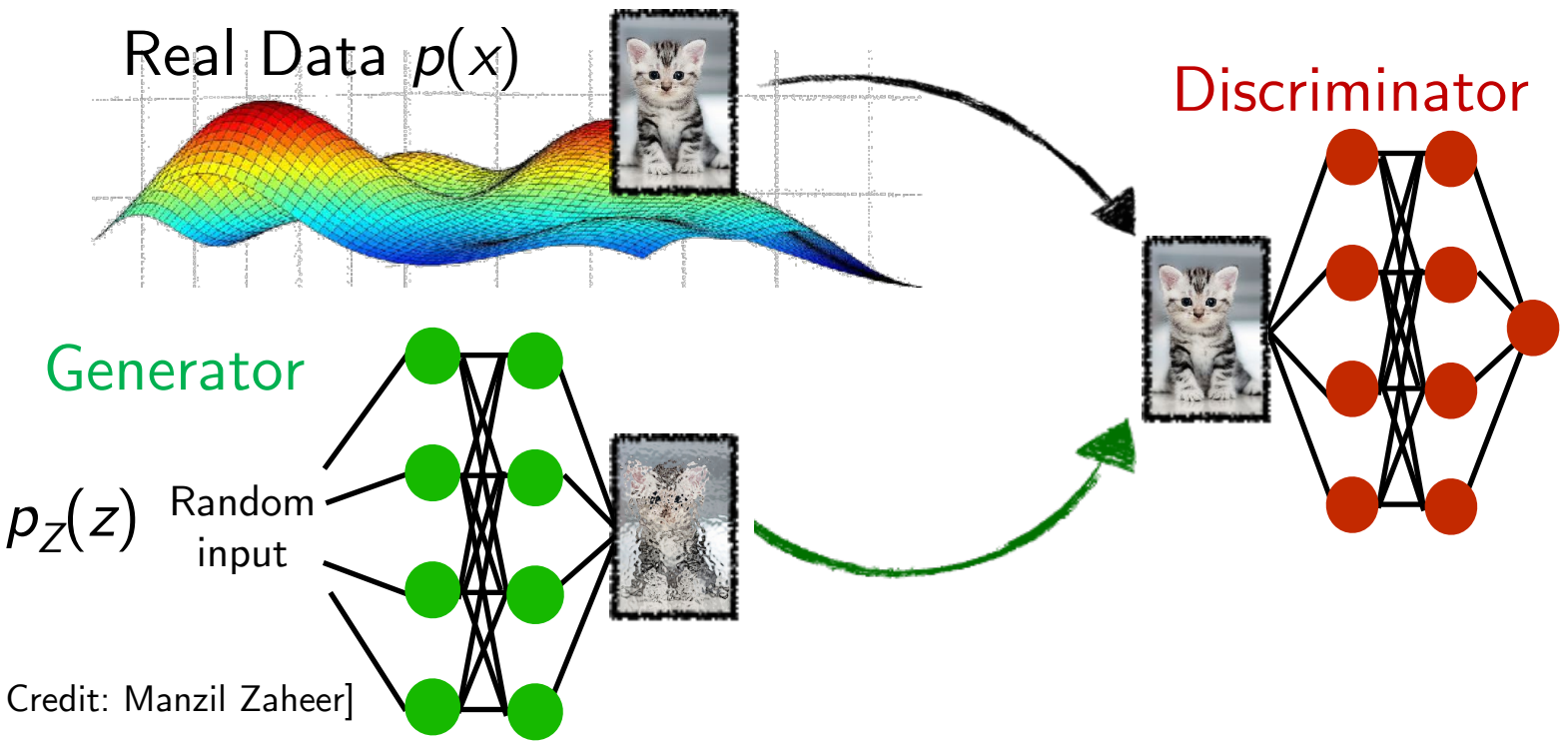
[Slide Credit: Manzil Zaheer]

Goodfellow et al, 2014

GAN Formulation: Generator

- Generator's objective: Fool the best discriminator

$$\min_{G} \max_{D} \mathbb{E}_{x \sim p}\left[\log D(x)\right] + \mathbb{E}_{z \sim p_Z}\left[\log\left(1 - D(G(z))\right)\right]$$



[Slide Credit: Manzil Zaheer]

Goodfellow et al, 2014

GAN Formulation: Optimization

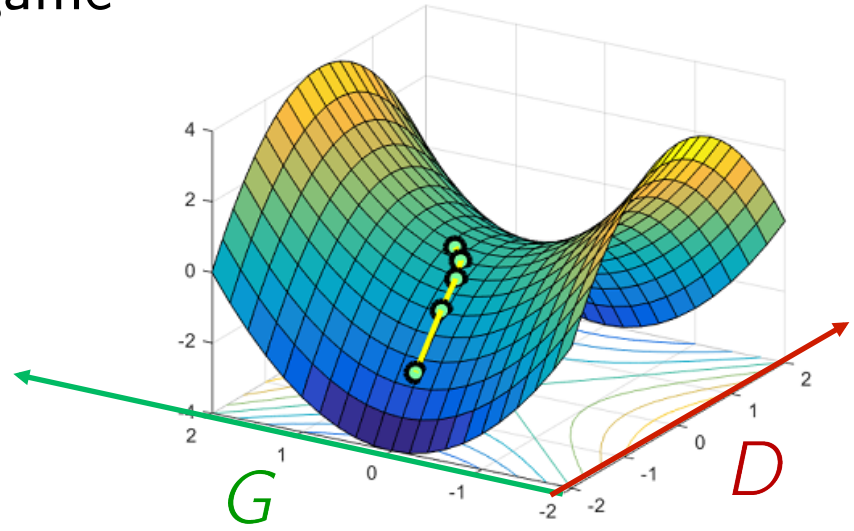
- Overall GAN optimization

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p}\left[\log D(x)\right] + \mathbb{E}_{z \sim p_Z}\left[\log\left(1 - D(G(z))\right)\right]$$

- The generator-discriminator are iteratively updated using SGD to find "equilibrium" of a "min-max objective" like a game

$$G \leftarrow G - \eta_G \nabla_G V(G, D)$$

$$D \leftarrow D - \eta_D \nabla_D V(G, D)$$



[Slide Credit: Manzil Zaheer]