Lecture 20: Deep Generative Models

CS 109B, STAT 121B, AC 209B, CSE 109B

Mark Glickman and Pavlos Protopapas

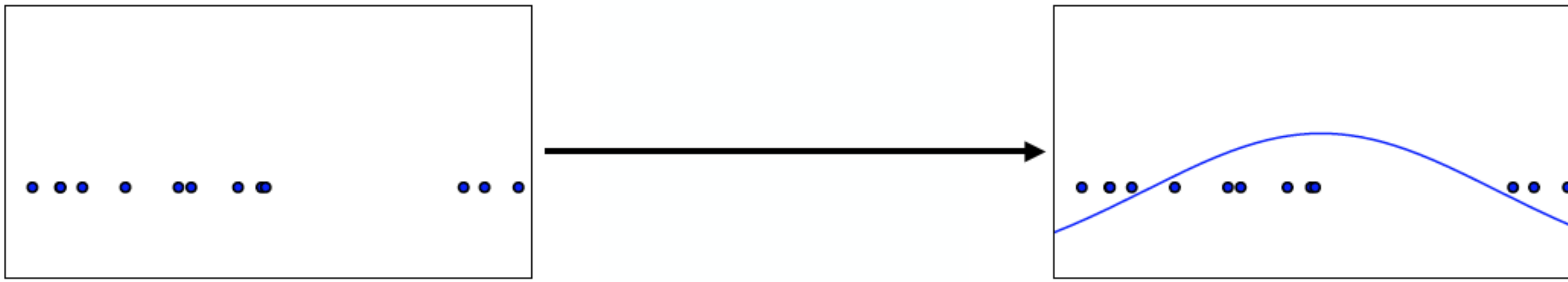
This lecture is based on the following tutorials:

I. Goodfellow, *Generative Adversarial Netwo*rks (GANs), NIPS 2016 Tutorial

Shenlong Wang, *Deep Generative Models*, http://www.cs.toronto.edu/~slwang/generative_model.pdf

Generative Model

Density Estimation



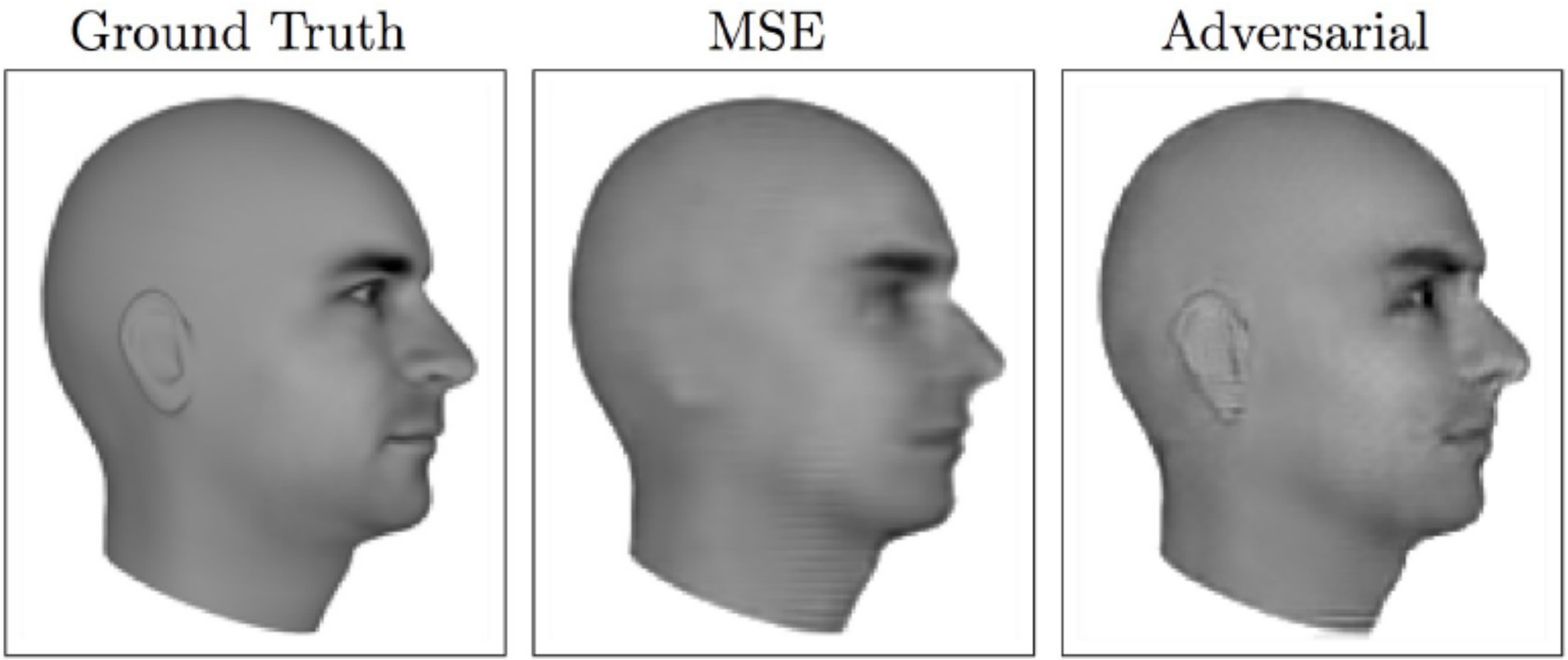
Sample Generation

Training examples   Model samples

Why generative modeling?

- Modeling high-dim probability distributions
- Denoising damaged samples
- Simulate future events for planning and RL
- Imputing missing data
  - Semi-supervised Learning
- Multi-modal data
  - Multiple correct answers

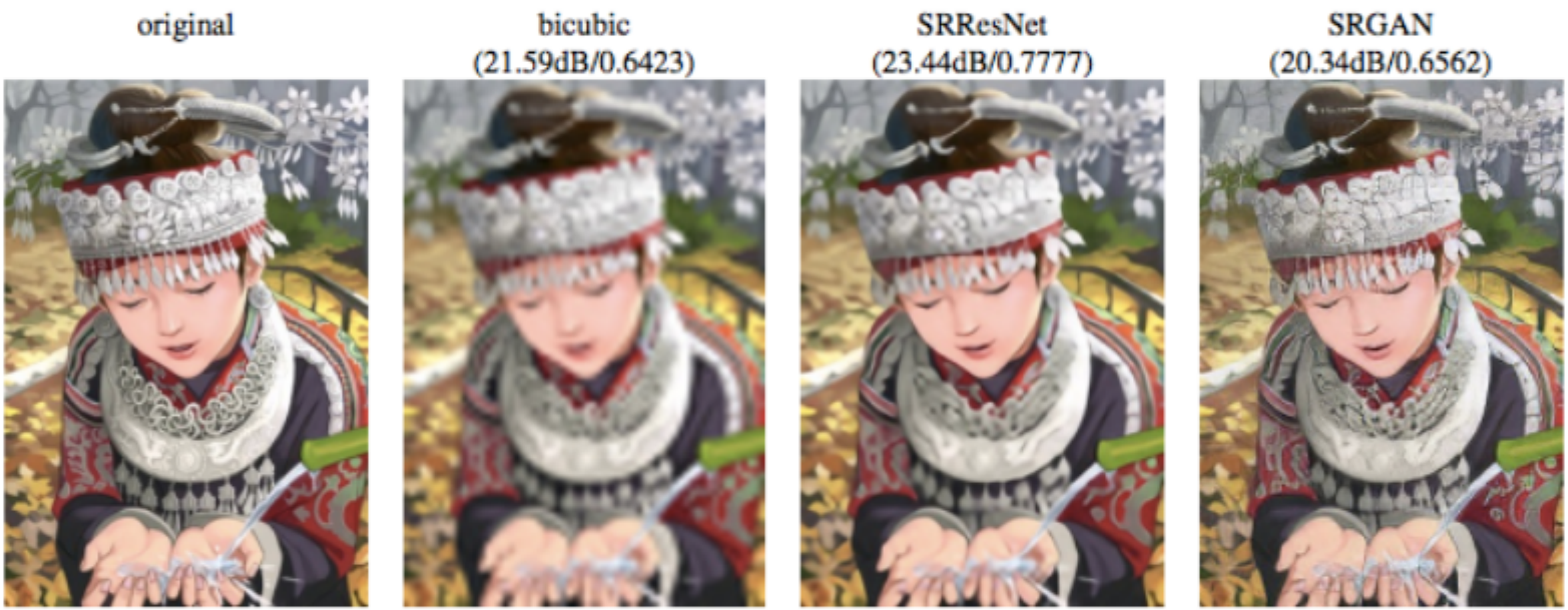
Next Video Frame Prediction



Lotter et al., 2016

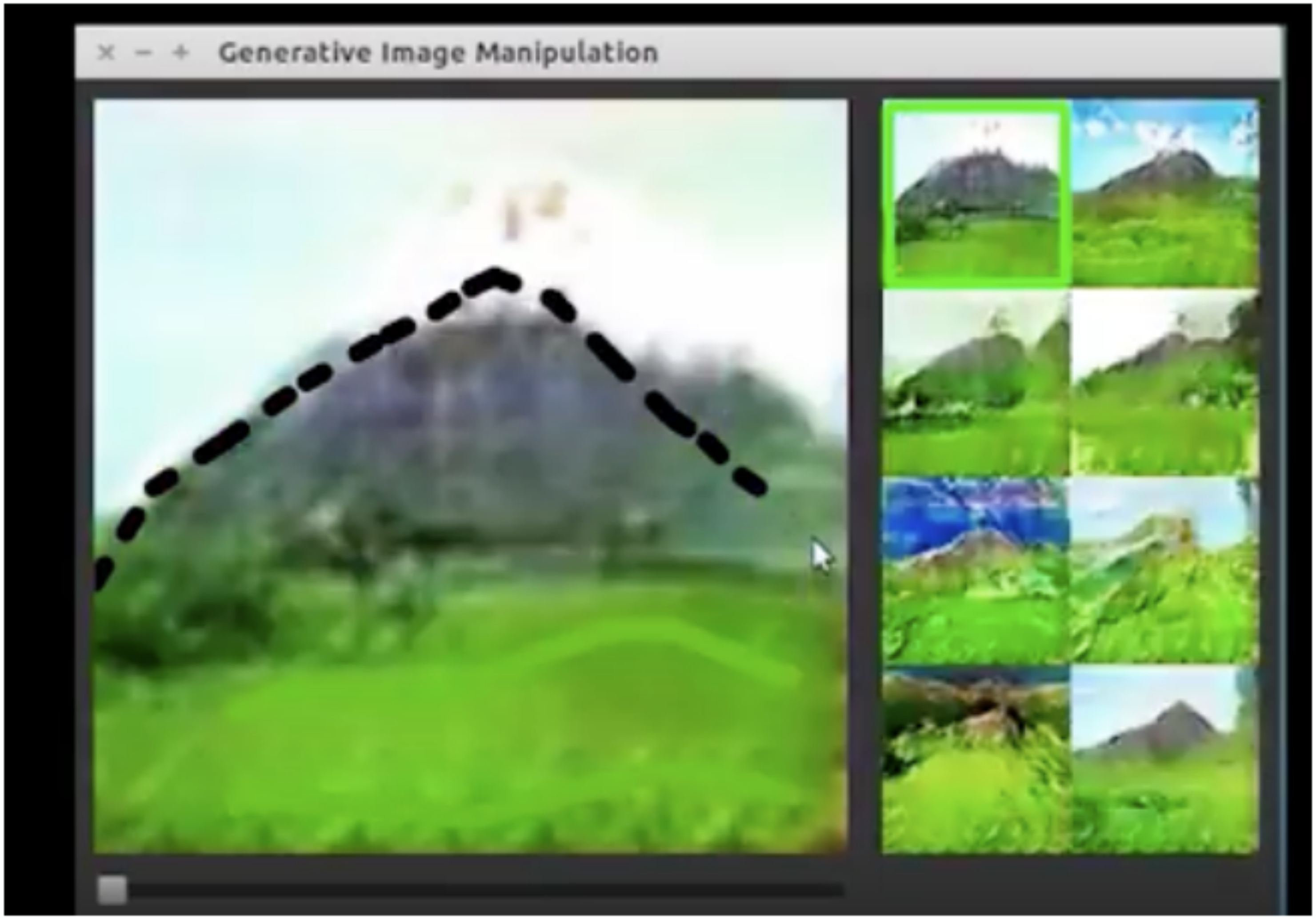
Single Image Super-resolution

- Synthesize a high-resolution equivalent from a low-resolution image



Ledig et al., 2016

Interactive Image Generation

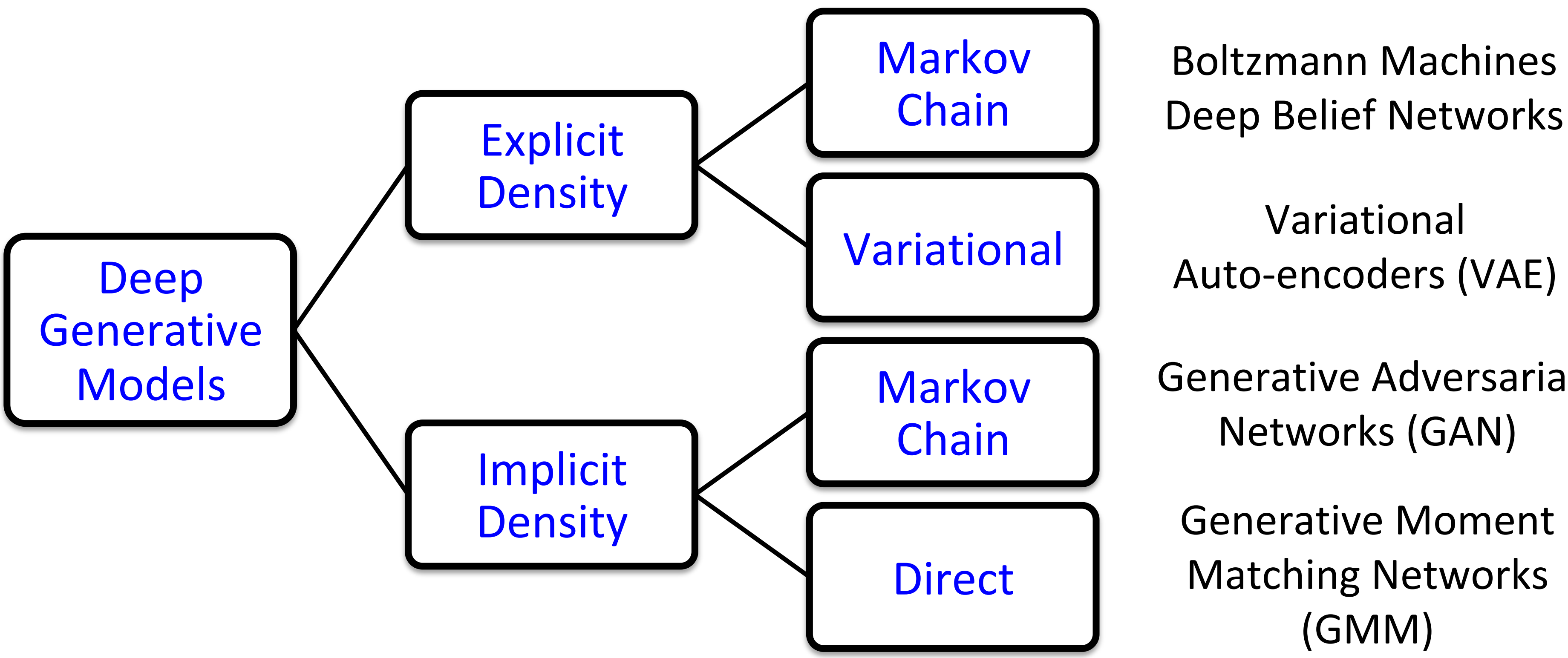


Zhu et al., 2016

Image-to-Image Translation

Isola et al., 2016

Taxonomy of Deep Generative Models



Outline

- *Explicit Density Models*
  - Boltzmann Machines & Deep Belief Networks (DBN)
  - Variational Auto-encoders (VAE)
- *Implicit Density Models*
  - Generative Adversarial Networks (GAN)

Boltzmann Machines

x: d-dimensional binary vector

$$P(x) = \frac{\exp(-E(x))}{Z}$$

$$Z = \sum_x \exp(-E(x))$$

Energy function given by:

$$E(x) = -x^T U x - b^T x$$

Visible and Hidden Units

x: d-dimensional binary vector

$$P(v,h) = \frac{\exp(-E(v,h))}{Z}$$

$$Z = \sum_{v,h} \exp(-E(v,h))$$

Energy function given by:

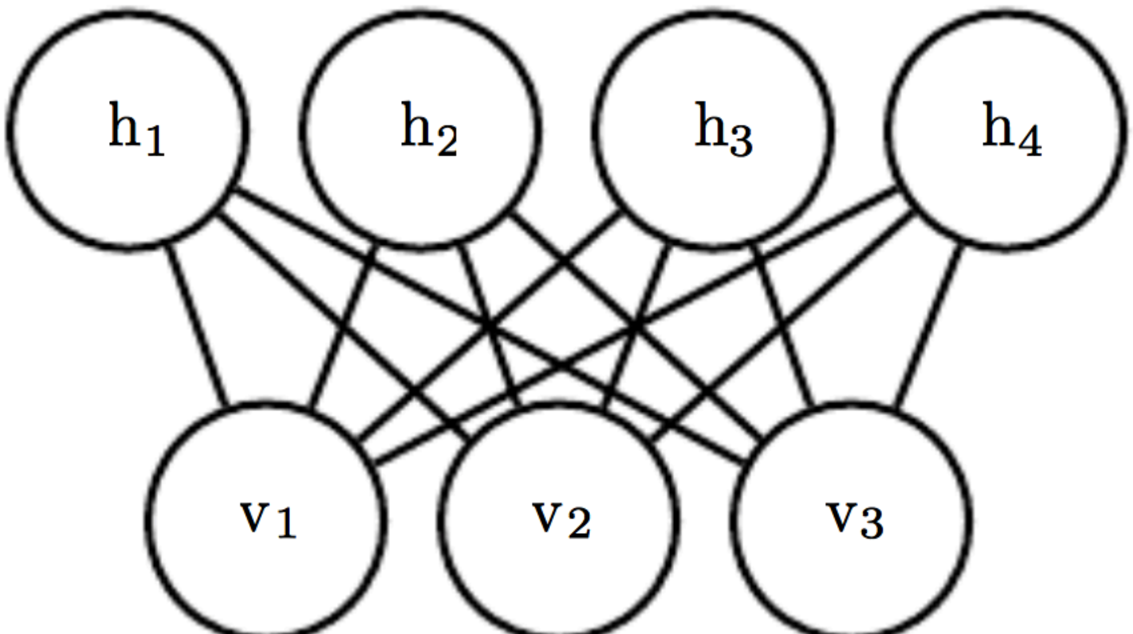
$$E(v,h) \ = \ -v^T R v - v^T W h - h^T S h - b^T v - c^T h$$

Training Boltzmann Machines

- Maximum-likelihood estimation
- Partition function is intractable
- May be estimated with MCMC methods

Restricted Boltzmann Machines

- No connections among hidden units, and among visible units (bipartite graph)



Restricted Boltzmann Machines

x: d-dimensional binary vector

$$P(v,h) = \frac{\exp(-E(v,h))}{Z}$$

$$Z = \sum_{v,h} \exp(-E(v,h))$$

Energy function given by:

$$E(v,h) \;=\; -b^T v - c^T h - v^T W h$$

Training RBMs

- Conditional distributions have simple form

$$P\left(h_j = 1 \middle| v\right) = \sigma\left(\mathrm{c}_j + (W^T v)_j\right)$$

$$P\left(v_j = 1 \middle| h\right) = \sigma\left(\mathrm{b}_j + (Wh)_j\right)$$

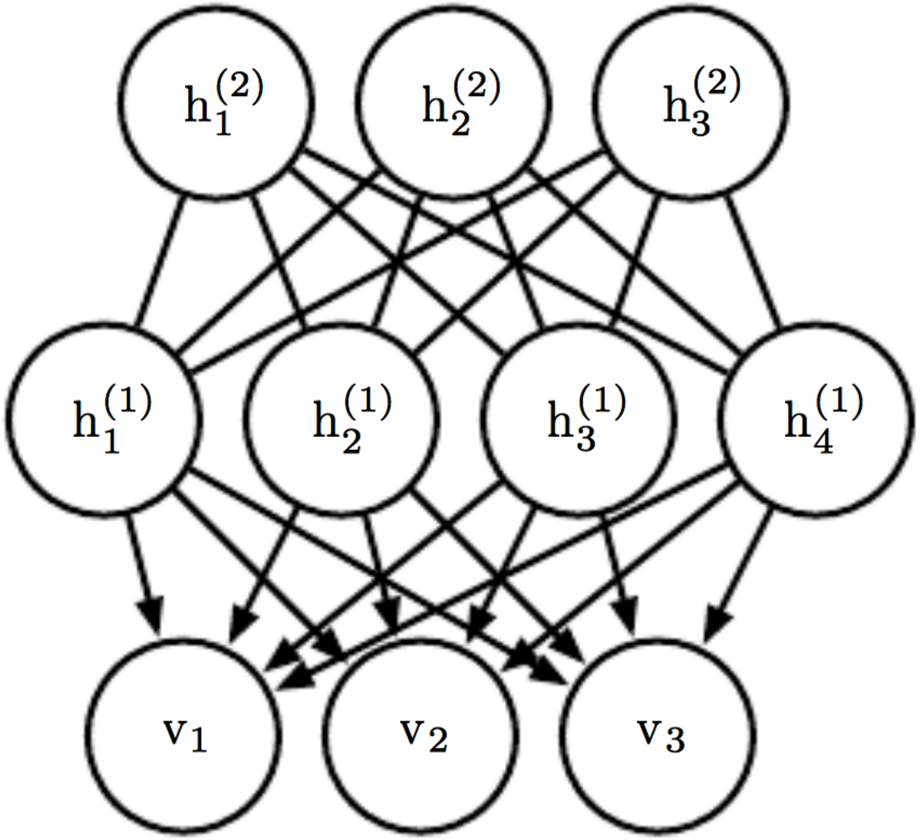
Sigmoid

- Efficient MCMC: block Gibbs sampling
  - Alternatively sample from P(h|v) and P(v|h)

Deep Belief Networks

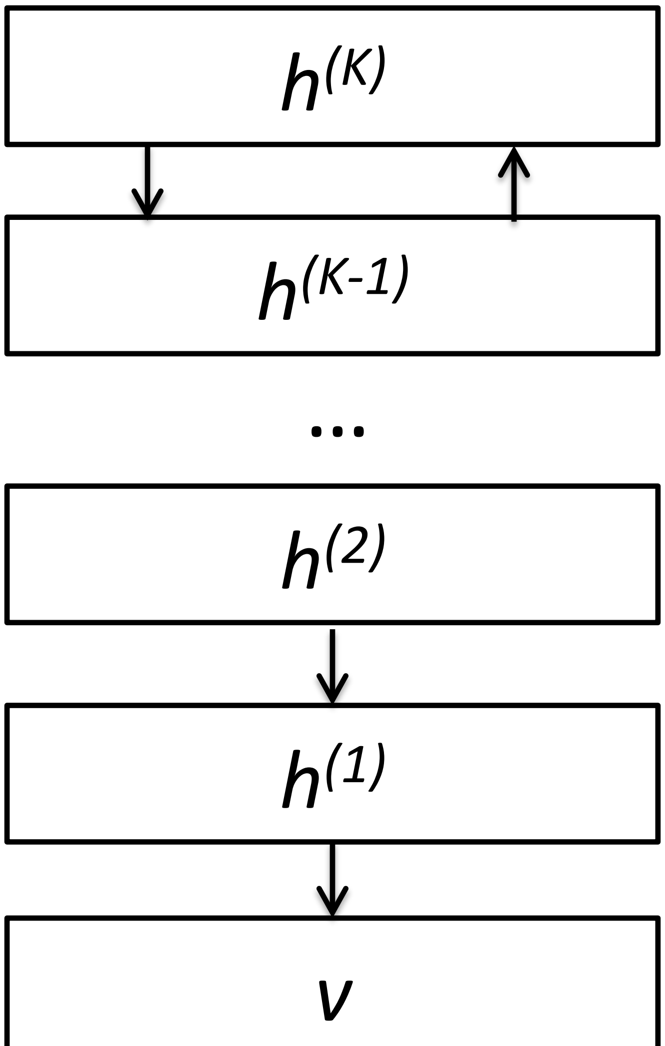
- Multiple hidden layers

Introduction of DBNs in 2006 began the current deep learning renaissance



Undirected connections between last two layers

Deep Belief Networks



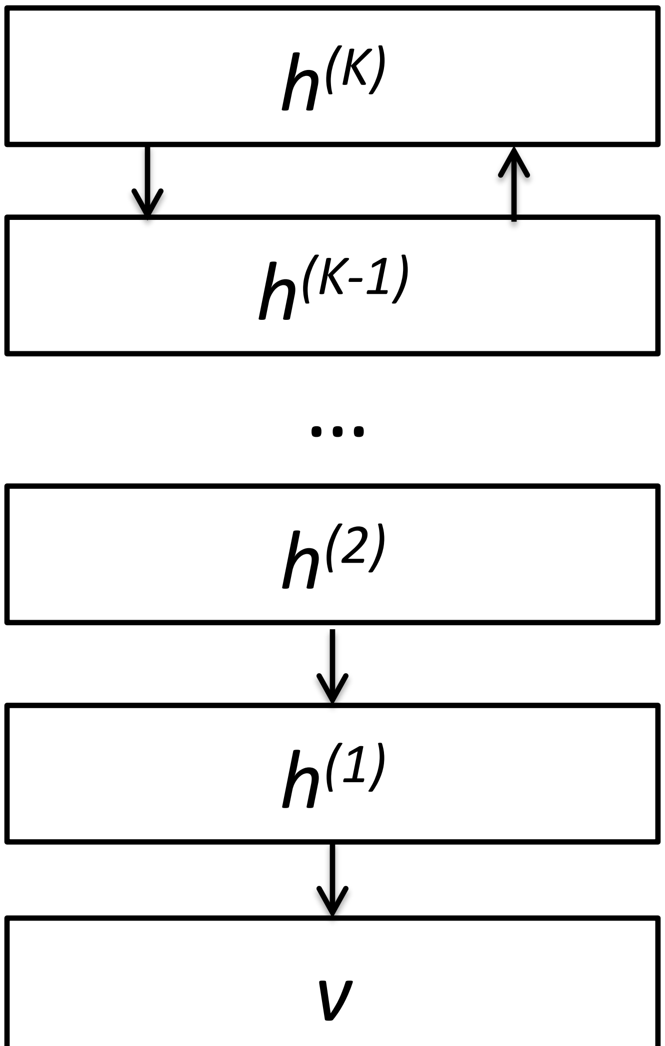
$$P(h^{(K-1)}, h^{(K)}) \;=\; \frac{1}{Z}\exp\left(E(h^{(K-1)}, h^{(K)})\right)$$

$$\vdots$$

$$P\left(h_i^{(1)} = 1 \middle| h^{(2)}\right) \;=\; \sigma\left(\mathrm{b}_i^{(2)} + w_i^{(2)} \bullet h^{(2)}\right)$$

$$P\left(v_i = 1 \middle| h^{(1)}\right) \;=\; \sigma\left(\mathrm{b}_i^{(1)} + w_i^{(1)} \bullet h^{(1)}\right)$$

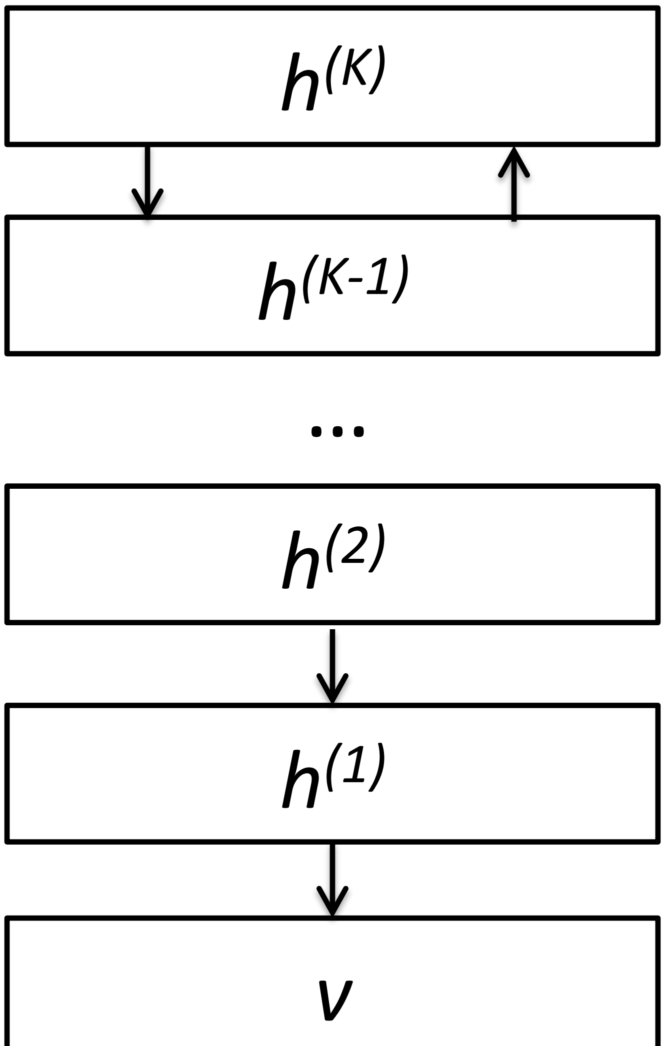
Layer-wise DBN Training



Train an RBM to maximize

$$E_{v \sim p_{data}} \log p_\theta(v)$$

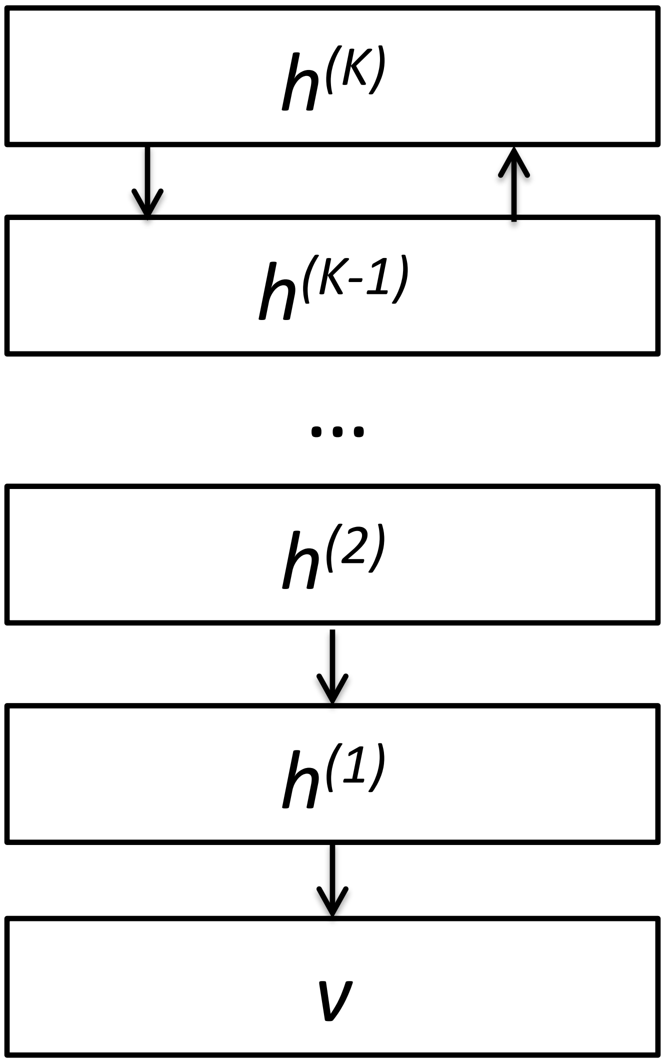
Layer-wise DBN Training



Train an RBM to maximize

$$E_{v\sim p_{data}} E_{h^{(1)}\sim p\left(h^{(1)}\middle|v\right)} \log p_{\theta_1}(h^{(1)})$$

Layer-wise DBN Training



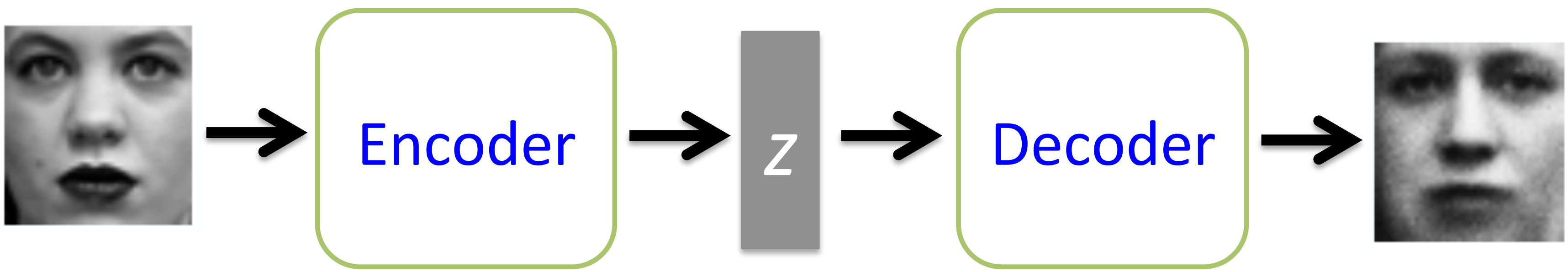
Train an RBM to maximize

$$E_{v\sim p_{data}} \cdots E_{h^{(K)}\sim p\left(h^{(K)}\middle|h^{(K-1)}\right)} \log p_{\theta_K}(h^{(K)})$$

Outline

- *Explicit Density Models*
  - Boltzmann Machines & Deep Belief Networks (DBN)
  - Variational Auto-encoders (VAE)
- *Implicit Density Models*
  - Generative Adversarial Networks (GAN)

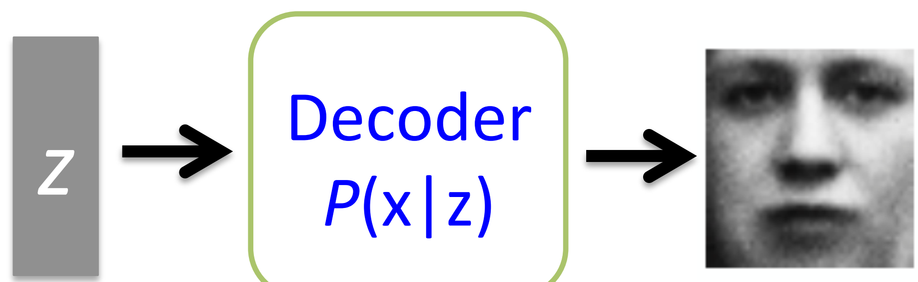
Recap: Autoencoder



Variational Autoencoder

- Easier to train using gradient-based methods
- VAE sampling:

Sample from $P(z)$
Standard Gaussian

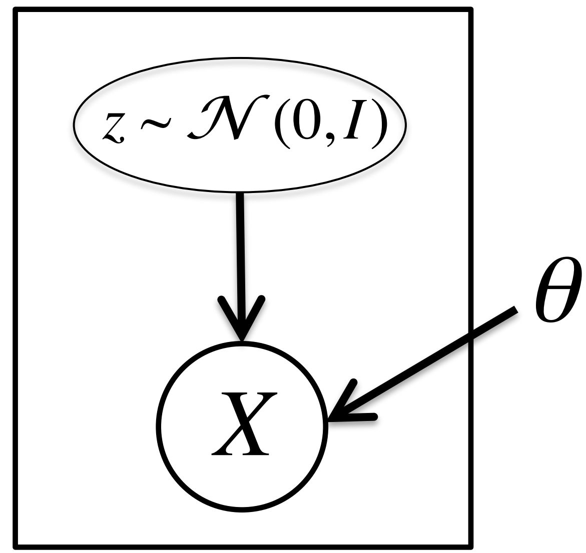


VAE Likelihood

Neural network

$$p_\theta(x) = \int_z p_\theta\left(x \middle| z\right) \boxed{p_\theta(z)}\, dz$$

Difficult to approximate in high dim through sampling



For most *z* values *p*(*x*|*z*) close to 0

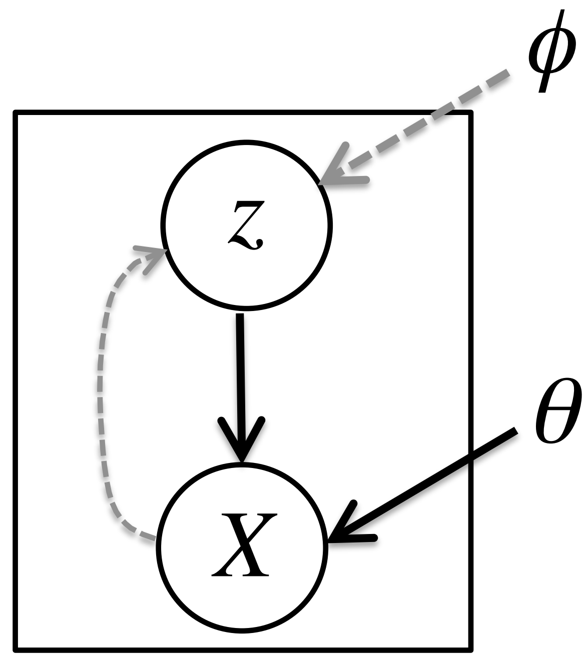
VAE Likelihood

Another neural net

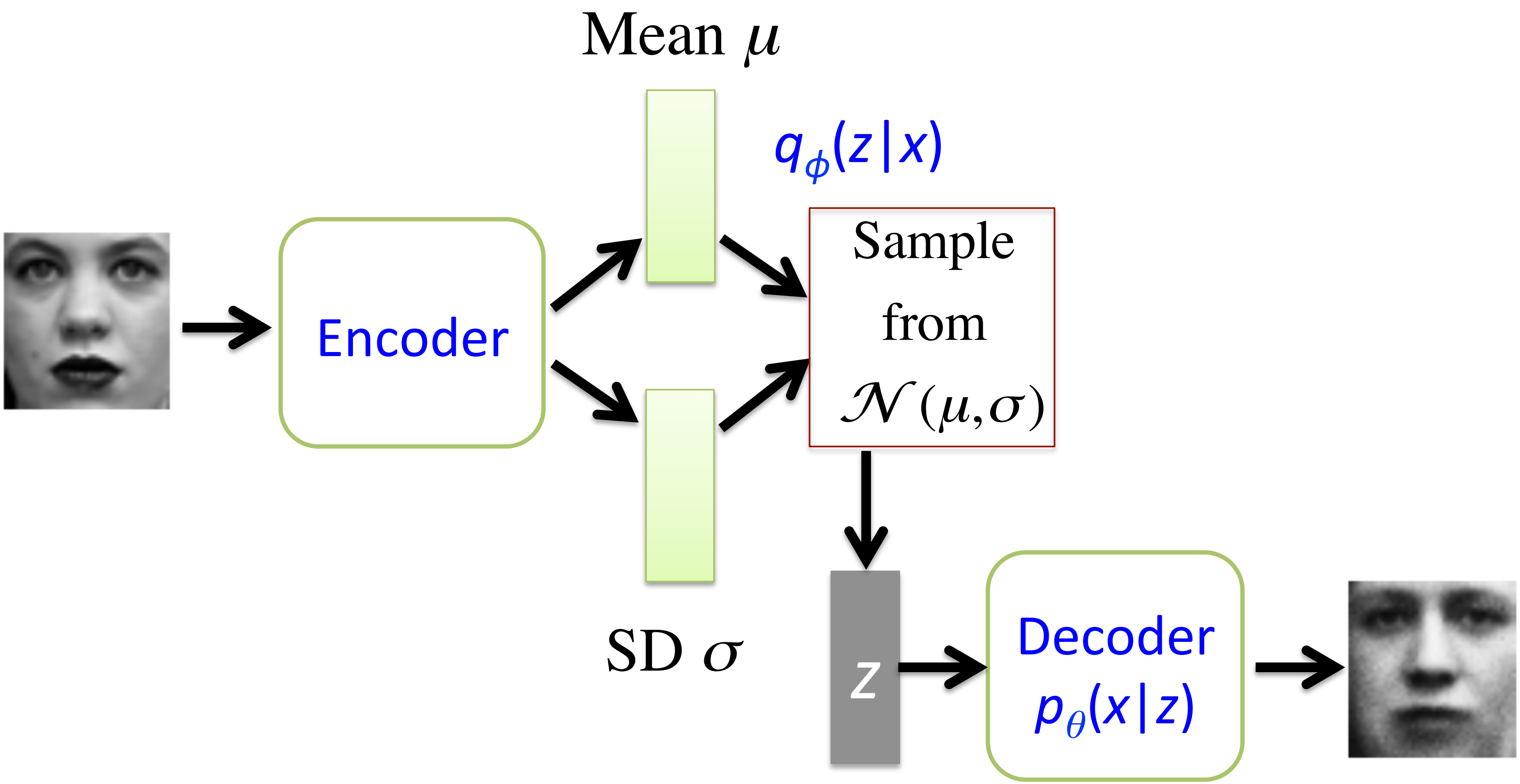
$$p_\theta(x) = \int_z p_\theta(x|z)\boxed{q_\phi(z|x)}dz$$

Proposal distribution:

Likely to produce values of *x* for which *p*(*x*|*z*) is non-zero



VAE Architecture



VAE Loss

Reconstruction Loss

$$-\mathbf{E}_{z\sim q_\phi(z|x)} \log\left(p_\theta\left(x|z\right)\right)$$

VAE Loss

Reconstruction Loss

Proposal distribution should resemble a Gaussian

$$-\mathbf{E}_{z\sim \boxed{q_\phi(z|x)}}\log\left(p_\theta\left(x|z\right)\right) + KL\left(\boxed{q_\phi\left(z|x\right)}\Vert p_\theta(z)\right)$$

VAE Loss

Reconstruction Loss

Proposal distribution should resemble a Gaussian

$$-\mathbf{E}_{z \sim \boxed{q_\phi(z|x)}} \log\left(p_\theta(x|z)\right) + KL\left(\boxed{q_\phi(z|x)} \| p_\theta(z)\right)$$

$$\geq -\log p_\theta(x)$$

Variational upper bound
on loss we care about!

Training VAE

- Apply stochastic gradient descent
- Sampling step not differentiable
- Use a re-parameterization trick
  - Move sampling to input layer, so that the sampling step is independent of the model

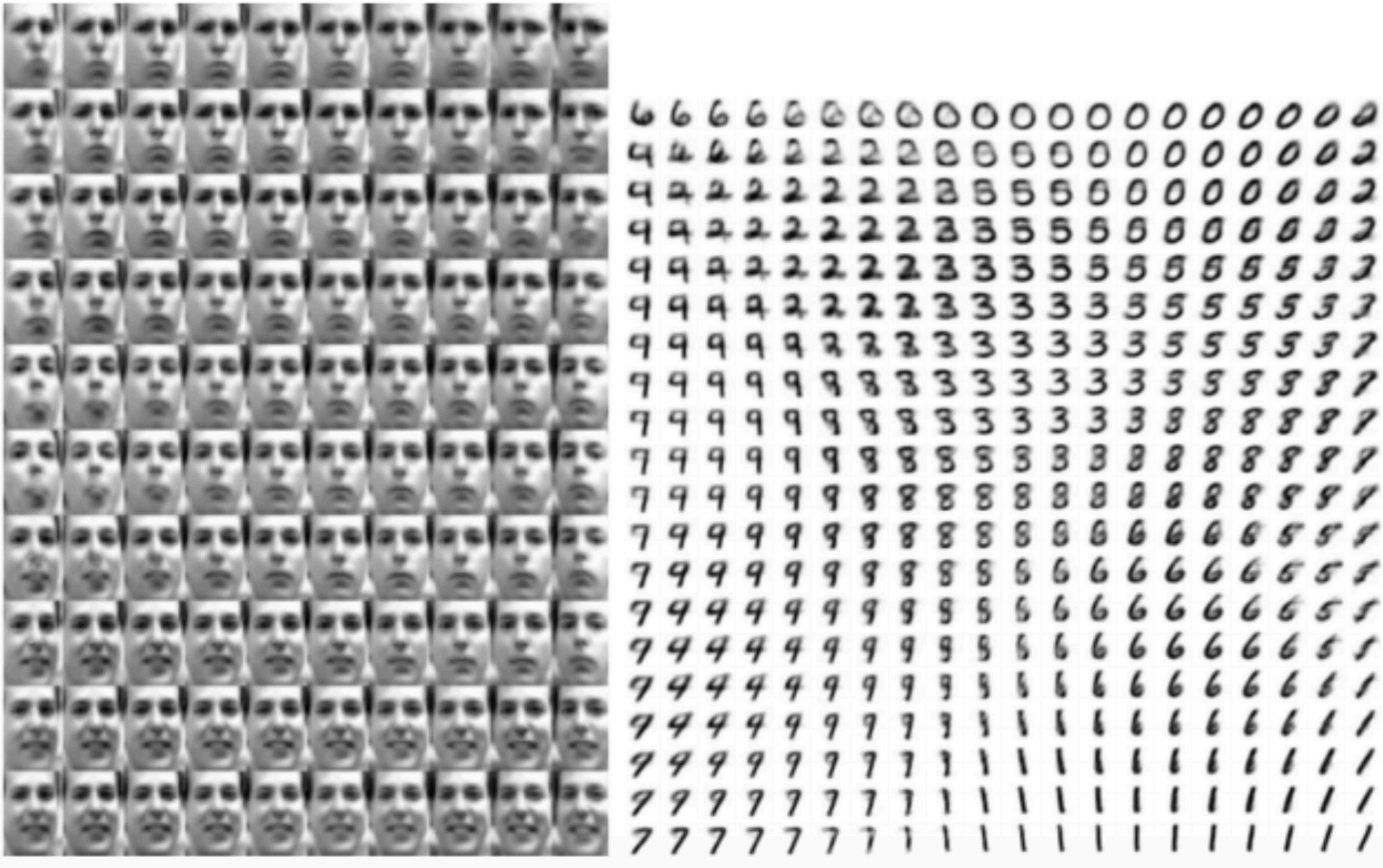
Boltzmann Machines vs. VAE

Pros:

- VAE is easier to train
- Does not require MCMC sampling
- Has theoretical backing
- Applicable to wider range of models

Cons:

- Samples from VAE tend to be blurry



Kingma and Welling, 2014

Outline

- *Explicit Density Models*
  - Boltzmann Machines & Deep Belief Networks (DBN)
  - Variational Auto-encoders (VAE)
- *Implicit Density Models*
  - Generative Adversarial Networks (GAN)

Generative Adversarial Networks

- No Markov chains needed
- No variational approximations needed
- Often regarded as producing better examples

Two Player Game

Generator *G*

Seeks to create samples from p_{data}

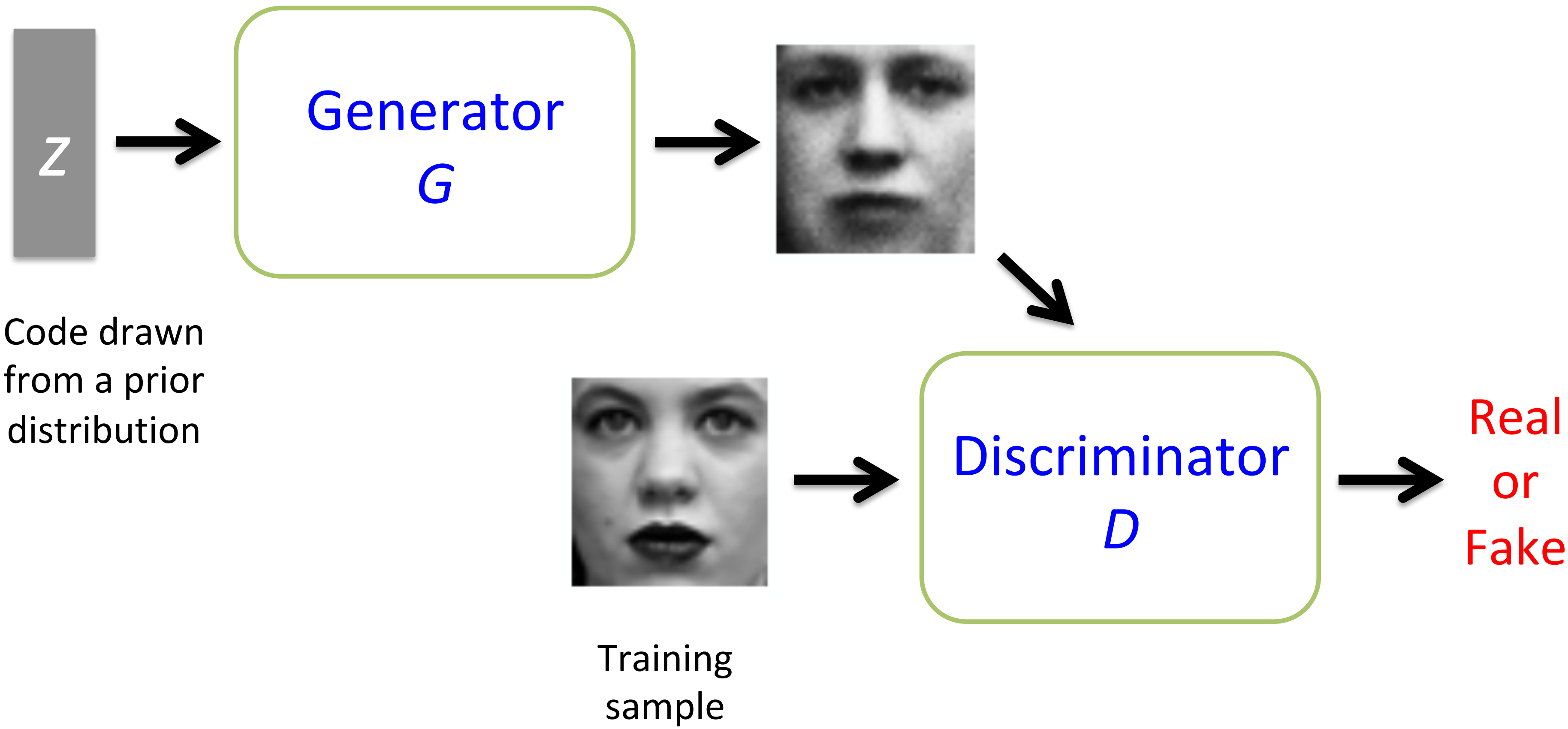
Seeks to trick discriminator into believing that its samples are genuine

Discriminator *D*

Classifies samples as real or fake

Seeks to not get tricked by the generator

GAN Overview



Min-max Cost

Discriminator seeks to predict 1 on training samples

Discriminator wants to predict 0 on samples generated by *G*

$$V(\theta^D, \theta^G) = \boxed{\mathbf{E}_{x\sim p_{data}} \log D(x)} + \boxed{\mathbf{E}_z \log(1 - D(G(z)))}$$

Generator wants *D* to not distinguish between original and generated samples!

$$J^{(G)} = V(\theta^{(D)}, \theta^{(G)}) \qquad J^{(D)} = -V(\theta^{(D)}, \theta^{(G)})$$

Min-max Cost

$$\min_{\theta^G} \max_{\theta^D} J(\theta^G, \theta^D)$$

- Equilibrium is a saddle-point
- Training is difficult in practice

Training GANs

- Sample mini-batch of training images x, and generator codes z
- Update G using back-prop
- Update D using back-prop

Optional: Run k steps of one player for every step of the other player

Radford et al., 2015