Machine Learning & Data Mining

CS/CNS/EE 155

Deep Generative Models

Recap: Generative Models

- Generative models vs. discriminative models
- Generative models: learn P(X,Y) close to P_{gt}(X,Y)

➔ HMM, naive Bayes, latent Dirichlet allocation, …

- Discriminative models: directly learn P(Y|X)

➔ Neural networks, random forests, logistic regression, SVM, …

- Why generative models?

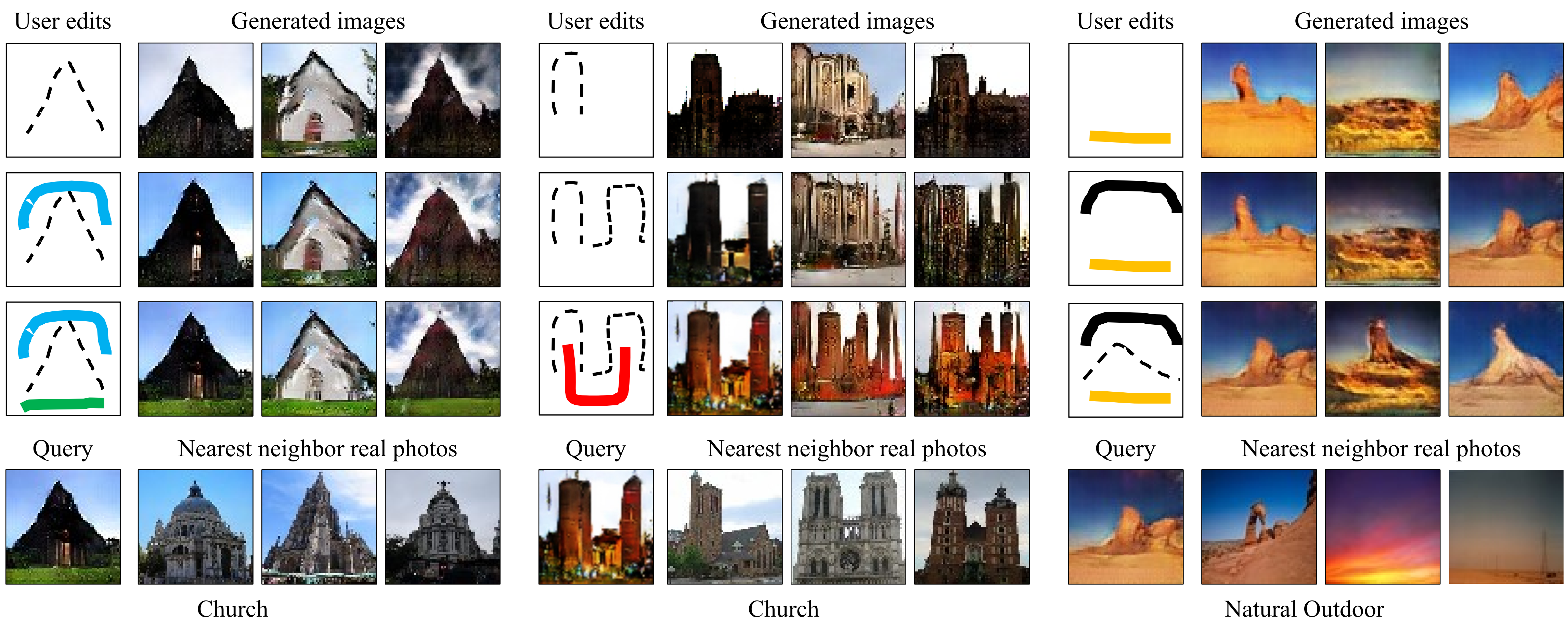
➔ Understanding data by generating them:
What I cannot create, I do not understand. —Richard Feynman

➔ Underlying causal relationship.

➔ Better predictions for future situations.

➔ Deep generative models have shown great potentials.

Interactive Image Generation

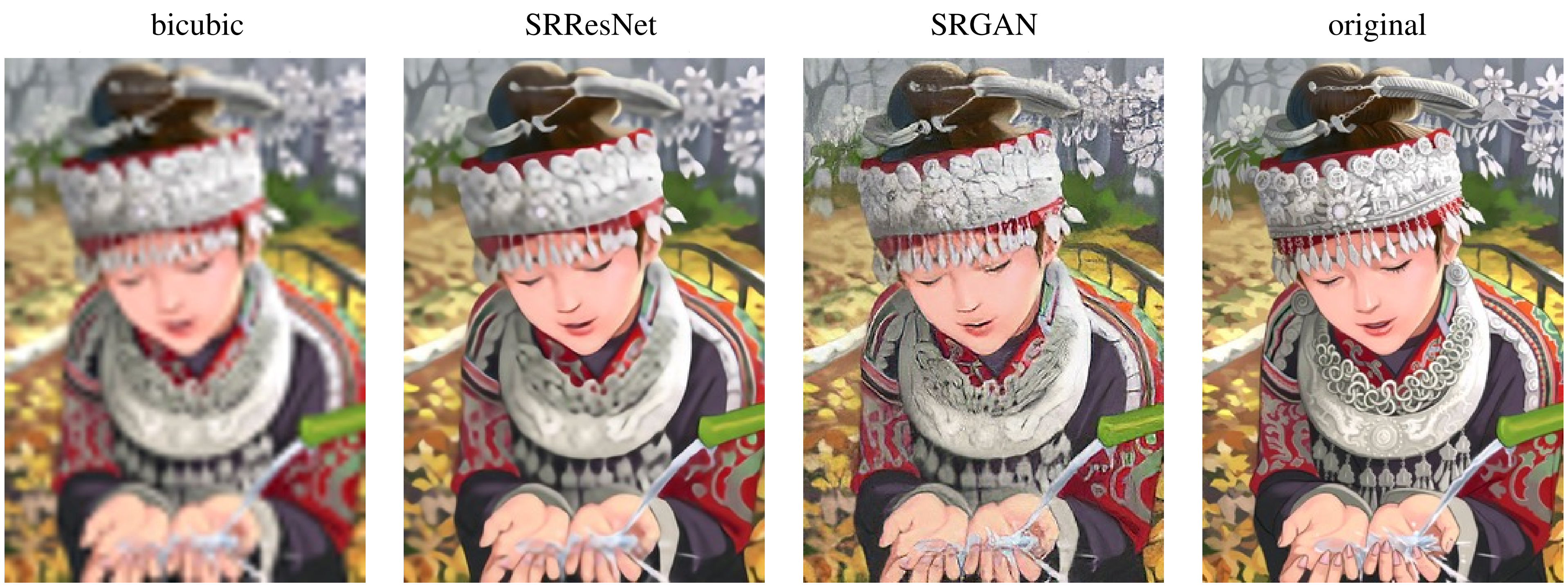


(Zhu et al 2016)

Image To Image Translation

(Isola et al 2016)

Single Image Super-Resolution



(Ledig et al 2016)

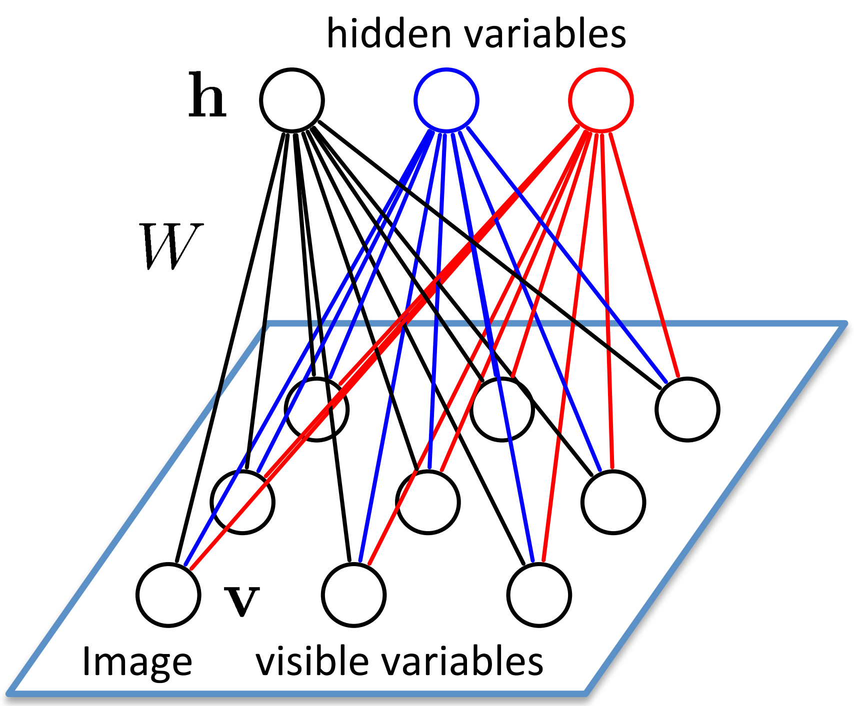
Deep Generative Models

<table>
<tr><td colspan="3">Explicit density</td><td colspan="2">Implicit density</td></tr>
<tr><td>Tractable</td><td colspan="2">Approximate</td><td>Direct</td><td>Markov Chain</td></tr>
<tr><td rowspan="2">Pixel RNN</td><td>Variational</td><td>Markov Chain</td><td rowspan="2">GAN</td><td rowspan="2">GSN</td></tr>
<tr><td>VAE</td><td>RBM,DBM</td></tr>
</table>

(adapted from Goodfellow's slide)

- Restricted Boltzmann Machine (RBM) (Smolensky 1986)
- Deep Boltzmann Machine (DBM) (Salakhutdinov & Hinton 2009)
- Variational Autoencoder (VAE) (Kingma & Welling 2013)
- Generative Adversarial Networks (GAN) (Goodfellow et al 2014)
- Generative Stochastic Networks (GSN) (Bengio et al 2014)
- Pixel RNN (van den Oord et al 2016)

Restricted Boltzmann Machines



(from R. Salakhutdinov's slide)

$$P(\mathrm{v} = \boldsymbol{v}, \mathrm{h} = \boldsymbol{h}) = \frac{1}{Z} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h})\right)$$

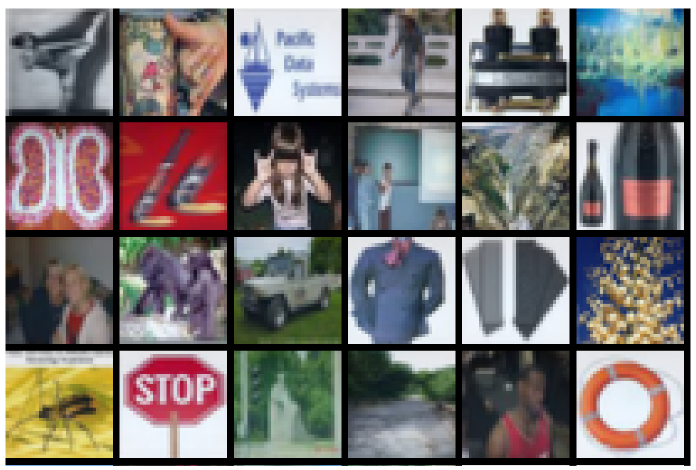
$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}$$

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

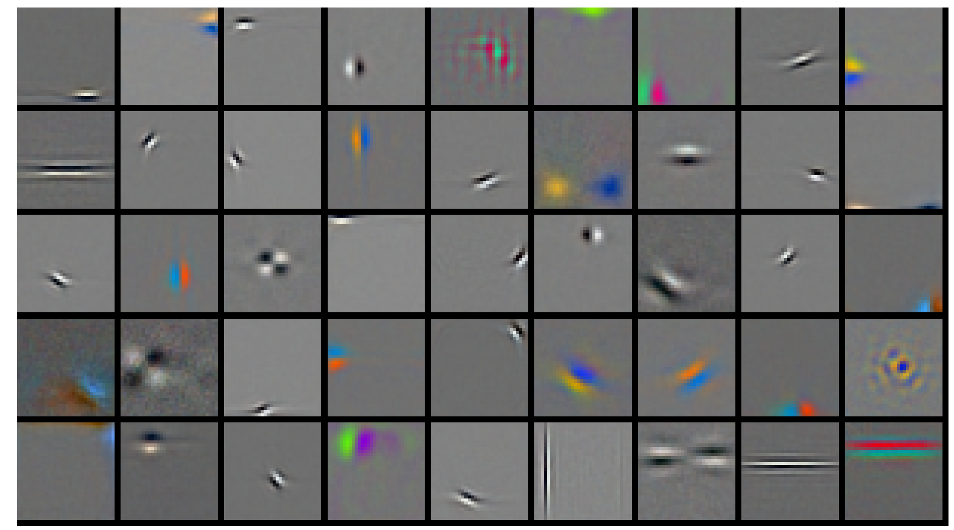
- Undirected model, only bipartite connections between v and h.
- Given N i.i.d. training examples, we want to learn model parameter θ = {W, b, c}.
- Training is done by maximizing log-likelihood: $L(\theta) = \frac{1}{N} \sum_{n=1}^{N} \log P_\theta(\mathbf{v}^{(n)})$
- Partition function is intractable so use Markov chain methods.

Restricted Boltzmann Machines (cont.)

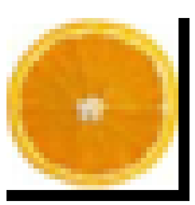
4 million **unlabelled** images



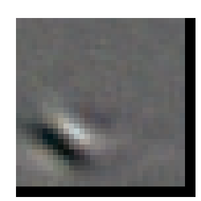
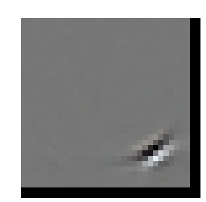
Learned features (out of 10,000)



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



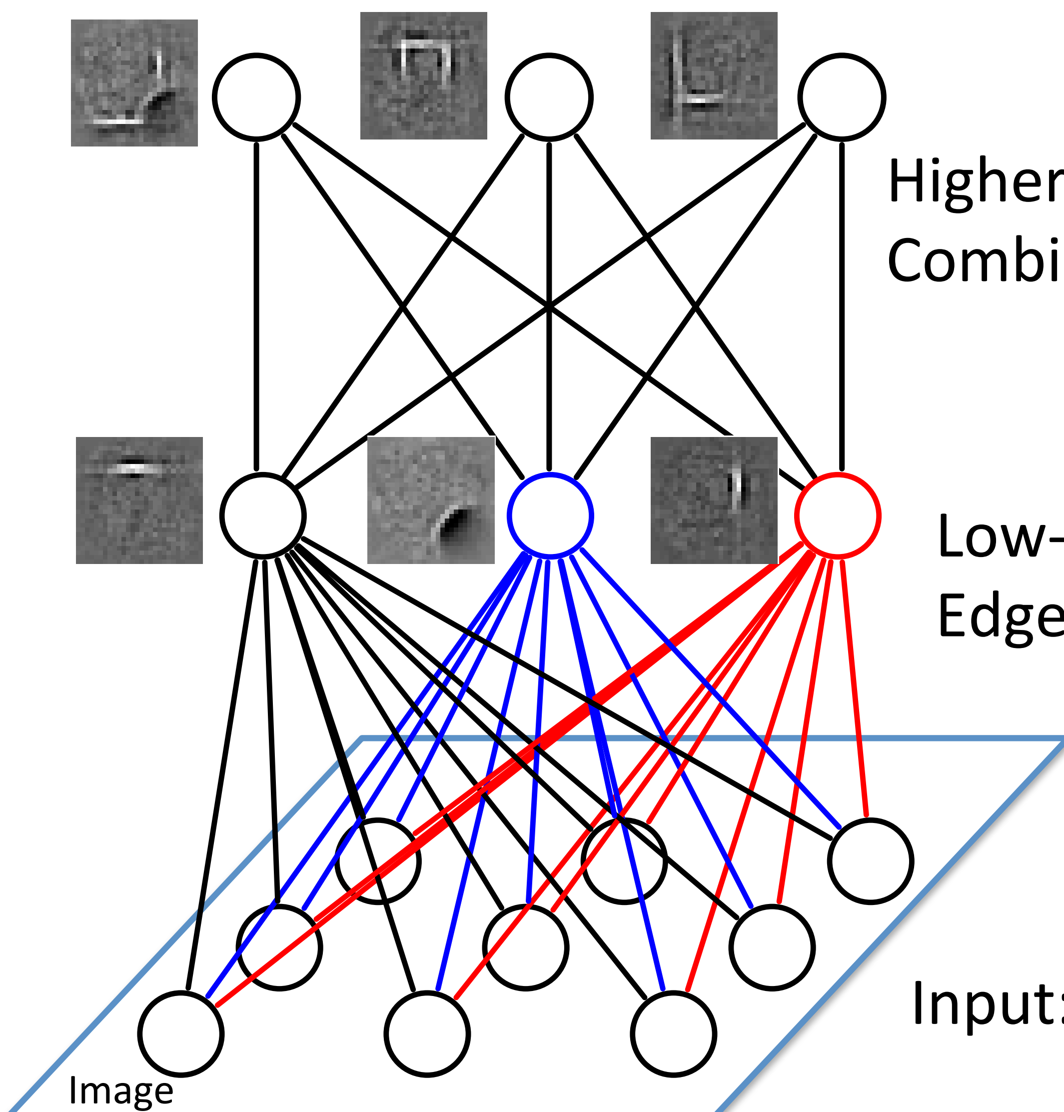
New Image

$= 0.9 *$  $+ 0.8 *$  $+ 0.6 *$  $\ldots$

(from R. Salakhutdinov's slide)

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.

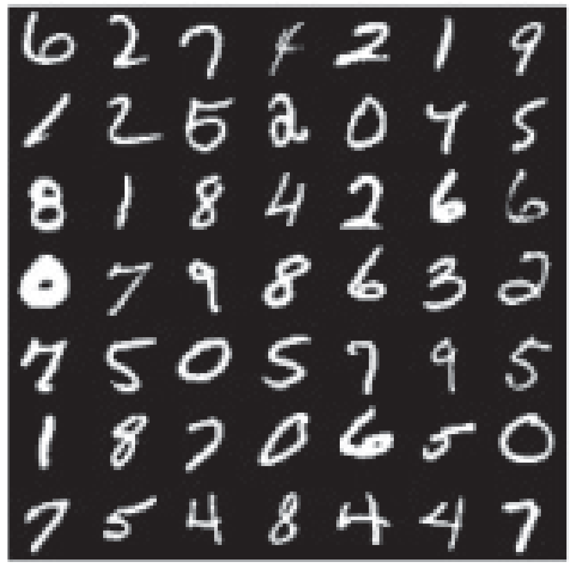
Input: Pixels

(Salakhutdinov 2008, Salakhutdinov & Hinton 2012)

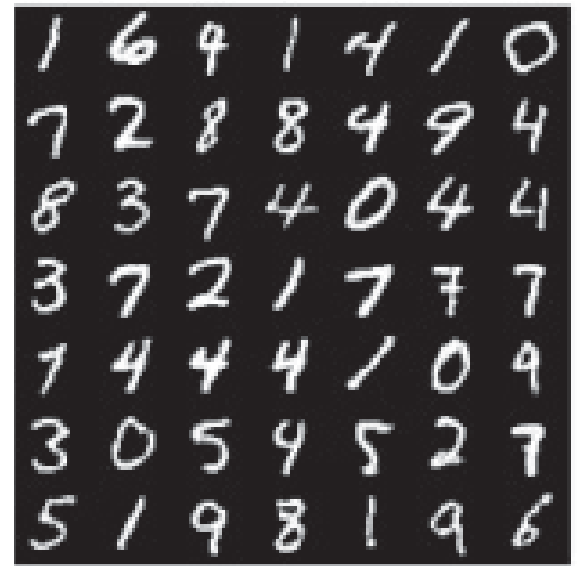
(from R. Salakhutdinov's slide)

Deep Boltzmann Machines (cont.)

Training samples



3-layer BM

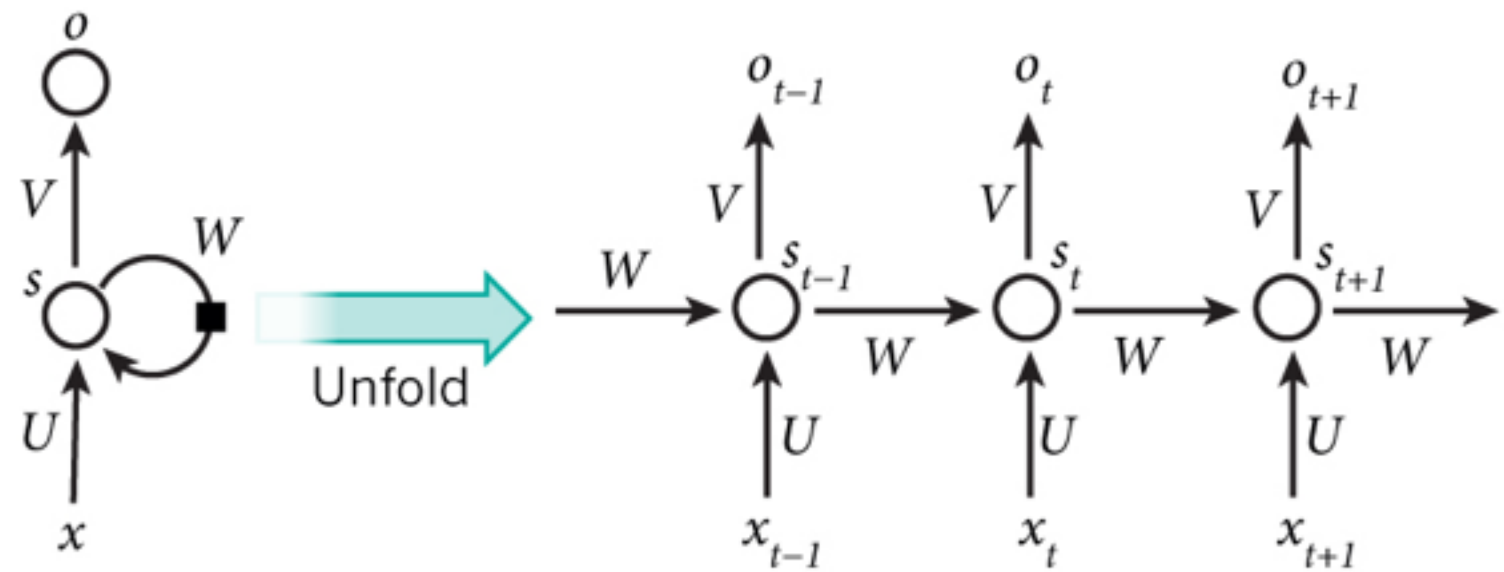


Training Samples

Generated Samples

(Salakhutdinov & Hinton 2012)

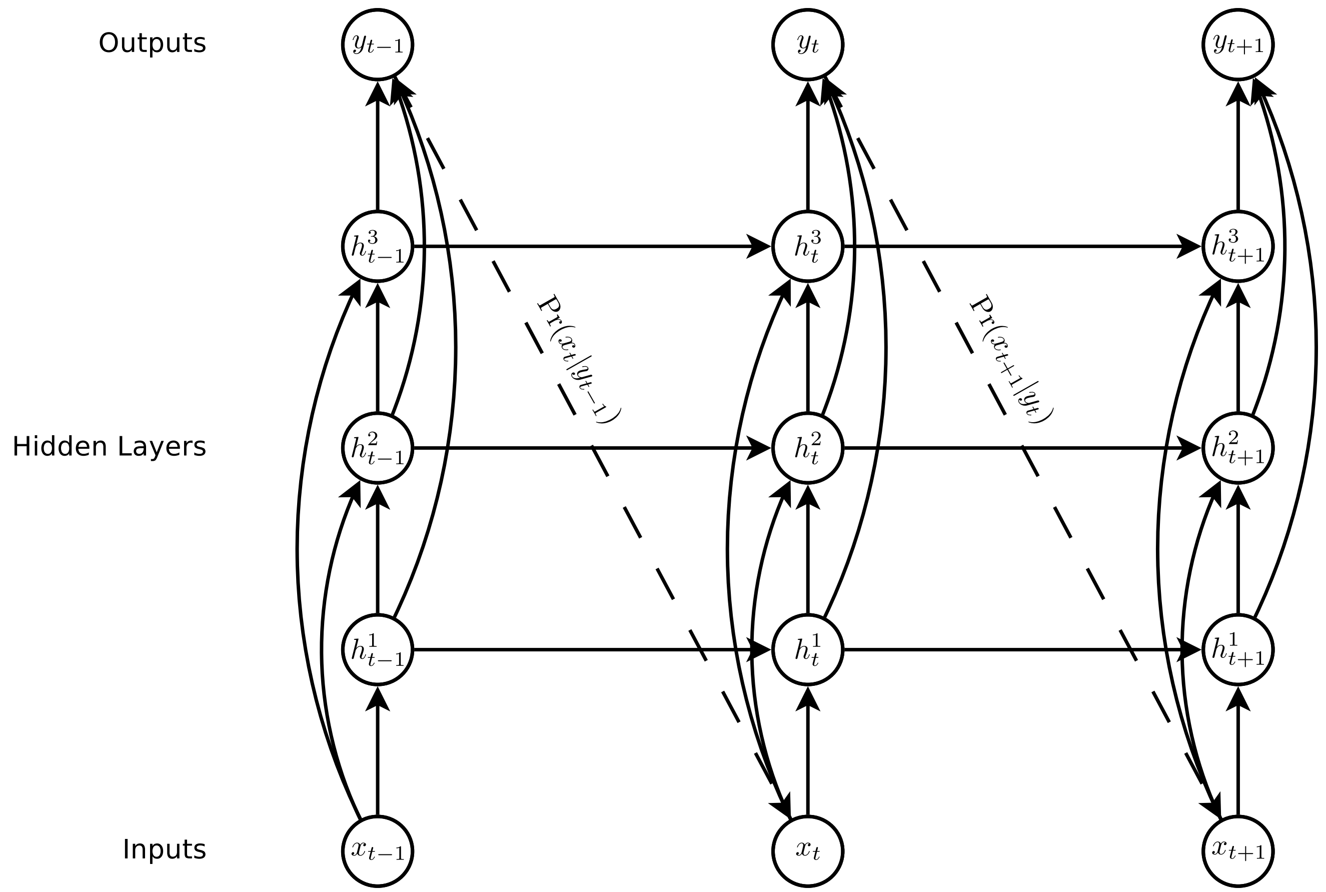
Recurrent Neural Networks (RNN)



(from Denny Britz's tutorial)

- Recurrent Neural Networks use sequential information by having recurrent connections.
- Input $x = \{x_1, x_2, \ldots, x_T\}$, output $o = \{o_1, o_2, \ldots, o_T\}$, hidden state $s = \{s_1, s_2, \ldots, s_T\}$.
- We compute: $s_t = h(Ws_{t-1} + Ux_t + b_s)$, $o_t = Vs_t + b_o$
- We want to learn $\theta = \{W, U, V, b\}$ and training is similar to traditional Neural Networks, but with Backpropagation Through Time (BPTT).
- We can stack more layers as Deep RNN.
- Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber,1997) is popular to solve gradient vanish/exploding issues.

RNN for Sequence Generation



(Graves, 2014)

Input x = $\{x_1, x_2, \ldots, x_T\}$

output y = $\{y_1, y_2, \ldots, y_T\}$

hidden state h = $\{h_1, h_2, \ldots, h_T\}$

$$\hat{y}_t = b_y + \sum_{n=1}^{N} W_{h^n y} h_t^n$$

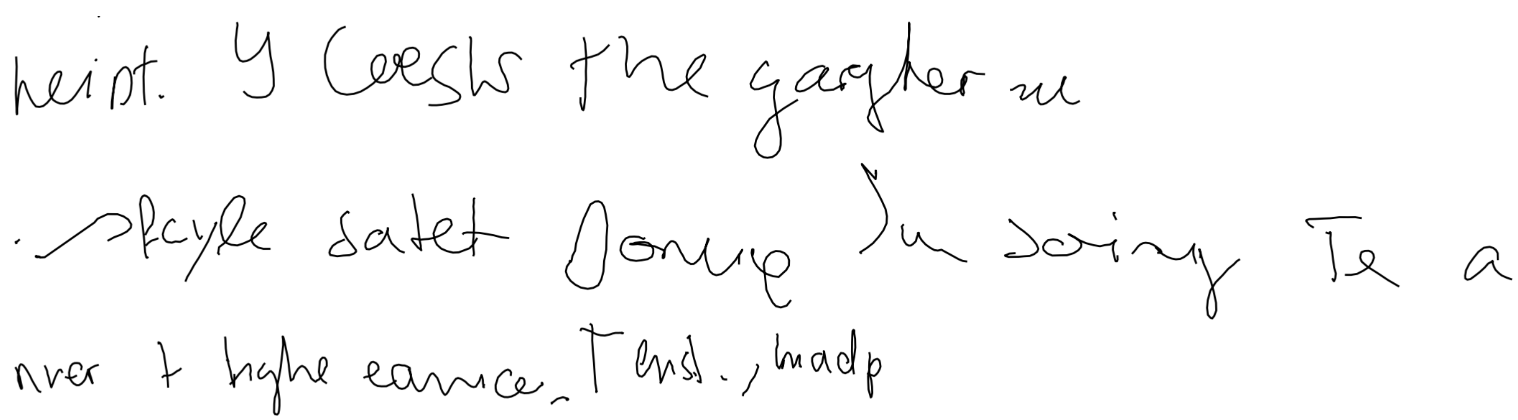
$$y_t = \mathcal{Y}(\hat{y}_t)$$

- Probability of input sequence x is: $\Pr(\mathbf{x}) = \prod_{t=1}^{T} \Pr(x_{t+1}|y_t)$
- Training is done by maximizing: $\mathcal{L}(\mathbf{x}) = -\sum_{t=1}^{T} \log \Pr(x_{t+1}|y_t)$
- In generation, it needs sampling from $\Pr(x_t|y_{t-1})$ and output is fed as next input.
- For synthesis, it can condition on additional inputs (i.e., text sentence).
- It needs to be careful for the form of $\Pr(x_t|y_{t-1})$.

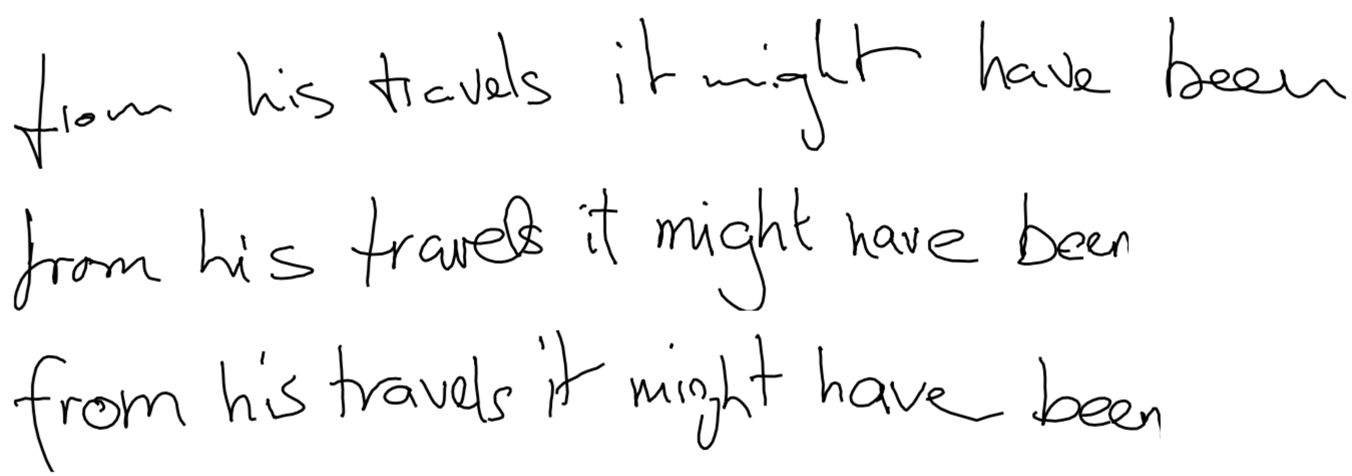
RNN for Sequence Generation (cont.)

```
Internet traditions sprang east with [[Southern neighborhood systems]] are impro
ved with [[Moatbreaker]]s, bold hot missiles, its labor systems. [[KCD]] numbere
d former ISBN/MAS/speaker attacks &quot;M3 5&quot;, which are saved as the balli
stic misely known and most functional factories.  Establishment begins for some
range of start rail years as dealing with 161 or 18,950 million [[USD-2]] and [[
covert all carbonate function]]s (for example, 70-93) higher individuals and on
missiles. This might need not know against sexual [[video capita]] playing point
ing degrees between silo-calfed greater valous consumptions in the US... header
can be seen in [[collectivist]].
```

Generated wikipedia data



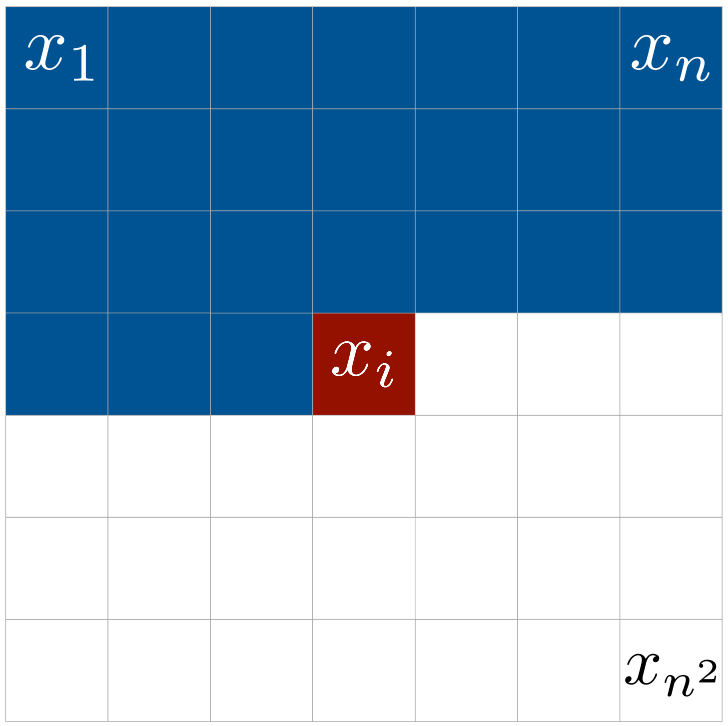
Generated handwriting

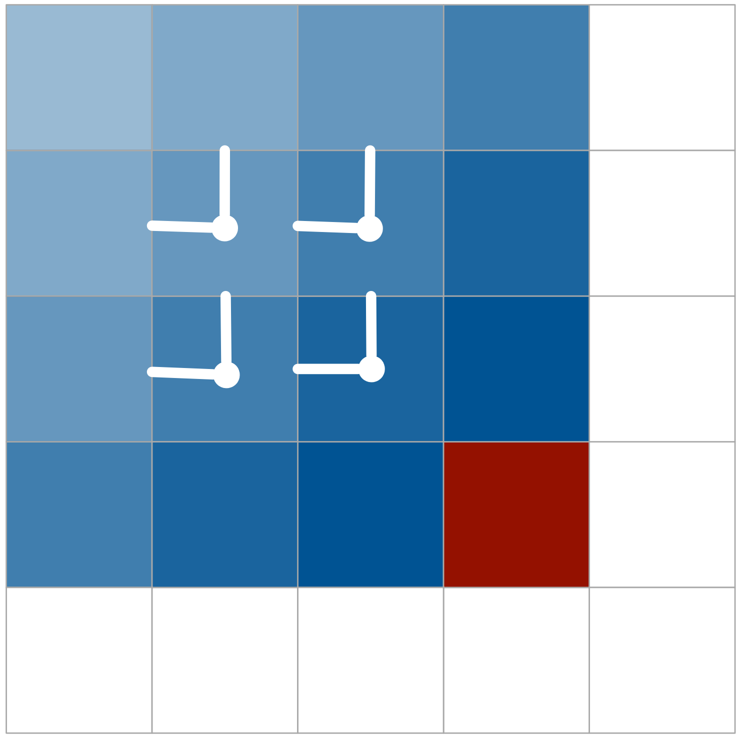


Synthesized handwriting

(Graves, 2014)

Pixel RNN (van den Oord et al 2016)

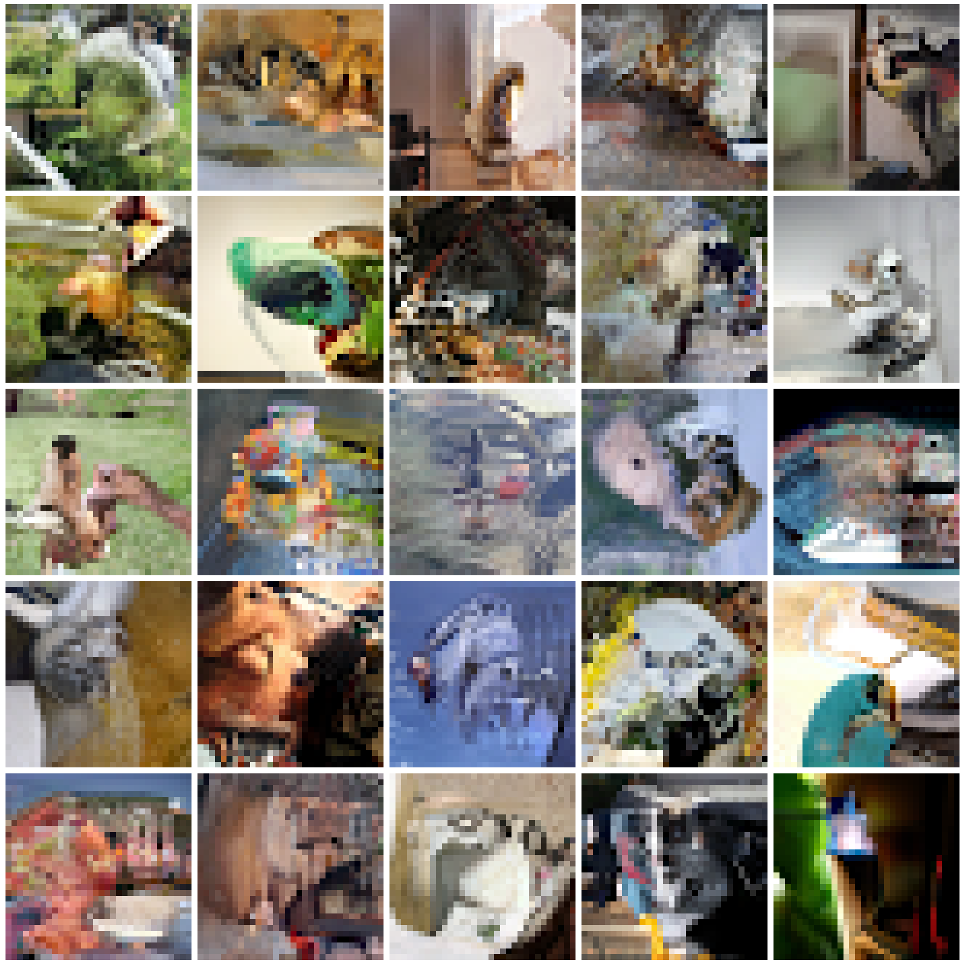




- Probability of an image x of n×n pixels: $p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, ..., x_{i-1})$
- Image is generated sequentially.

Pixel RNN (van den Oord et al 2016)

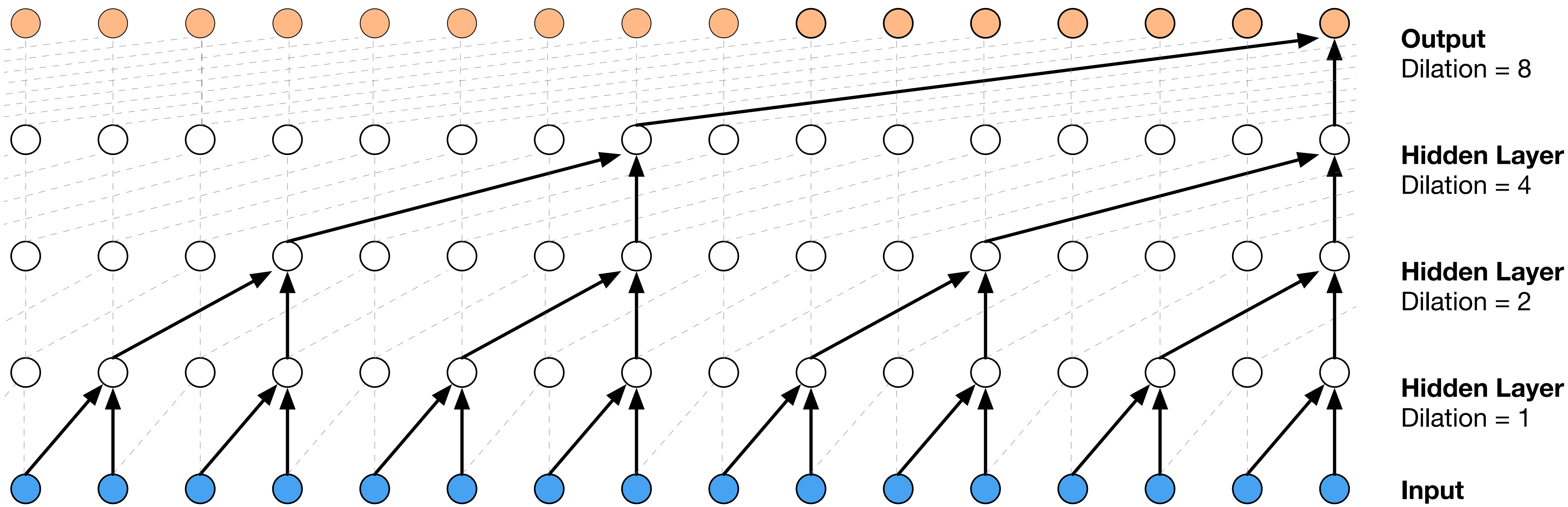
Image completion



Generated samples

- They achieved the best log-likelihood and training is simple.
- However, sampling is inefficient and do not provide simple low-dimensional codes for images.

WaveNet (van den Oord et al 2016)

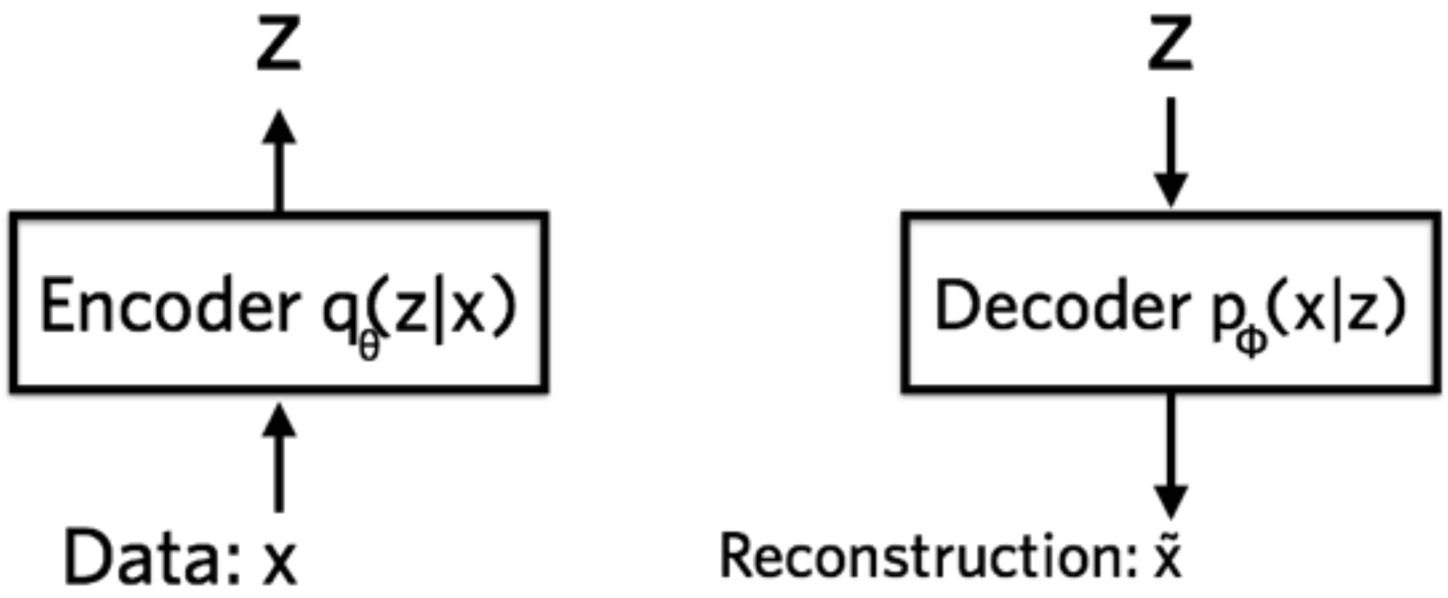


- We model joint distribution of a wave form x = {x_1,x_2,…,x_T }:

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1})$$

- Input: {x_1,x_2,…,x_{t-1}}, output: softmax unit for the next x_t
- Network structure: stacks of convolutional layers.
- Training is done by maximizing log-likelihood.
- It takes two minutes to synthesize one second of audio.

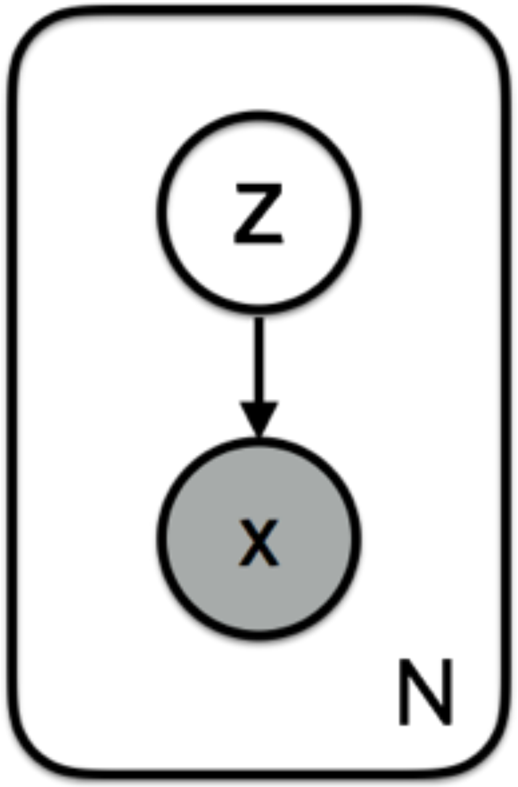
Variational Autoencoder



(from Jaan Altosaar's tutorial)

- Proposed by (Kingma and Welling, 2013).
- Encoder (inference nets) takes data x as input and outputs parameters to $q_\theta(z|x)$.
- Decoder (generative nets) takes latent variable z and outputs parameters to $p_\phi(x|z)$.

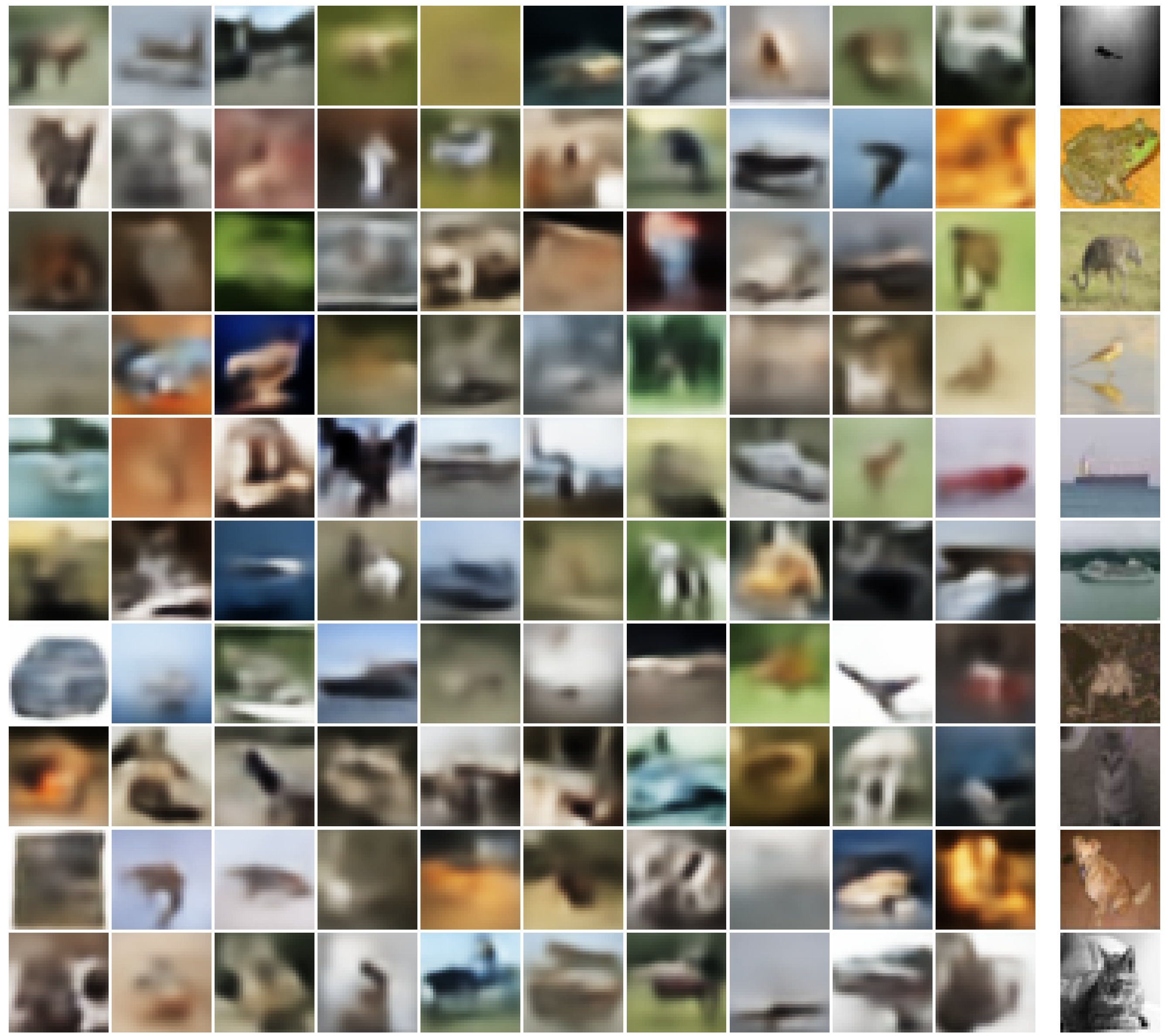
Variational Autoencoder (cont.)



- We have joint distribution p(x,z)=p(x|z)p(z).
- Draw $z_i \sim$ p(z) and draw datapoint $x_i \sim$ p(x|z).
- For inference, p(z|x)=p(x|z)p(z)/p(x), and p(x)=∫p(x|z)p(z)dz but it is intractable.
- We use q_λ(z|x) to approximate p(z|x).

- We want q_λ(z|x) = argmin_λ KL(q_λ(z|x)||p(z|x)).
- log p(x) = E_q[log p(x,z)] - E_q[log q_λ(z|x)] + KL(q_λ(z|x)||p(z|x)).
- E_q[log p(x,z)] - E_q[log q_λ(z|x)] is a lower bound and we want to maximize it.
- For each datapoint x_i, it becomes
$$E_{q_\theta(z|x_i)}[\log p_\phi(x_i|z)] - KL(q_\theta(z|x_i)||p(z)) = -l_i(\theta, \phi).$$
- Training is done by backpropagation.

Variational Autoencoder (cont.)



(Gregor et al., 2015)

- But the generated images are blurry.

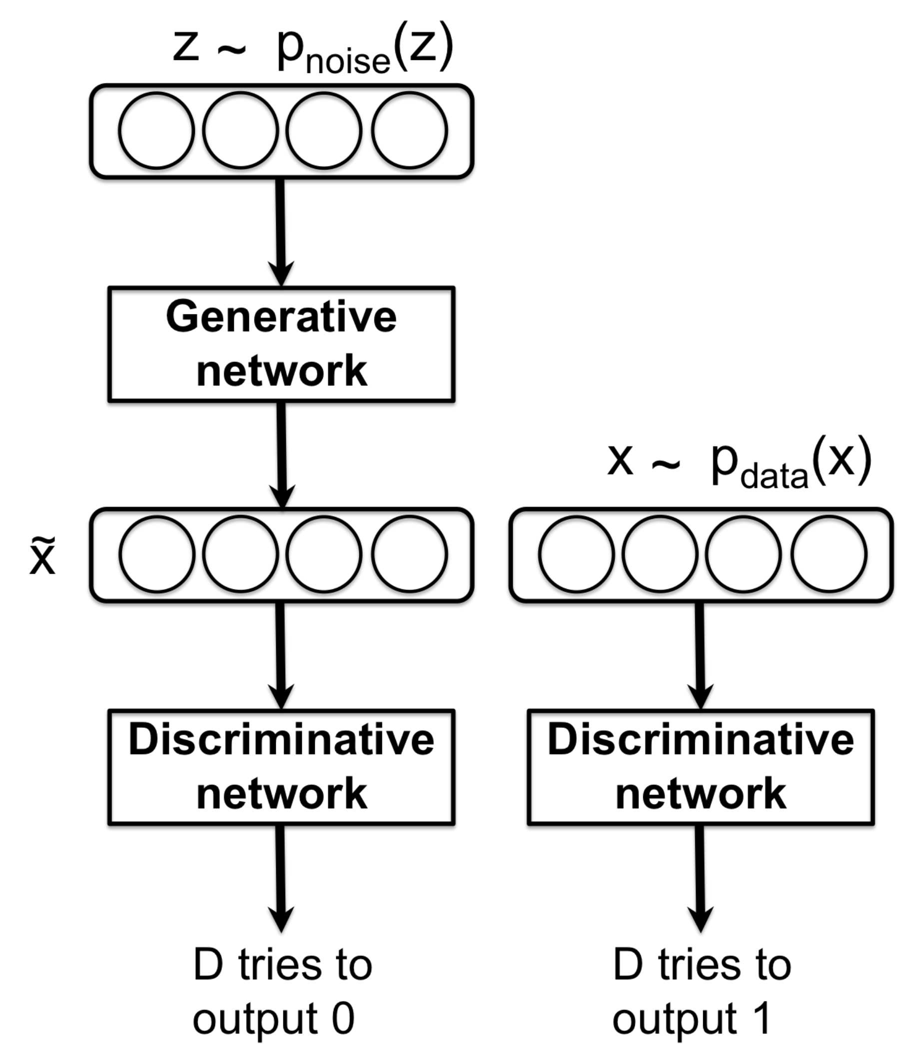
Generative Adversarial Networks (GAN)

- Why do we need p(x)? Just learn to sample directly.

- Minimax game between two players.

➔ Discriminative model D: distinguishes between real and fake samples generated from G.

➔ Generative model G: try to fool D by generating fake samples.

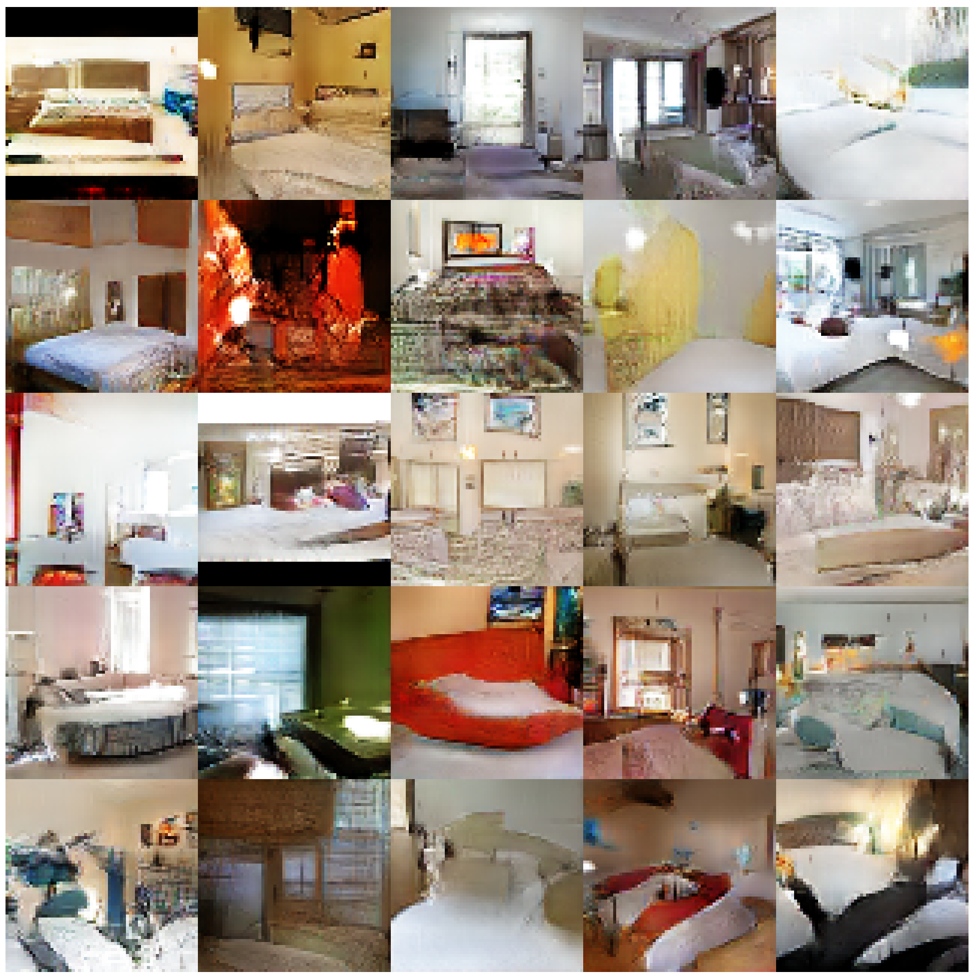


(from Emily Denton's slides)

- Optimize w.r.t. D and G

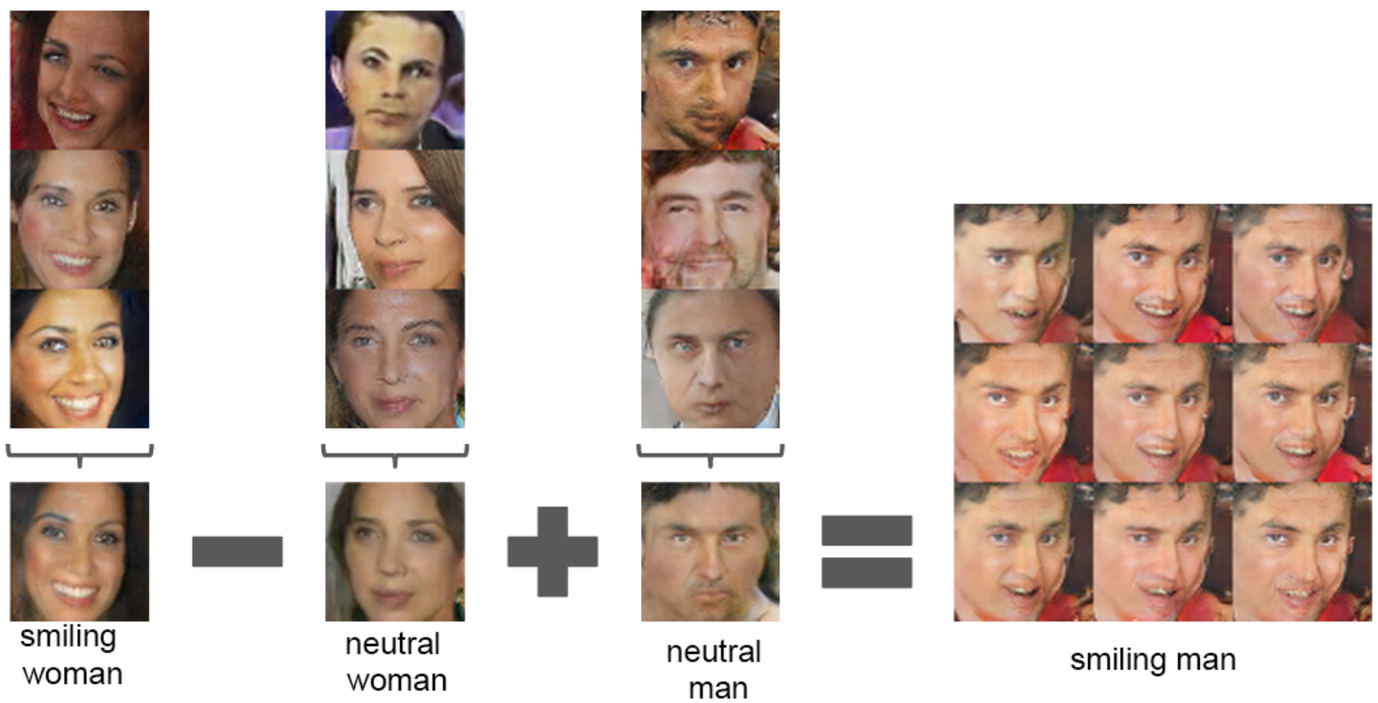
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1-D(G(\boldsymbol{z})))].$$

Image Generation



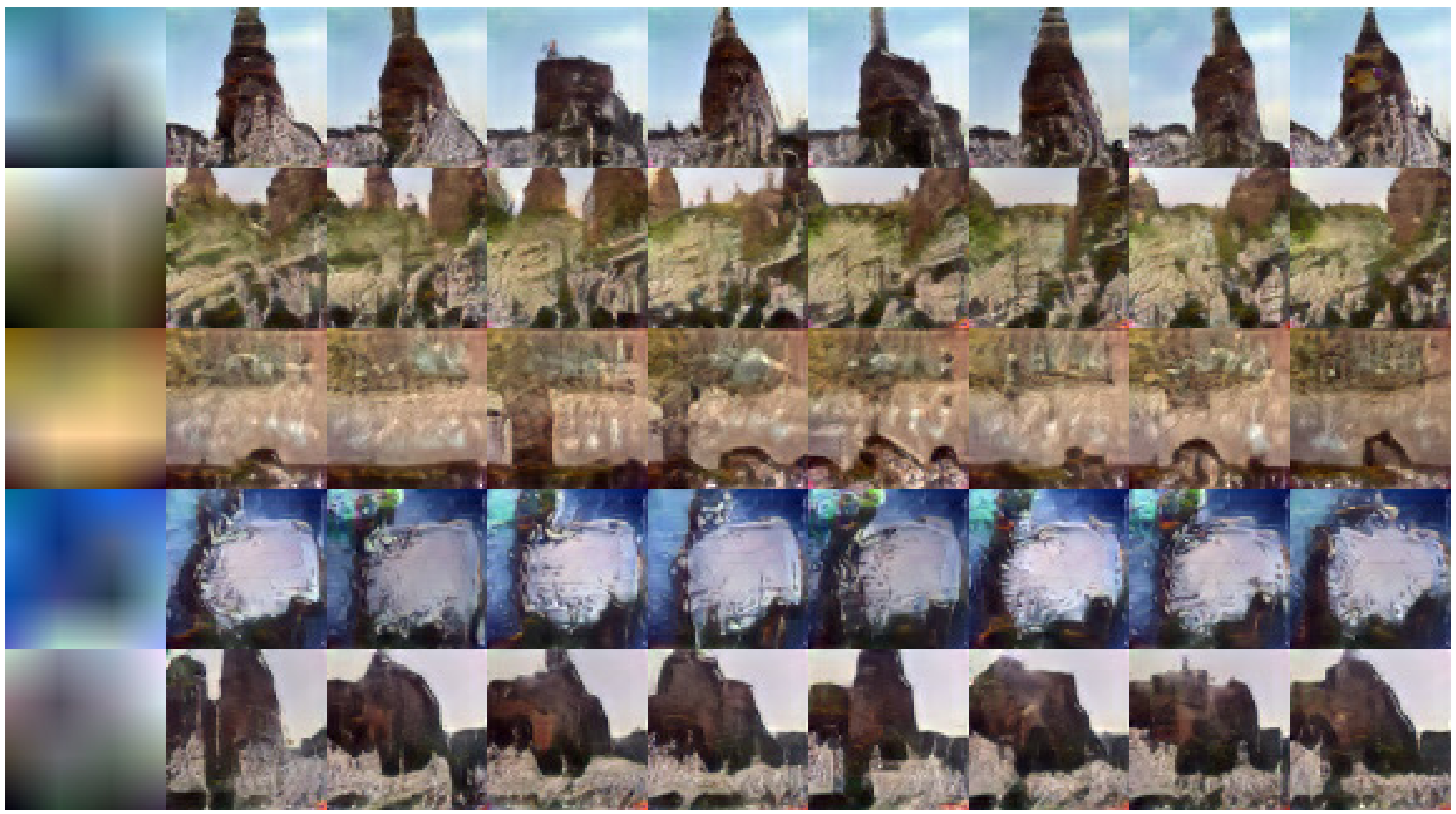
(Radford et al 2016)

We have latent codes z

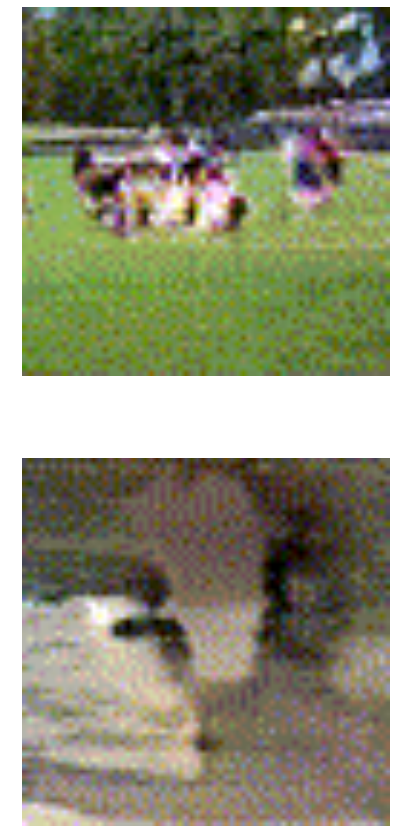


(Radford et al 2016)

Other Extensions to GAN

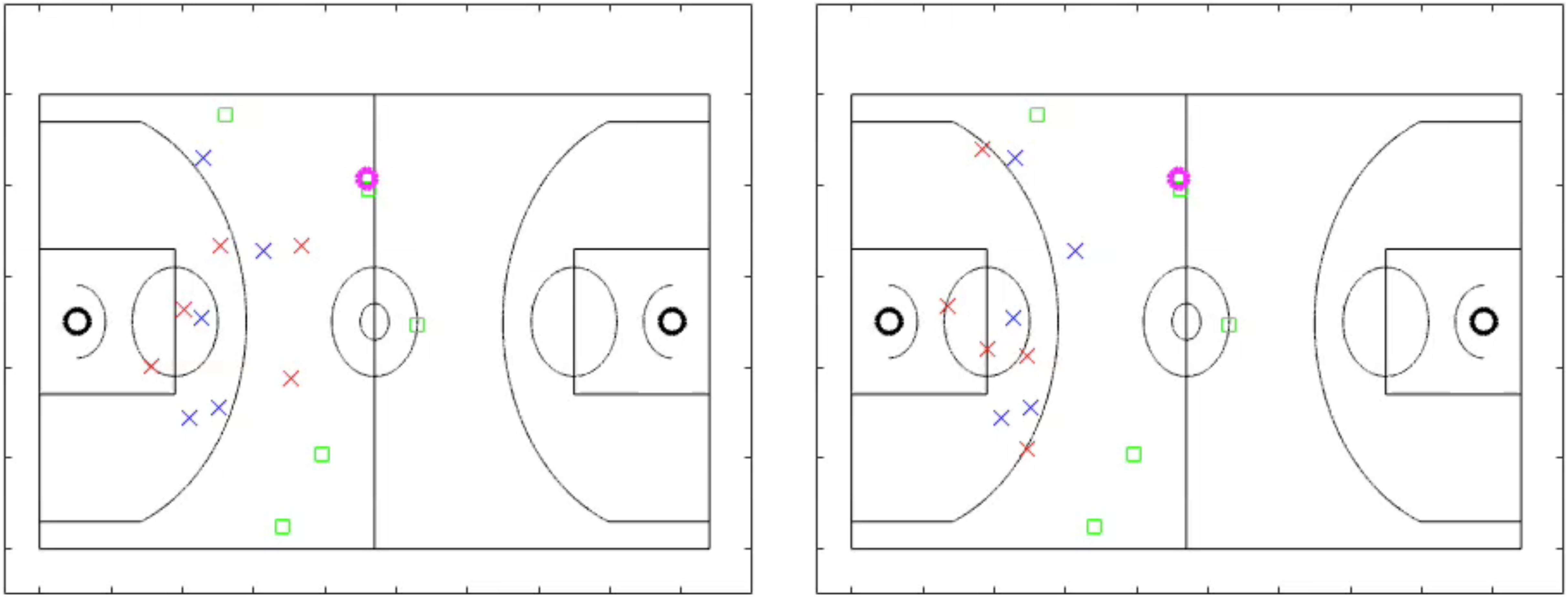


Conditional GAN
(Denton et al 2015)



Video generation
(Vondrick et al 2016)

GAN for Spatio-temporal data



NBA basketball data

GAN (cont.)

- We do not need Markov chains.
- We may use latent codes z to control generated samples.
- Generates most crisp images.
- Training is done by backpropagation but difficult and usually unstable.
- There is no log-likelihood to measure.

Evaluating Generative Models

- How to measure model qualities?
- Log-likelihood, Parzen window estimates, and visual fidelity of generated samples.
- But they are largely independent of each other when the data is high-dimensional (Theis et al 2016).

Conclusion

- There are great potentials for deep generative models.
- Learning to generate data may be best way to understand them.
- GAN seems to generate best image samples.