DEEP PROBABILISTIC MODELS

LECTURE 1 - INTRODUCTION

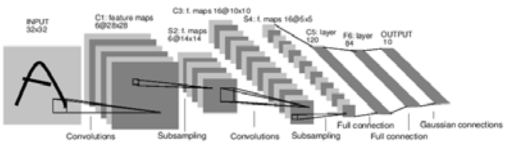
DEEP PROBABILISTIC MODELS

implemented using deep neural networks

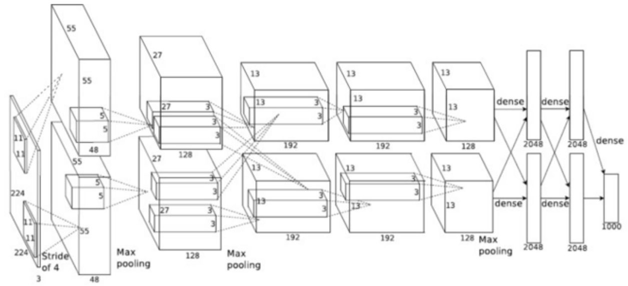
expressed using probability & statistics

an approximation of a real phenomenon

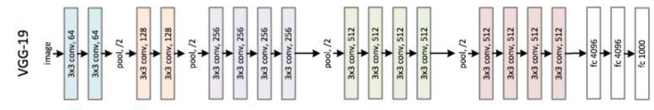
convolutional neural networks for *classification*



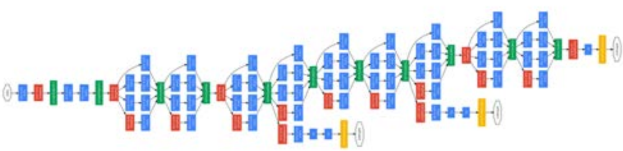
LeNet



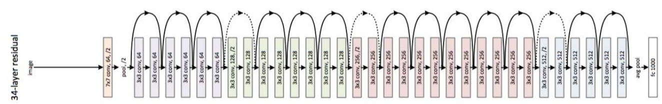
AlexNet



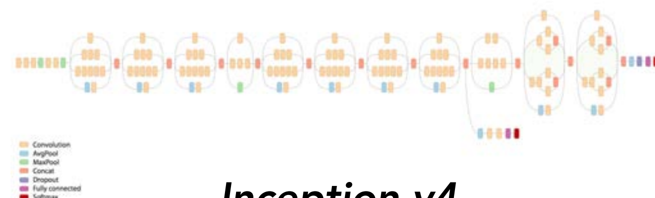
VGG



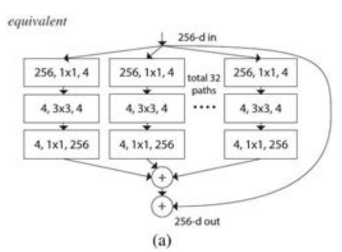
GoogLeNet



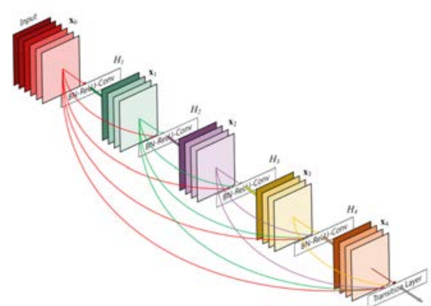
ResNet



Inception v4

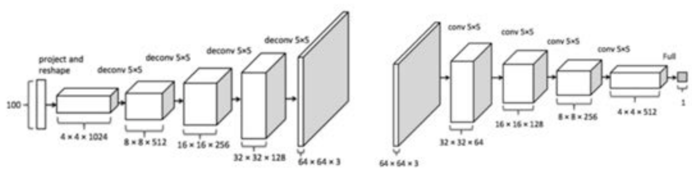


ResNeXt

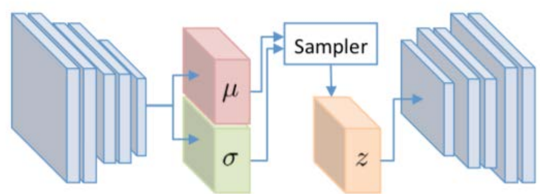


DenseNet

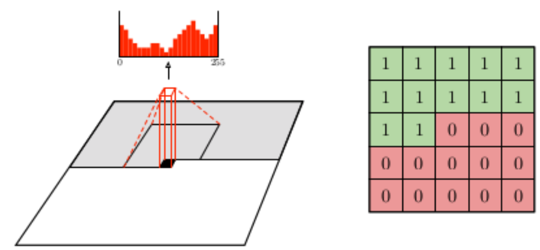
convolutional models for *image generation*



DC-GAN



convolutional VAE



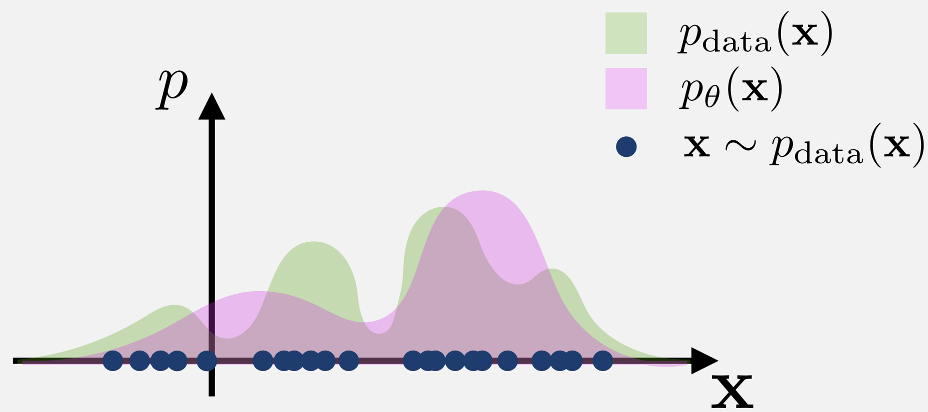
Pixel CNN

modeling the data distribution

data: $p_{\text{data}}(\mathbf{x})$

model: $p_\theta(\mathbf{x})$

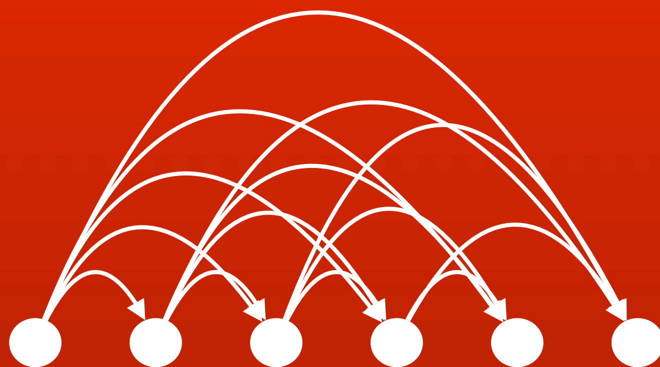
parameters: θ



maximum likelihood estimation

find the model that assigns the *maximum likelihood* to the data

$$
\begin{aligned}
\theta^* &= \arg\min_\theta \; D_{KL}(p_{\text{data}}(\mathbf{x}) || p_\theta(\mathbf{x})) \\
&= \arg\min_\theta \; \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[\log p_{\text{data}}(\mathbf{x}) - \log p_\theta(\mathbf{x}) \right] \\
&= \arg\max_\theta \; \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[\log p_\theta(\mathbf{x}) \right] \approx \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)})
\end{aligned}
$$

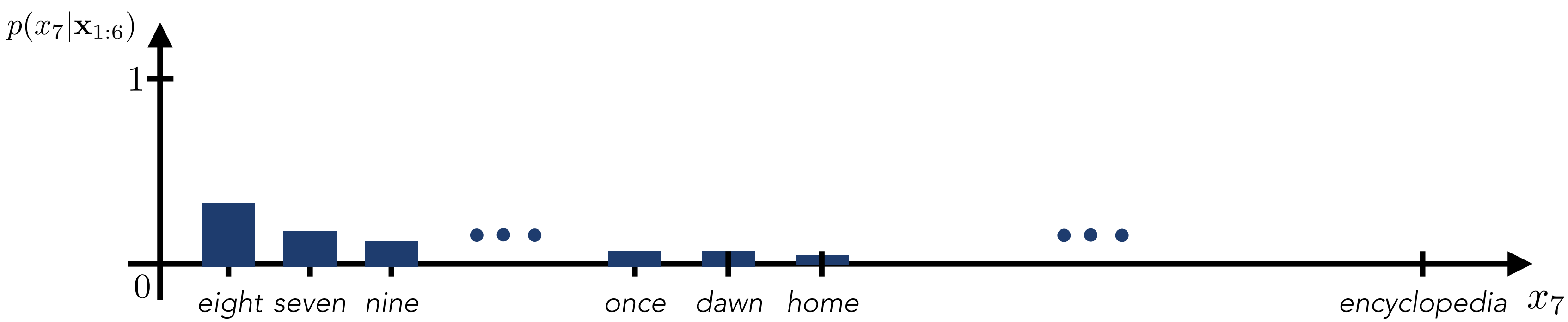


autoregressive models

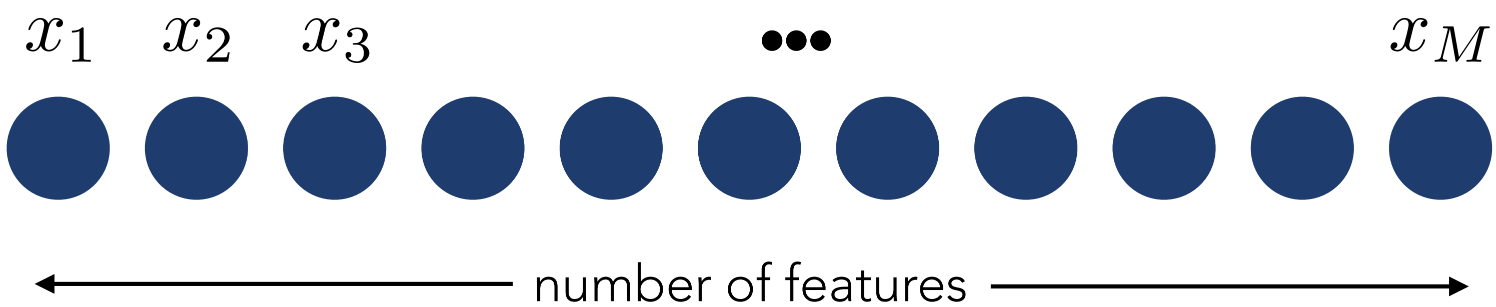
conditional probability distributions

This	*morning*	*I*	*woke*	*up*	*at*	
x_1	x_2	x_3	x_4	x_5	x_6	x_7

What is $p(x_7|\mathbf{x}_{1:6})$?



a data example



$$p(\mathbf{x}) = p(x_1, x_2, \ldots, x_M)$$

chain rule of probability

split the joint distribution into a product of conditional distributions

$$p(\mathbf{x}) = p(x_1, x_2, \ldots, x_M)$$

$$p(a|b) = \frac{p(a,b)}{p(b)} \longrightarrow p(a,b) = p(a|b)p(b)$$

definition of conditional probability

recursively apply to $p(x_1, x_2, \ldots, x_M)$:

$$p(x_1, x_2, \ldots, x_M) = p(x_1)p(x_2, \ldots, x_M|x_1)$$

$$\vdots$$

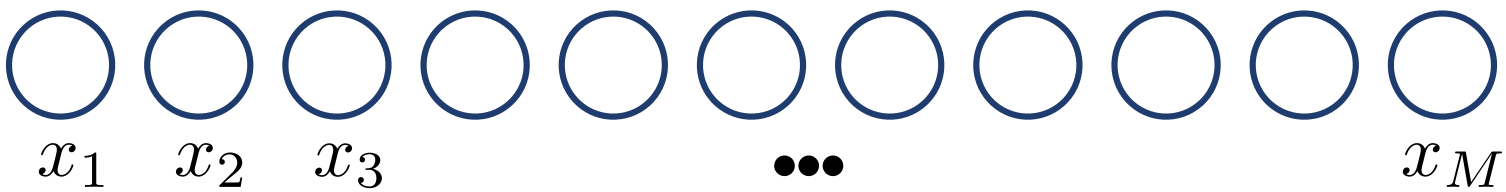
$$= p(x_1)p(x_2|x_1)\ldots p(x_M|x_1, \ldots, x_{M-1})$$

$$\boxed{p(x_1, \ldots, x_M) = \prod_{j=1}^{M} p(x_j|x_1, \ldots, x_{j-1})}$$

note: conditioning order is arbitrary

model the conditional distributions of the data

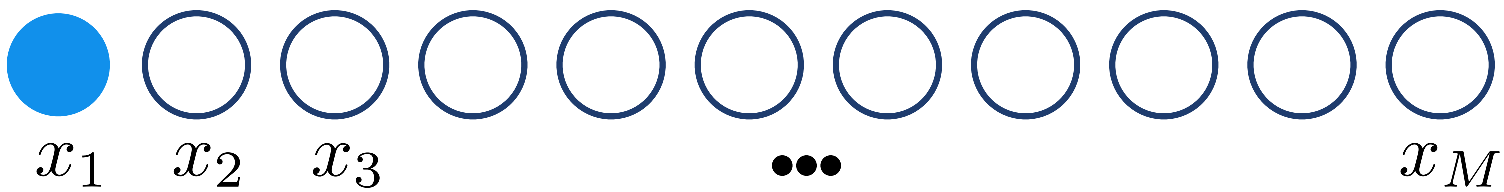
learn to ***auto-regress*** each value



model the conditional distributions of the data

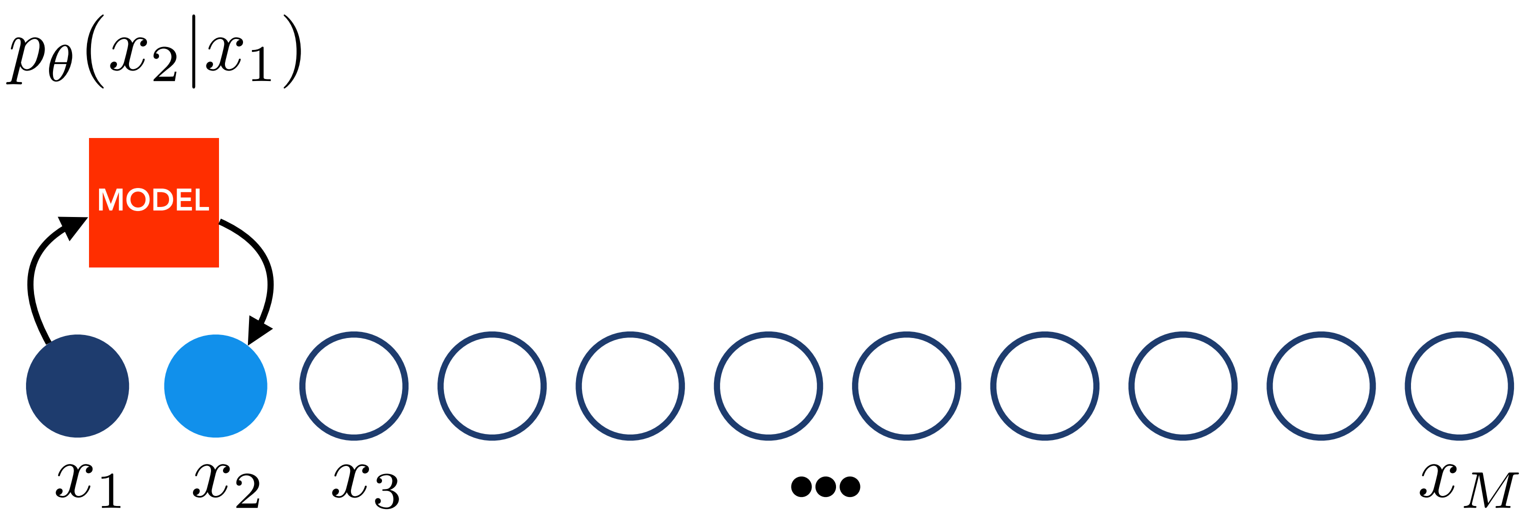
learn to ***auto-regress*** each value

$$p_\theta(x_1)$$



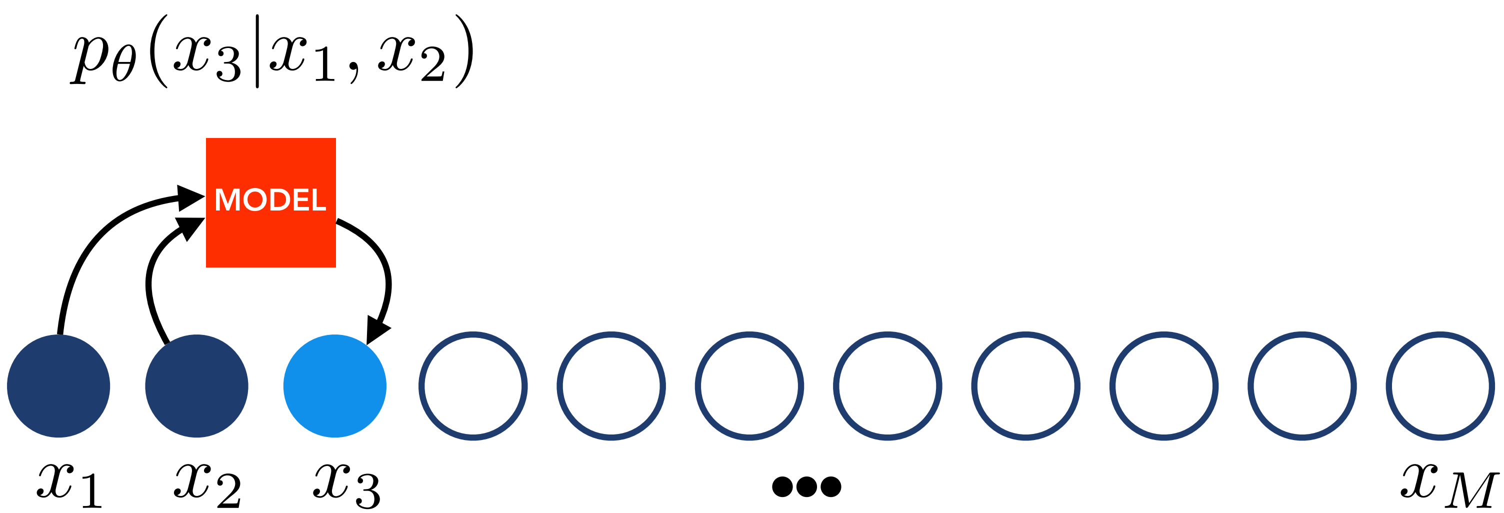
model the conditional distributions of the data

learn to ***auto-regress*** each value



model the conditional distributions of the data

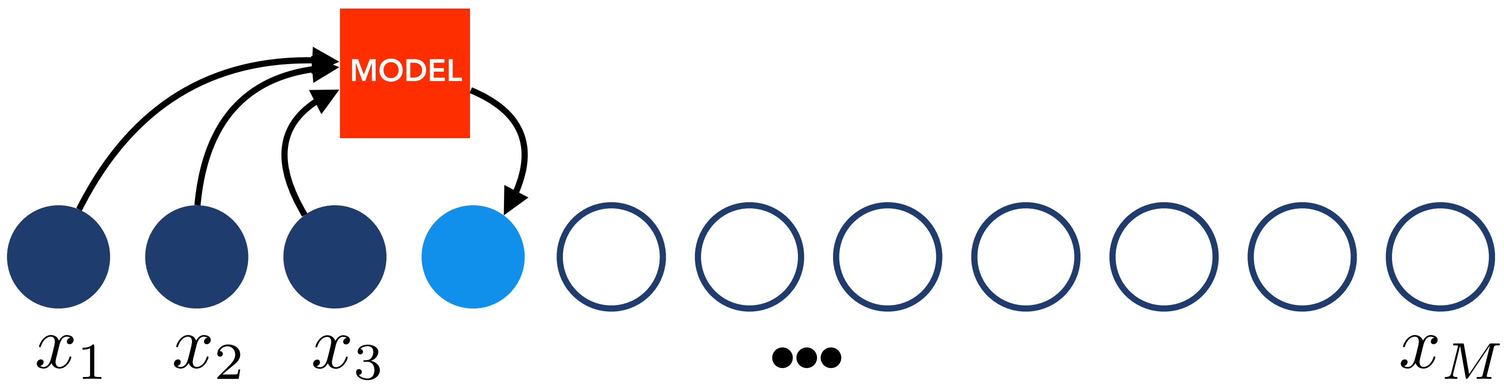
learn to ***auto-regress*** each value



model the conditional distributions of the data

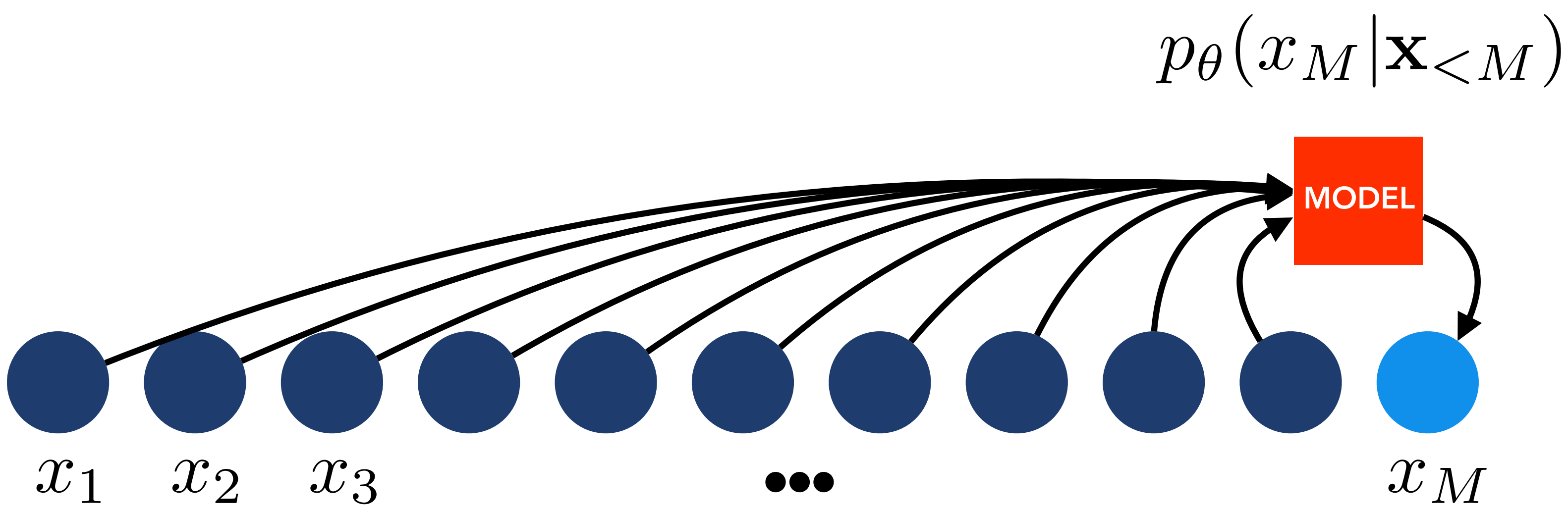
learn to ***auto-regress*** each value

$$p_\theta(x_4 | x_1, x_2, x_3)$$



model the conditional distributions of the data

learn to ***auto-regress*** each value



maximum likelihood estimation

maximize the log-likelihood (under the model) of the true data examples

$$\theta^* = \arg\max_\theta \; \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[\log p_\theta(\mathbf{x})\right] \approx \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)})$$

for auto-regressive models:

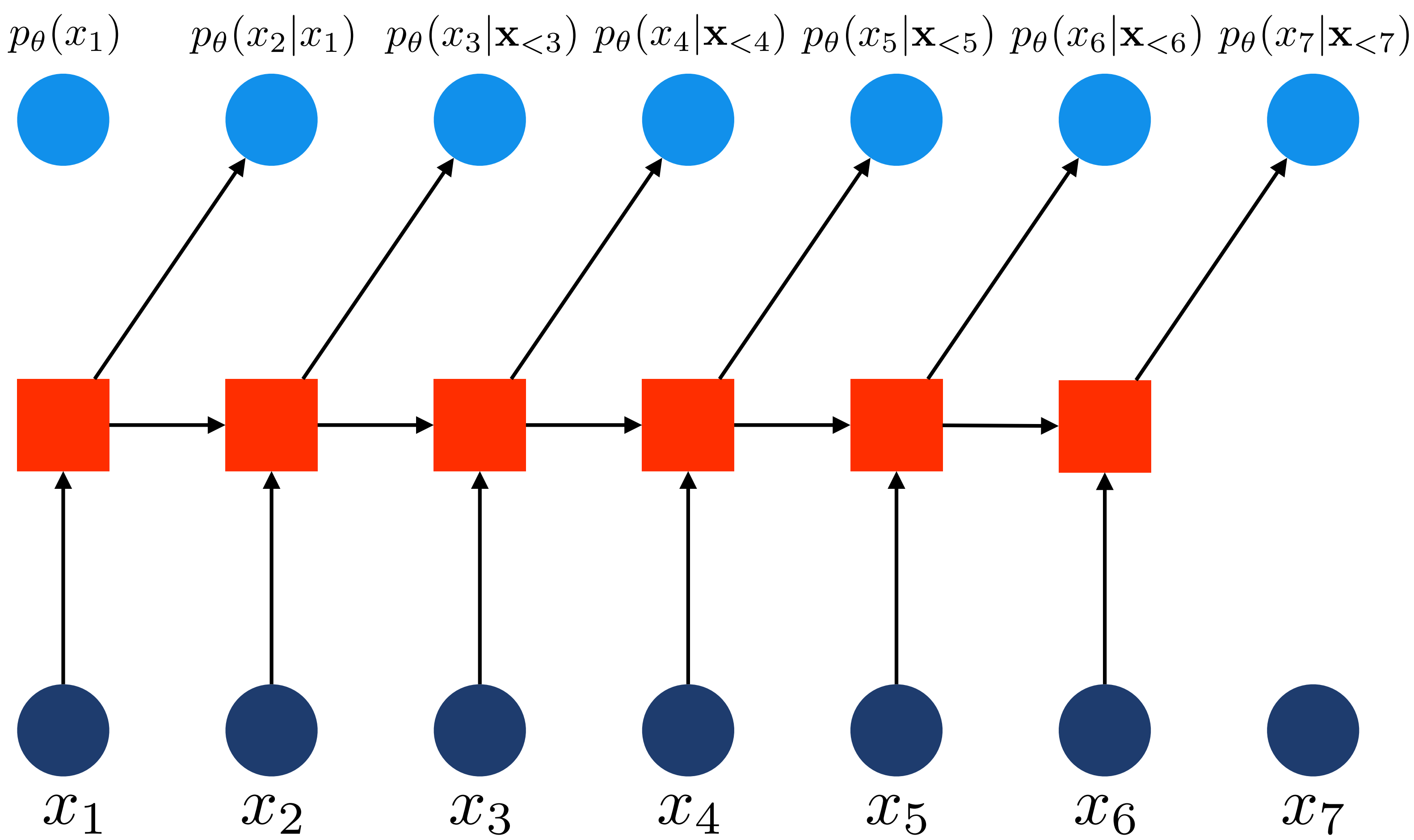
$$\log p_\theta(\mathbf{x}) = \log \left(\prod_{j=1}^{M} p_\theta(x_j | \mathbf{x}_{<j}) \right)$$

$$= \sum_{j=1}^{M} \log p_\theta(x_j | \mathbf{x}_{<j})$$

$$\theta^* = \arg\max_\theta \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} \log p_\theta(x_j^{(i)} | \mathbf{x}_{<j}^{(i)})$$

models

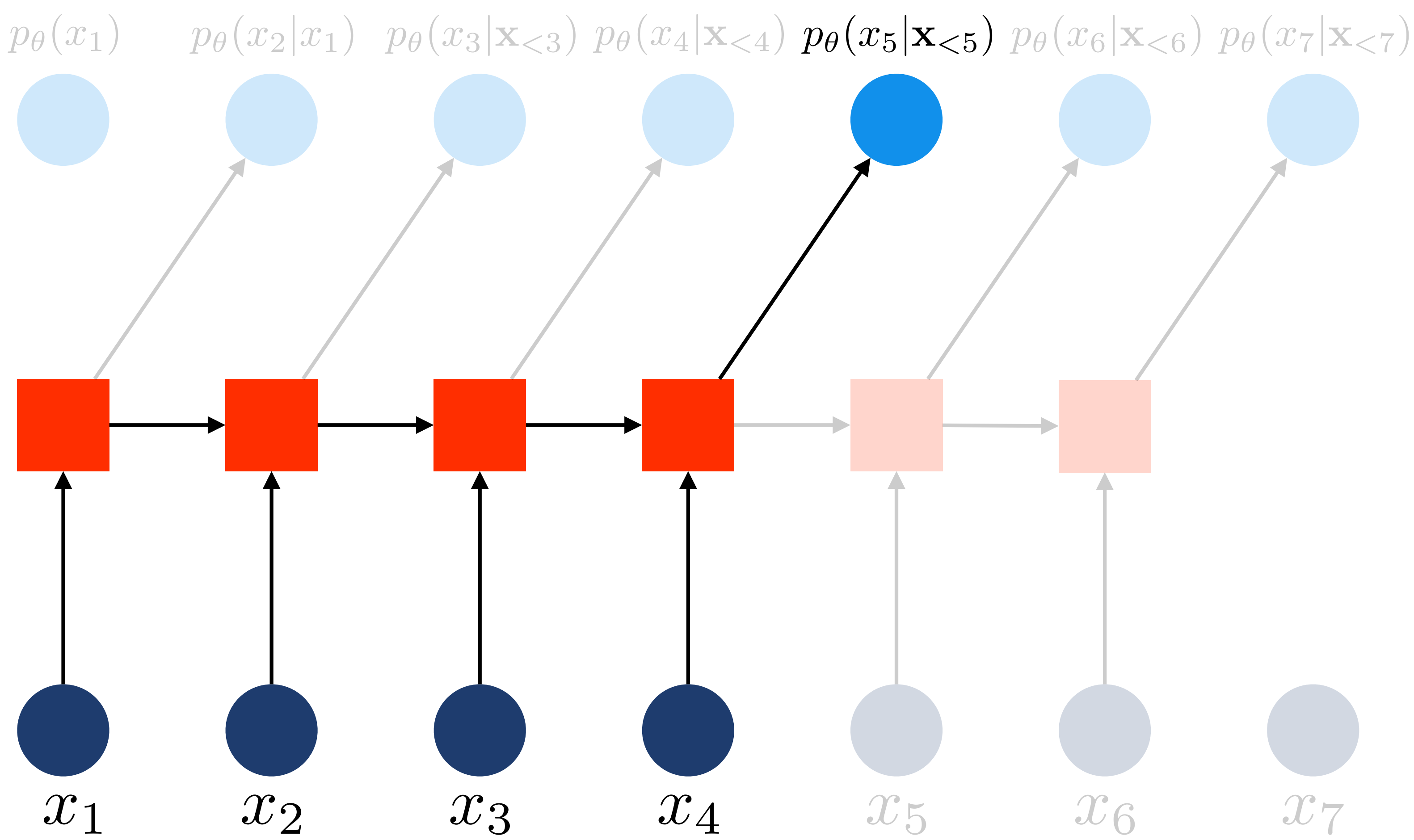
can parameterize conditional distributions using a **recurrent neural network**



see **Deep Learning** (Chapter 10), *Goodfellow et al.*, 2016

models

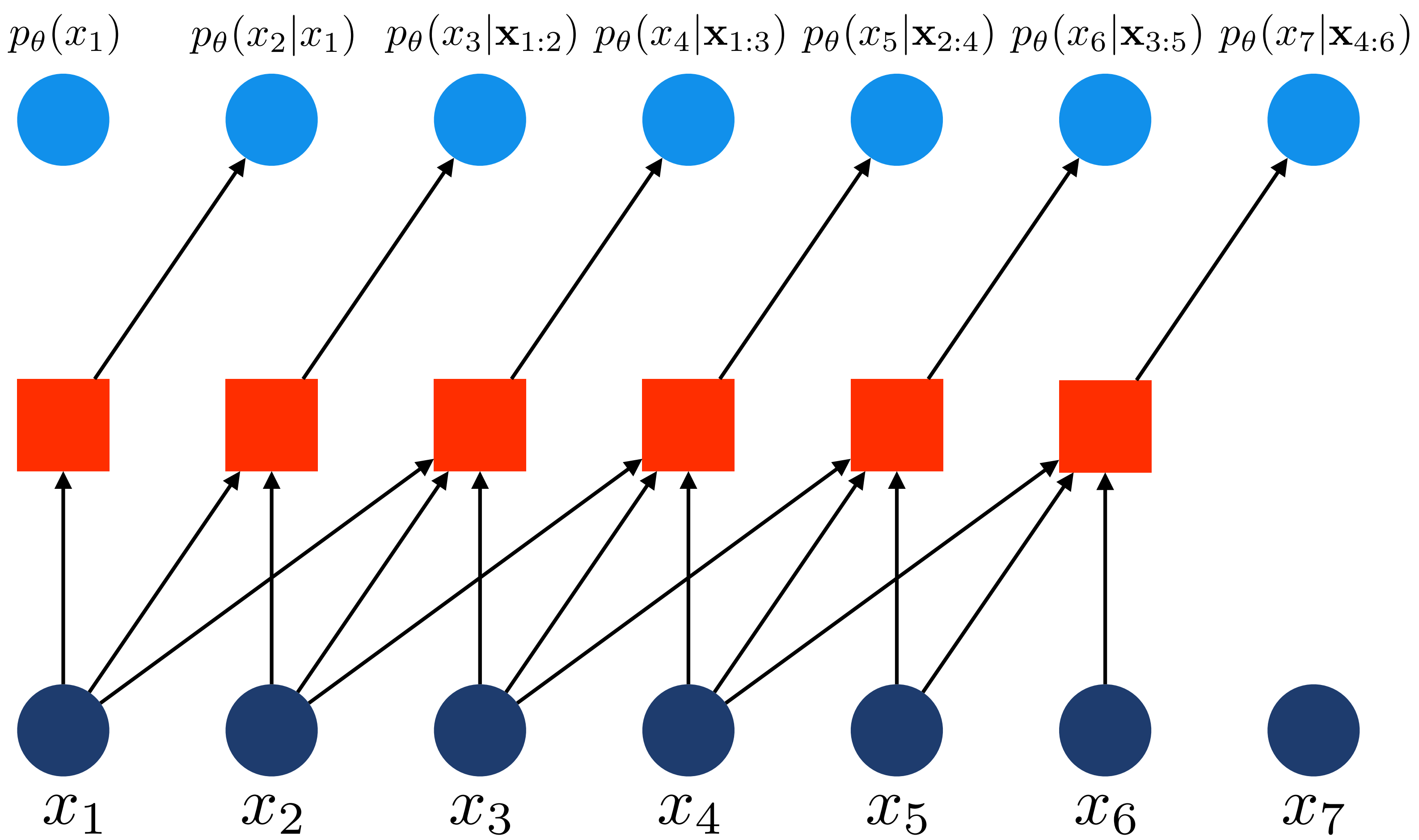
can parameterize conditional distributions using a **recurrent neural network**



see **Deep Learning** (Chapter 10), *Goodfellow et al.*, 2016

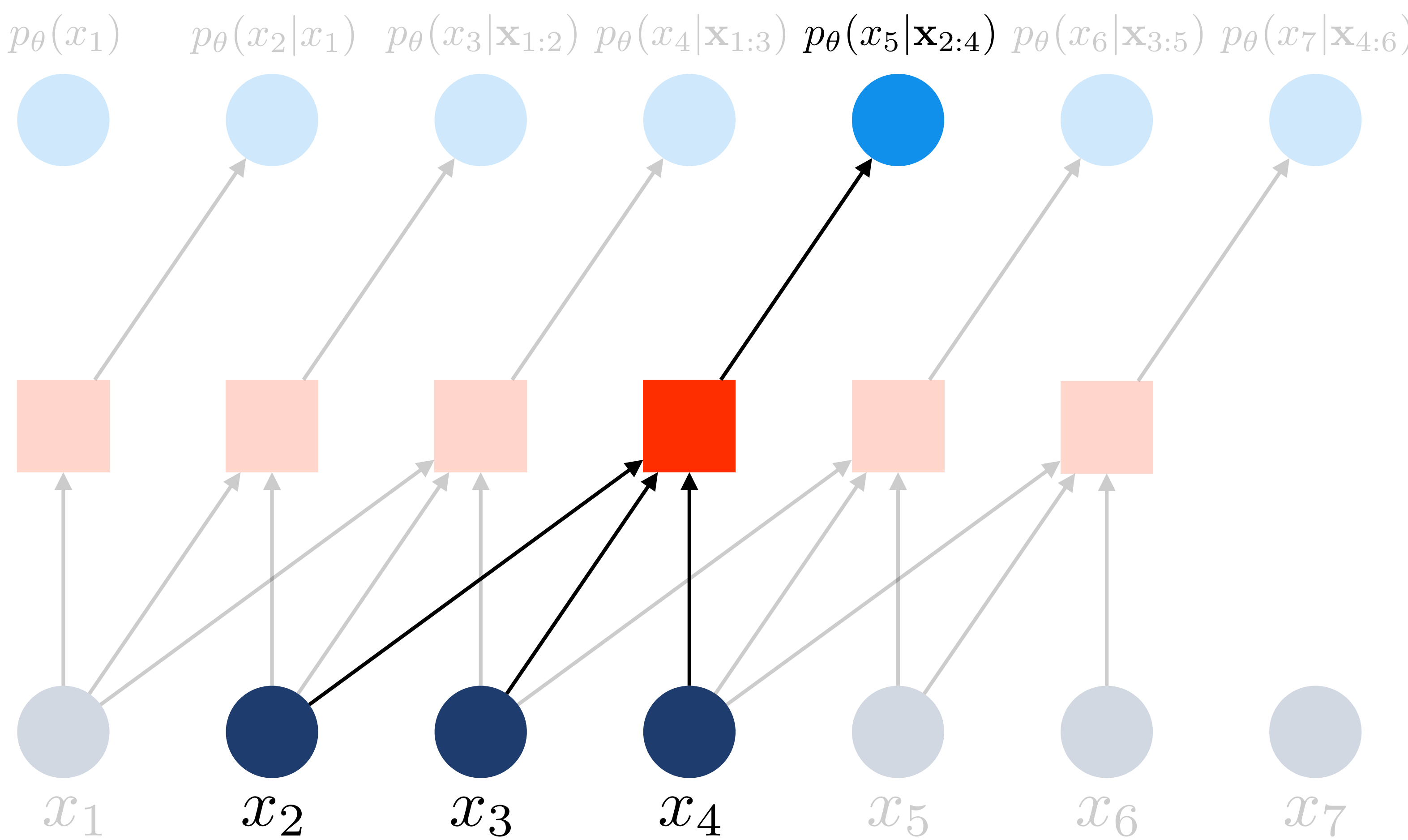
models

can condition on a local window using **convolutional neural networks**



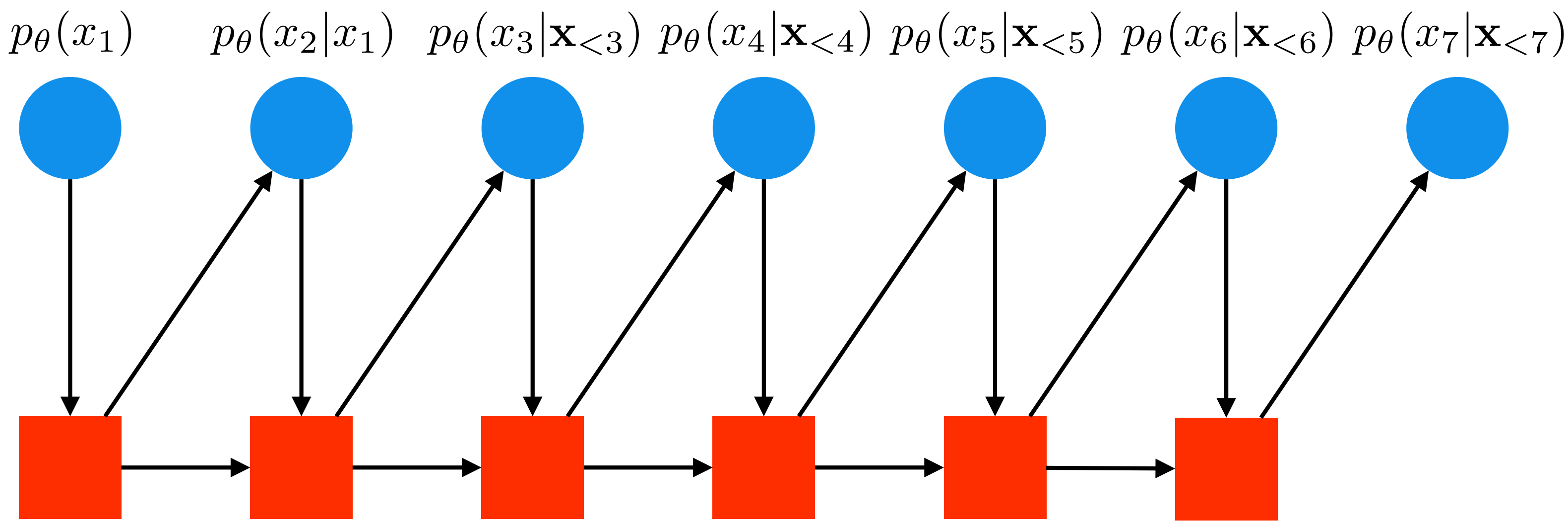
models

can condition on a local window using **convolutional neural networks**



sampling

sample from the model by drawing from the output distribution



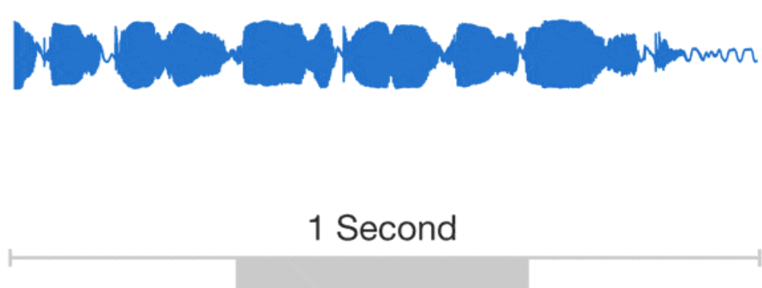
example applications

text

images

Pixel Recurrent Neural Networks, *van den Oord et al.*, 2016

speech



WaveNet: A Generative Model for Raw Audio, *van den Oord et al.*, 2016

SYSTEM PROMPT (HUMAN-WRITTEN)

A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.

MODEL COMPLETION (MACHINE-WRITTEN, FIRST TRY)

The incident occurred on the downtown train line, which runs from Covington and Ashland stations.

In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.

“The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation,” said Tom Hicks, the U.S. Energy Secretary, in a statement. “Our top priority is to secure the theft and ensure it doesn’t happen again.”

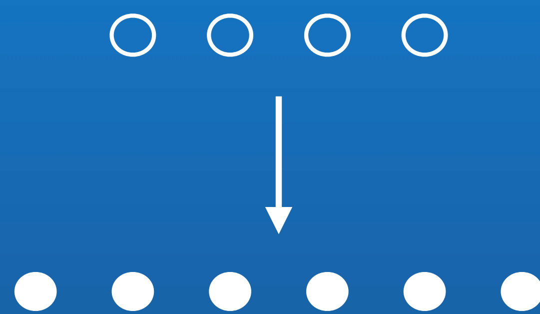
The stolen material was taken from the University of Cincinnati’s Research Triangle Park nuclear research site, according to a news release from Department officials.

The Nuclear Regulatory Commission did not immediately release any information.

According to the release, the U.S. Department of Energy’s Office of Nuclear Material Safety and Security is leading that team’s investigation.

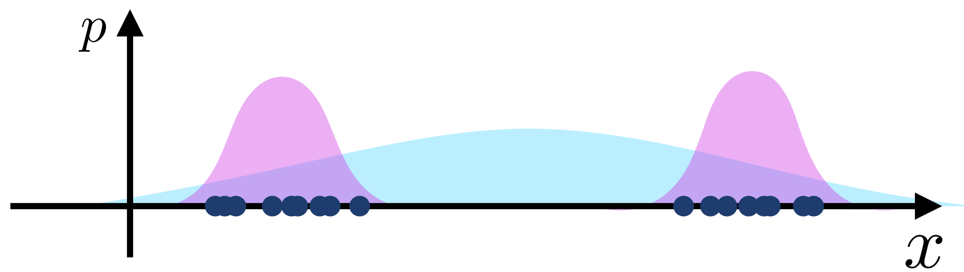
“The safety of people, the environment and the nation’s nuclear stockpile is our highest priority,” Hicks said. “We will get to the bottom of this and make no excuses.

Attention is All You Need, Vaswani *et al.*, 2017
Improving Language Understanding by Generative Pre-Training, Radford *et al.*, 2018
Language Models as Unsupervised Multi-task Learners, Radford *et al.*, 2019



explicit
latent variable models

latent variables result in mixtures of distributions



approach 1

directly fit a distribution to the data

$$p_\theta(x) = \mathcal{N}(x; \mu, \sigma^2)$$

approach 2

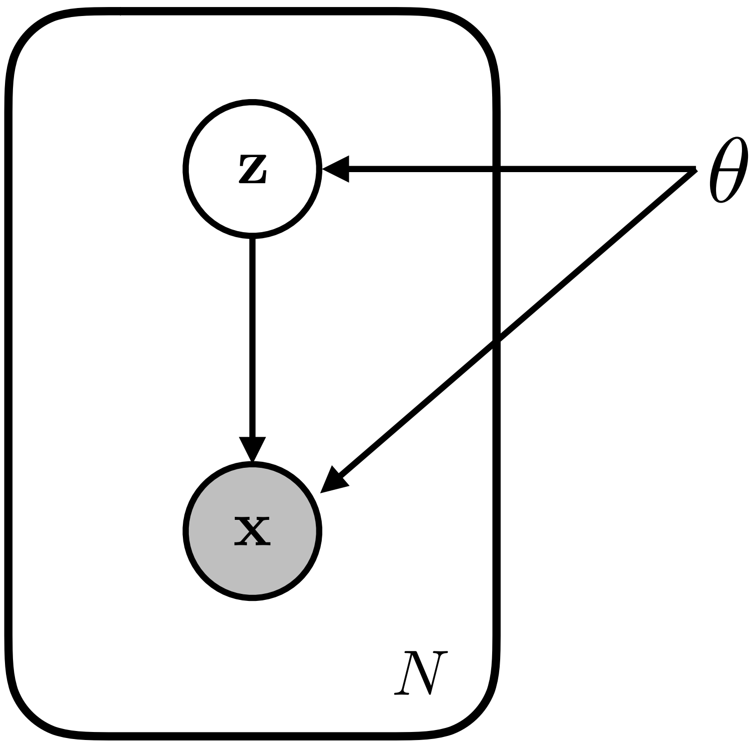
use a latent variable to model the data

$$p_\theta(x, z) = p_\theta(x|z)p_\theta(z) = \mathcal{N}(x; \mu_x(z), \sigma_x^2(z))\mathcal{B}(z; \mu_z)$$

$$p_\theta(x) = \sum_z p_\theta(x, z)$$

$$= \underbrace{\mu_z \cdot \mathcal{N}(x; \mu_x(1), \sigma_x^2(1))}_{\text{mixture component}} + \underbrace{(1 - \mu_z) \cdot \mathcal{N}(x; \mu_x(0), \sigma_x^2(0))}_{\text{mixture component}}$$

directed latent variable model



Generation

GENERATIVE MODEL

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$$

joint — *conditional likelihood* — *prior*

1. sample $\mathbf{z}$ from $p(\mathbf{z})$
2. use $\mathbf{z}$ samples to sample $\mathbf{x}$ from $p(\mathbf{x}|\mathbf{z})$

intuitive example: graphics engine

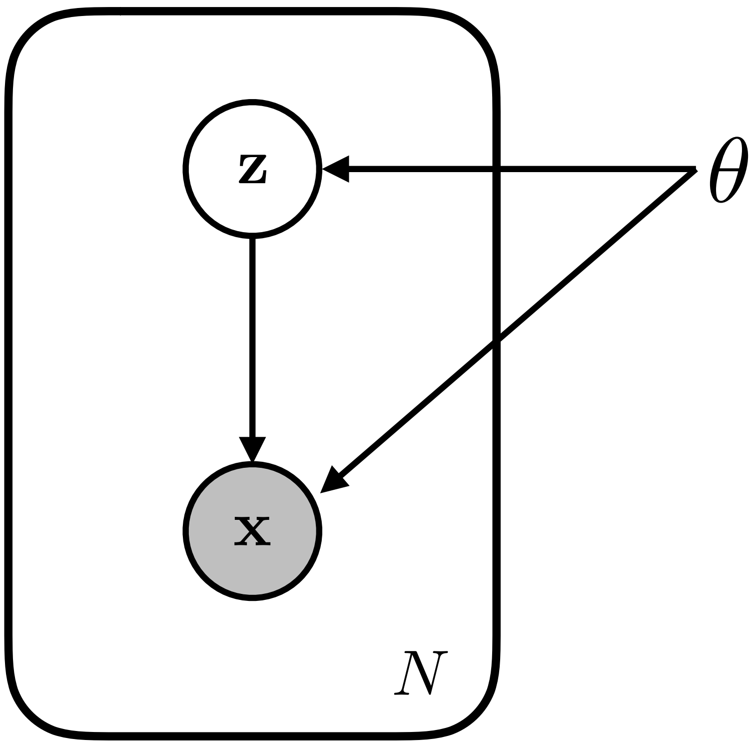
object ~ p(objects)

lighting ~ p(lighting)

background ~ p(bg)

RENDER →

<u>directed</u> latent variable model



Posterior Inference

INFERENCE

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})}$$

posterior

joint

marginal likelihood

use Bayes' rule

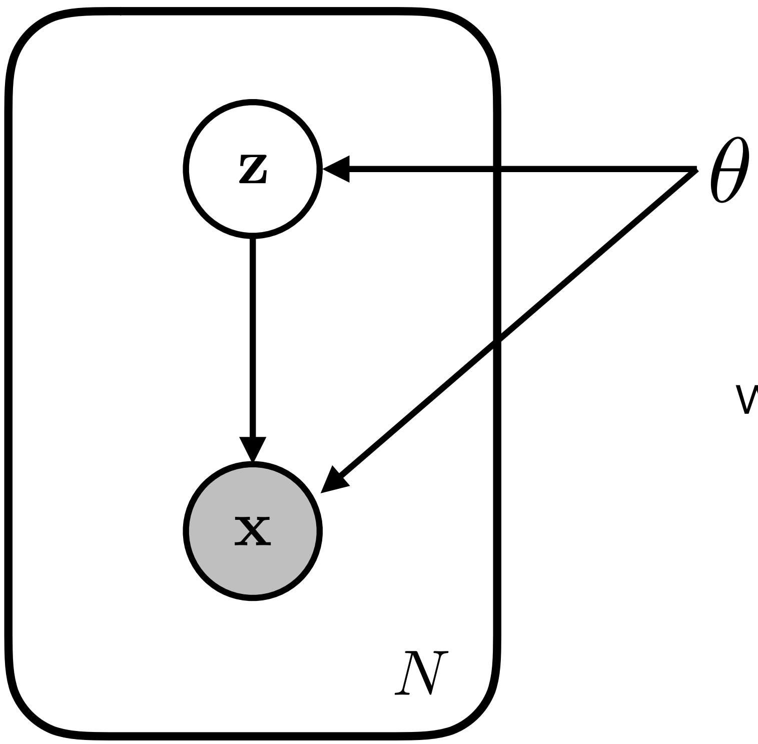
provides conditional distribution over latent variables

intuitive example

what is the probability that I am observing a cat given these pixel observations?

observation

<u>directed</u> latent variable model



Model Evaluation

MARGINALIZATION

marginal likelihood

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

joint

to evaluate the likelihood of an observation, we need to *marginalize* over all latent variables

i.e. consider all possible underlying states

intuitive example

observation

how likely is this observation under my model?
(what is the probability of observing this?)

for all objects, lighting, backgrounds, etc.:

how plausible is this example?

maximum likelihood estimation

maximize the log-likelihood (under the model) of the true data examples

$$\theta^* = \arg\max_\theta \; \mathbb{E}_{p_{\text{data}}(\mathbf{x})} [\log p_\theta(\mathbf{x})] \approx \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)})$$

for latent variable models:

discrete

$$\log p_\theta(\mathbf{x}) = \log \sum_z p_\theta(\mathbf{x}, \mathbf{z})$$

or

continuous

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

marginalizing is often intractable in practice

variational inference

lower bound the log-likelihood by introducing an approximate posterior

introduce an ***approximate posterior*** $q(\mathbf{z}|\mathbf{x})$

$$\log p_\theta(\mathbf{x}) = \mathcal{L}(\mathbf{x}) + D_{KL}(q(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z}|\mathbf{x}))$$

where $\mathcal{L}(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right]$

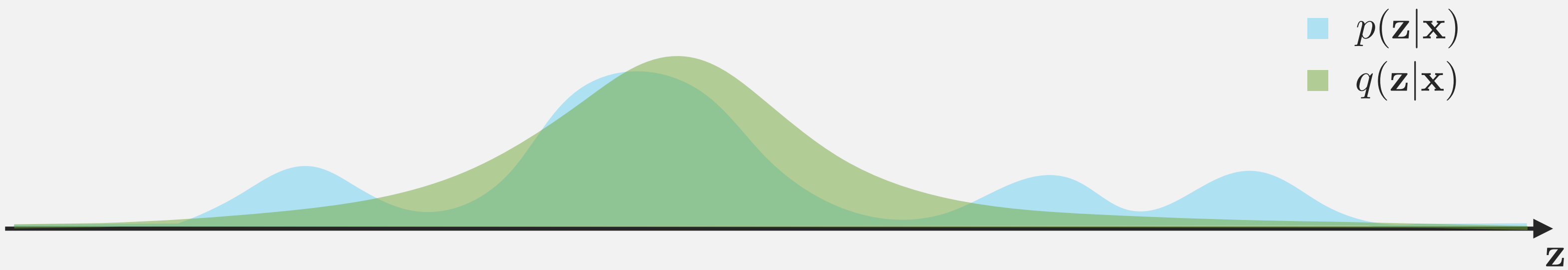
$$D_{KL} \geq 0 \longrightarrow \mathcal{L}(\mathbf{x}) \leq \log p_\theta(\mathbf{x})$$ (lower bound)

variational expectation maximization (EM)

E-Step: optimize $\mathcal{L}(\mathbf{x})$ *w.r.t.* $q(\mathbf{z}|\mathbf{x})$

M-Step: optimize $\mathcal{L}(\mathbf{x})$ *w.r.t.* θ

the E-Step indirectly minimizes $D_{KL}(q(\mathbf{z}|\mathbf{x}) || p_\theta(\mathbf{z}|\mathbf{x}))$



interpreting the lower bound

we can write the lower bound as

$$\mathcal{L} \equiv \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right]$$

$$= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right]$$

$$= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right]$$

$$= \underbrace{\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right]}_{\text{reconstruction}} - \underbrace{D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))}_{\text{regularization}}$$

$q(\mathbf{z}|\mathbf{x})$ is optimized to represent the data while staying close to the prior

connections to *compression, information theory*

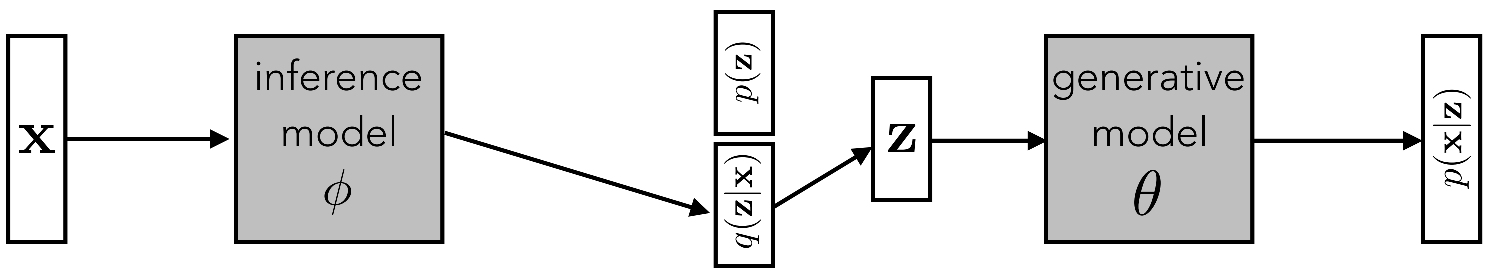
variational autoencoder (VAE)

variational expectation maximization (EM)

E-Step: optimize $\mathcal{L}(\mathbf{x})$ *w.r.t.* $q(\mathbf{z}|\mathbf{x})$

M-Step: optimize $\mathcal{L}(\mathbf{x})$ *w.r.t.* θ

use a separate ***inference model*** to directly output approximate posterior estimates



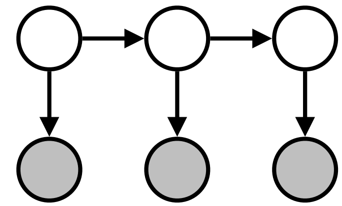
learn both models jointly using *stochastic backpropagation*

reparametrization trick: $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

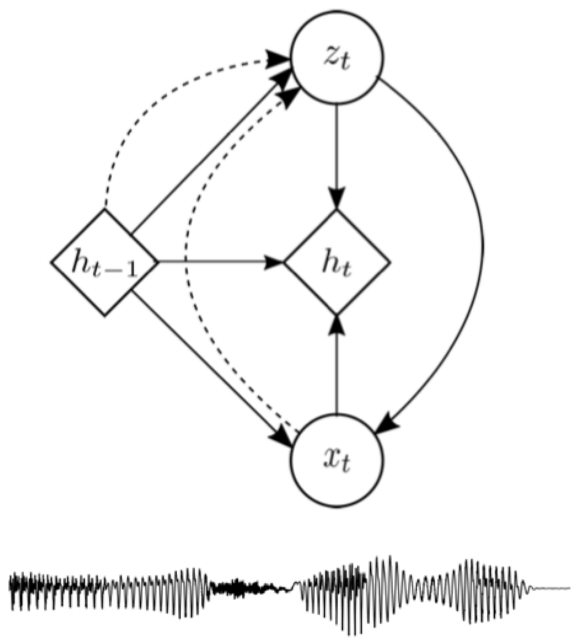
Autoencoding Variational Bayes, Kingma & Welling, 2014
Stochastic Backpropagation, Rezende *et al.*, 2014

sequential latent variable models

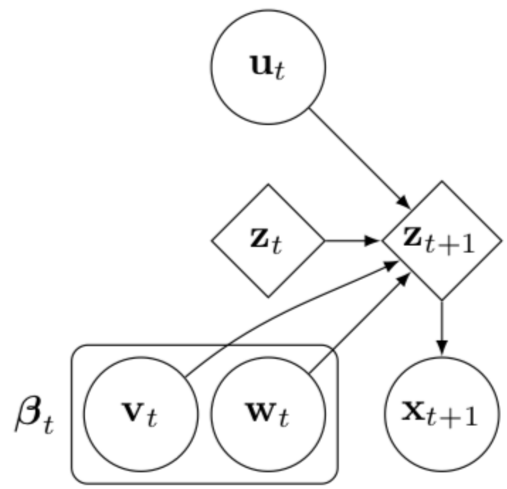
can use the same techniques to train
sequential latent variable models



some examples:



A Recurrent Latent Variable Model for Sequential Data, *Chung et al.*, 2015

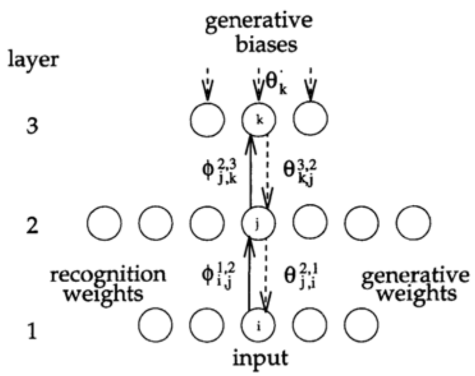


Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data, *Karl et al.*, 2016

discrete latent variable models

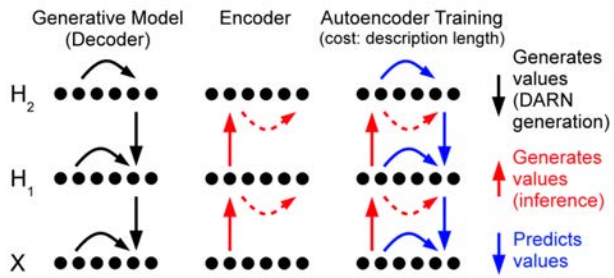
with discrete latent variables, cannot easily backprop through sampling $\mathbf{z}$

Helmholtz Machine / Wake-Sleep



Dayan *et al.*, 1995

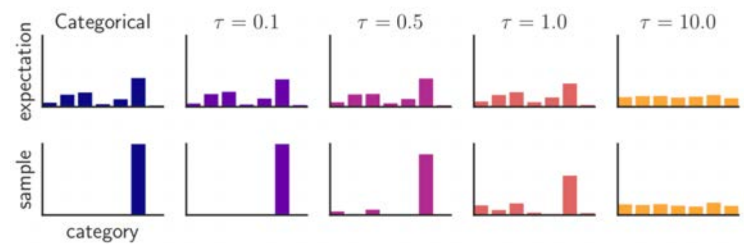
REINFORCE Gradients



Gregor *et al.*, 2014

Mnih & Gregor, 2014

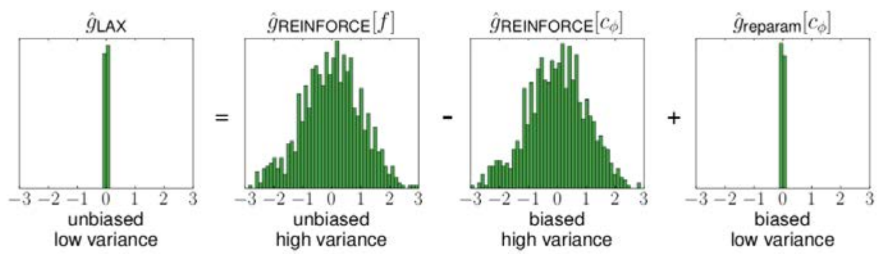
Relaxed Distributions



Jang *et al.*, 2017

Maddison *et al.*, 2017

Combinations

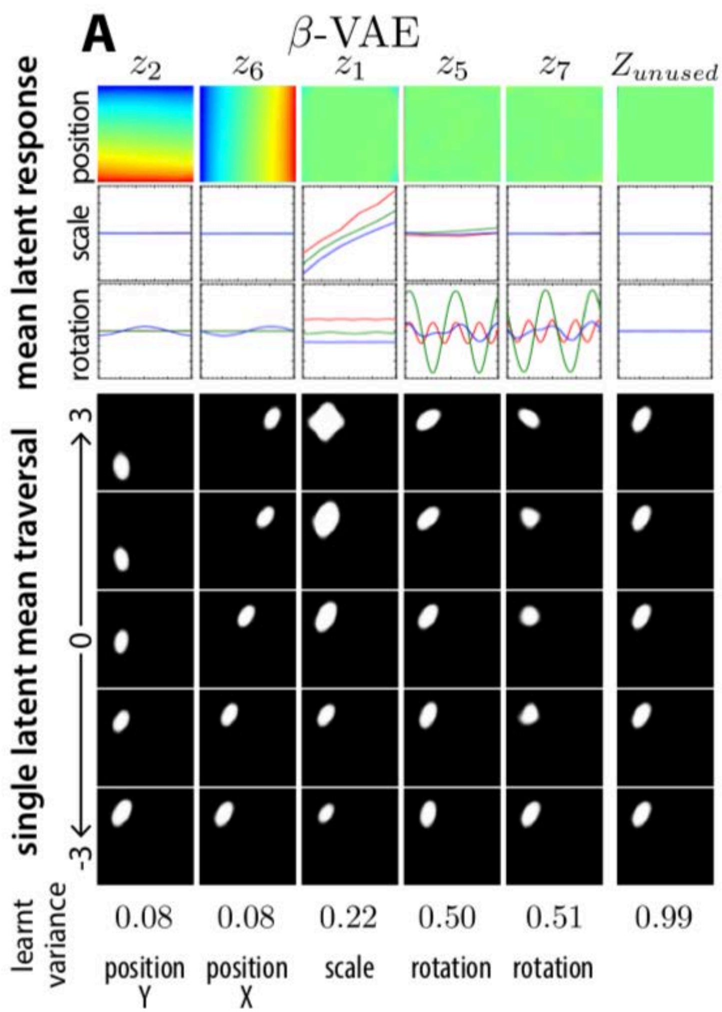


Tucker *et al.*, 2017
Grathwohl *et al.*, 2018

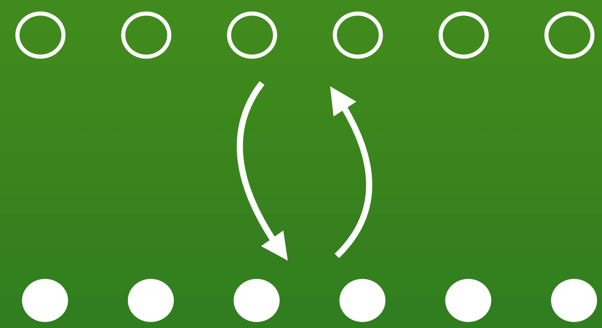
representation learning

latent variables provide a natural representation for downstream tasks

beta-VAE



Higgins *et al.*, 2017

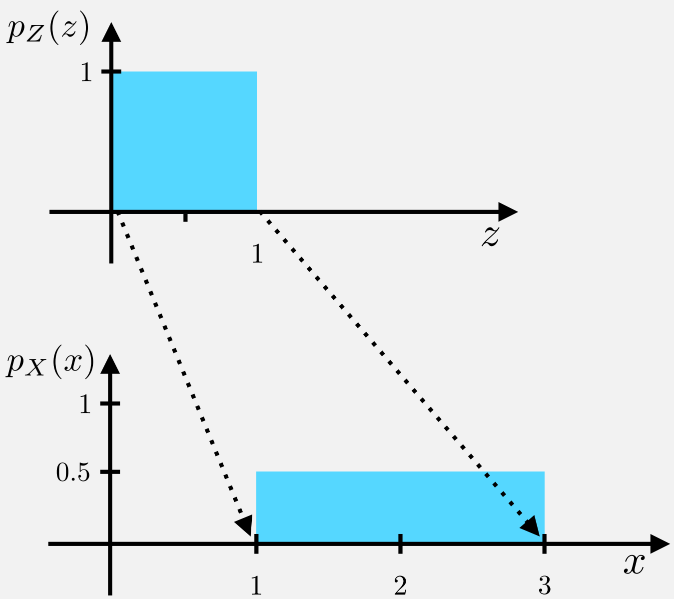


invertible / flow-based models

change of variables

use an invertible mapping to directly evaluate the log likelihood

simple example



sample z from a *base distribution*

$$z \sim p_Z(z) = \text{Uniform}(0, 1)$$

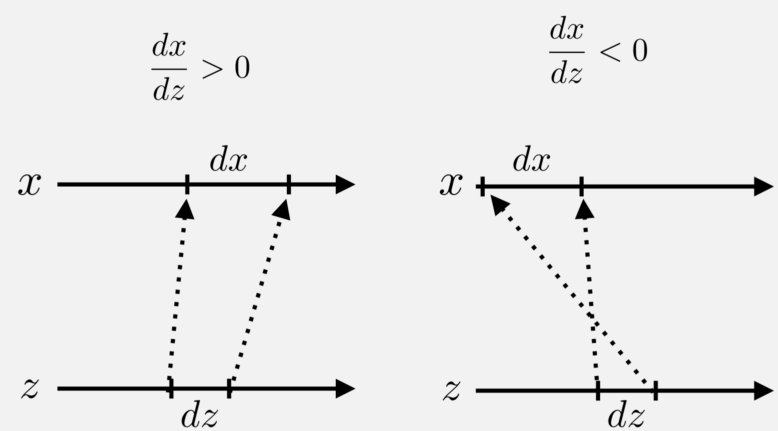
apply a transform to z to get a *transformed distribution*

$$x = f(z) = 2z + 1$$

$$p_X(x)dx = p_Z(z)dz$$

$$p_X(x) = p_Z(z) \left| \frac{dz}{dx} \right|$$

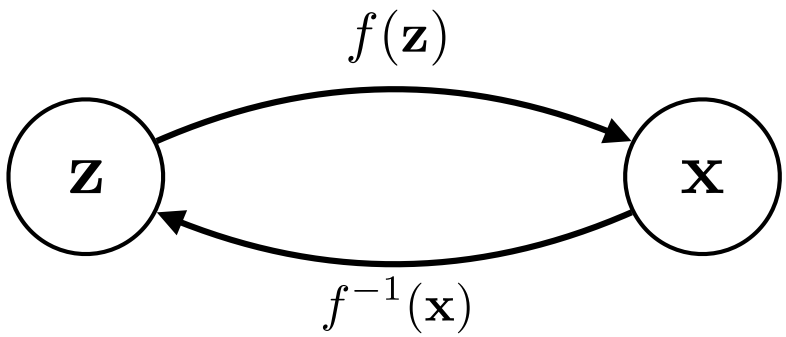
conservation of probability mass



change of variables

$p_Z(\mathbf{z})$

base distribution



$p_X(\mathbf{x})$

transformed distribution

change of variables formula

$$p_X(\mathbf{x}) = p_Z(\mathbf{z}) \left| \det \mathbf{J}(f^{-1}(\mathbf{x})) \right|$$

or

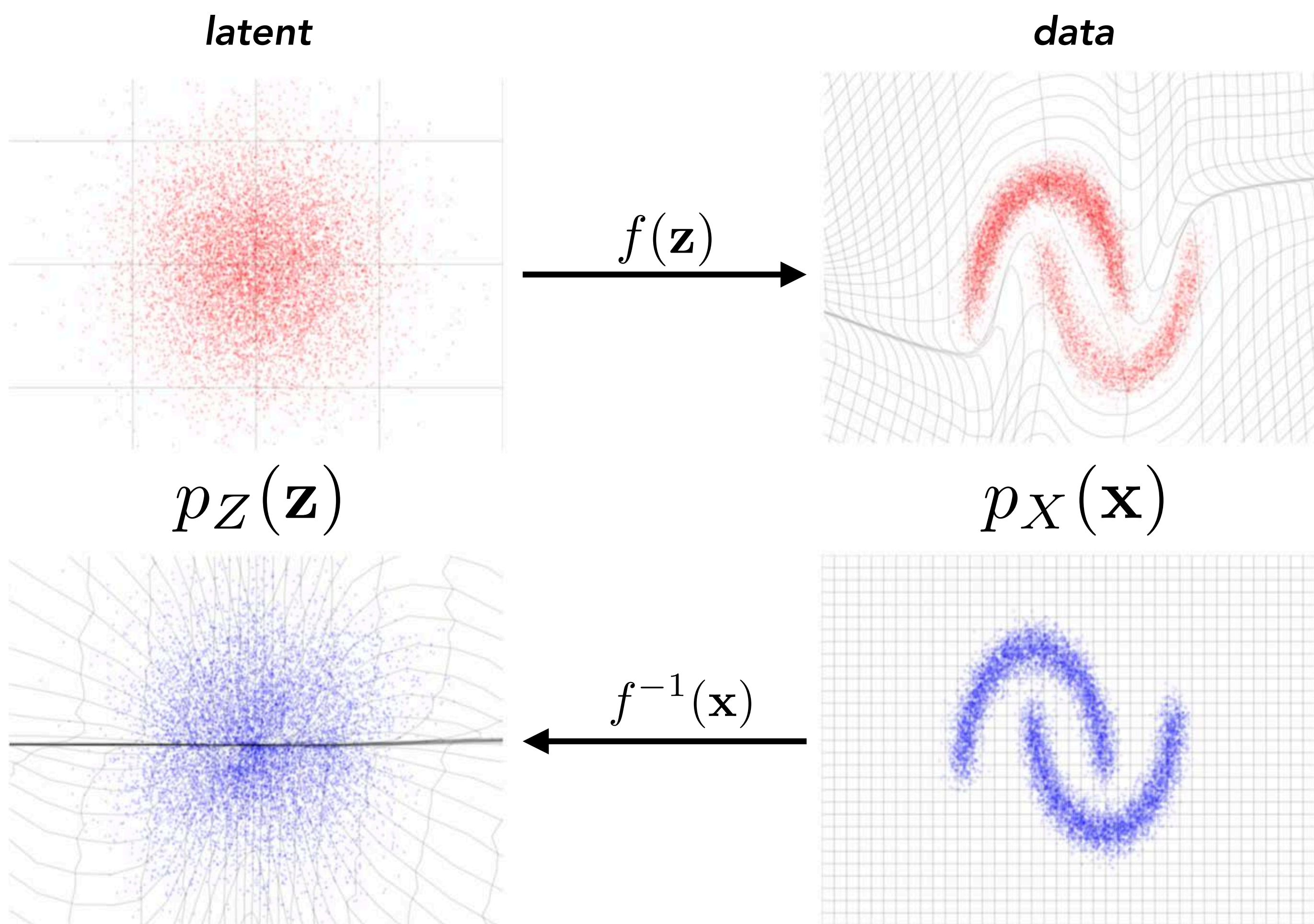
$$\log p_X(\mathbf{x}) = \log p_Z(\mathbf{z}) + \log \left| \det \mathbf{J}(f^{-1}(\mathbf{x})) \right|$$

$\mathbf{J}(f^{-1}(\mathbf{x}))$ is the *Jacobian* matrix of the inverse transform

$\det \mathbf{J}(f^{-1}(\mathbf{x}))$ *is the local distortion in volume from the transform*

change of variables

transform the data into a space that is easier to model



Density Estimation Using Real NVP, Dinh *et al.*, 2016

maximum likelihood estimation

maximize the log-likelihood (under the model) of the true data examples

$$\theta^* = \arg\max_\theta \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[\log p_\theta(\mathbf{x})\right] \approx \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)})$$

for invertible latent variable models:

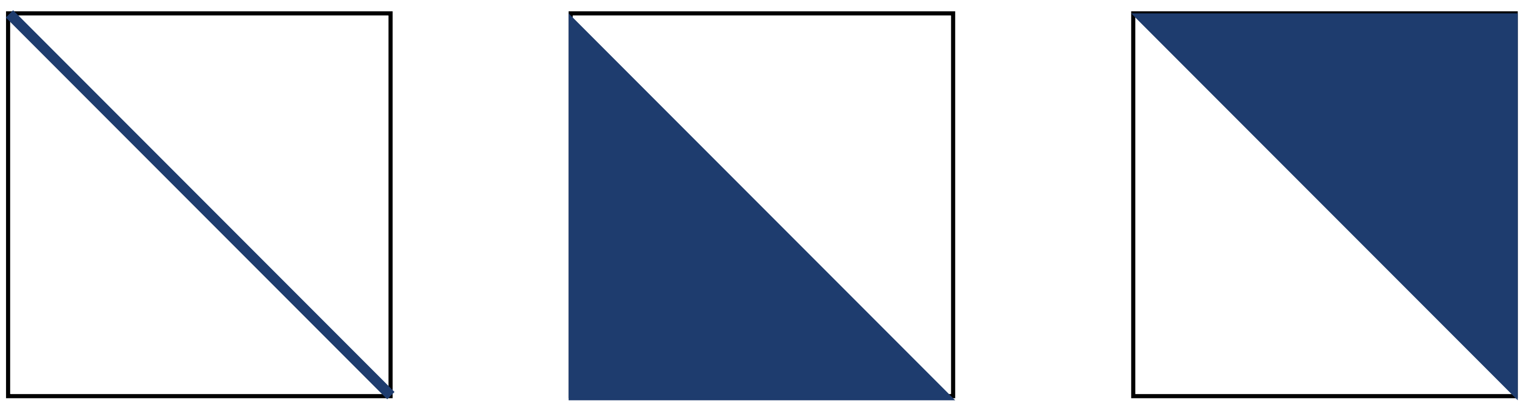
$$\log p_\theta(\mathbf{x}) = \log p_\theta(\mathbf{z}) + \log \left|\det \mathbf{J}(f_\theta^{-1}(\mathbf{x}))\right|$$

$$\theta^* = \arg\max_\theta \frac{1}{N}\sum_{i=1}^{N} \left[\log p_\theta(\mathbf{z}^{(i)}) + \log \left|\det \mathbf{J}(f_\theta^{-1}(\mathbf{x}^{(i)}))\right|\right]$$

change of variables

to use the change of variables formula, we need to evaluate $\det \mathbf{J}(f^{-1}(\mathbf{x}))$

for an arbitrary $N \times N$ Jacobian matrix, this is worst case $O(N^3)$



restrict the transforms to those with diagonal or triangular inverse Jacobians

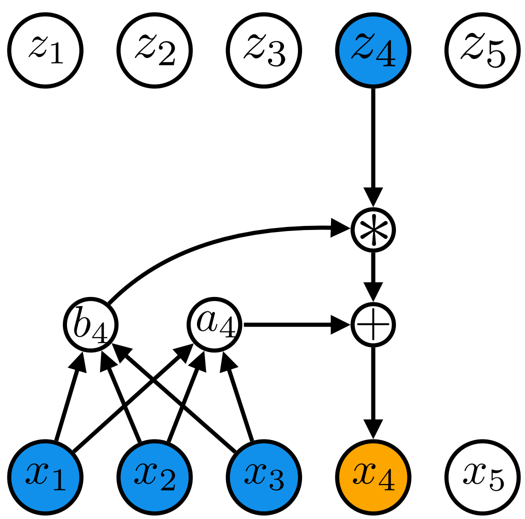
allows us to compute $\det \mathbf{J}(f^{-1}(\mathbf{x}))$ in $O(N)$

→ *product of diagonal entries*

masked autoregressive flow (MAF)

TRANSFORM

base distribution

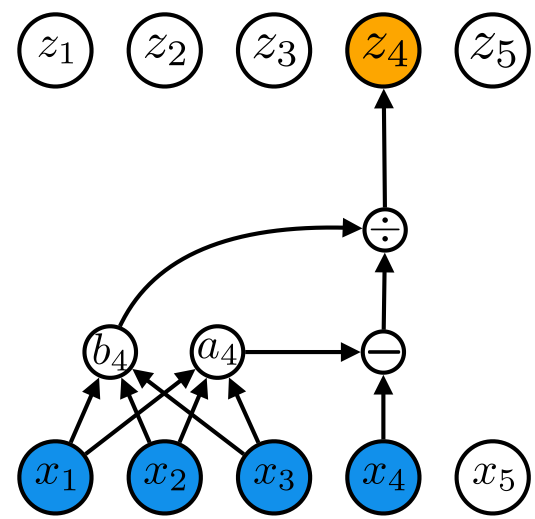


transformed distribution

$$x_4 = a_4(\mathbf{x}_{1:3}) + b_4(\mathbf{x}_{1:3}) \cdot z_4$$

INVERSE TRANSFORM

base distribution



transformed distribution

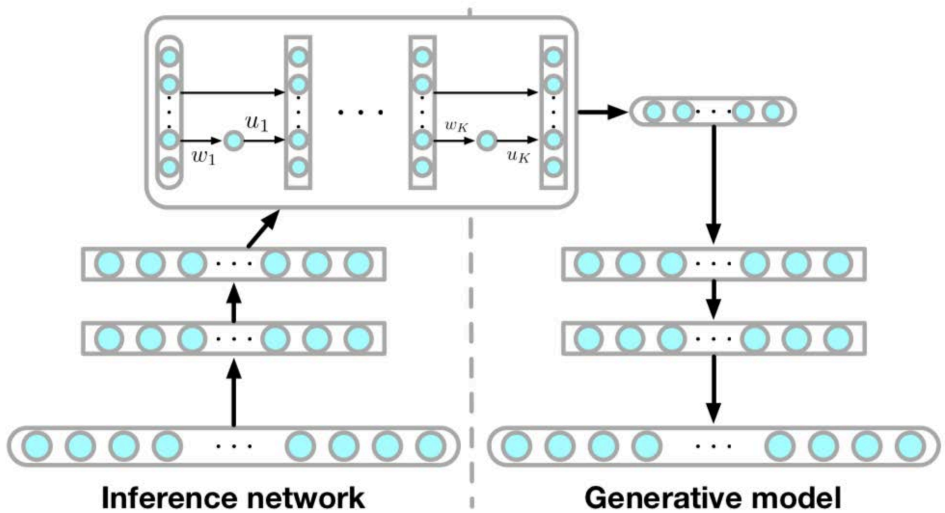
$$z_4 = \frac{x_4 - a_4(\mathbf{x}_{1:3})}{b_4(\mathbf{x}_{1:3})}$$

Masked Autoregressive Flow, Papamakarios *et al.*, 2017

normalizing flows (NF)

can also use the change of variables formula for variational inference

parameterize $q(\mathbf{z}|\mathbf{x})$ as a transformed distribution

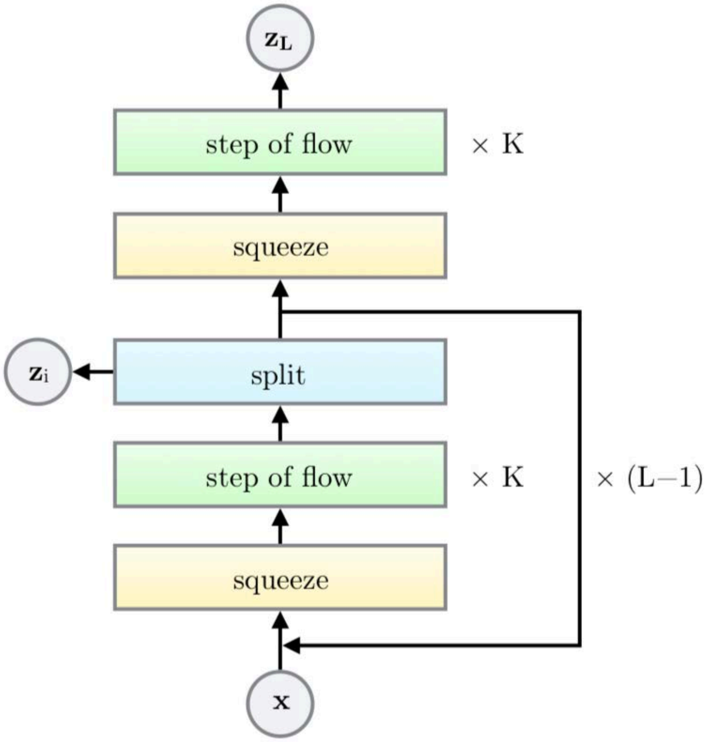


use more complex approximate posterior, but evaluate a simpler distribution

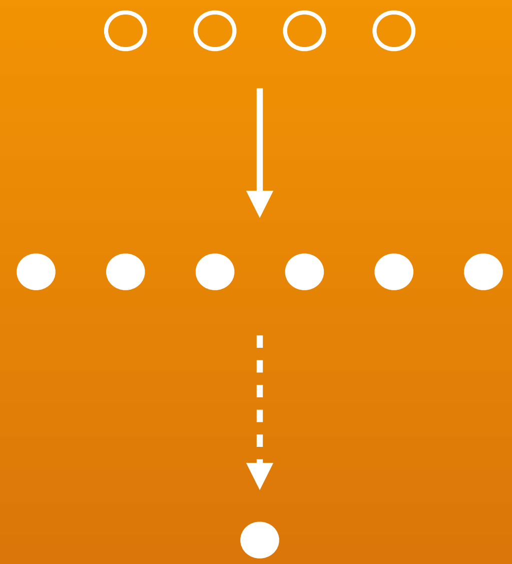
Normalizing Flows, Rezende & Mohamed, 2015

Glow

use 1 x 1 convolutions to perform transform



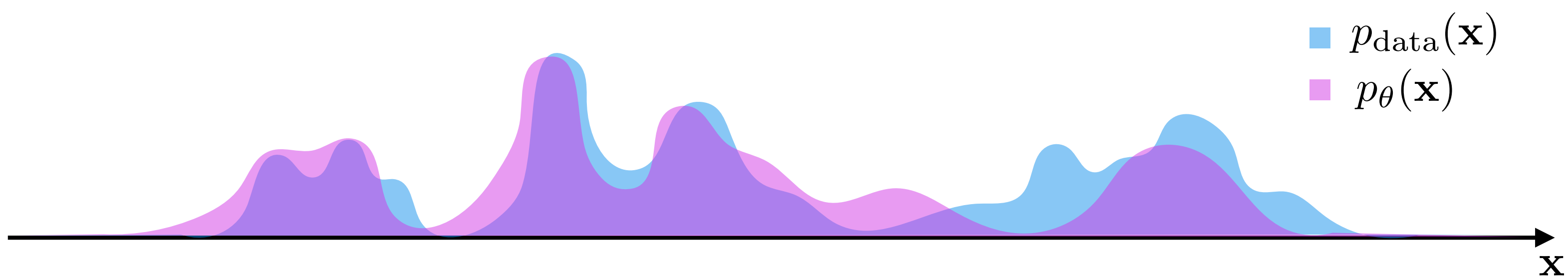
Glow, Kingma & Dhariwal, 2018



implicit

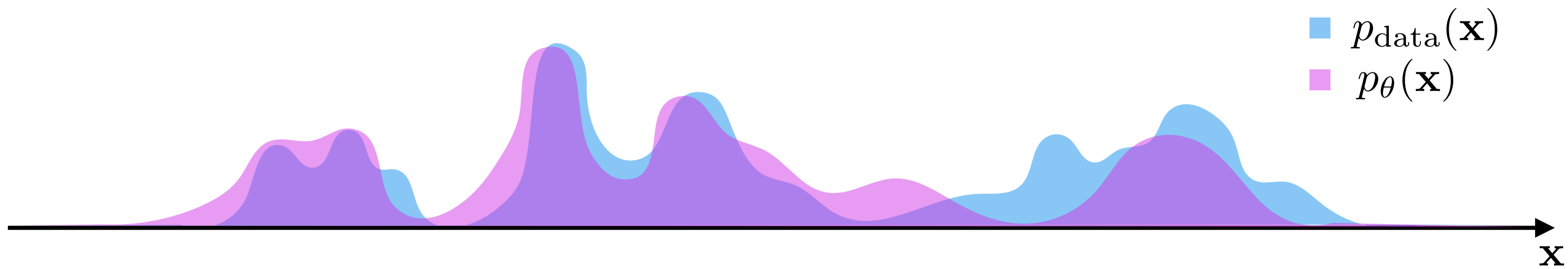
latent variable models

instead of using an *explicit* probability density,
learn a model that defines an *implicit density*



specify a stochastic procedure for generating the data
that does not require an explicit likelihood evaluation

Learning in Implicit Generative Models, Mohamed & Lakshminarayanan, 2016



estimate density ratio through *hypothesis testing*

data distribution $p_{\text{data}}(\mathbf{x})$

generated distribution $p_\theta(\mathbf{x})$

$$\frac{p_{\text{data}}(\mathbf{x})}{p_\theta(\mathbf{x})} = \frac{p(\mathbf{x}|y = \text{data})}{p(\mathbf{x}|y = \text{model})}$$

$$\frac{p_{\text{data}}(\mathbf{x})}{p_\theta(\mathbf{x})} = \frac{p(y = \text{data}|\mathbf{x})p(\mathbf{x})/p(y = \text{data})}{p(y = \text{model}|\mathbf{x})p(\mathbf{x})/p(y = \text{model})}$$

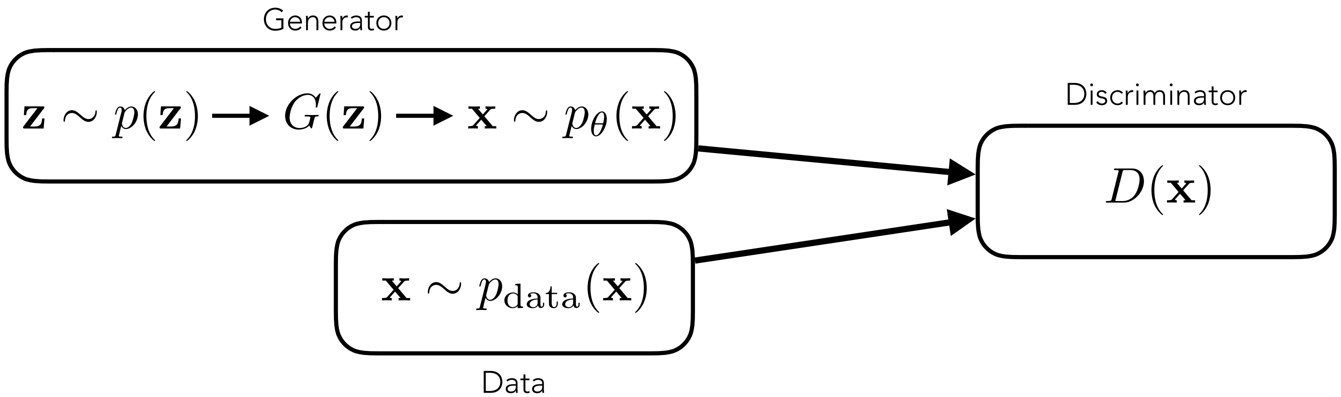
(Bayes' rule)

$$\frac{p_{\text{data}}(\mathbf{x})}{p_\theta(\mathbf{x})} = \frac{p(y = \text{data}|\mathbf{x})}{p(y = \text{model}|\mathbf{x})}$$

(assuming equal dist. prob.)

density estimation becomes a sample discrimination task

Generative Adversarial Networks (GANs)



Generator: $G(\mathbf{z})$

Discriminator: $D(\mathbf{x}) = \hat{p}(y = \text{data}|\mathbf{x}) = 1 - \hat{p}(y = \text{model}|\mathbf{x})$

Log-Likelihood:

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\log \hat{p}(y = \text{data}|\mathbf{x})\right] + \mathbb{E}_{p_\theta(\mathbf{x})}\left[\log \hat{p}(y = \text{model}|\mathbf{x})\right]$$

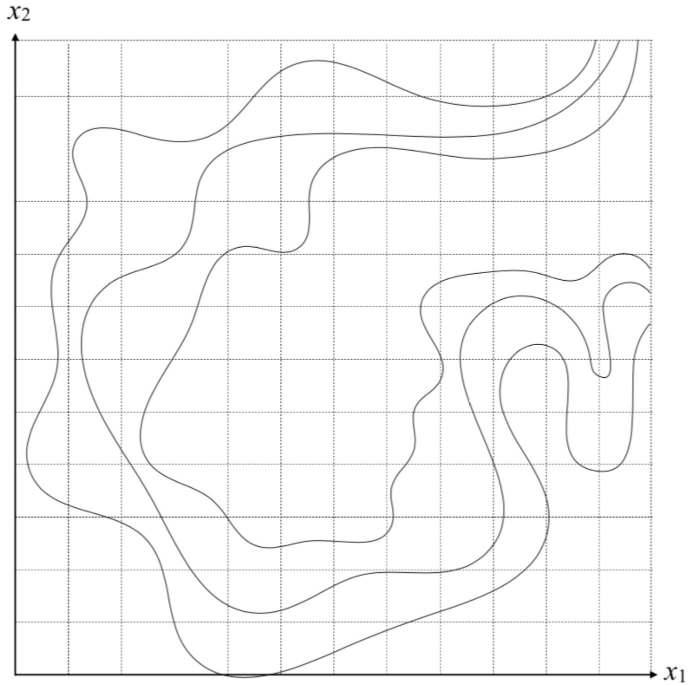
$$= \mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\log D(\mathbf{x})\right] + \mathbb{E}_{p_\theta(\mathbf{x})}\left[\log(1 - D(\mathbf{x}))\right]$$

$$= \mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\log D(\mathbf{x})\right] + \mathbb{E}_{p(\mathbf{z})}\left[\log(1 - D(G(\mathbf{z})))\right]$$

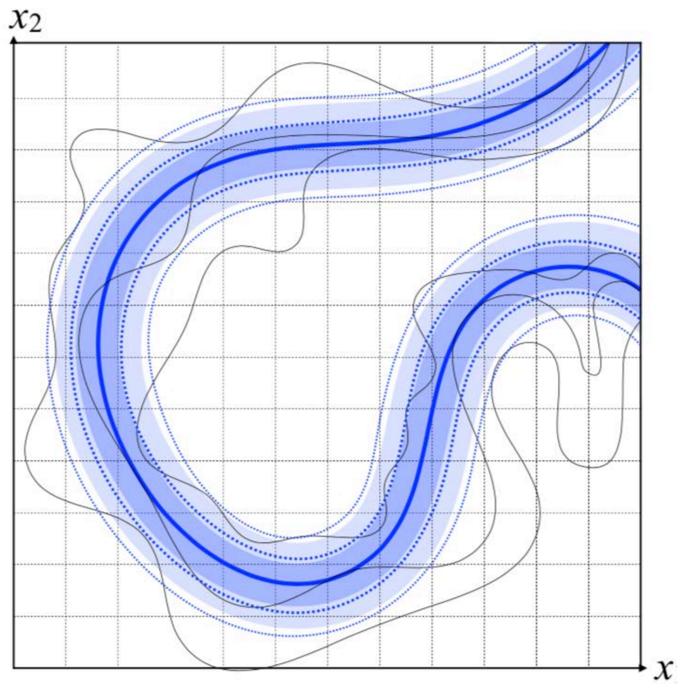
Minimax: $$\min_G \max_D \mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\log D(\mathbf{x})\right] + \mathbb{E}_{p(\mathbf{z})}\left[\log(1 - D(G(\mathbf{z})))\right]$$

Ian *Goodfellow*, 2016
Shakir *Mohamed*, 2016

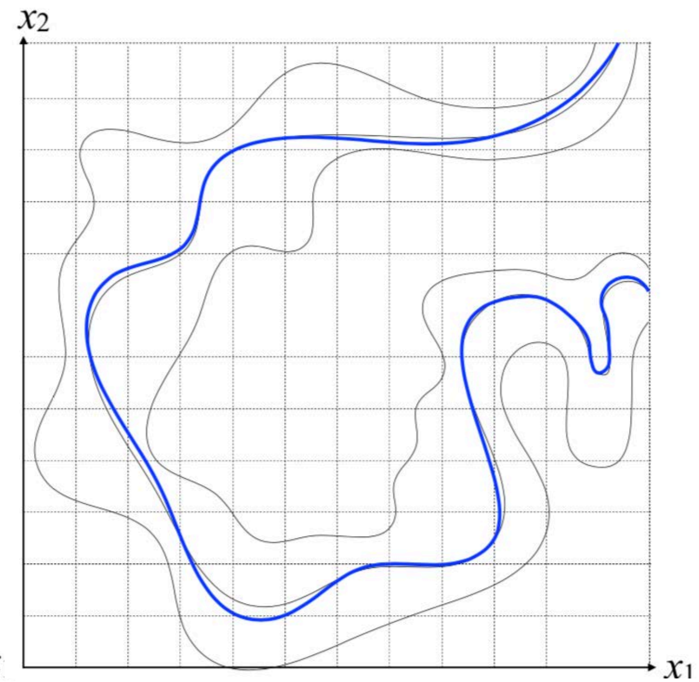
interpretation



data manifold



explicit model



implicit model

explicit models tend to cover the entire data manifold, but are constrained

implicit models tend to capture part of the data manifold, but can neglect other parts

→ *"mode collapse"*

Generative Adversarial Networks (GANs)

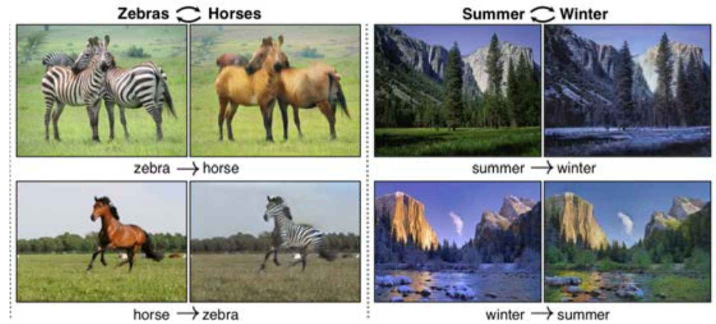
GANs can be difficult to optimize

Improved Training of Wasserstein GANs, Gulrajani et al., 2017

applications

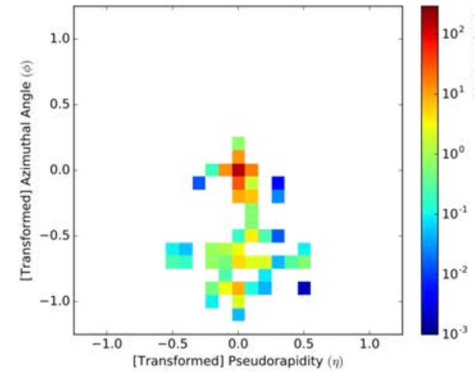
image to image translation

Image-to-Image Translation with Conditional Adversarial Networks, *Isola et al.*, 2016



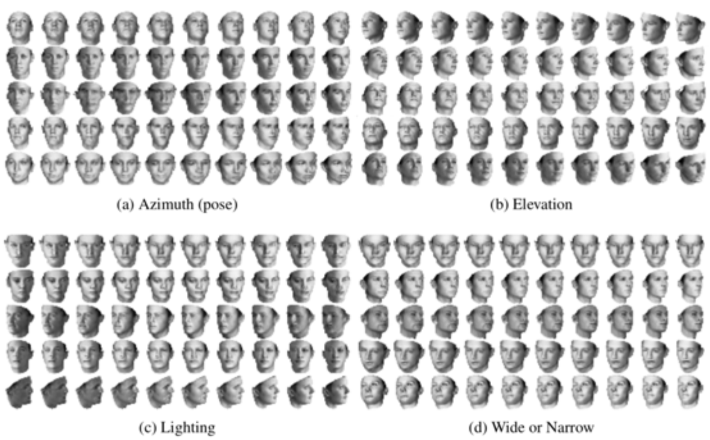
Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, *Zhu et al.*, 2017

experimental simulation



Learning Particle Physics by Example, *de Oliveira et al.*, 2017

interpretable representations

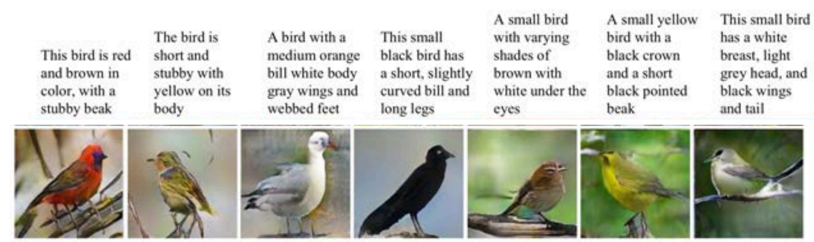


InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets, *Chen et al.*, 2016

music synthesis

MIDINET: A CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORK FOR SYMBOLIC-DOMAIN MUSIC GENERATION, *Yang et al.*, 2017

text to image synthesis



StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, *Zhang et al.*, 2016


2014

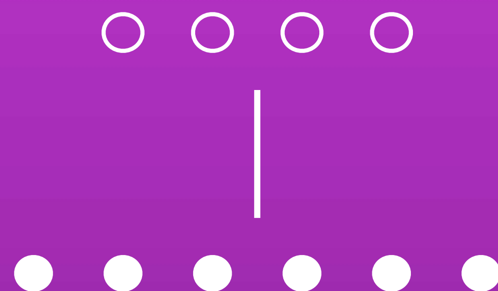

2015


2016


2017


2018

arxiv.org/abs/1406.2661
arxiv.org/abs/1511.06434
arxiv.org/abs/1606.07536
arxiv.org/abs/1710.10196
arxiv.org/abs/1812.04948



energy-based models

energy-based models

express a normalized distribution in terms of an *unnormalized* distribution

$$p(\mathbf{x}) = \frac{1}{Z}\tilde{p}(\mathbf{x})$$

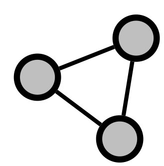
(partition function) $$Z = \int \tilde{p}(\mathbf{x}) d\mathbf{x}$$

energy-based models (or *Boltzmann machines*) define the unnormalized density as

$$\tilde{p}(\mathbf{x}) = \exp(-E(\mathbf{x}))$$

$E(\mathbf{x})$ is an *energy function*

this is a special case of an undirected graphical model



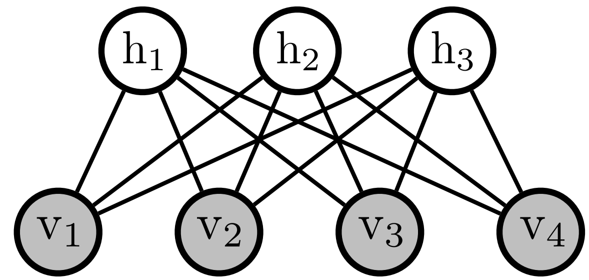
Deep Learning (Chapter 16), *Goodfellow et al.*, 2016

restricted Boltzmann machines (RBMs)

structure

restricted Boltzmann machines consist of visible (observed) units $\mathbf{v}$ and hidden (latent) units $\mathbf{h}$

connections are *restricted* to a bipartite graph:



the restricted graph structure allows us to express

$$p(\mathbf{h}|\mathbf{v}) = \prod_i p(\mathrm{h}_i|\mathbf{v}) \qquad p(\mathbf{v}|\mathbf{h}) = \prod_j p(\mathrm{v}_j|\mathbf{h})$$

functional form

define the energy function as

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

where $\mathbf{b}, \mathbf{c}, \mathbf{W}$ are learnable parameters

Deep Learning (Chapter 16), *Goodfellow et al.*, 2016

restricted Boltzmann machines (RBMs)

training

the linear energy function, $E(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^\intercal \mathbf{v} - \mathbf{c}^\intercal \mathbf{h} - \mathbf{v}^\intercal \mathbf{W} \mathbf{h}$, has simple derivatives, e.g.

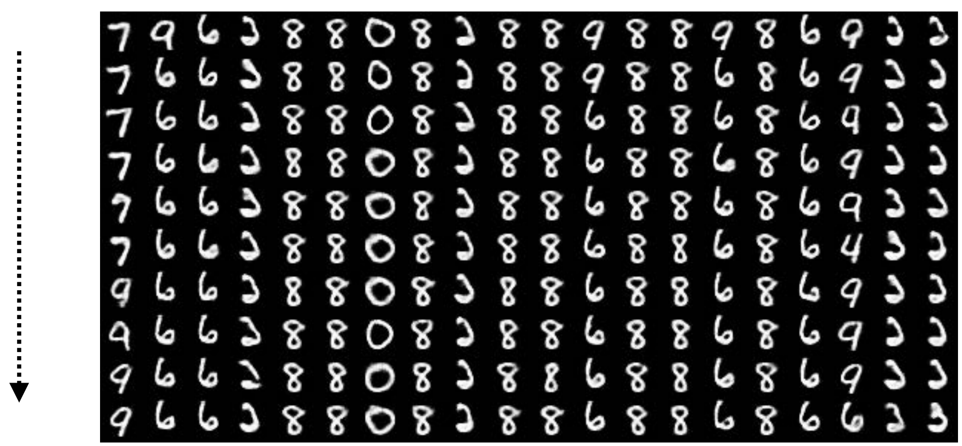
$$\frac{\partial}{\partial W_{i,j}} E(\mathbf{v}, \mathbf{h}) = -\mathrm{v}_i \mathrm{h}_j$$

can use of a variety of sampling-based training algorithms (see Chapter 18 of Goodfellow *et al.)*

⟶ contrastive divergence, stochastic maximum likelihood, score matching

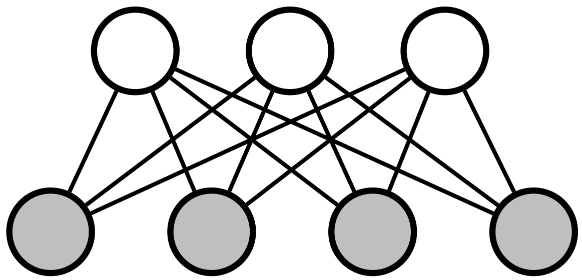
⟶ based on estimating $\nabla_\theta \log p_\theta(\mathbf{x})$ through sampling

sampling

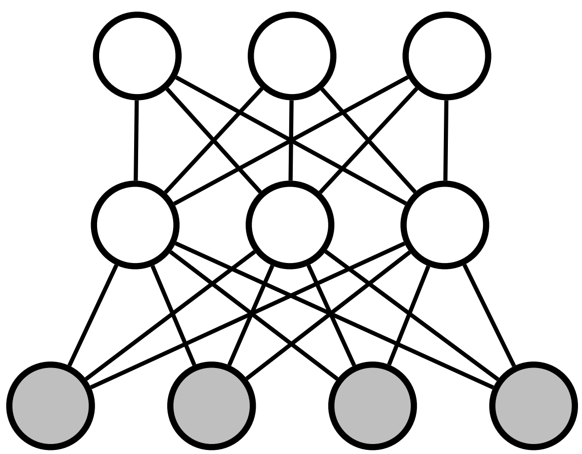


Deep Learning (Chapters 16, 18), *Goodfellow et al.*, 2016

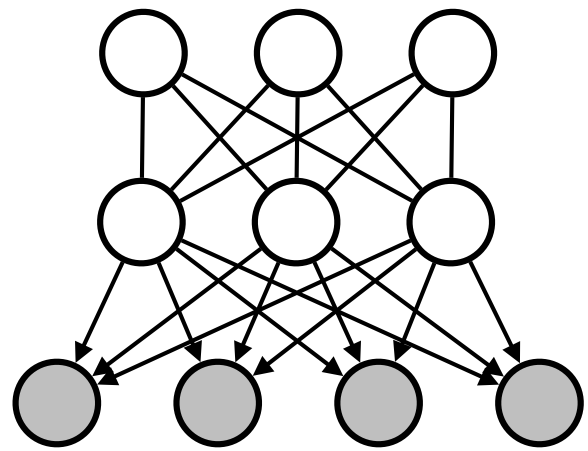
deep energy-based models



Restricted Boltzmann Machine



Deep Boltzmann Machine

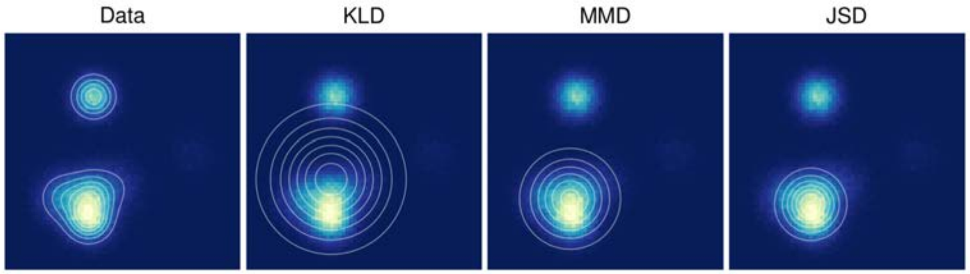


Deep Belief Network

Deep Learning (Chapter 20), *Goodfellow et al.*, 2016

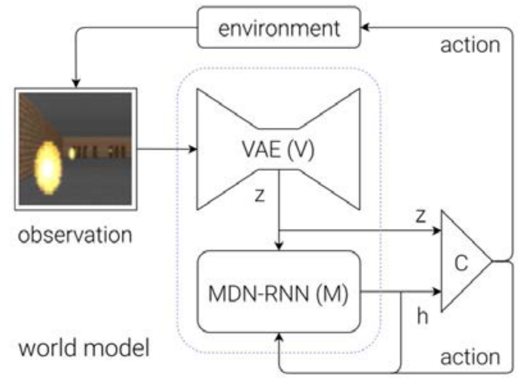
other topics

Generative Model Evaluation, Training Criteria



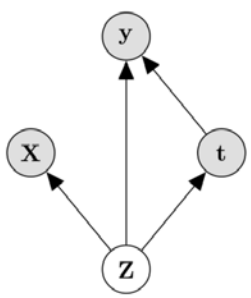
Theis *et al.*, 2016

Generative Models + RL



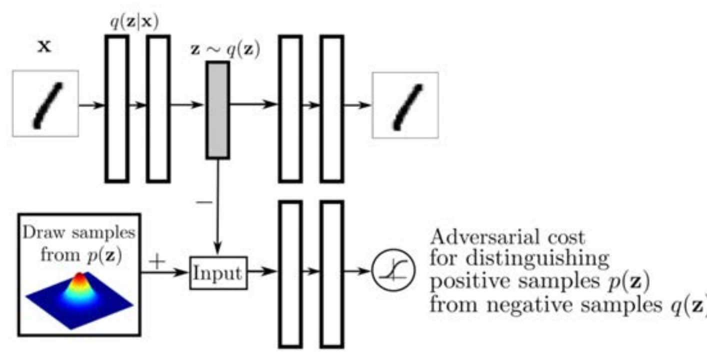
Ha & Schmidhuber, 2018

Causal Models



Louizos *et al.*, 2017

Combinations of Models



Makhzani *et al.*, 2016