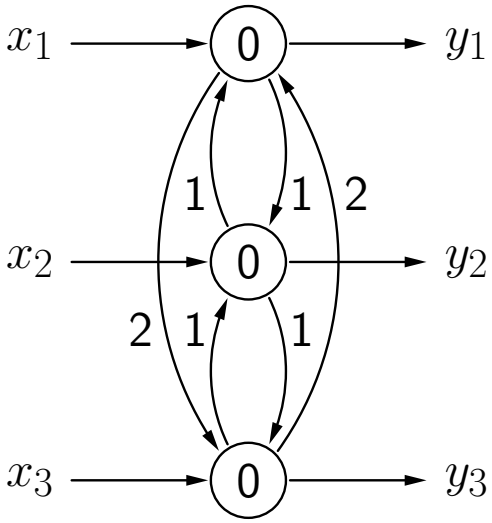
# Neuronale Netze
# Generative Modelle

Prof. Dr.-Ing. Sebastian Stober

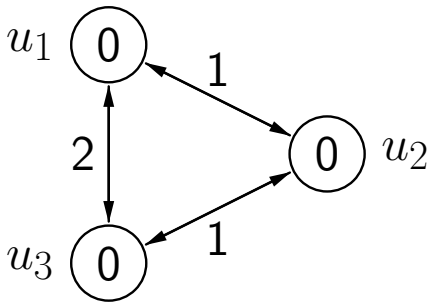
Artificial Intelligence Lab
Institut für Intelligente Kooperierende Systeme
Fakultät für Informatik
stober@ovgu.de


FACULTY OF
COMPUTER SCIENCE

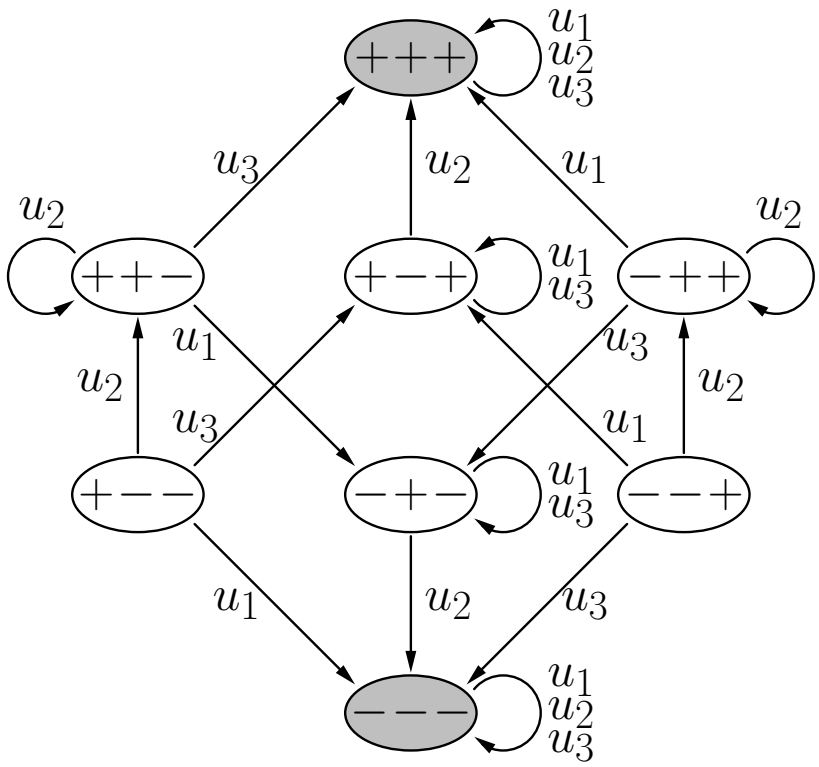
# Recap: Hopfield-Netze




vereinfachte Darstellung

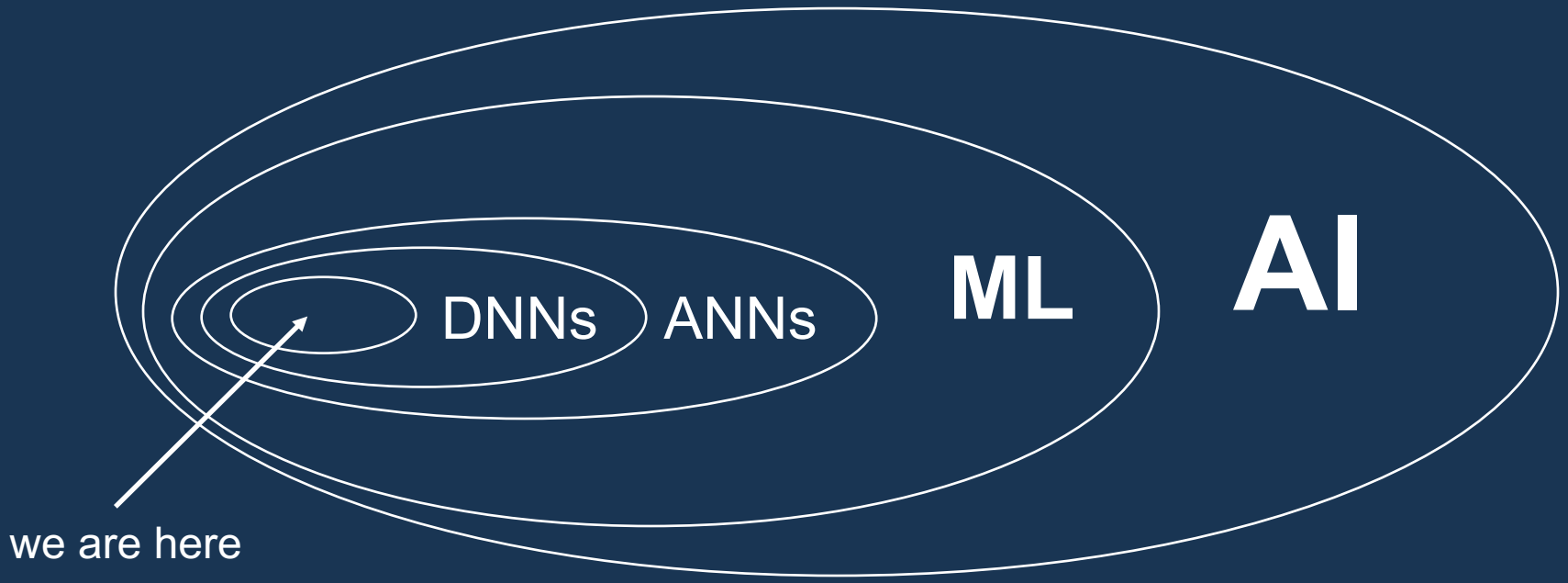



Zustandsgraph




- Konvergenz bei asynchroner Verarbeitung (in fester Reihenfolge)
- Endzustand (lokales) Energieminimum
- Verwendung als assoziativer Speicher oder zur Optimierung

# Generative Models



# 3 Phases of Deep Learning

1. unsupervised pre-training
   (~ 10 years ago)

2. end-to-end supervised training
   (~ 5 years ago)

3. generative training
   (today)

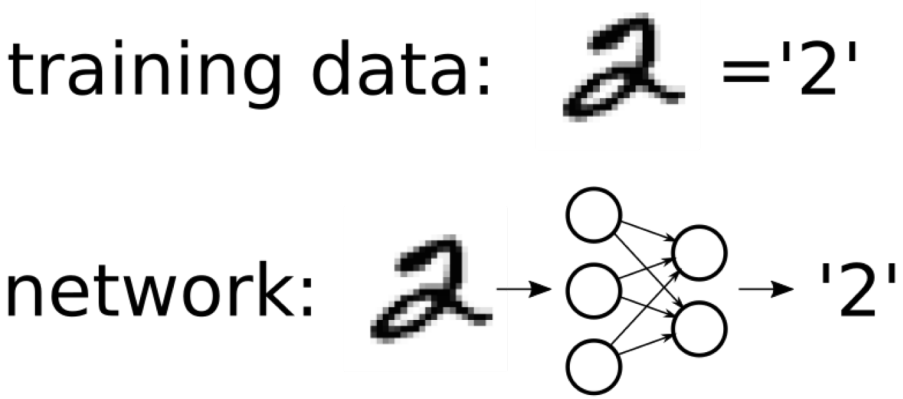
"What I cannot create,
I do not understand."

*Richard Feynman*

- very promising approach for
  - unsupervised learning
  - semi-supervised learning

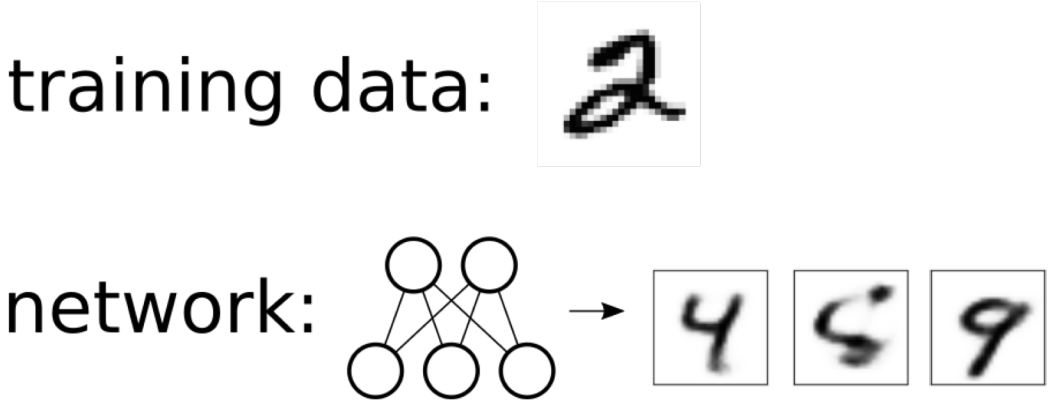
# Motivation

- mostly *supervised* learning in ML/DL
  = learning to predict true labels
  - need a lot of labeled data
  - discards information unrelated to prediction
  - need examples for all predictable outcomes



# Motivation

- generative models are *unsupervised*
  = learning to describe the data structure
  - labels are not necessary
  - all characteristics of the data need to be learned
  - anomalies in the data can be detected



# Generative Training

- Given training data, generate new samples from same distribution!

training data ~ $p_{data}(x)$

generated samples ~ $p_{model}(x)$

=> train $p_{model}(x)$ to approximate $p_{data}(x)$

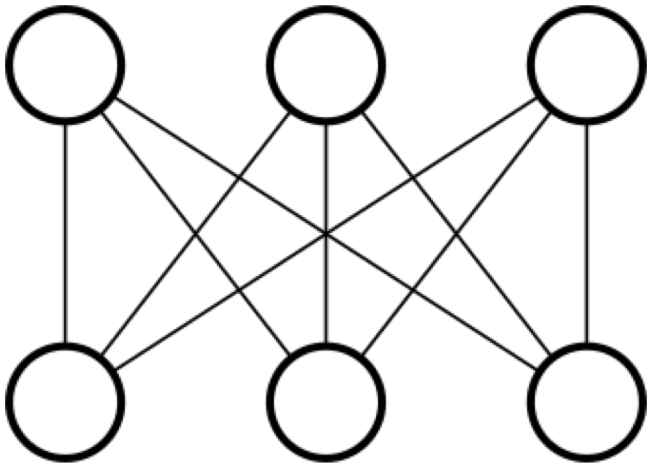
- here: capture dependencies between pixels

images from Denton et al. 2015

# Graphical Models

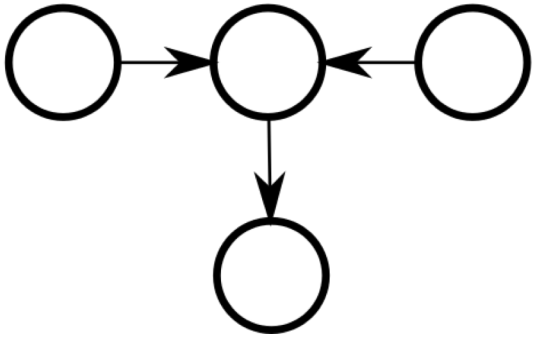
- full joint probability distribution not tractable
- model structure of the probability space, i.e. (conditional) independencies

undirected model



(energy-based)

directed model

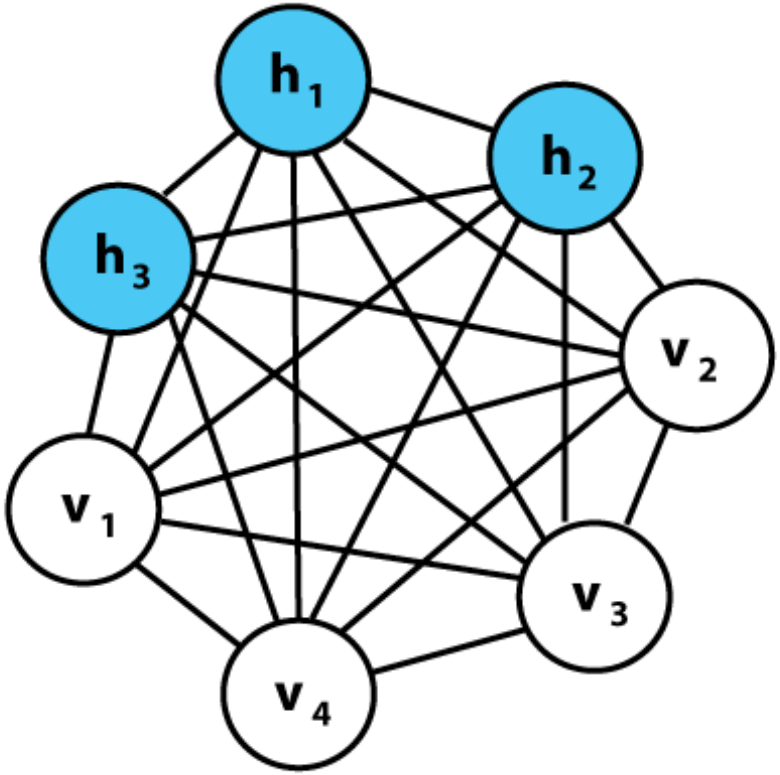


(causal structure)

undirected graphical models

# Boltzmann Machines

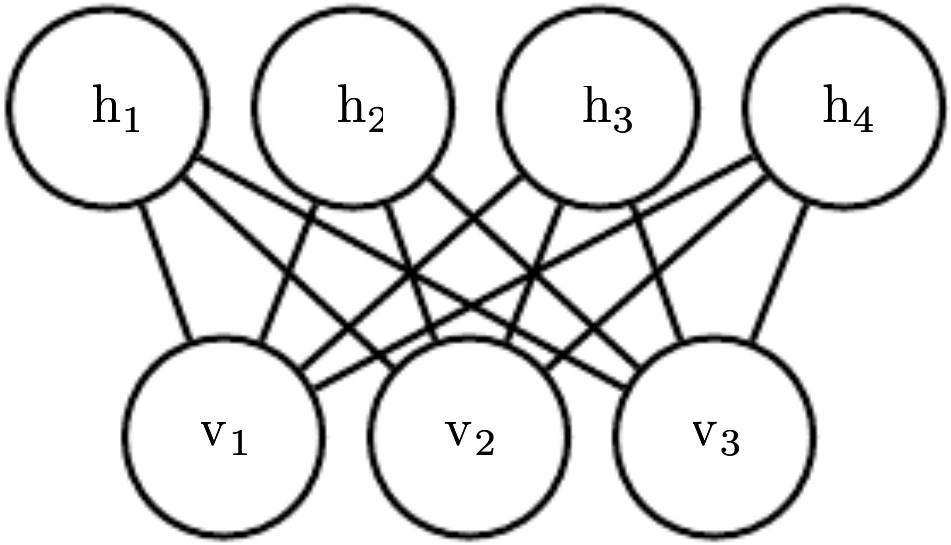
# Boltzmann-Maschinen (1985)




Keine Neuronen sondern Zufallsvariablen,
die sich gegenseitig beeinflussen!

Geoffrey Hinton
(Univ. of Toronto / Google)

# Restricted Boltzmann Machine (RBM)




**hidden variables**
(conditionally independent given visible variables)

**visible variables**
(conditionally independent given hidden variables)

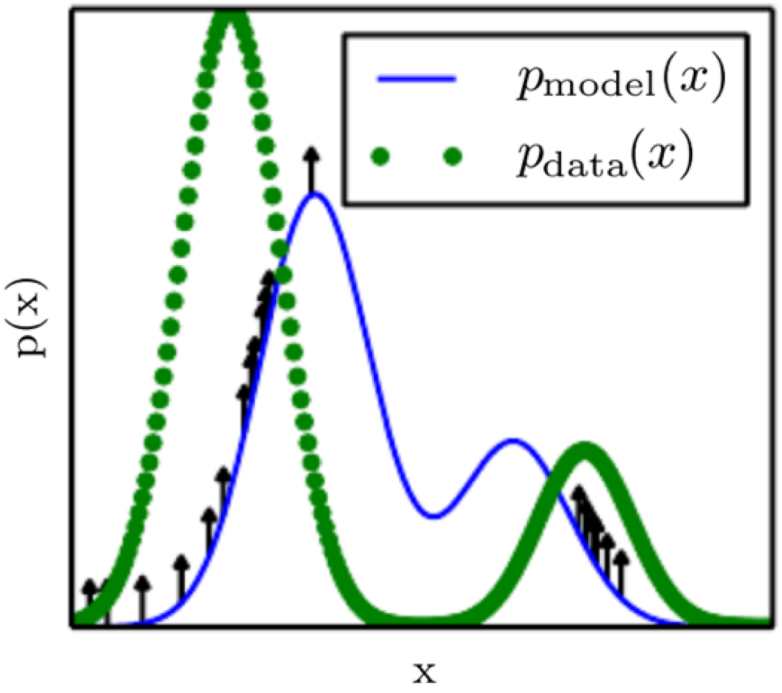
$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}$$

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

"partition function" – for probability normalization not tractable (sum over **many** values)

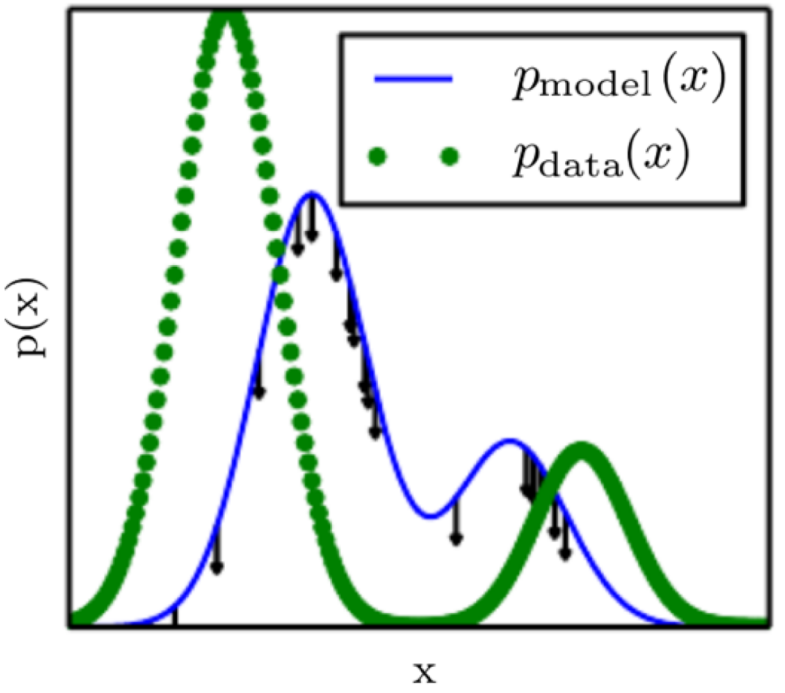
# RBM Training

positive phase




make samples from training data more likely

negative phase

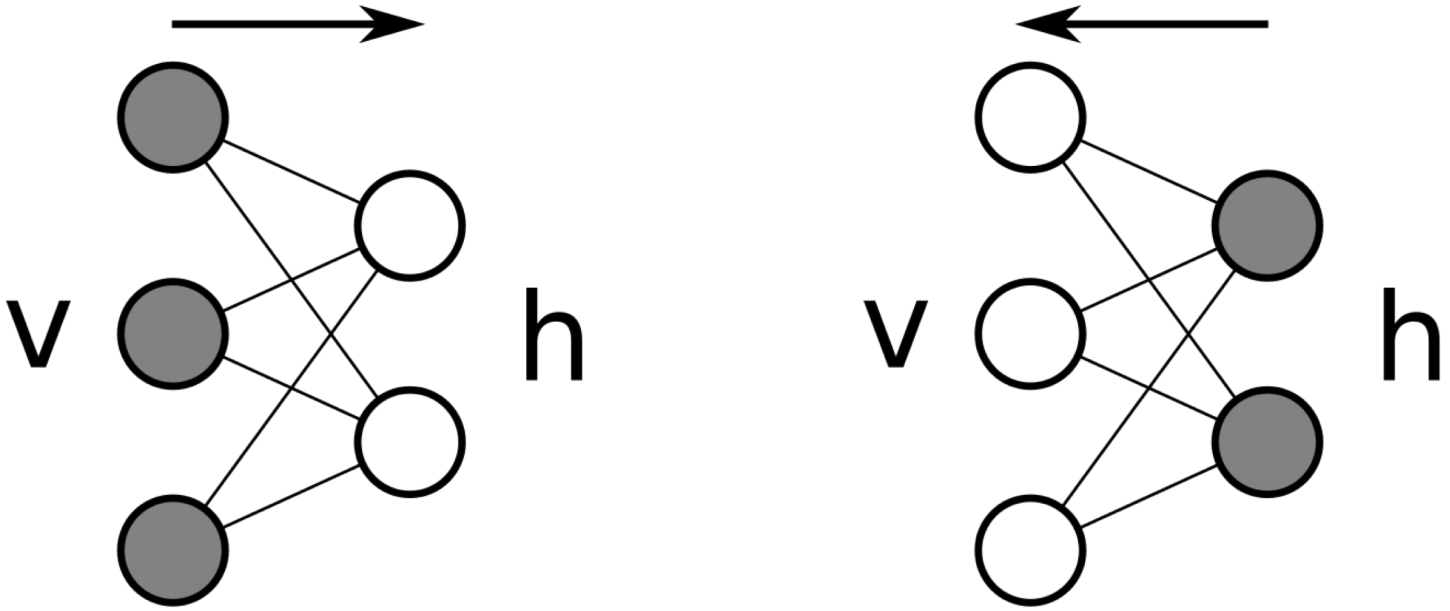



make samples from model less likely

[deeplearningbook.org]

# RBM Inference

infer unknown variable values given others

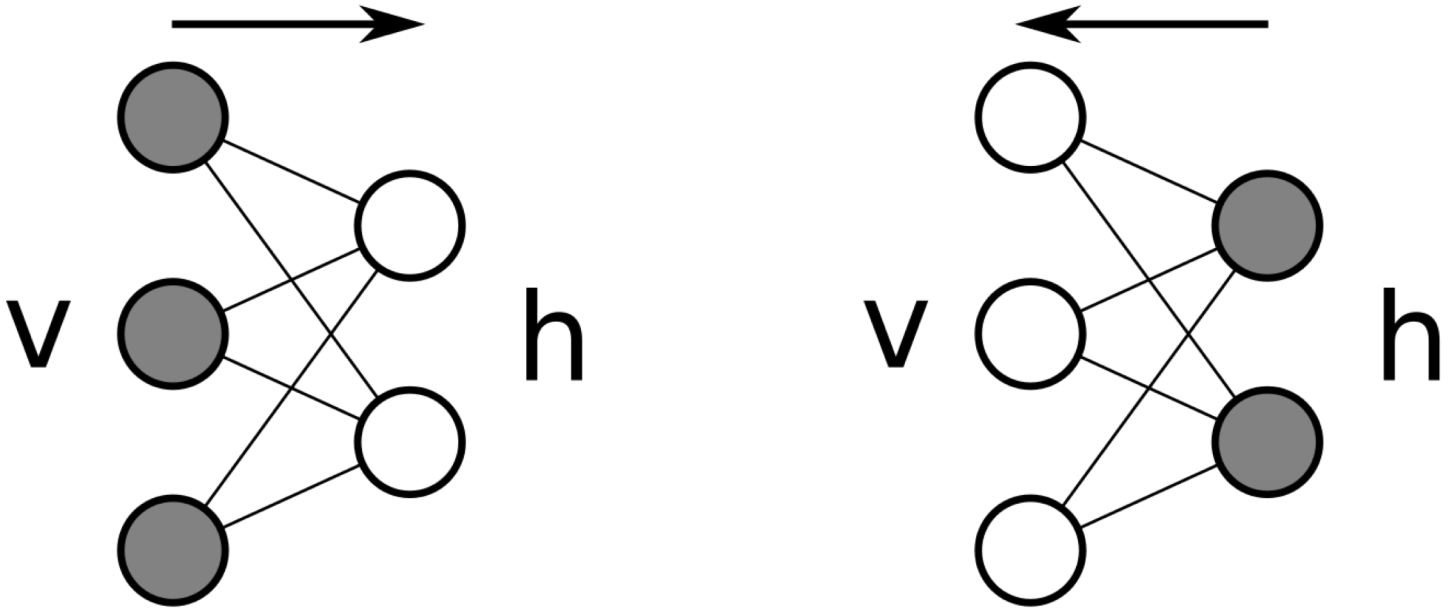



tractable (can be computed in parallel in 1 step because of conditional independence)

# RBM Gibbs Sampling

generate a sample (likely observed data)

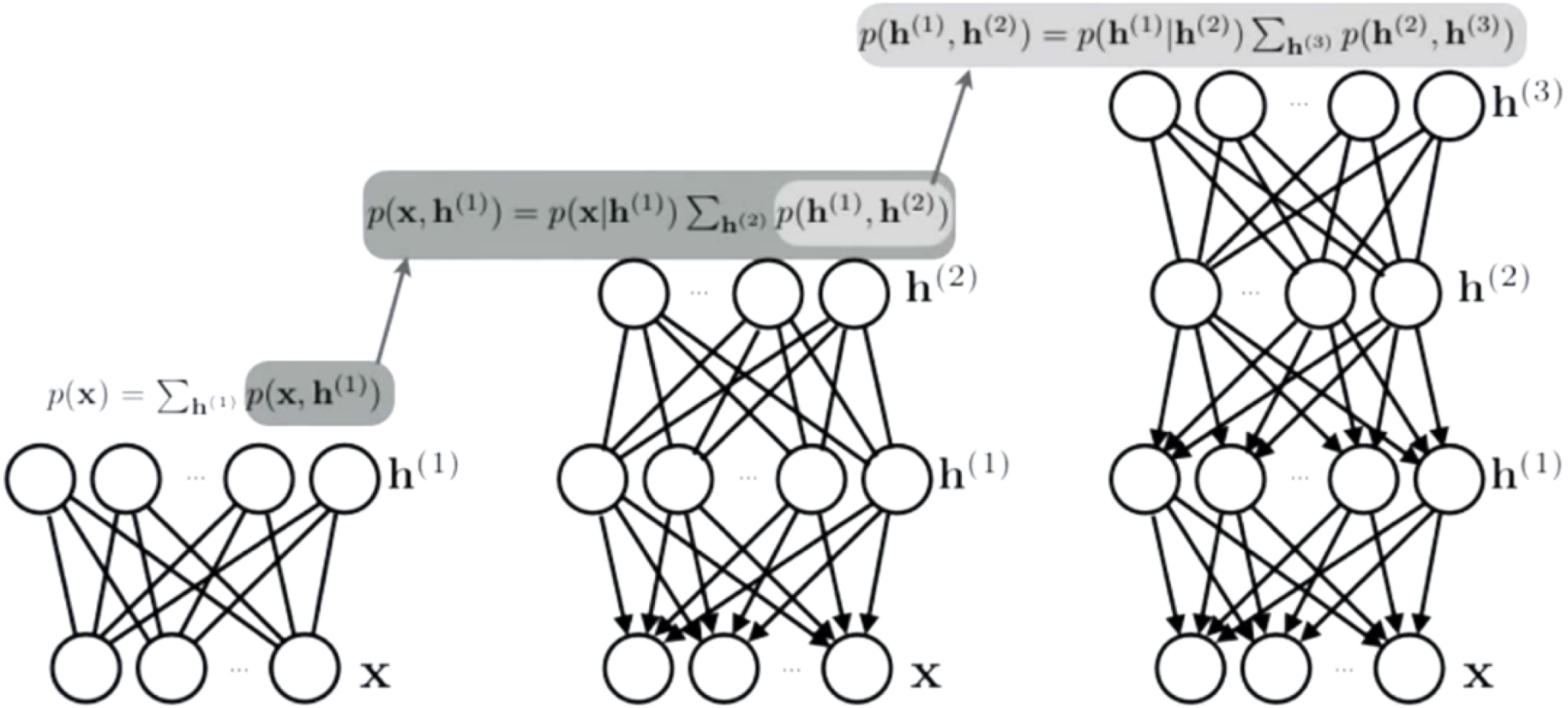
special case of Monte-Carlo Markov Chain (MCMC)




start with random initialization and go back and forth
(alternating parallel update of v given h and h given h)
until convergence (equilibrium)

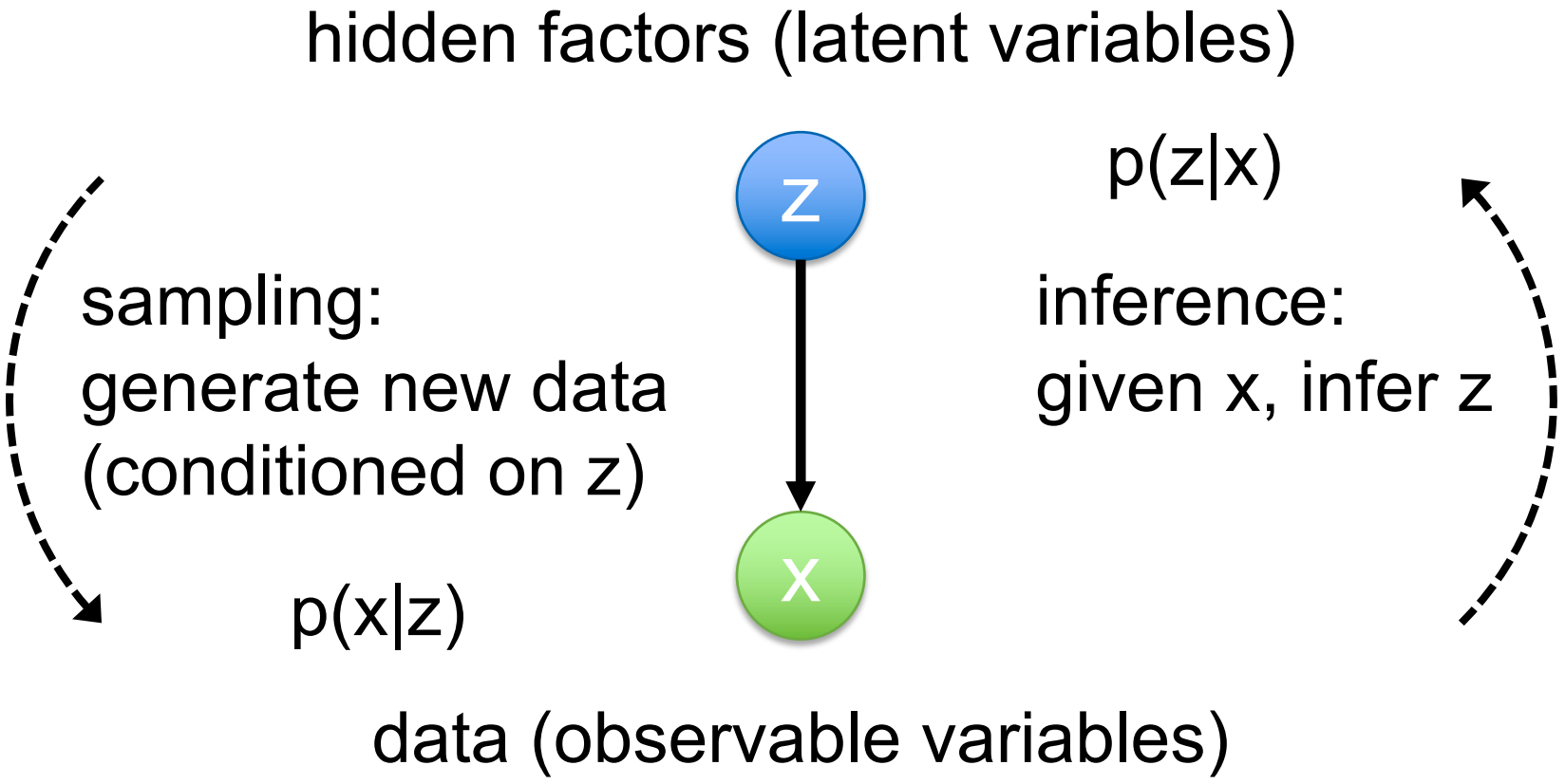
# Deep Belief Networks (DBNs)

stacked pre-training of RBMs



# Directed Graphical Models

# Directed Graphical Model




Use DNNs to parameterize and represent conditional distributions!

# Common Problems

- **strong assumptions** about the structure in the data
- **severe approximations**, leading to suboptimal models
- **computationally expensive** inference procedures like Markov Chain Monte Carlo (MCMC) during training

# Variational Autoencoders (VAEs)

# Common Problems

- **weak assumptions** about the structure in the data
- **approximation** (inference) introduces only small error given high-capacity model
- **fast training** using back-propagation

# Variational Autoencoder

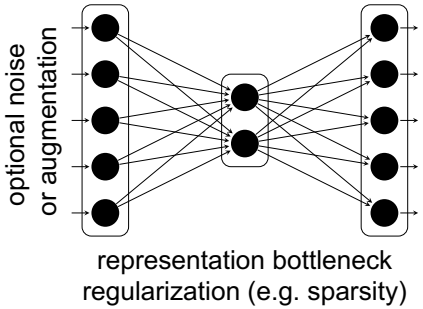
- basic idea:
  - combine the principle of auto-encoders with **variational Bayesian learning**
  - inputs, latent representations, and reconstructed outputs treated as **probabilistic, random variables** within a directed graphical model


- Kingma and Welling, *Auto-Encoding Variational Bayes*, *International Conference on Learning Representations (ICLR)* 2014.
- Rezende, Mohamed and Wierstra, *Stochastic back-propagation and variational inference in deep latent Gaussian models*. ArXiv, 2014.

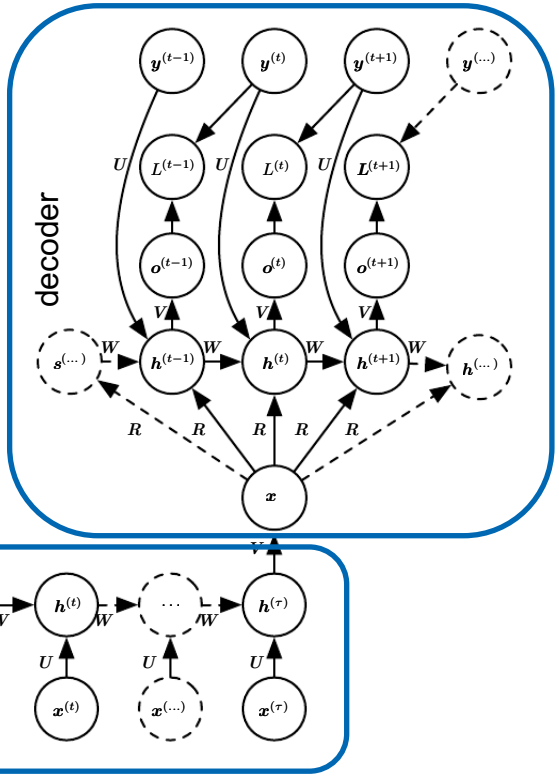
# Recap: Autoencoders

**structure:** encoder + decoder
- arbitrarily complex
- often symmetrical

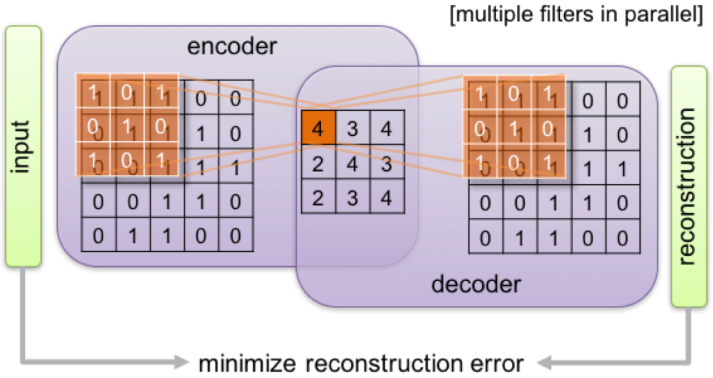
**objective:** reconstruct inputs
- often under constraints (capacity, sparsity etc.)



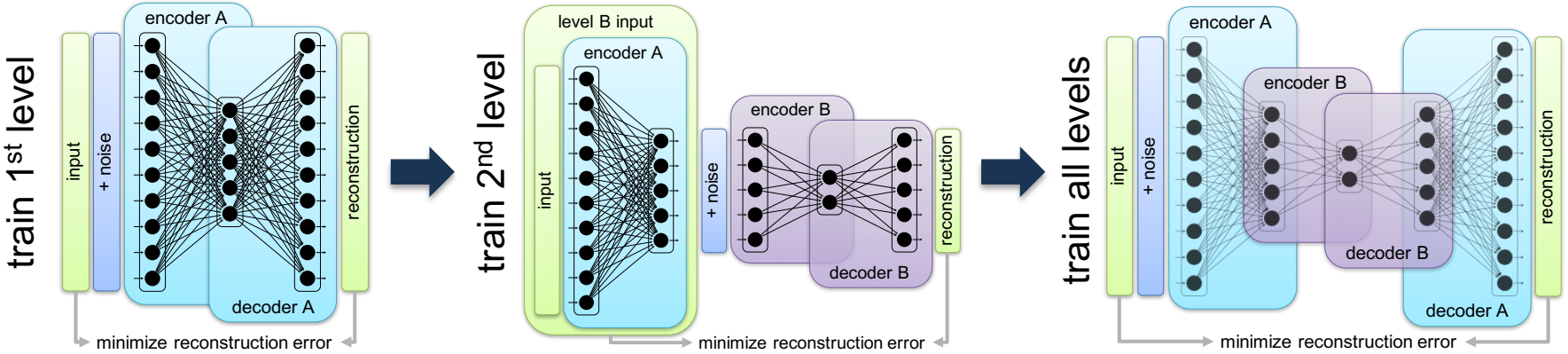

## Recurrent Autoencoder



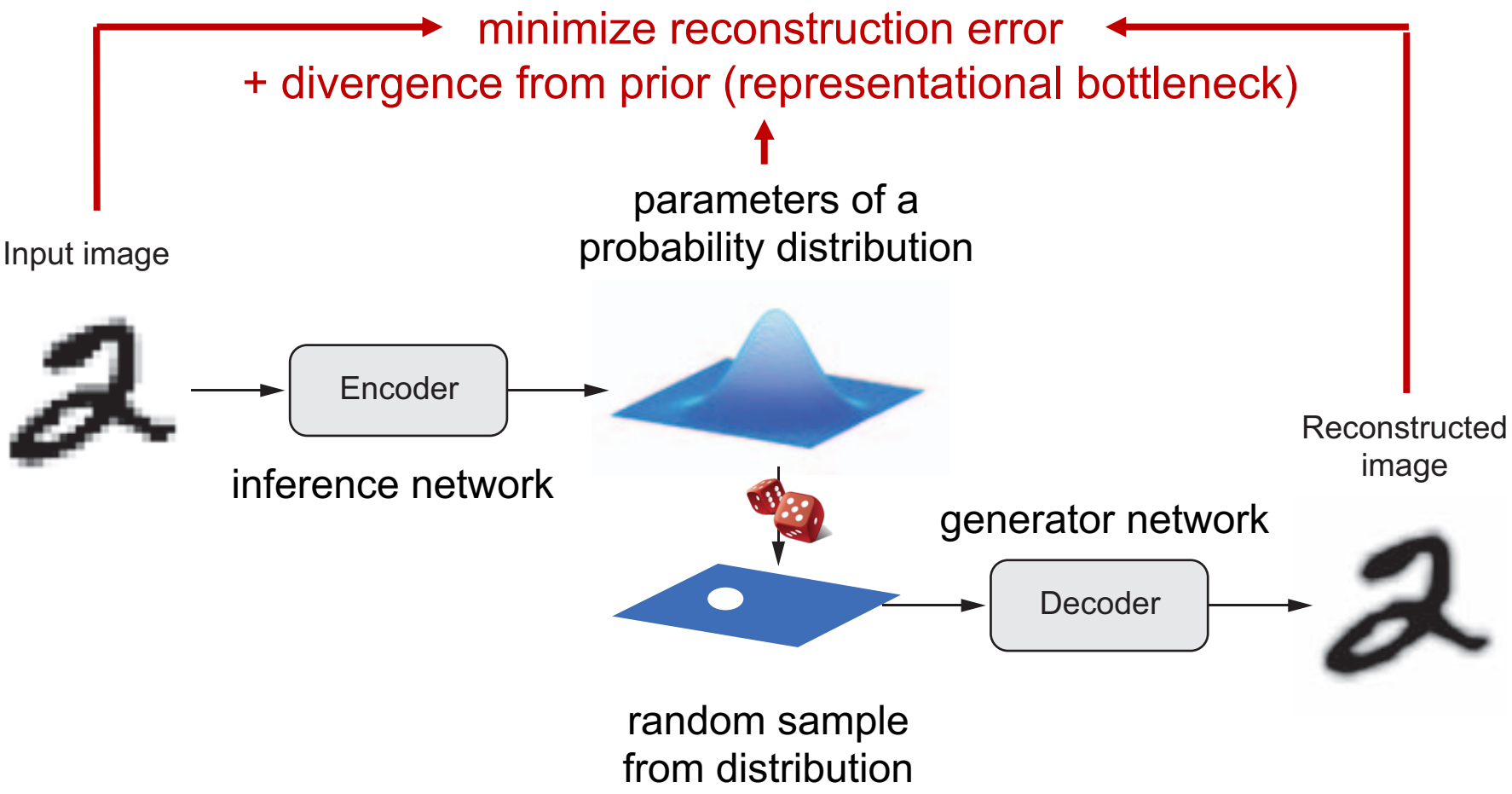

## Convolutional Autoencoder




## Stacked (Denoising) Autoencoder



# Variational Autoencoder (VAE)




adapted from F. Chollet (2017) “Deep Learning with Python”, Manning

# Training Objective: Maximum Likelihood

maximize data probability for training set under entire generative process:

$$p(x) = \int p_{\boldsymbol{\theta}}(x \mid z)\, p(z)\, dz$$

generator net parameters

practical problems:

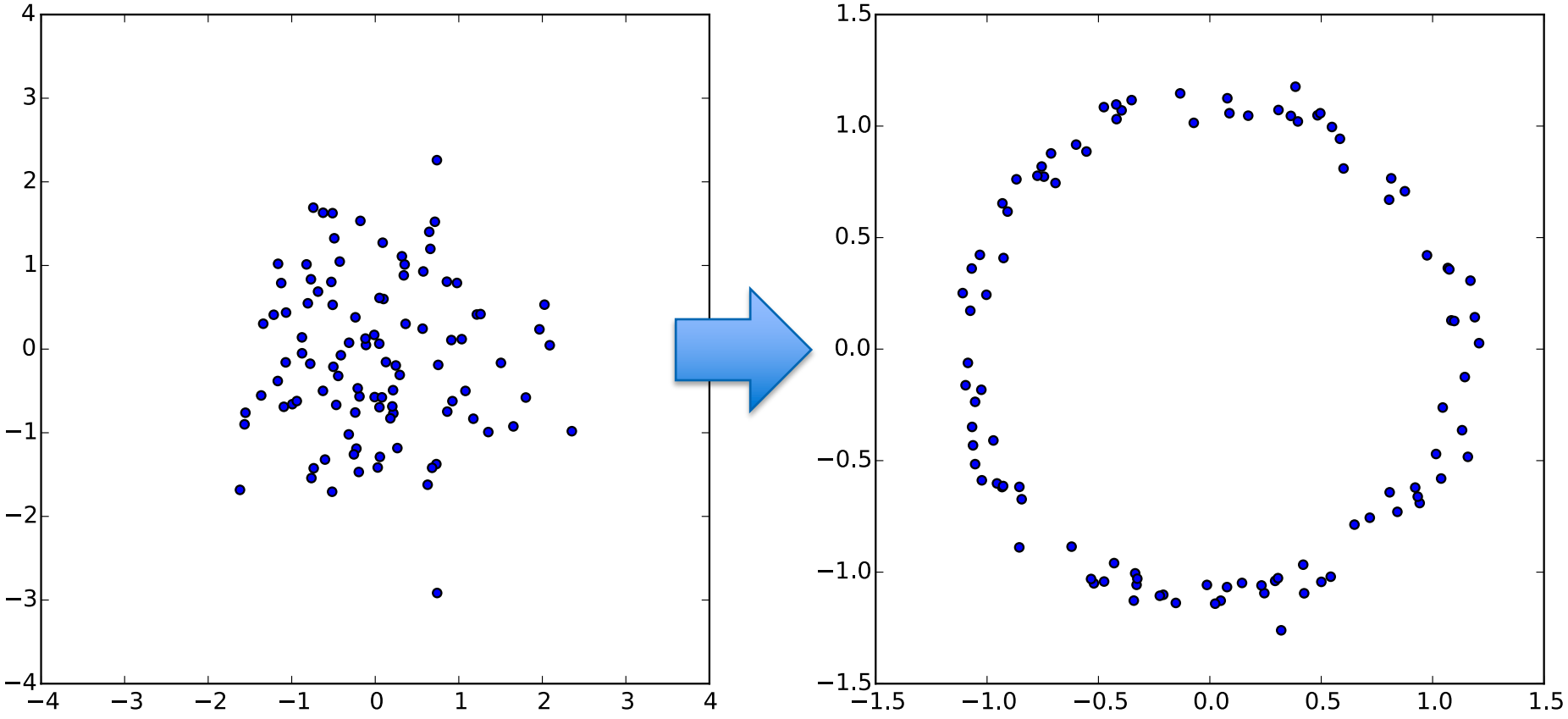
1. How to define latent variables z?
2. How to deal with the integral over z?

# 1. Latent Variables

- assume that there is no simple interpretation of the dimensions of z
- instead assert that samples of z can be drawn from a **simple distribution**
  e.g. N(0,I)

Note: This is the (rather) weak assumption we need to make and a potential weak spot.

# Transforming Distributions

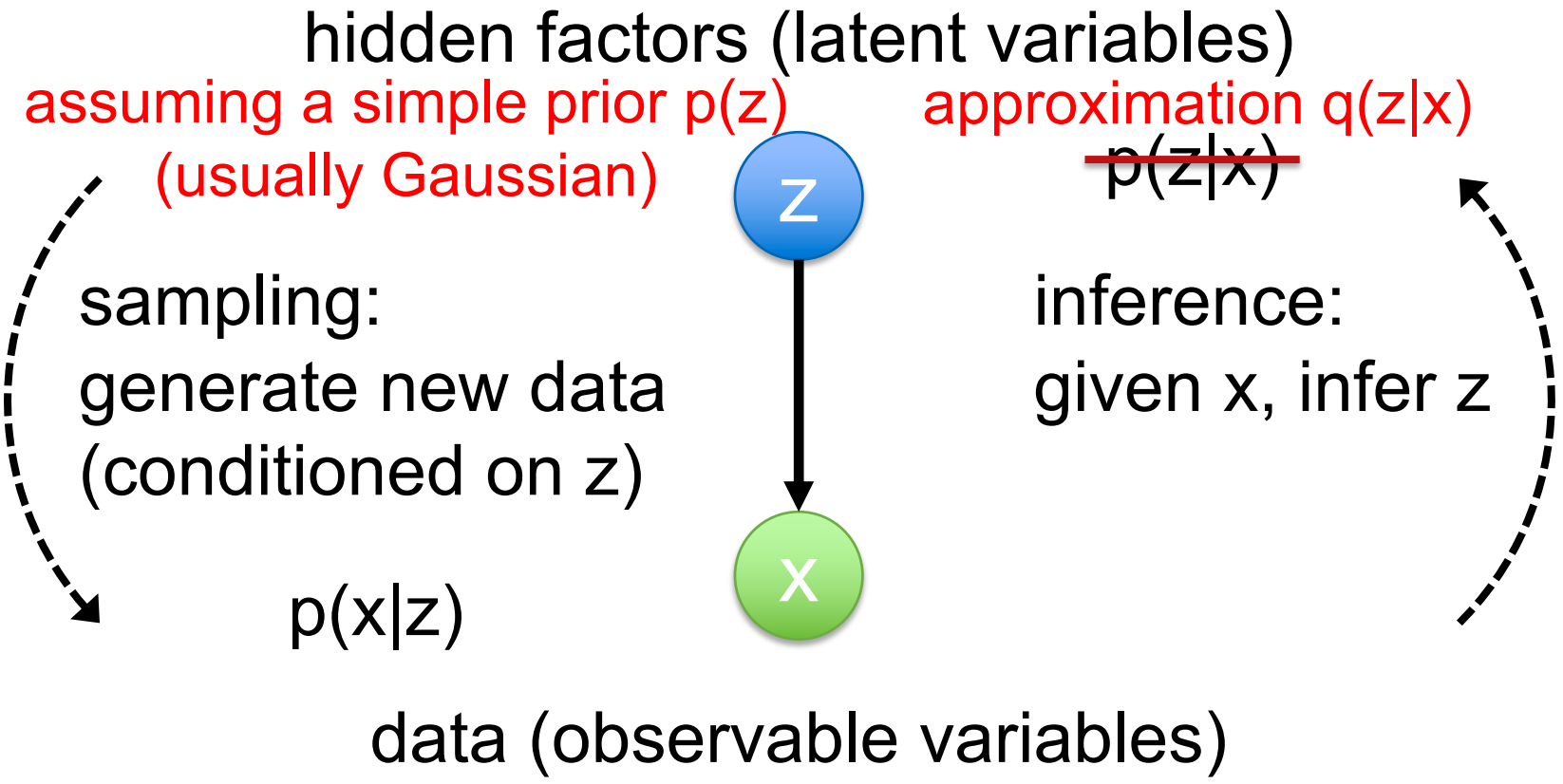



$$g(z) = z/10 + z/||z||$$

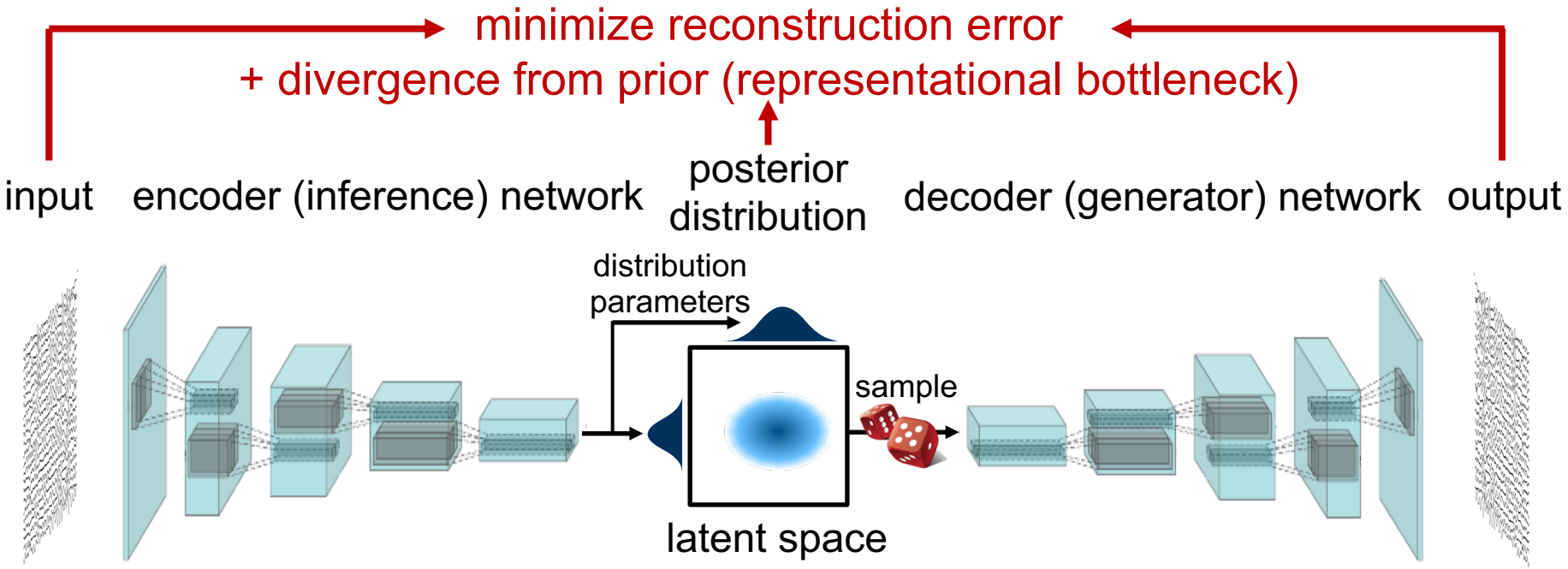
# 2. Intractability

- $p(x) = \int p_{\theta}(x \mid z)\, p(z)\, dz$ is intractable
- but $p_{\theta}(x \mid z) \approx 0$ for most z
- idea: attempt to sample **values of z that are likely to have produced x**, and compute p(x) just from those
- **new function $q_{\phi}(z \mid x)$** to return likely z for given x (learned approximate inference)

# Directed Graphical Model



# Variational Autoencoder

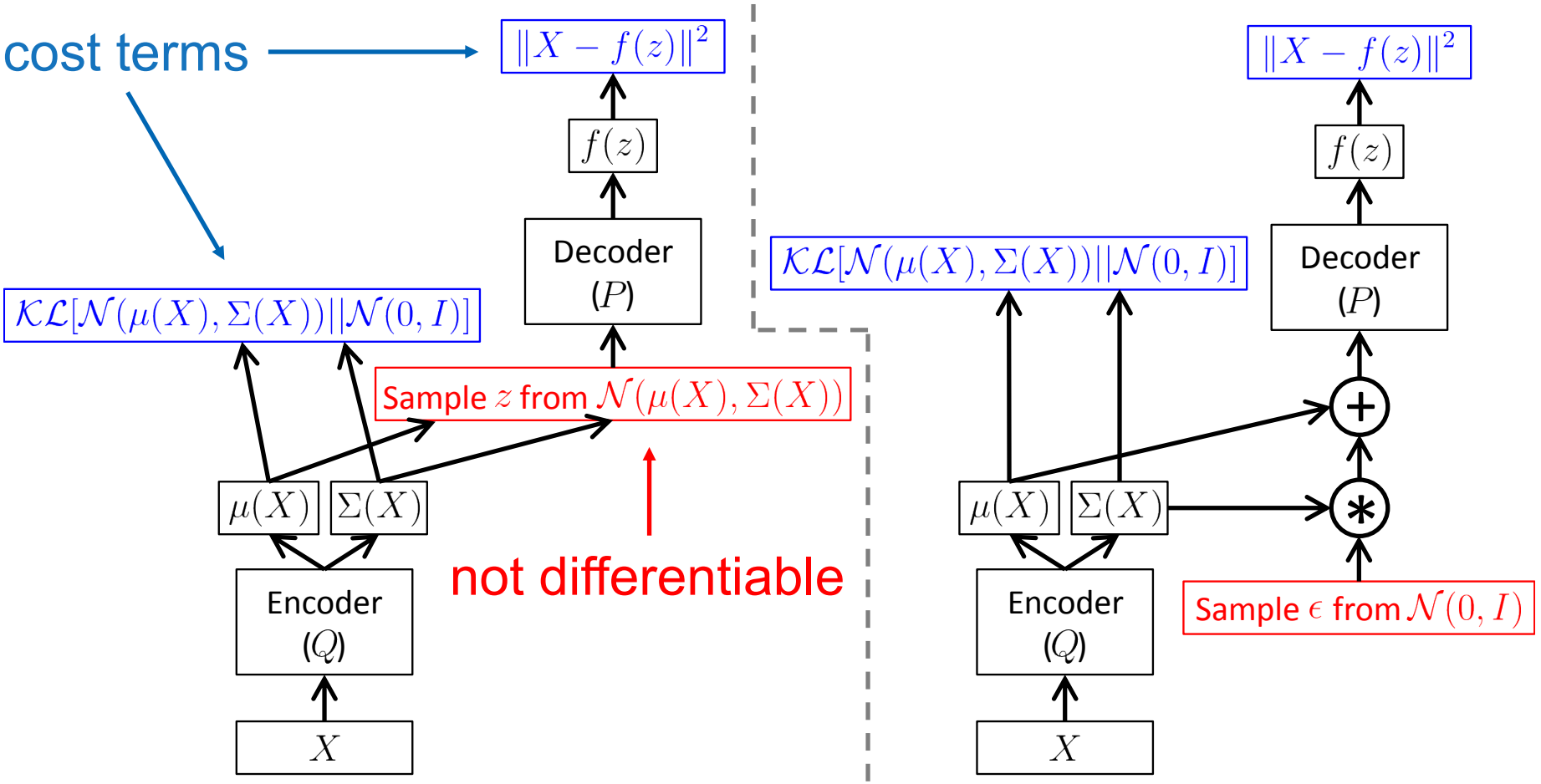



x q(z|x) p(x|z)

variational approximation
of the true posterior p(z|x)

problem for backpropagation: sampling operation (🎲) not differentiable

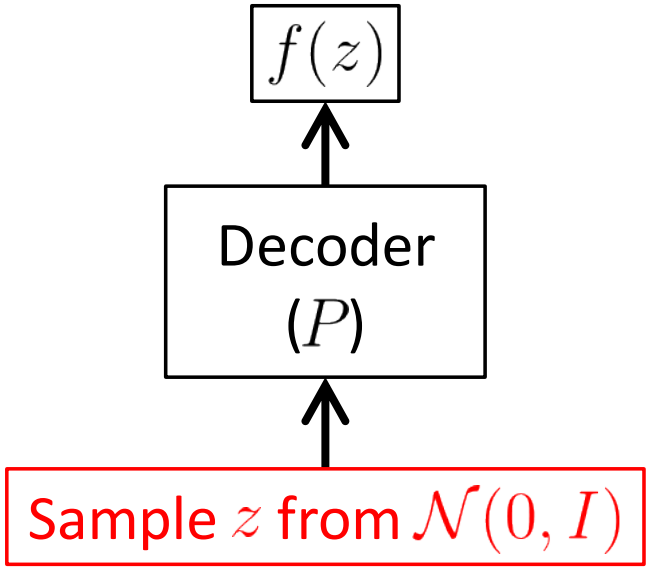
# Reparametrization Trick




https://arxiv.org/abs/1606.05908

adapted from Doersch 2016

# VAE Sampling



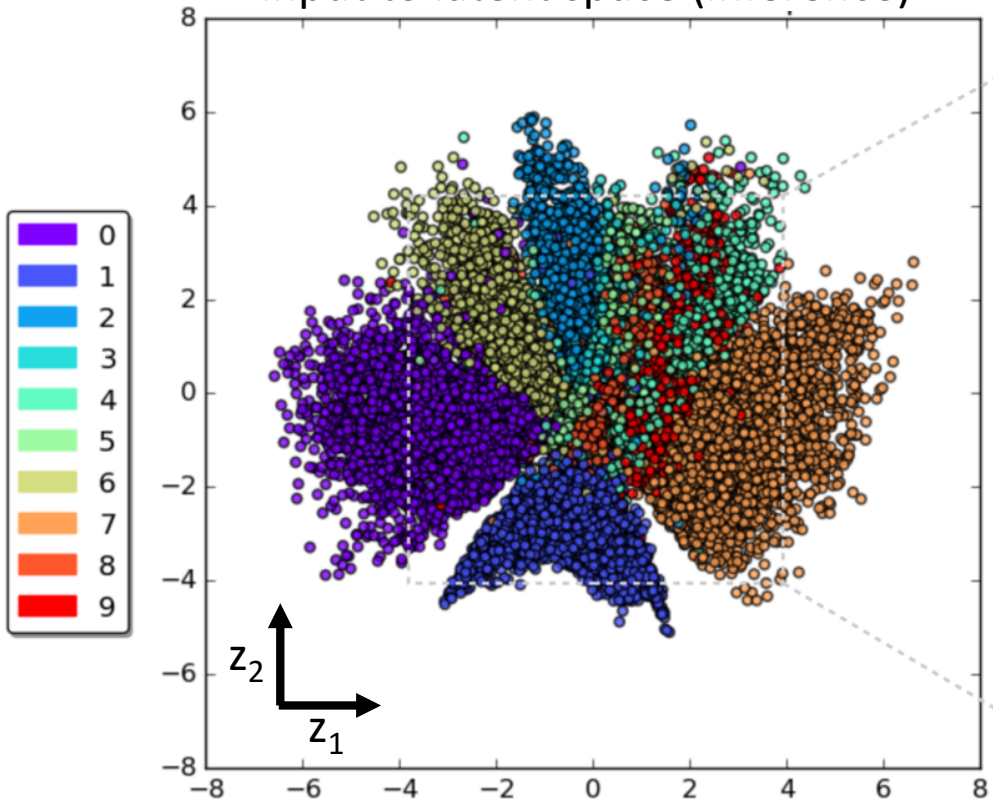

https://arxiv.org/abs/1606.05908

Doersch 2016

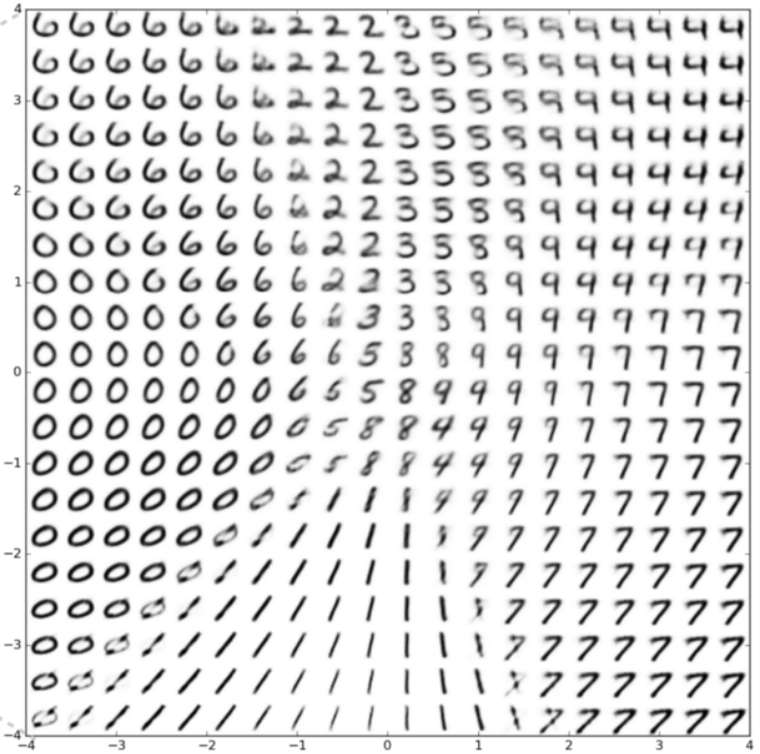
# VAE Introspection

## Latent Space Visualization (for MNIST dataset)

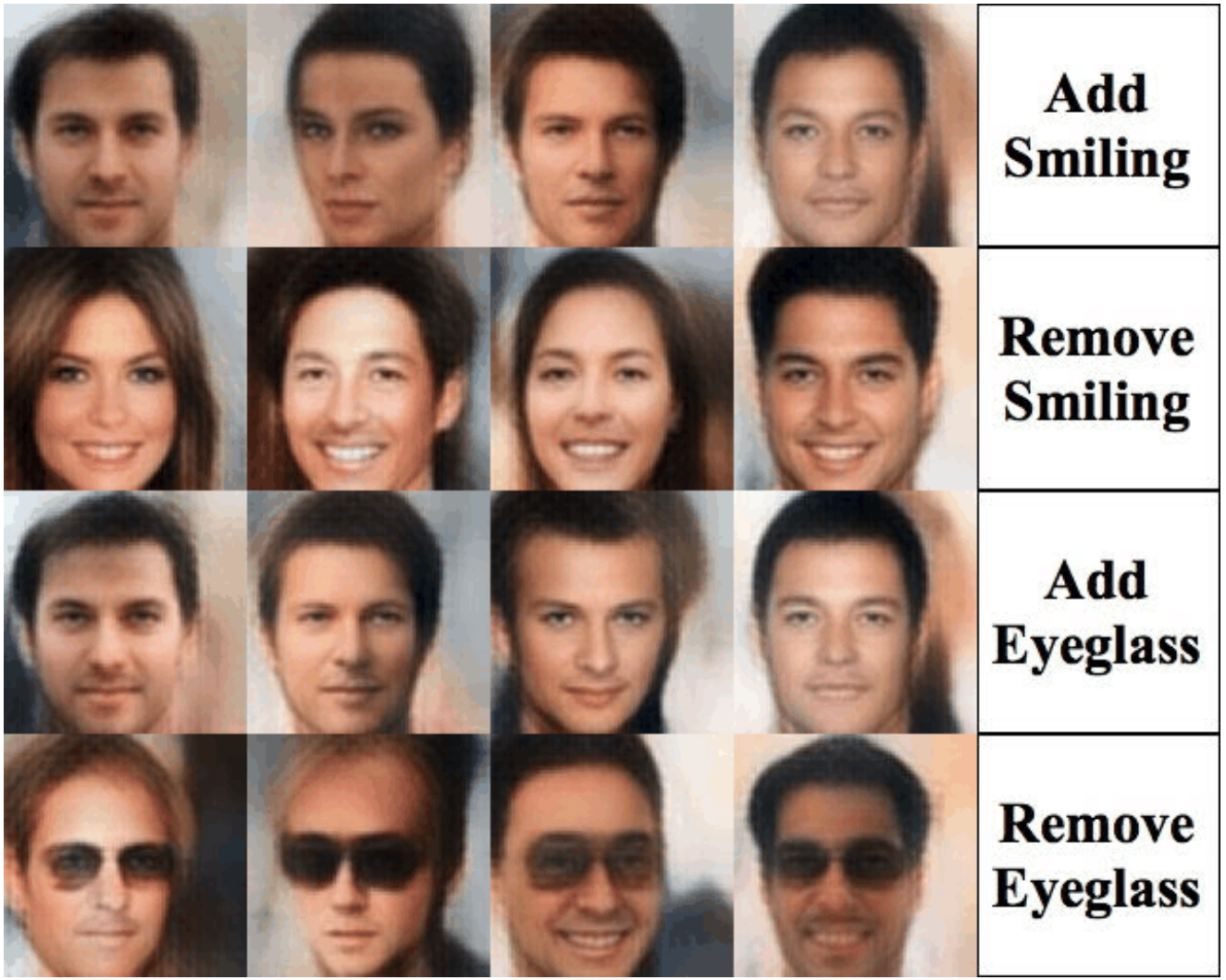
input to latent space (inference)




latent space to output (generation)



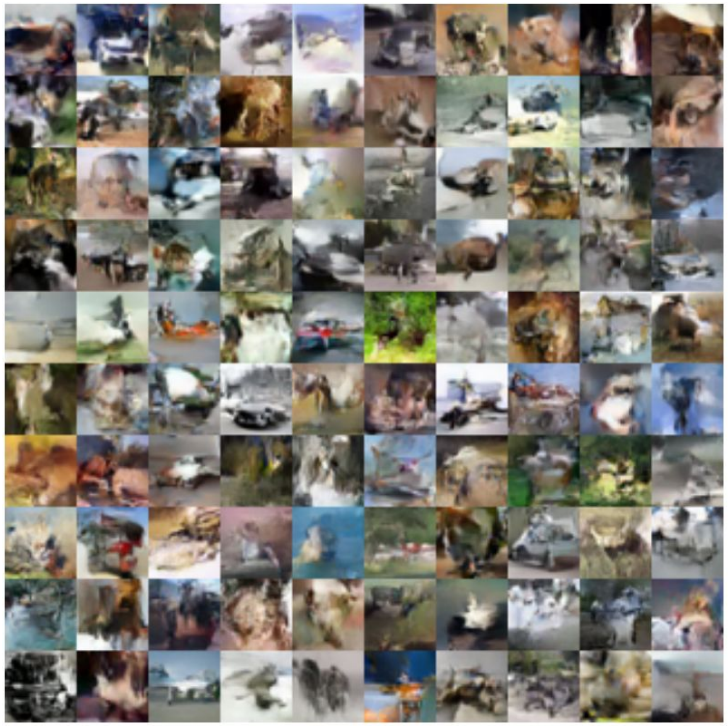
# Fun with Latent Vectors




https://houxianxu.github.io/assets/project/dfcvae

# Room for Improvements

- generated images tend to be blurry



Kingma et al. 2016

Larsen et al. 2017

# VAEs Summary

- probabilistic spin to traditional auto-encoders
- optimize a (variational) lower bound

**pros:**

- principled approach to generative models
- inference of q(z|x) can be useful feature representation

**cons:**

- maximizes only approximation (lower bound) of likelihood
- samples blurrier and lower quality compared to GANs

**active areas of research:**

- more flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian
- incorporating structure in latent variables

# Generative Adversarial Networks (GANs)

# Generative Adversarial Net

- basic idea:
  - if we don't care about inference, we can try to sample from training distribution directly
  - game-theoretic approach

Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

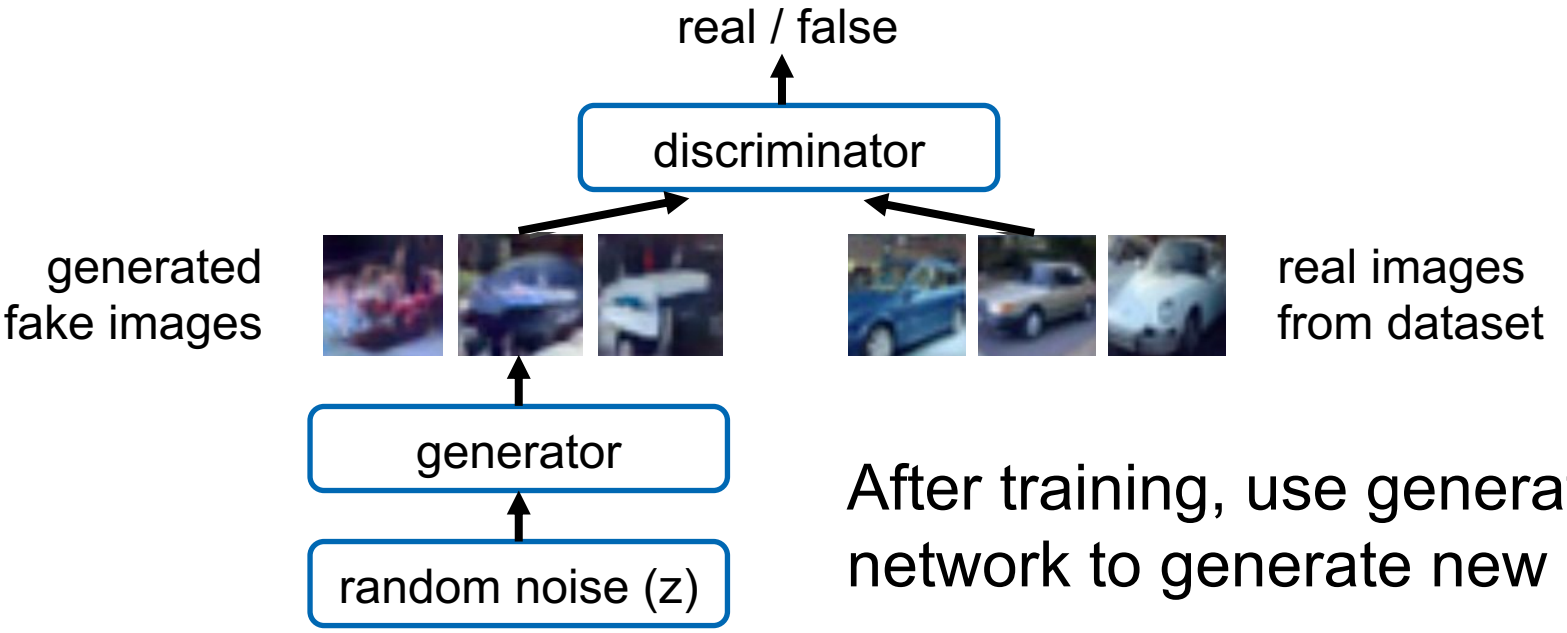
# Generative Adversarial Net

- problem: we want to sample from complex, high-dimensional training distribution
  No direct way to do this!
- solution:
  - sample from a simple distribution, e.g. random noise
  - learn transformation to training distribution
  - use DNN for the complex transformation

# Two-Player Game

**generator net** (like VAE decoder):
try to fool the discriminator by generating real-looking data

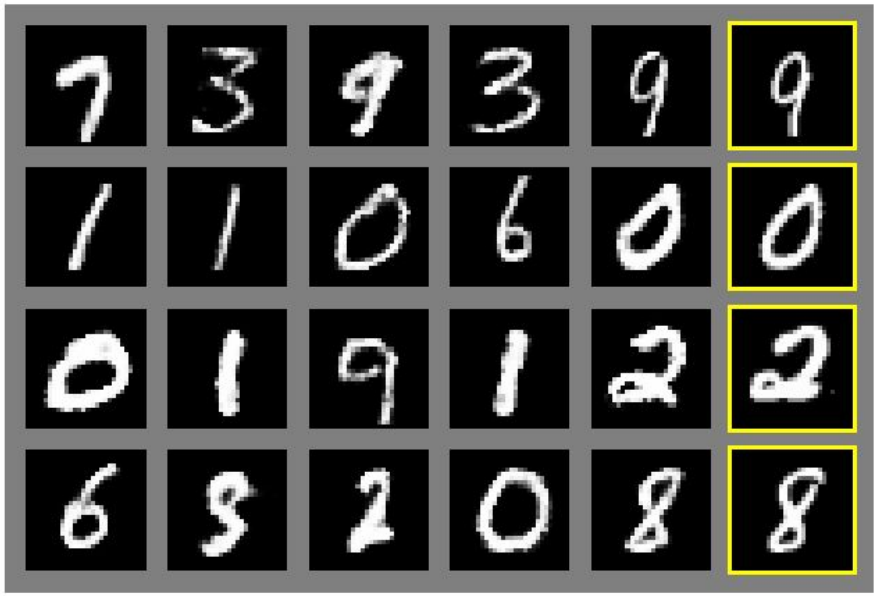
**discriminator net**:
try to distinguish between real and fake data

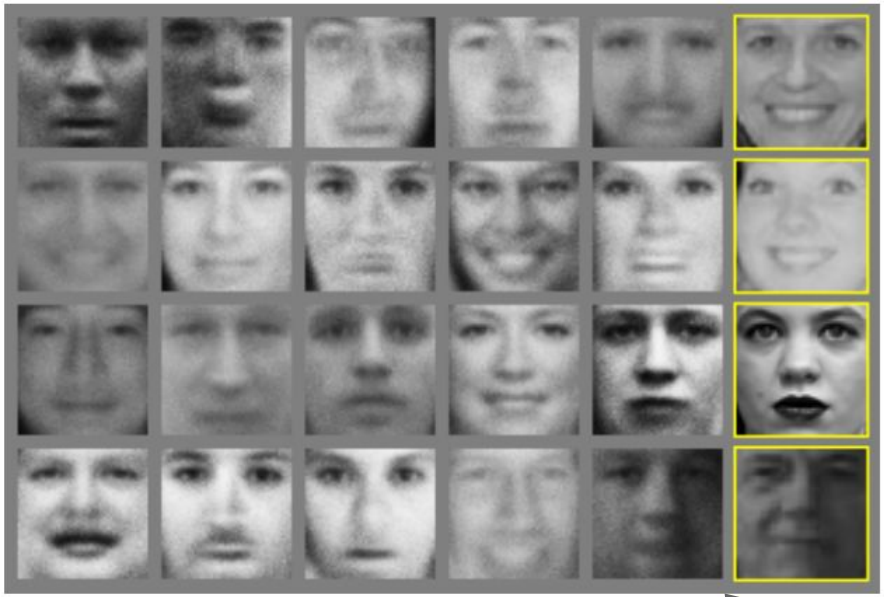



After training, use generator network to generate new data.

images from Denton et al. 2015

# Generated Samples



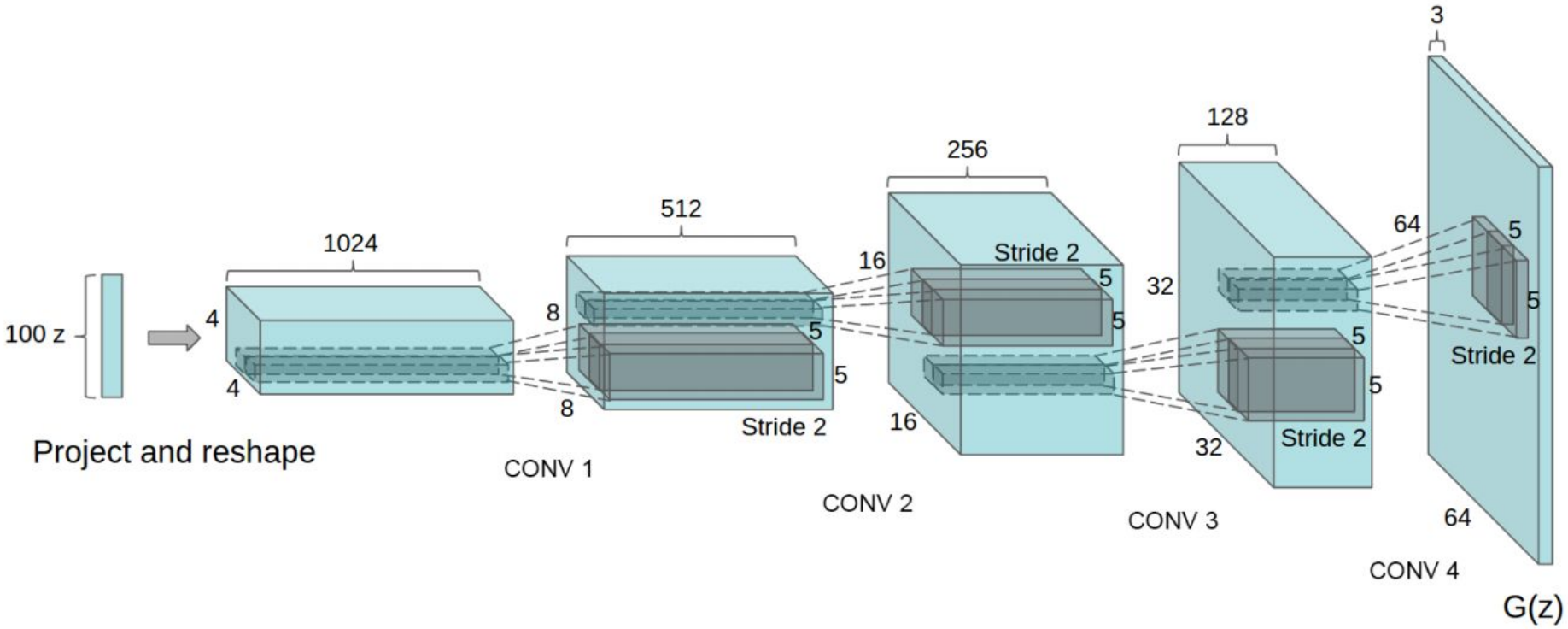


(from the original publication)

https://arxiv.org/abs/1406.2661

Goodfellow et al. 2014

# CNN Architectures (DCGAN)




Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

# DCGAN Samples

https://github.com/Newmu/dcgan_code

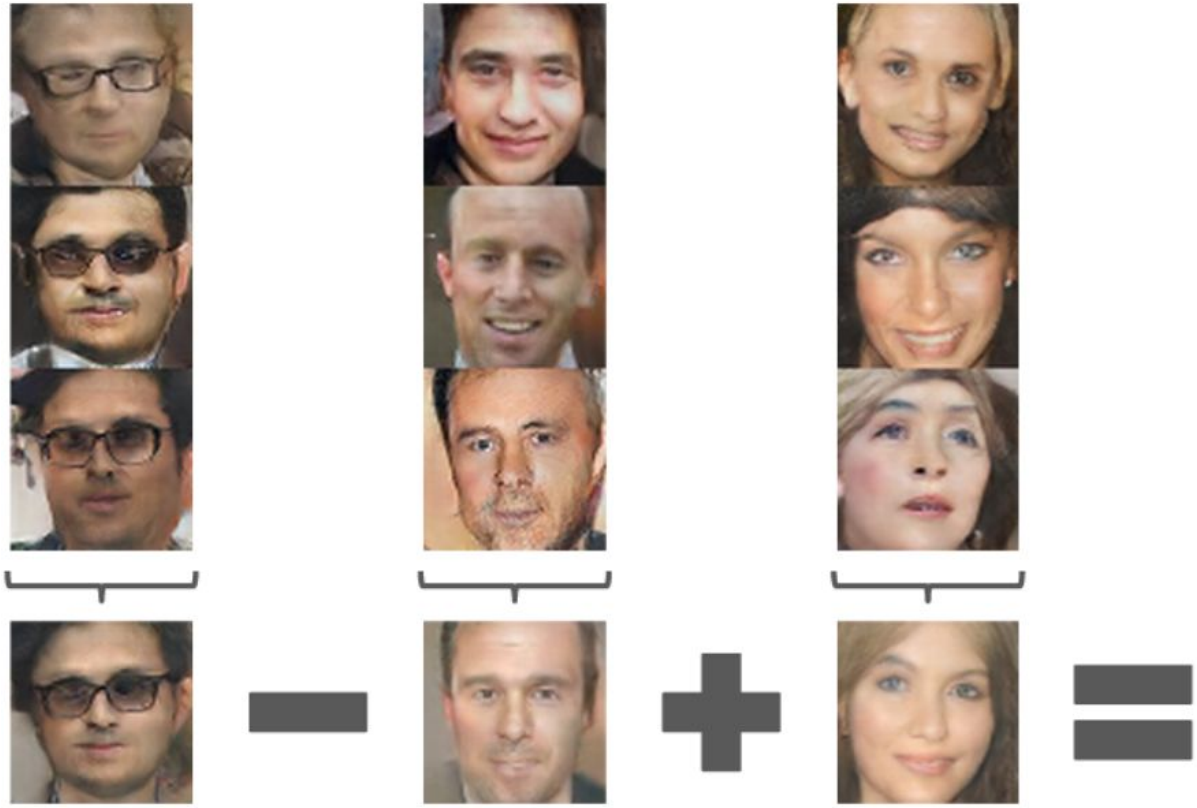
Radford et al. 2016

# Fun with Latent Vectors

man
with glasses

man
no glasses

woman
no glasses



woman
whit glasses

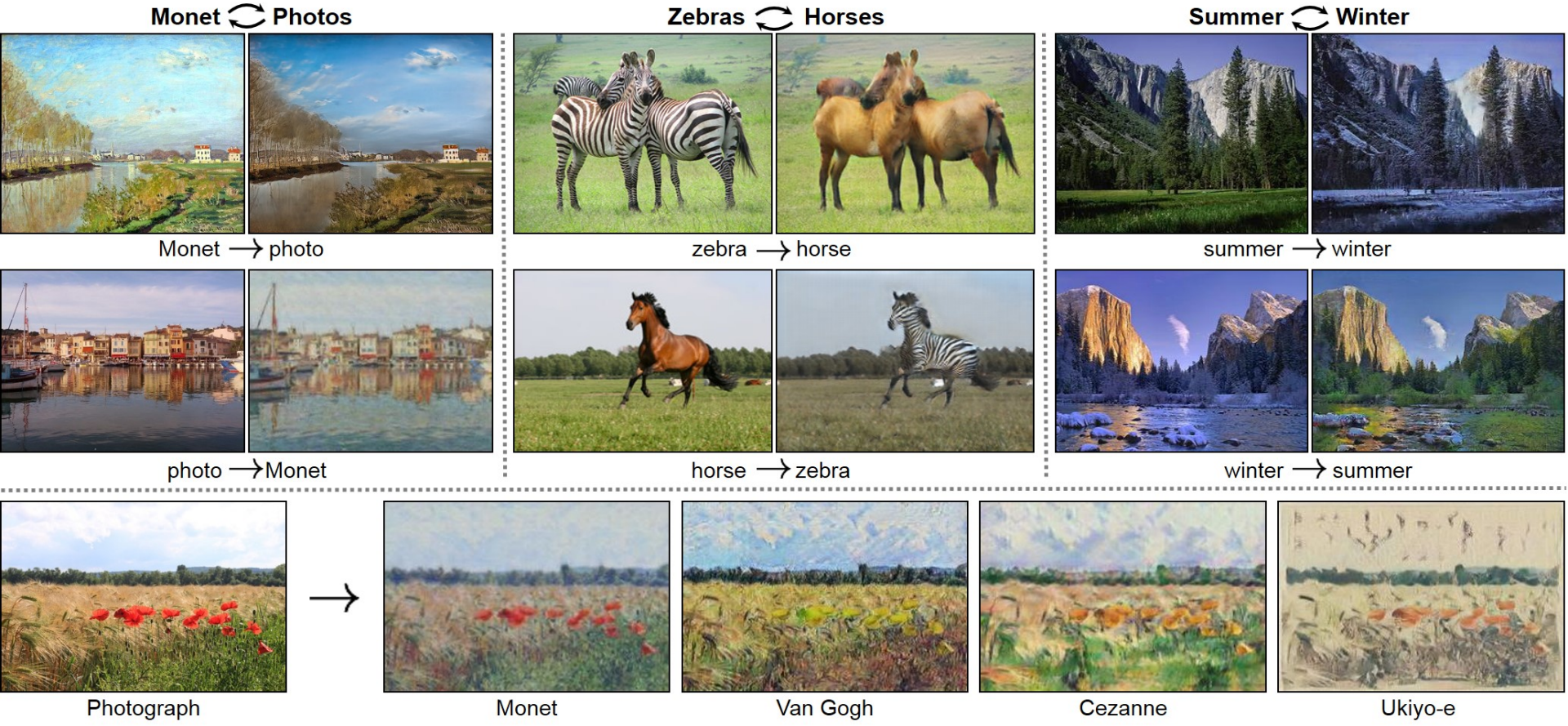
https://github.com/Newmu/dcgan_code

Radford et al. 2016

# Domain Transfer

CycleGAN. Zhu et al. 2017 https://github.com/junyanz/CycleGAN

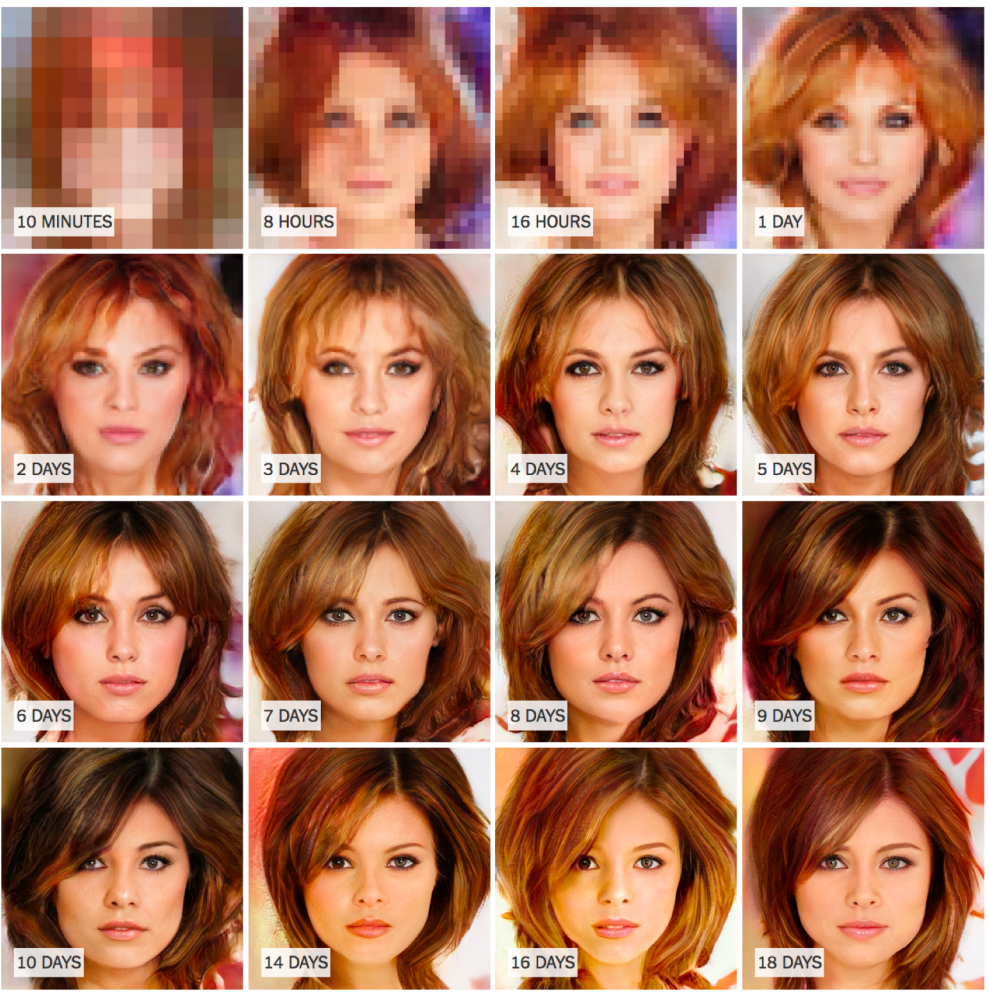
# Domain Transfer




CycleGAN. Zhu et al. 2017

https://github.com/junyanz/CycleGAN

# Generated Faces in HD




http://research.nvidia.com/publication/2017-10_Progressive-Growing-of

# GANs Summary

- no inference / explicit density function
- game-theoretic approach: learn to generate from training distribution through 2-player game

**pros:**

- beautiful, state-of-the-art samples!

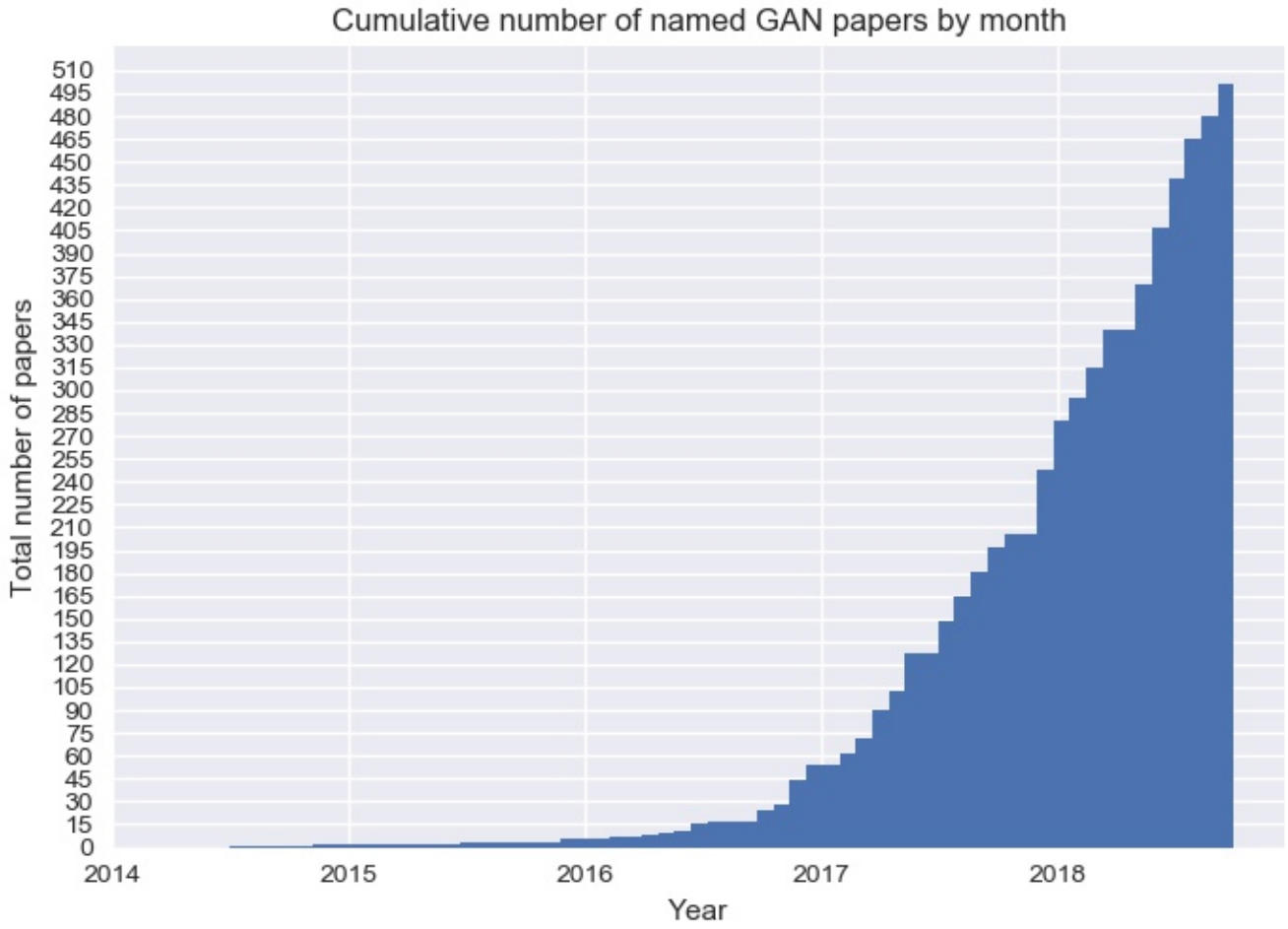
**cons:**

- trickier / more unstable to train
- cannot solve inference queries such as p(x), p(z|x)

**active areas of research:**

- better loss
- more stable training
- conditional GANs / many variants (cf. “GAN Zoo”)

# The GAN Hype




https://github.com/hindupuravinash/the-gan-zoo

# VAEs vs GANs

VAEs:

- optimize variational lower bound on likelihood
- useful latent representation, inference queries
- current sample quality not the best

GANs:

- game-theoretic approach, best samples!
- can be tricky and unstable to train
- no inference queries

Combinations:

- VAE-GAN, Adversarially Learned Inference