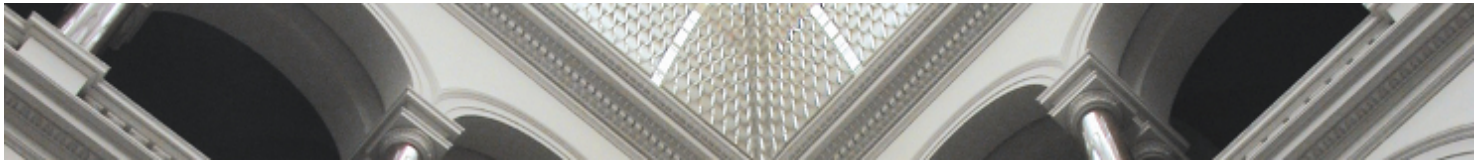
Technische Universität Berlin

Generative Neural Networks

Philipp Seegerer | Technische Universität Berlin - Machine Learning Group | Deep Neural Networks SS18



Technische Universität Berlin

Agenda

Generative Models

- Introduction
- Applications

Boltzmann Machines

Deep Belief Nets

Generative Adversarial Networks

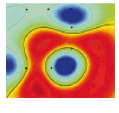
- Analysing GANs
- Examples

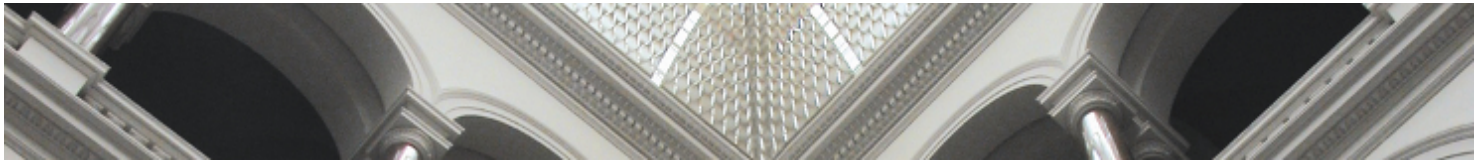
Variational Autoencoders

Autoregressive Networks

- PixelRNN and PixelCNN
- WaveNet

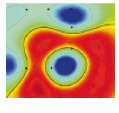
Summary

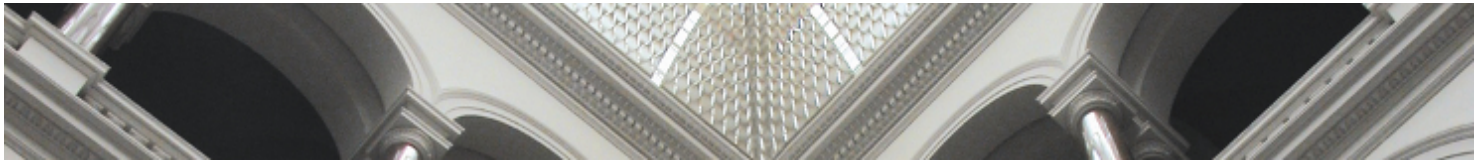




Technische Universität Berlin

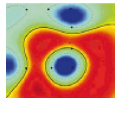
Agenda

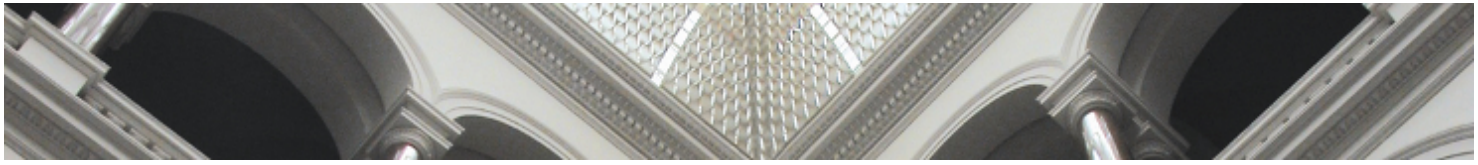




Technische Universität Berlin

Agenda





Technische Universität Berlin

Introduction

- World is represented by a vector of random variables

$$\vec{x} = (x_1, \ldots, x_d) \ ,$$

  where d is the dimensionality of the feature space.
- $\vec{x}$ is governed by probability distribution $p(\vec{x})$
- Goal: learn a model $p_\Theta(\vec{x})$ with parameters Θ that is close to $p(\vec{x})$
- $\vec{x}$ often on manifold with dimensionality $\ll d$

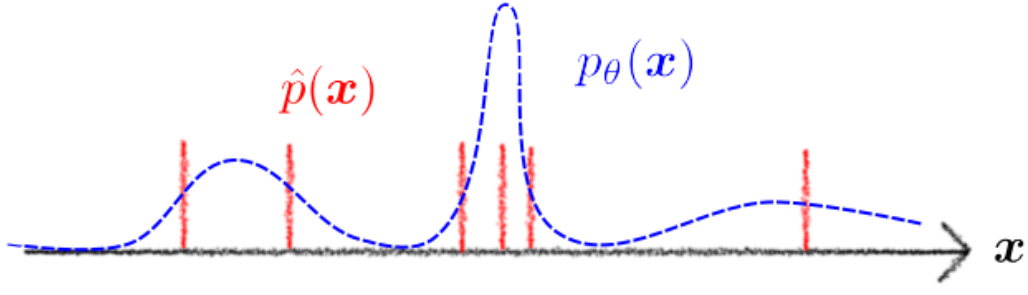
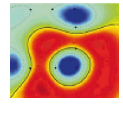
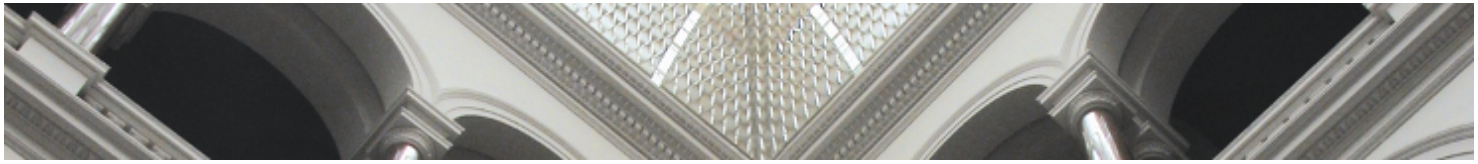


Figure: True and modeled data distribution.





Technische Universität Berlin

Generative vs. Discriminative Models

- Discriminative model: models the *posterior distribution* of output y given input $\vec{x}$:

$$p(y|\vec{x})$$

- Generative model: models the *joint distribution* of input $\vec{x}$ and output y:

$$p(\vec{x}, y)$$

- $p(\vec{x}, y)$ can be modelled explicitly or implicitly
- Examples: mixture models (e. g. Gaussian mixtures), Hidden Markov models

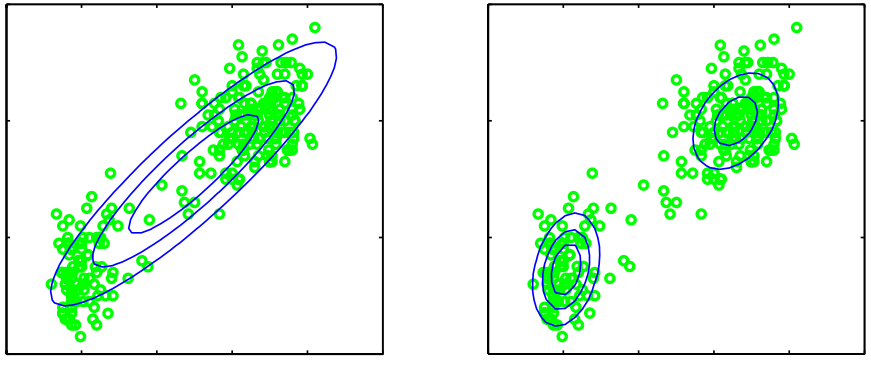
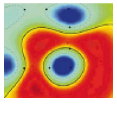
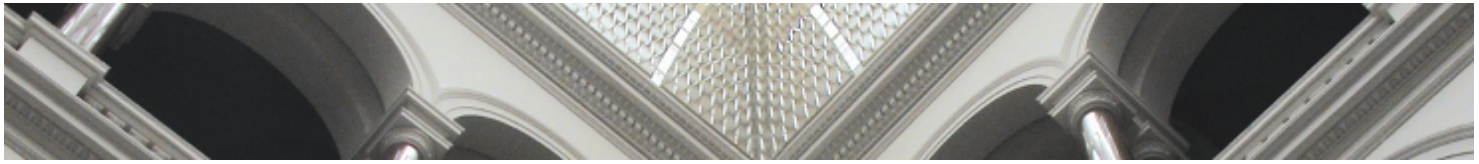


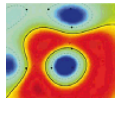
Figure: Example of Gaussian mixture model [Bis06].

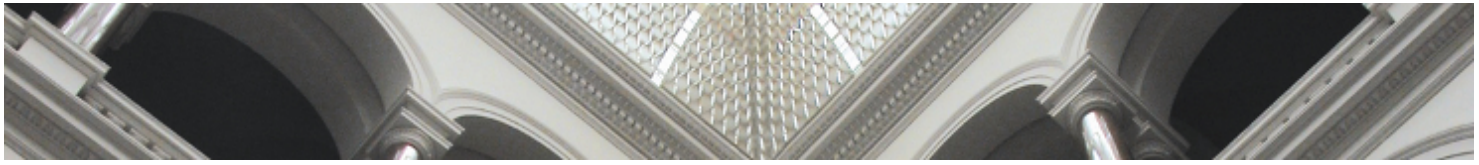




Technische Universität Berlin

Agenda





Technische Universität Berlin

Artifact Removal

Example: complete or denoise input before processing it with a discriminative model

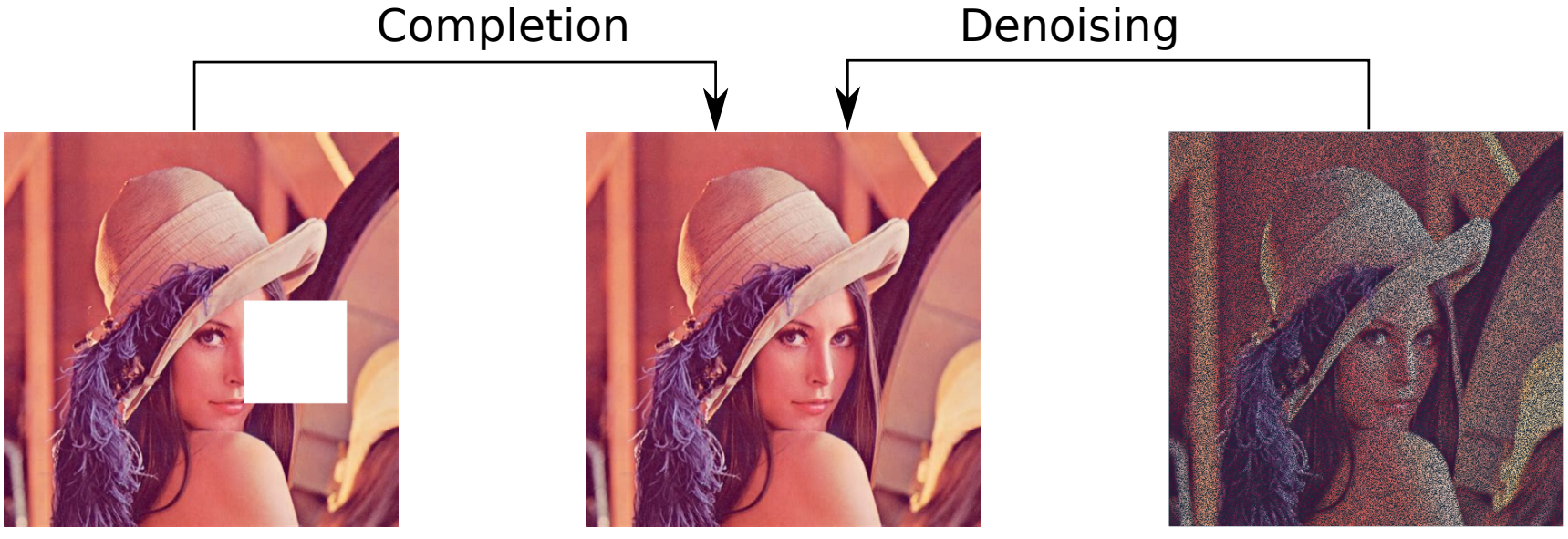
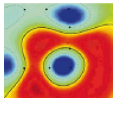
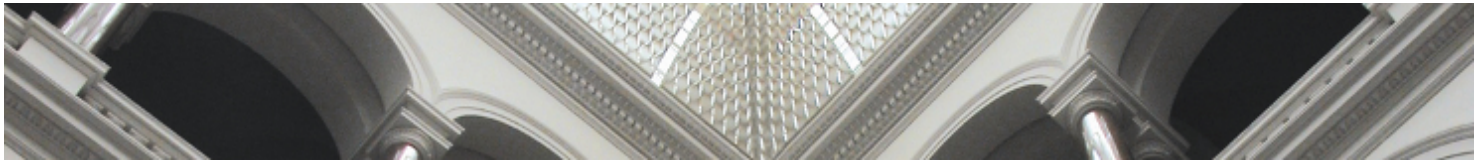


Figure: Data completion and denoising.

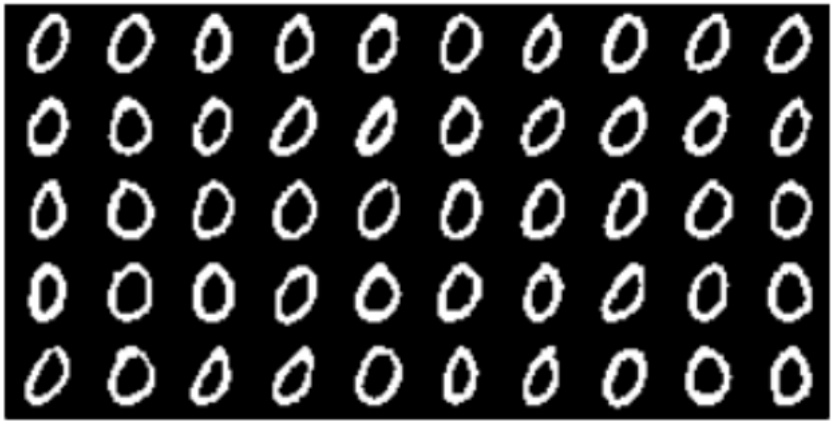




Technische Universität Berlin

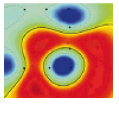
Novelty Detection

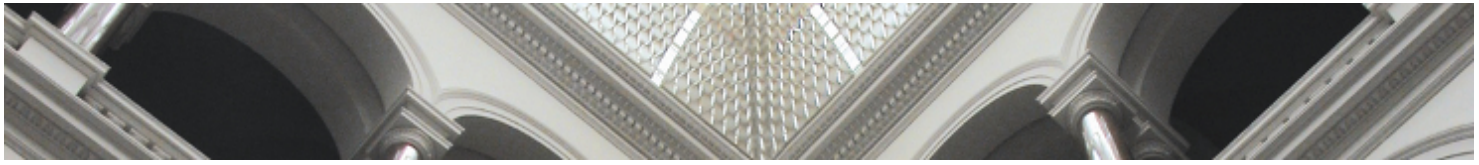
Classify as “outlier” a. k. a. “novelty” if new data point has low $p_\Theta(\vec{x})$



(a) High model probability

(b) Low model probability

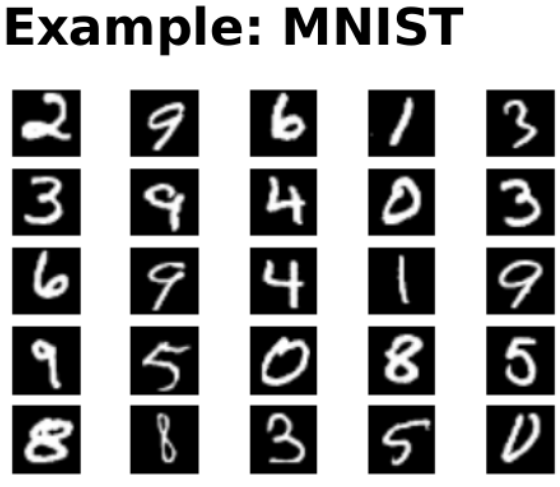


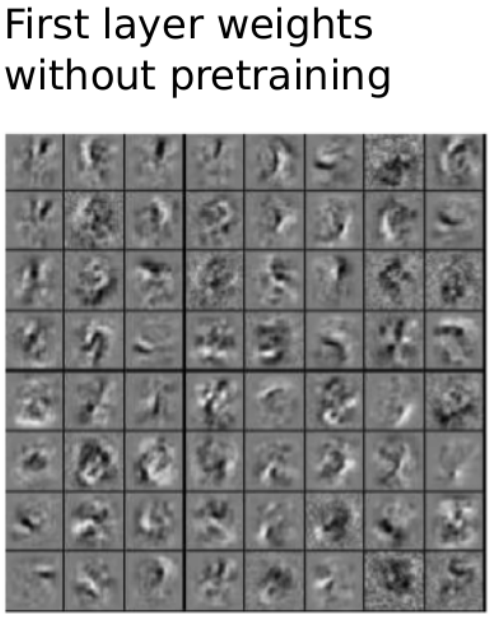


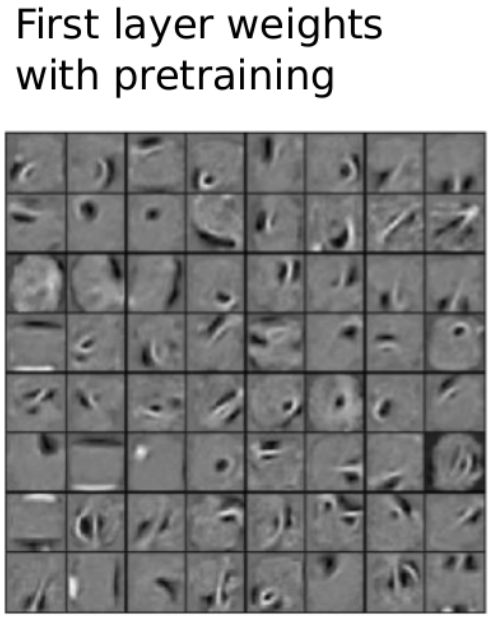
Technische Universität Berlin

Unsupervised Pretraining

Parameters (“features”) learned by a generative model as initialization for discriminative model

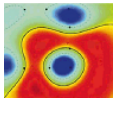


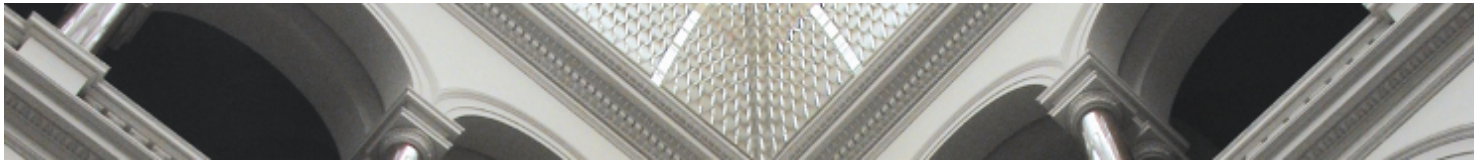




Source: Erhan, JMLR, 2010

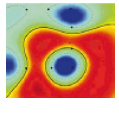
Figure: Pretrained features on MNIST digits [Erh+10].

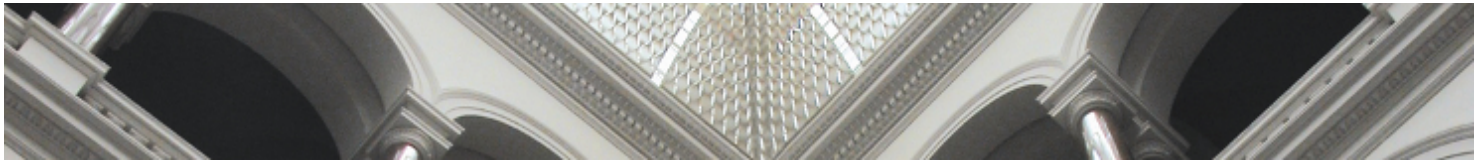




Technische Universität Berlin

Agenda





Technische Universität Berlin

Graphical Models

- Representation of probability distributions
- Each variable is a node in the graph
- Dependencies are represented by (directed or undirected) edges in the graph

Example:

$$p(a, b, c, d) = p(a)p(b)p(c|a, b)p(d|c)$$

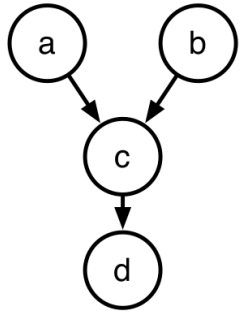
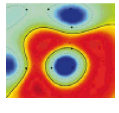
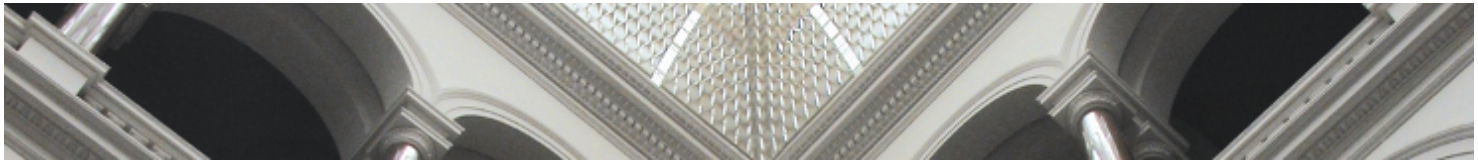


Figure: Example of a directed graphical model.





Technische Universität Berlin

Boltzmann Machines

- Hidden $\vec{h}$ and visible states $\vec{v}$ are binary variables
- Definition by energies $E(\vec{v}, \vec{h})$ of joint configurations of $\vec{v}$ and $\vec{h}$
- Energies and probabilities are related by $p(\vec{v}, \vec{h}) \propto \exp\left(-E(\vec{v}, \vec{h})\right)$
- Energy of joint configuration:

$$-E(\vec{v}, \vec{h}) = \sum_{\text{visible } i} v_i b_i + \sum_{\text{hidden } k} h_k b_k + \sum_{i<j} v_i v_j w_{ij} + \sum_{i,k} v_i h_k w_{ik} + \sum_{k<l} h_k h_l w_{kl} \quad , \qquad (1)$$

with weights w_{ij} and biases b_i.

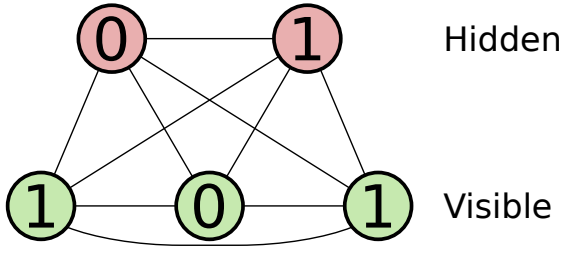
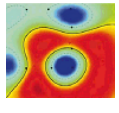
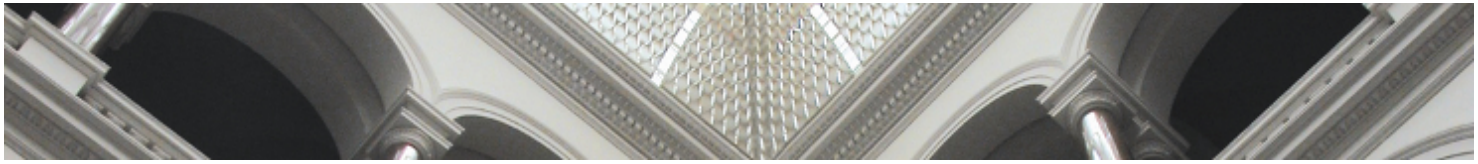


Figure: Boltzmann machine.

git push





Technische Universität Berlin

Energies and Probabilities

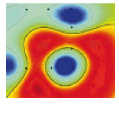
– Joint probability:

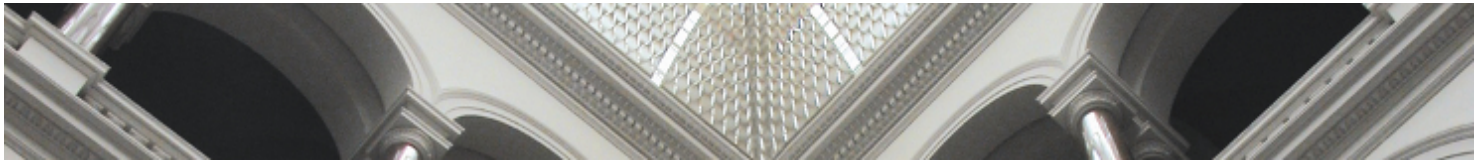
$$p(\vec{v},\vec{h}) = \frac{\exp\left(-E(\vec{v},\vec{h})\right)}{\sum_{\vec{u},\vec{g}}\exp\left(-E(\vec{u},\vec{g})\right)} = \frac{\exp\left(-E(\vec{v},\vec{h})\right)}{Z} \quad , \tag{2}$$

where Z is the partition function.

– Probability of a visible vector:

$$p(\vec{v}) = \sum_{\vec{h}} p(\vec{h})p(\vec{v}|\vec{h}) \tag{3}$$

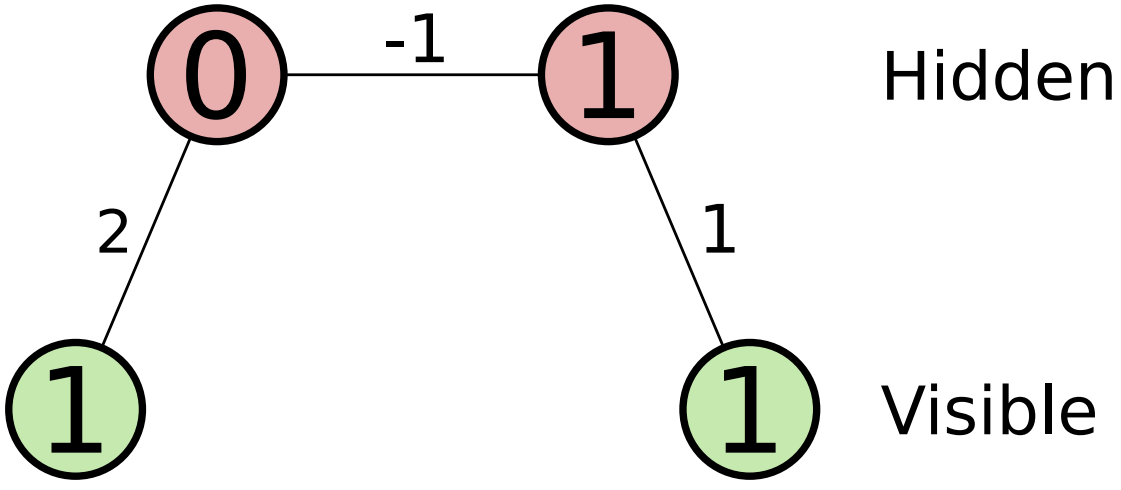




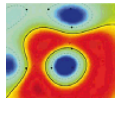
Technische Universität Berlin

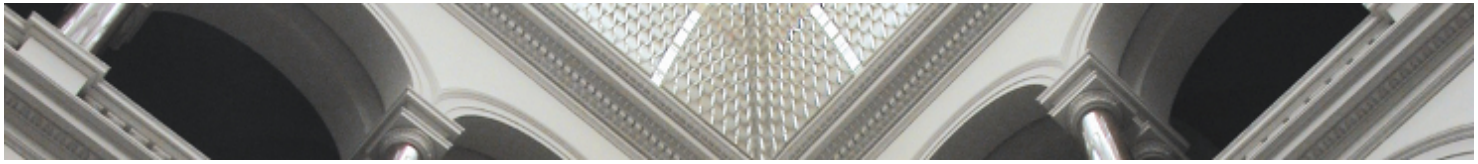
Exercise

Taken from [Hin12]



What are the associated energies and probabilities?





Technische Universität Berlin

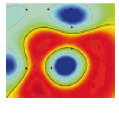
Sampling from Boltzmann Machine

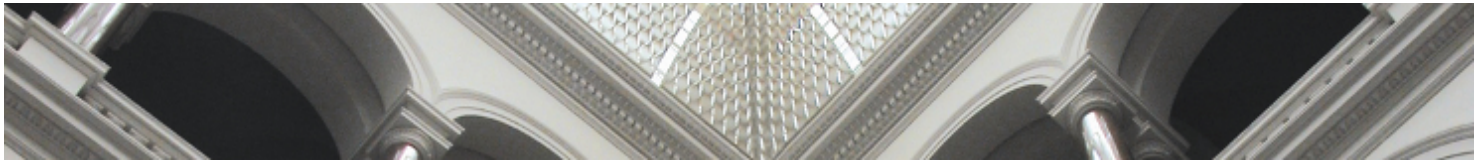
Global configuration $p(\vec{v}, \vec{h})$:

- More than a few hidden units $\rightarrow$ partition function is intractable
- Markov Chain Monte Carlo (MCMC) to sample from the model
  - Initialize random configuration
  - Pick units randomly and update their states based on the energy until equilibrium
  - Probability of configuration is defined by Boltzmann distribution

Posterior $p(\vec{h}|\vec{v})$:

- Needed for learning
- Same as sampling global configuration but fix visible state to training vector $\rightarrow$ change only hidden units





Technische Universität Berlin

Learning a Boltzmann Machine

- Goal: maximize probabilities of training data under the model

$$\hat{\Theta} = \arg\max_{\Theta} p_{\Theta}(X) = \arg\max_{\Theta} \sum_{i=1}^{N} \log p_{\Theta}(\vec{x}_i) \ , \tag{4}$$

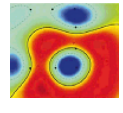
  with training set $X = (\vec{x}_1, \ldots, \vec{x}_N)$.
- Update of a weight depends only on difference of two correlations at equilibrium:

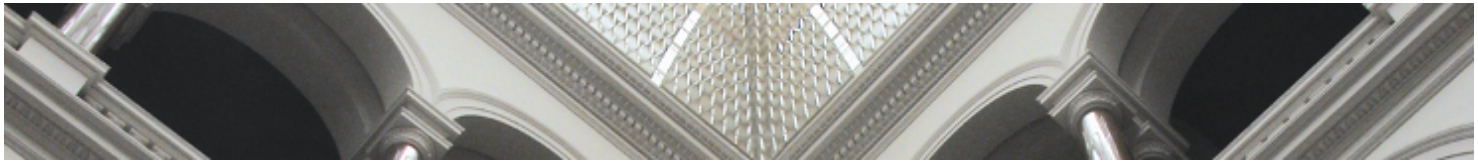
$$\frac{\partial p(\vec{v})}{\partial w_{ij}} = \underbrace{\mathbb{E}[s_i s_j]|_{\vec{v}}}_{\vec{v} \text{ is clamped on visible units}} - \underbrace{\mathbb{E}[s_i s_j]}_{\text{no clamping}} \ , \tag{5}$$

  where the s_i's are the states at equilibrium.
- Or alternatively:

$$\Delta w_{ij} \propto \underbrace{\mathbb{E}[s_i s_j]|_{\text{data}}}_{\text{positive phase}} - \underbrace{\mathbb{E}[s_i s_j]|_{\text{model}}}_{\text{negative phase}} \tag{6}$$

- Observation: Information about the parameters is conveyed by running the Markov chain until equilibrium $\rightarrow$ backpropagation not needed





Technische Universität Berlin

Deep Boltzmann Machine

- Updates to units have to be sequential
- Therefore, restriction to special architecture: “Deep Boltzmann Machine”
- No within-layer connections
- No connections between non-adjacent layers
  $\rightarrow$ efficient training

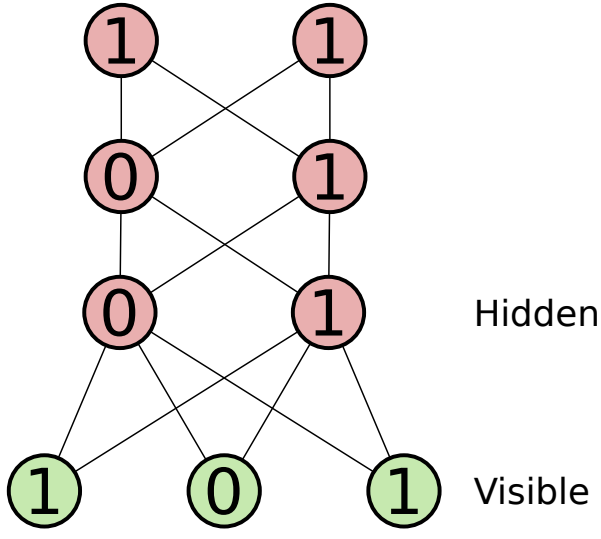
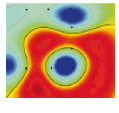
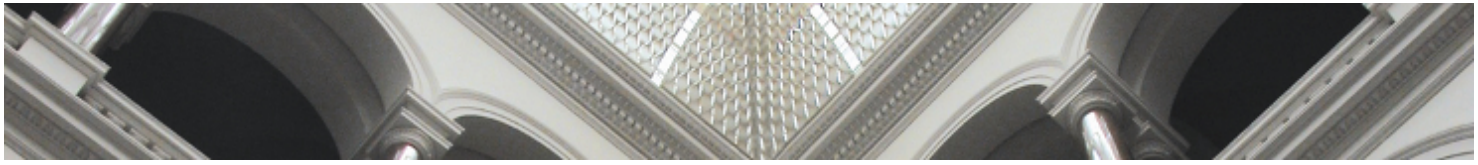


Figure: Deep Boltzmann machine.





Technische Universität Berlin

Restricted Boltzmann Machines (RBM)

- Restrict connectivity:
  - Only one hidden layer
  - No within-layer connections

  $\rightarrow$ only one step needed to reach equilibrium if visible layer is clamped!
- Quickly get $\mathbb{E}[v_i h_j]$'s in parallel:

$$p(h_j = 1) = \overbrace{\frac{1}{1 + \exp\left(-\left(b_j + \sum_{\text{visible } i} v_i w_{ij}\right)\right)}}^{\text{sigmoid}} \tag{7}$$

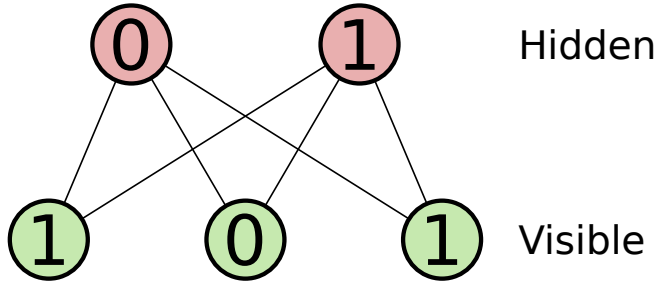
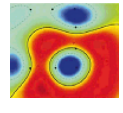
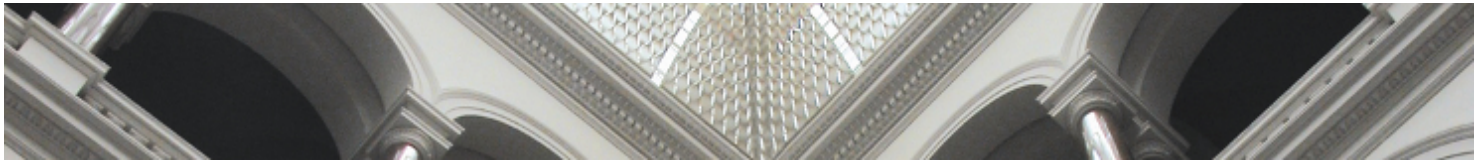


Figure: Restricted Boltzmann machine.





Technische Universität Berlin

RBM Training – Contrastive Divergence

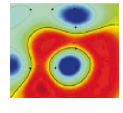
- Clamp training vector to visible units
- Alternate between updating hidden and visible units (in parallel)
- The learning rule is then:

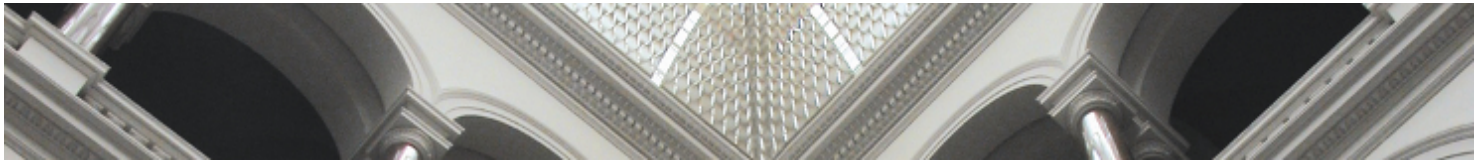
$$\Delta w_{ij} \propto \underbrace{\mathbb{E}[v_i h_j]^0}_{\text{Initial state}} - \underbrace{\mathbb{E}[v_i h_j]^\infty}_{\text{Equilibrium}} \tag{8}$$

- $\vec{v}$ at equilibrium: “fantasy”
- In practice, one update is usually sufficient (“CD1”):

$$\Delta w_{ij} \propto \mathbb{E}[v_i h_j]^0 - \mathbb{E}[v_i h_j]^1 \tag{9}$$

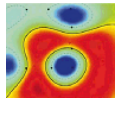
- $\vec{v}$ after one MC step: “reconstruction”
- This is wrong but it still works!
- Later in training: CD3-CD10

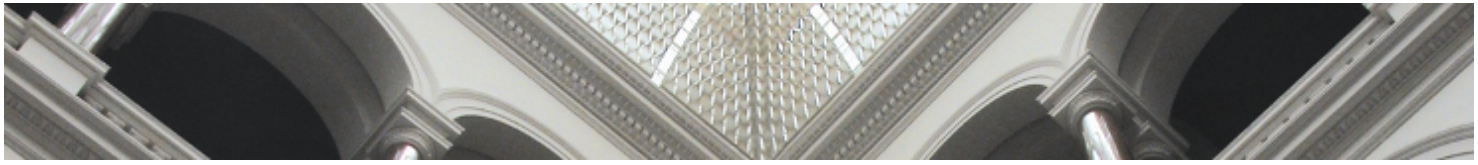




Technische Universität Berlin

Agenda





Technische Universität Berlin

Belief Net

- Directed acyclic graph of stochastic binary variables
- Some of them are visible (effects), some are hidden (causes)
- Inference: infer states of unobserved variables
- Learning: make training data more likely under the model
- Boltzmann machines: energy-based with symmetric connections
- Belief Nets: causal with directed acyclic connections

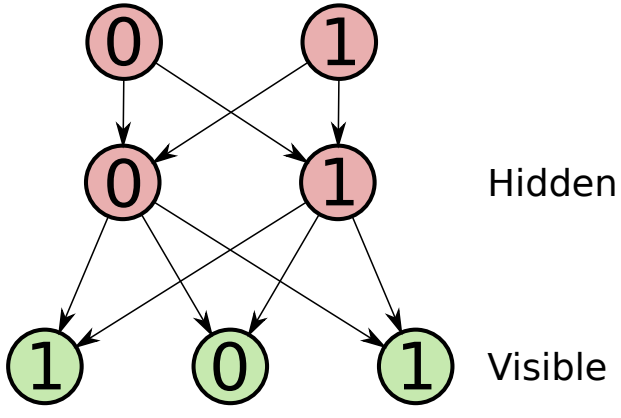
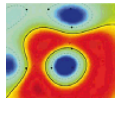
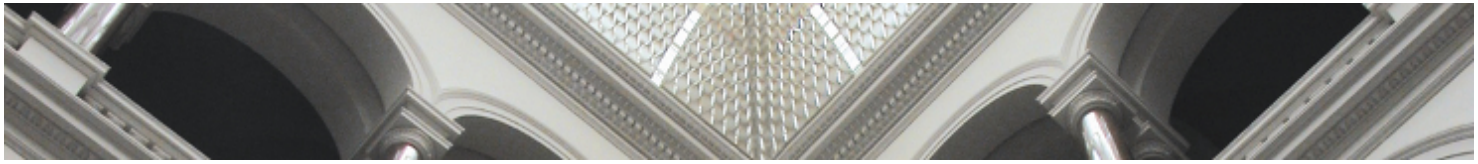


Figure: Deep Belief Net.

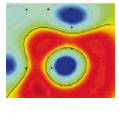


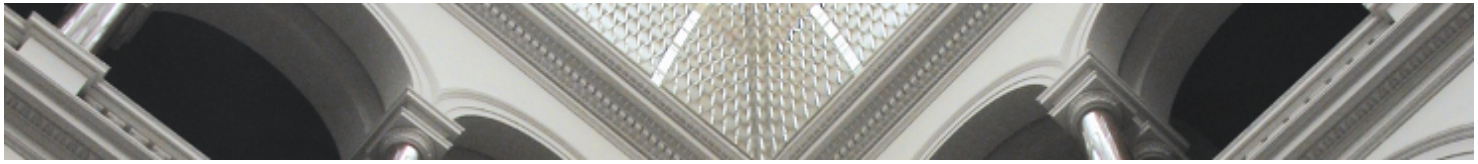


Technische Universität Berlin

Problems for Learning

- Sampling visible configurations is easy (top-down layer by layer)
- Inferring the posterior over hidden states (even sampling from it) is hard:
  - Number of possible hidden states is exponential in number of hidden units (typically millions)
  - Explaining away $\rightarrow$ posterior can not be factorized
  - Posterior depends on prior and likelihood
  - Marginalization over all possible configurations in higher layers to get prior for first layer





Technische Universität Berlin

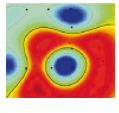
Learning if posterior sample available

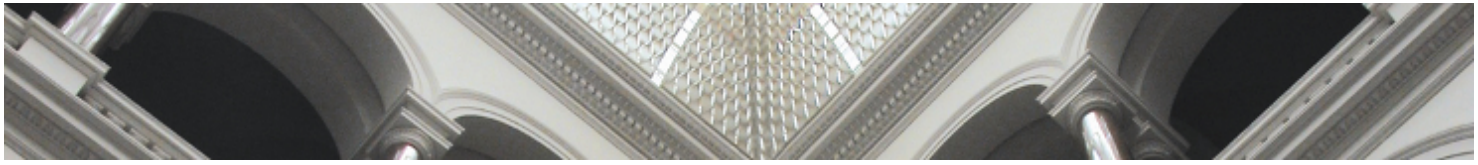
– If we had sample from posterior, learning can be done layer by layer:
maximize (for each unit) the probability that its state in this posterior sample would be generated by the sampled states of its parents.

$$p(\underbrace{s_i = 1}_{\text{unit } i \text{ is on}}) = \frac{1}{1 + \exp\left(-b_i - \underbrace{\sum_j s_j w_{ji}}_{\text{parents}}\right)} \tag{10}$$

– Maximum-likelihood learning rule:

$$\Delta w_{ij} \propto s_j(s_i - p(s_i = 1)) \tag{11}$$





Technische Universität Berlin

Wake-sleep Algorithm

- Idea: Sample from a wrong posterior distribution but still use the ML learning rule (Eq. 11)
- Wake phase: bottom-up pass of data
  - generate “posterior” sample
  - learn generative weights using ML learning rule
- Sleep phase: top-down pass
  - generate sample from model
  - learn reconstruction weights

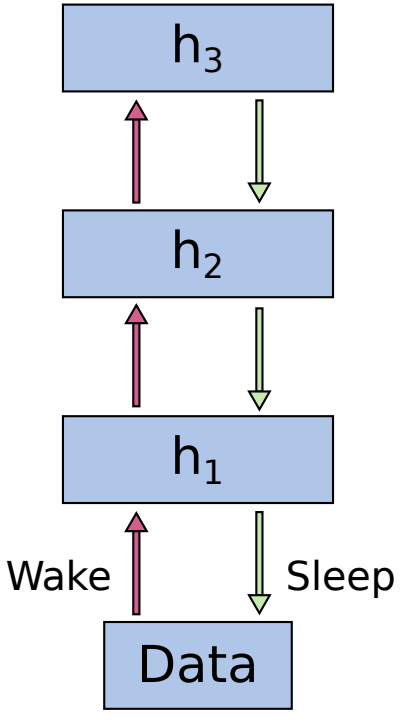
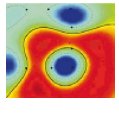
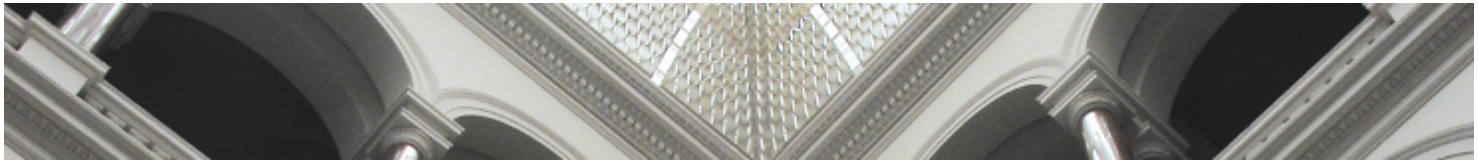


Figure: Wake-sleep algorithm.

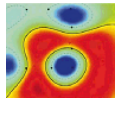


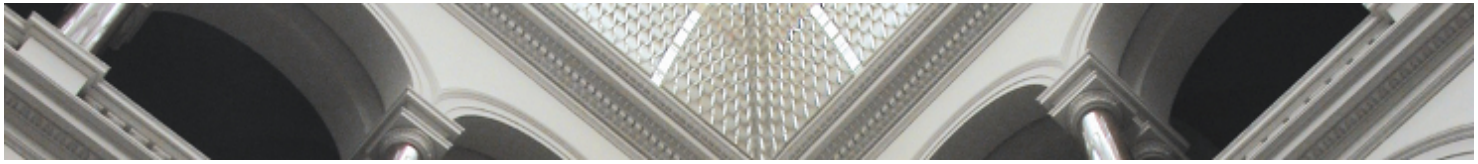


Technische Universität Berlin

Generative Pretraining

- Greedy learning: After training an RBM, treat the features as input to another RBM $\rightarrow$ deep architecture
- Improving variational lower bound on probability of the training data under the model by adding layers
- DBN $\equiv$ stack of RBMs
- Generative process
  1. Equilibrium sample in top RBM (alternating Gibbs sampling)
  2. Top-down pass to input layer
- Finetuning for generation with contrastive version of wake-sleep algorithm
- Finetuning for discrimination by backpropagation

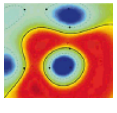


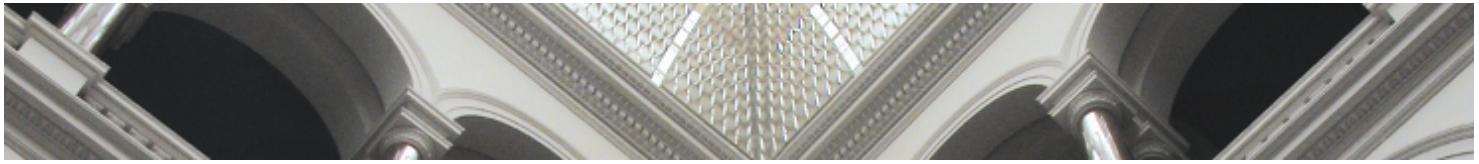


Technische Universität Berlin

Discriminative Finetuning

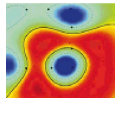
- Easier learning
- Better generalization
- Pretraining for informative features
- Finetuning for correct decision boundaries (modifies features only slightly)
- Deep networks are more effective than shallow ones
- Disadvantage: you learn features that you do not need for your actual task

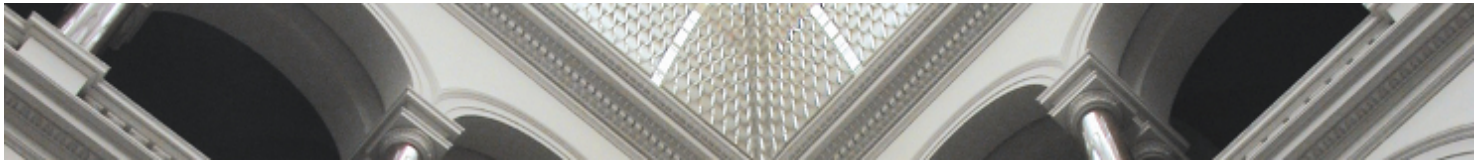




Technische Universität Berlin

Agenda

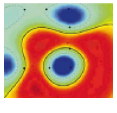


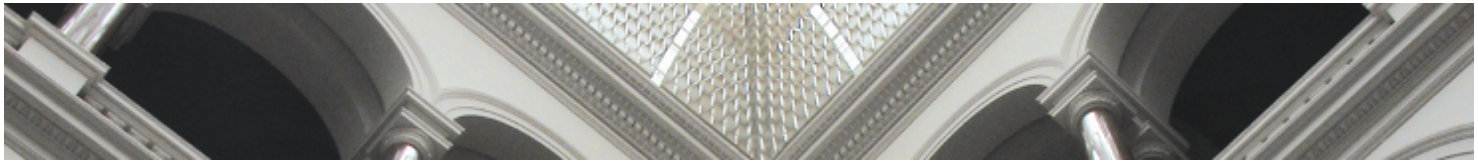


Technische Universität Berlin

Concept

- Goal: generate samples from same distribution as input data
- Interplay between Generator G and Discriminator D network
- Generator G and Discriminator D are deep neural networks
- D has to decide whether sample came from generator distribution p_G
- G tries to fool D by generating samples similar to p_{data}
- Input to G is white noise
- Competition G vs. $D \rightarrow$ simultaneous training improves both adversaries
- G should learn: $p_G = p_{\text{data}}$





Formulation as Minimax Game

- G and D are adversaries in a Minimax game
- Optimize the following value function V:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\vec{x}\sim p_{\text{data}}(\vec{x})}\left(\log D(\vec{x})\right) + \mathbb{E}_{\vec{z}\sim p_{\vec{z}}(\vec{z})}\left(\log(1 - D(G(\vec{z}))\right) \quad , \tag{12}$$

  where $\vec{z}$ is random noise vector sampled from latent space Z
- Optimization is done simultaneously and oppositely by G and D
- Nash equilibrium: optimal, stable condition for both players

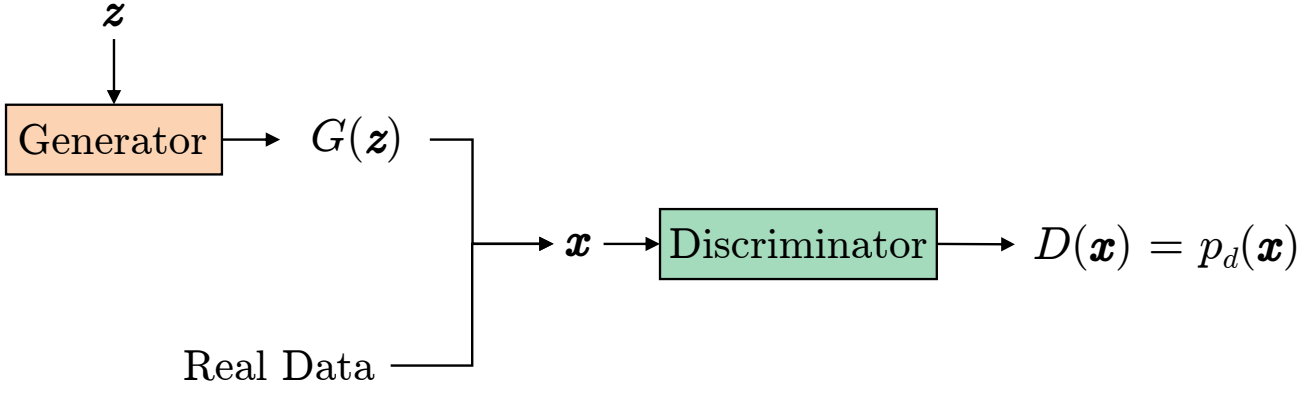
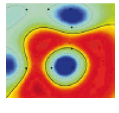
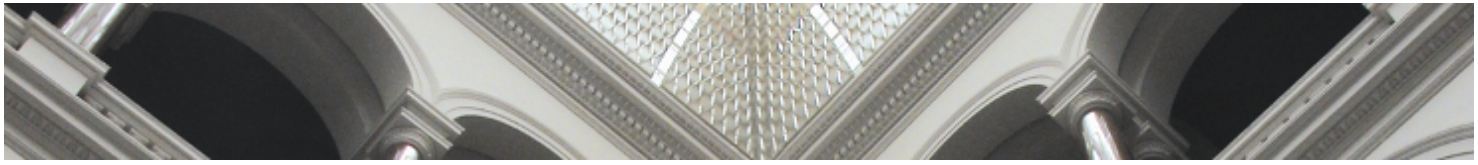


Figure: Schematic architecture of GAN.





Technische Universität Berlin

Optimal Solution of min-max Game

Global optimum if $p_G = p_{\text{data}}$

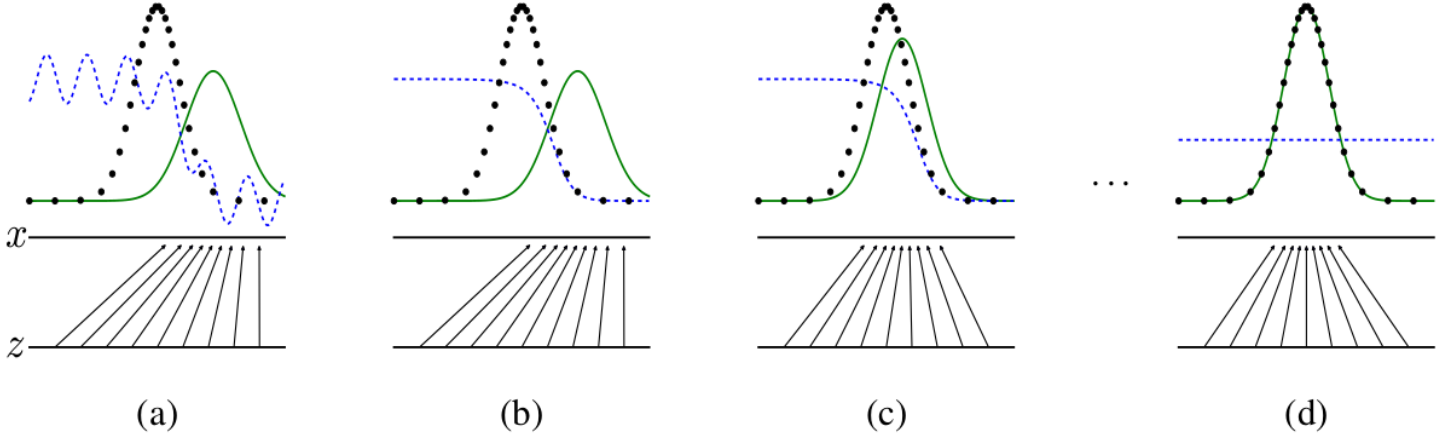
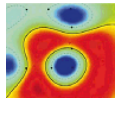
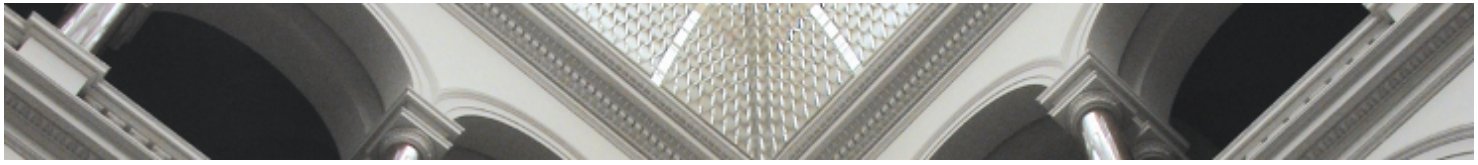


Figure: D: blue, G: green, data: black [Goo+14]

(a) Adversarial pair near convergence
(b) D near convergence
(c) G near convergence
(d) Equilibrium: $D(\vec{x}) = \frac{1}{2}$





Technische Universität Berlin

Theoretical Results

Optimal D (for fixed G):

$$D_G^*(\vec{x}) = \frac{p_{\text{data}}(\vec{x})}{p_{\text{data}}(\vec{x}) + p_G(\vec{x})} \tag{13}$$

Training objective for D:

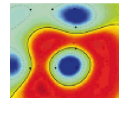
$$\begin{aligned} C(G) &= \max_D V(G, D) = \\ &= \mathbb{E}_{\vec{x}\sim p_{\text{data}}(\vec{x})} \left(\log \frac{p_{\text{data}}(\vec{x})}{p_{\text{data}}(\vec{x}) + p_G(\vec{x})} \right) + \mathbb{E}_{\vec{x}\sim p_G(\vec{x})} \left(\log \frac{p_G(\vec{x})}{p_{\text{data}}(\vec{x}) + p_G(\vec{x})} \right) \end{aligned} \tag{14}$$

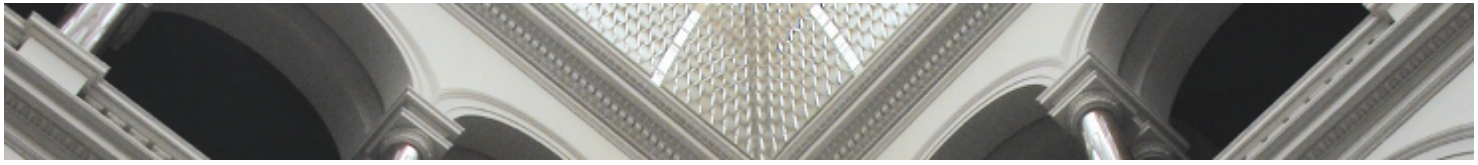
Global optimum if and only if $p_G = p_{\text{data}}$:

$$C(G) = -\log(4) + \underbrace{\text{JSD}(p_{\text{data}} || p_G)}_{\geq 0} ,$$

where JSD denotes the Jensen-Shannon divergence.

$\rightarrow$ GAN optimizes JSD between true and fake distribution





Training of GANs

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

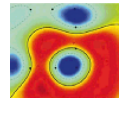
  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

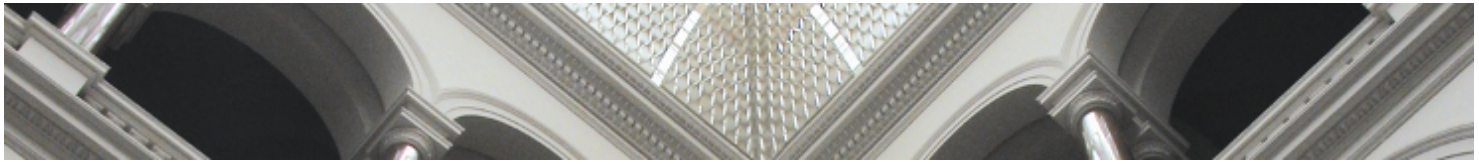
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Figure: GAN training algorithm proposed by [Goo+14].

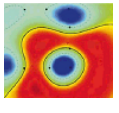


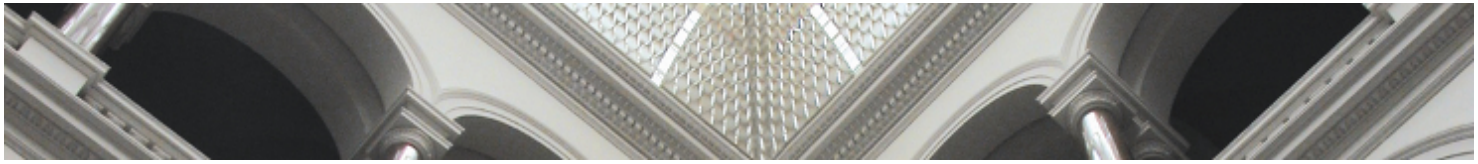


Technische Universität Berlin

Difficulties in Training

- Instable training
- Mode collapse
- No quantitative evaluation $\rightarrow$ difficult model selection





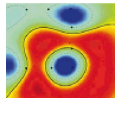
Conditional GANs

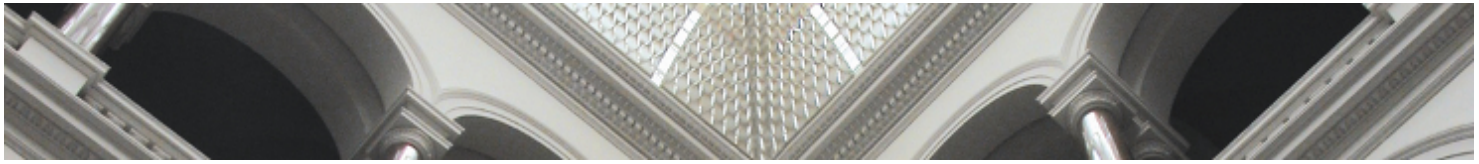
- Unsupervised training on unlabeled data $\vec{x}$: learn $p(\vec{x})$
- Labels ("conditions") $\vec{y}$ improve the results

  $\rightarrow$ conditional model: learn $p(\vec{x}|\vec{y})$ instead
- Alternative: learn joint probability $p(\vec{x}, \vec{y})$ (see [Sal+16])
- Modified loss function

$$\min_G \max_D V_{\text{cond}}(D, G) = \mathbb{E}_{(\vec{x},\vec{y})\sim p_{\text{data}}(\vec{x},\vec{y})} \left(\log D(\vec{x}, \vec{y})\right) + \\ + \mathbb{E}_{\vec{z}\sim p_{\vec{z}}(\vec{z}),\vec{y}\sim p_{\vec{y}}(\vec{y})} \left(\log(1 - D(G(\vec{z}, \vec{y}), \vec{y})\right) \quad , \tag{15}$$

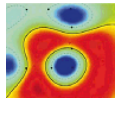
- Examples: Gauthier (2014) [Gau14], Salimans et al. (2016) [Sal+16]

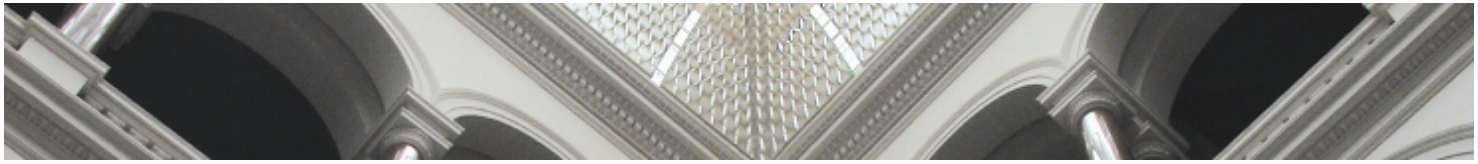




Technische Universität Berlin

Agenda

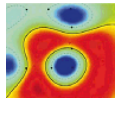


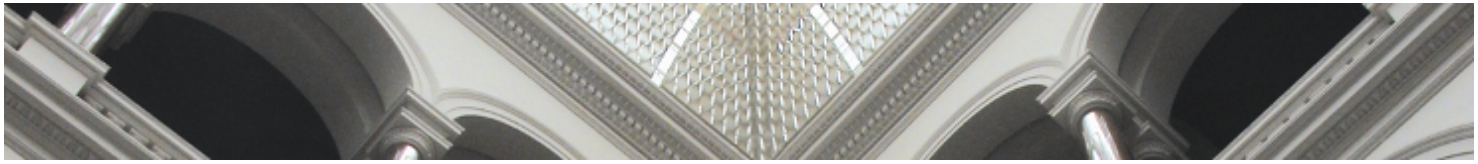


Technische Universität Berlin

Motivation

- Does the generator just memorize training images?
- Does the learned manifold have sensible properties?
- How is information coded in latent space?
- Can we quantify the performance of our model?
- Does the model train as expected?





Technische Universität Berlin

Latent Space Walking

Interpolate between two points $\vec{z}$ in noise space

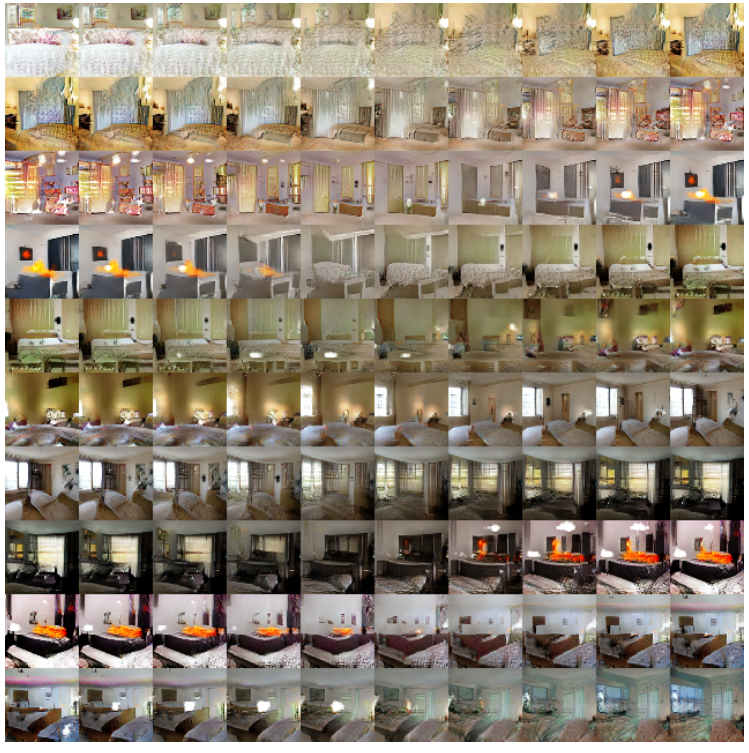
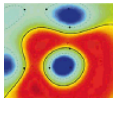
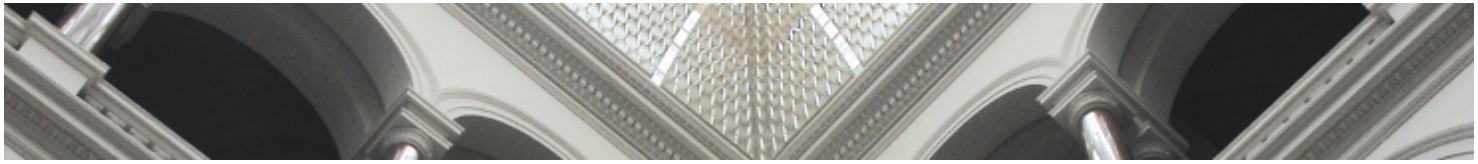


Figure: Interpolation between a series of 9 points in Z showing smooth transitions [RMC15].



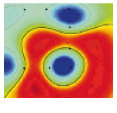


Technische Universität Berlin

Training Progress

Fix the noise vector and generate at different training stages

Example from [Kar+17]: `https://youtu.be/X0xxPcy5Gr4?t=1m17s`



Vector Arithmetics

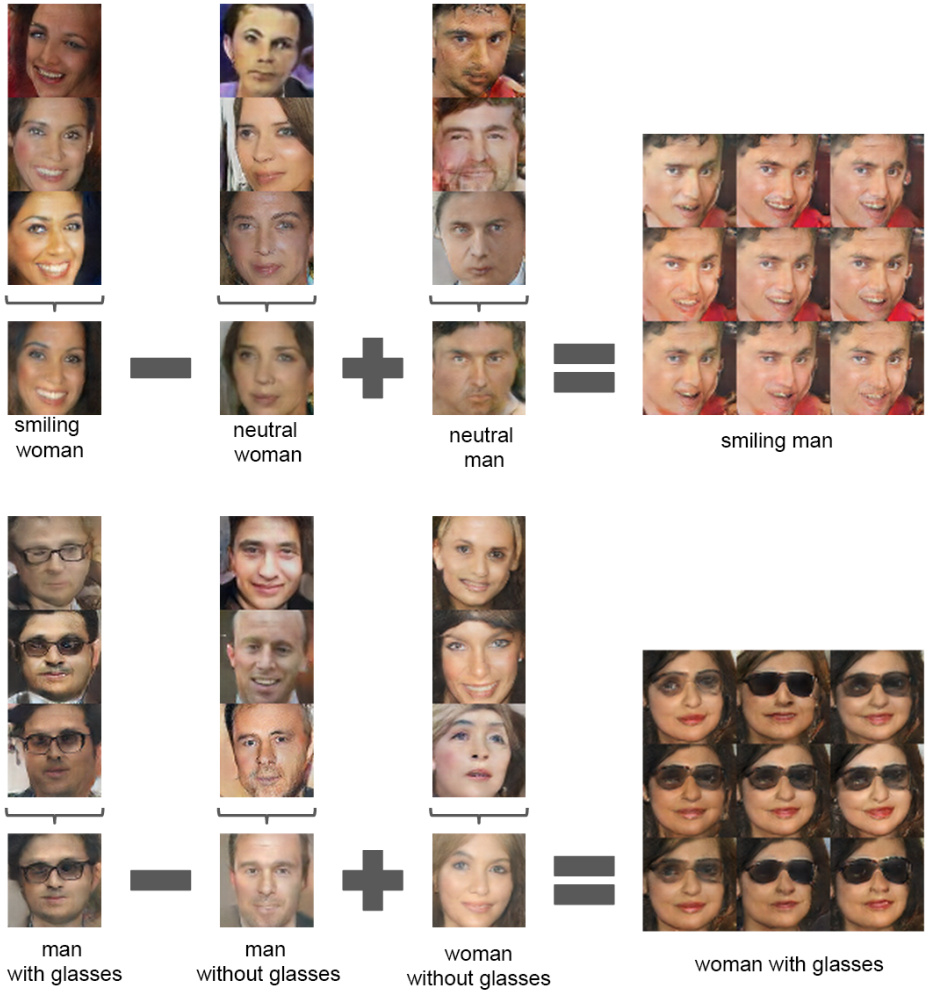
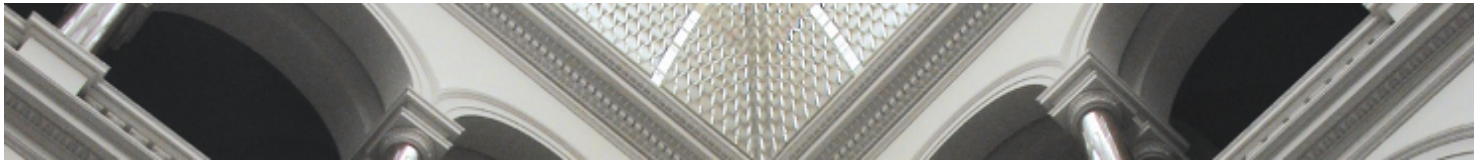
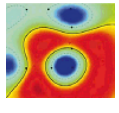


Figure: Vector operations in latent space [RMC15].



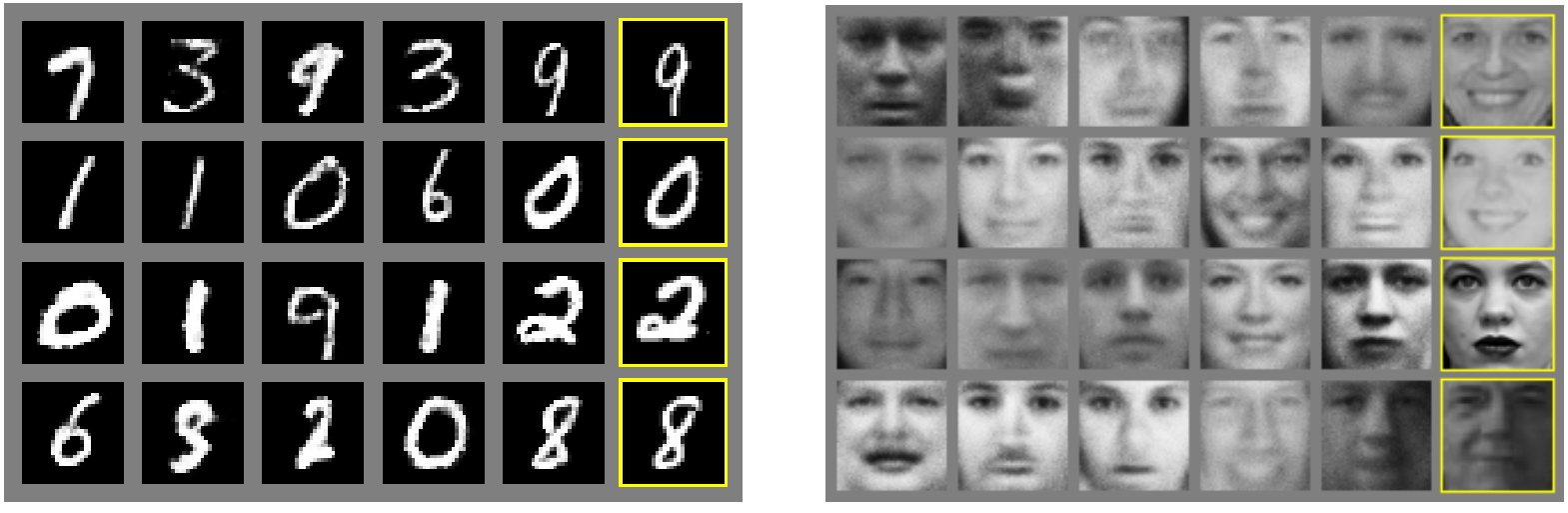
Technische Universität Berlin

Agenda



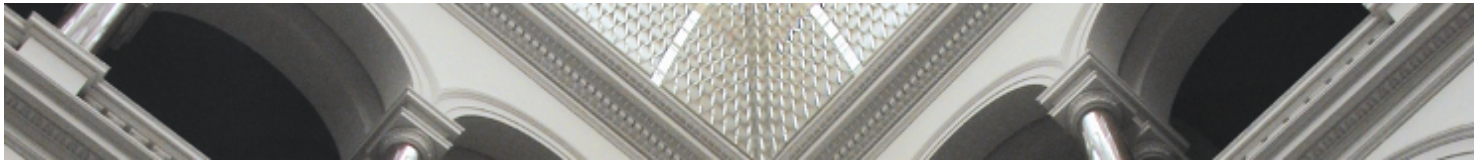
Original GAN

Goodfellow et al. (2014) [Goo+14]



(a) MNIST (b) TFD

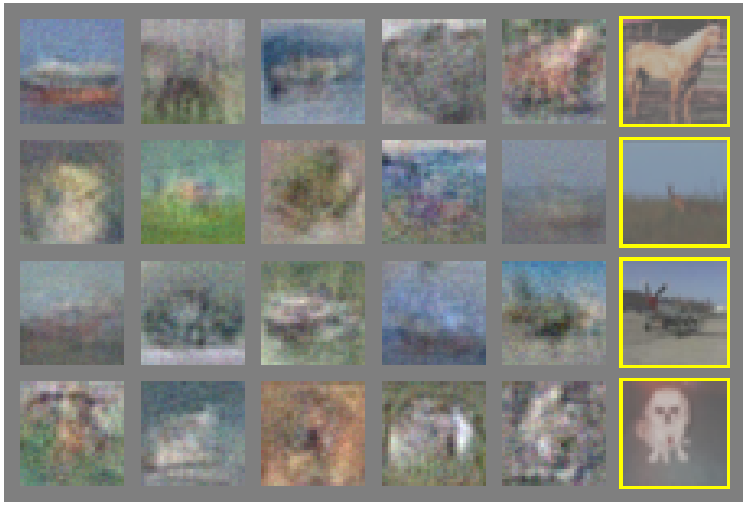
Figure: Generated images from [Goo+14].



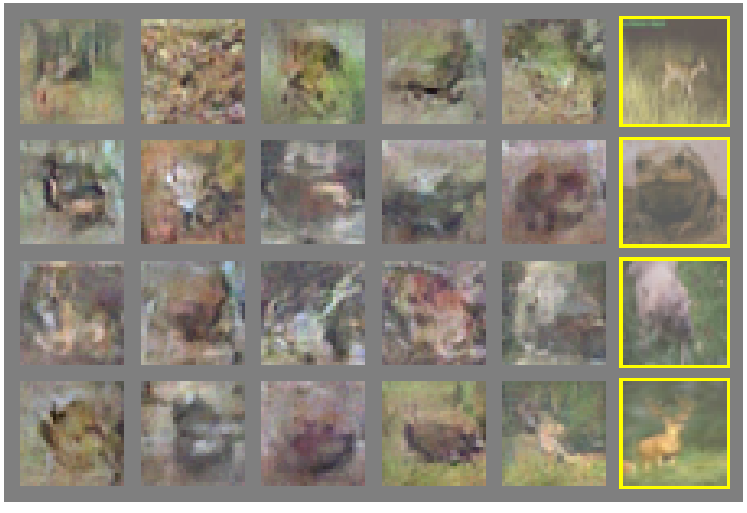
Technische Universität Berlin

Original GAN (cont.)

Goodfellow et al. (2014) [Goo+14]

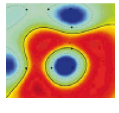


(a) CIFAR-10 (MLP)



(b) CIFAR-10 (CNN)

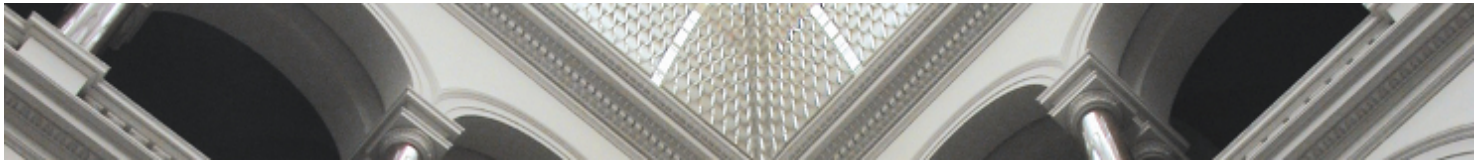
Figure: Generated images from [Goo+14].



Deep Convolutional GAN

Radford et al. (2015) [RMC15]

Figure: Generated images from [RMC15].



Technische Universität Berlin

Wasserstein GAN

Arjovski et al. (2017) [ACB17]

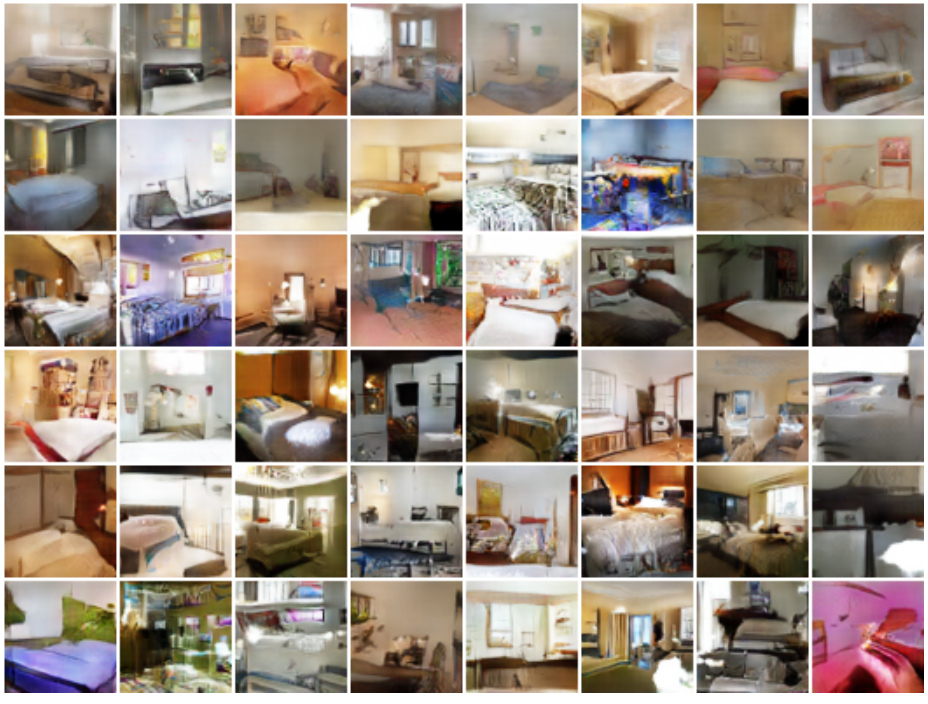
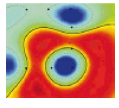
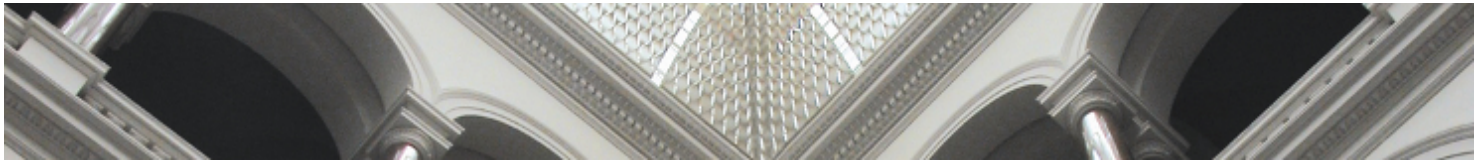


Figure: Generated images from [ACB17].





Technische Universität Berlin

Boundary Equilibrium GAN

Berthelot et al. (2017) [BSM17]

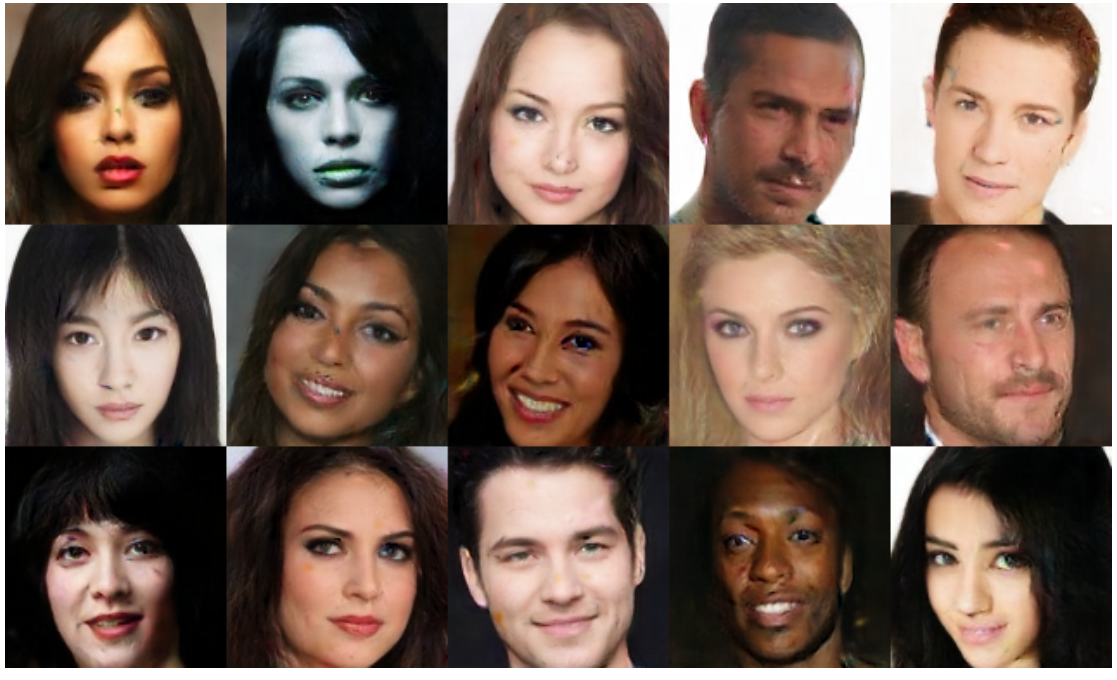
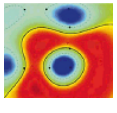
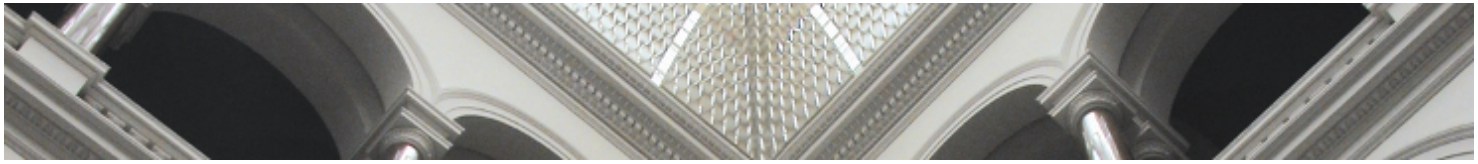


Figure: Generated images from [BSM17].





Technische Universität Berlin

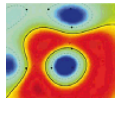
Strengths and Weaknesses

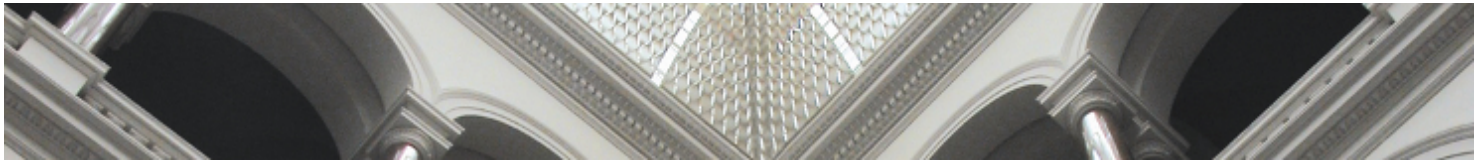
Strengths:

- Generation of visually appealing images
- Relatively efficient
- Simple concept
- No Markov chains required
- Sampling is straight forward

Weaknesses:

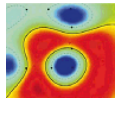
- Instable training
- Does not explicitly model density $\rightarrow$ no quantitative evaluation (except for visualizing samples)

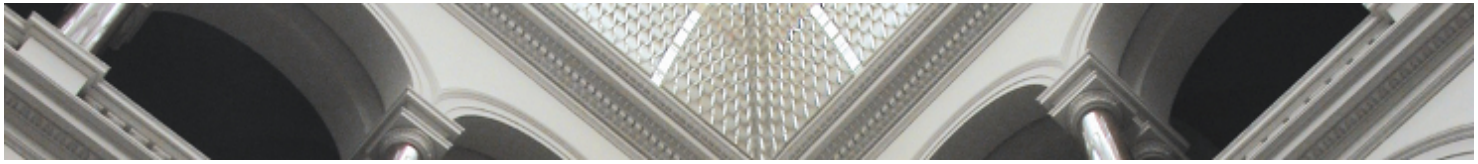




Technische Universität Berlin

Agenda





Technische Universität Berlin

Variational Auto-encoder

Proposed by Kingma et Welling (2014) [KW13]

Auto-encoder:

- Encoder: Maps input to nunmerical value in latent code space
- Decoder: Reconstruct the input from latent space

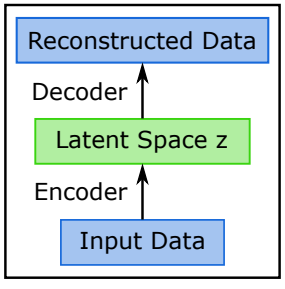
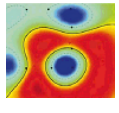
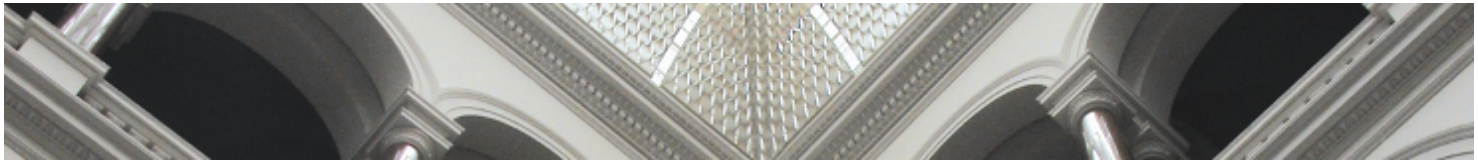


Figure: Auto-encoder.

Probabilistic extension: Variational Auto-encoder

- Encoder $q_\phi(z|x)$ maps inputs to latent distribution
- Decoder $p_\theta(x|z)$ reconstructs data distribution from latent distribution
- Prior distribution of code $p(z)$ is easy to sample, e. g. Gaussian





Technische Universität Berlin

VAE Forward Pass

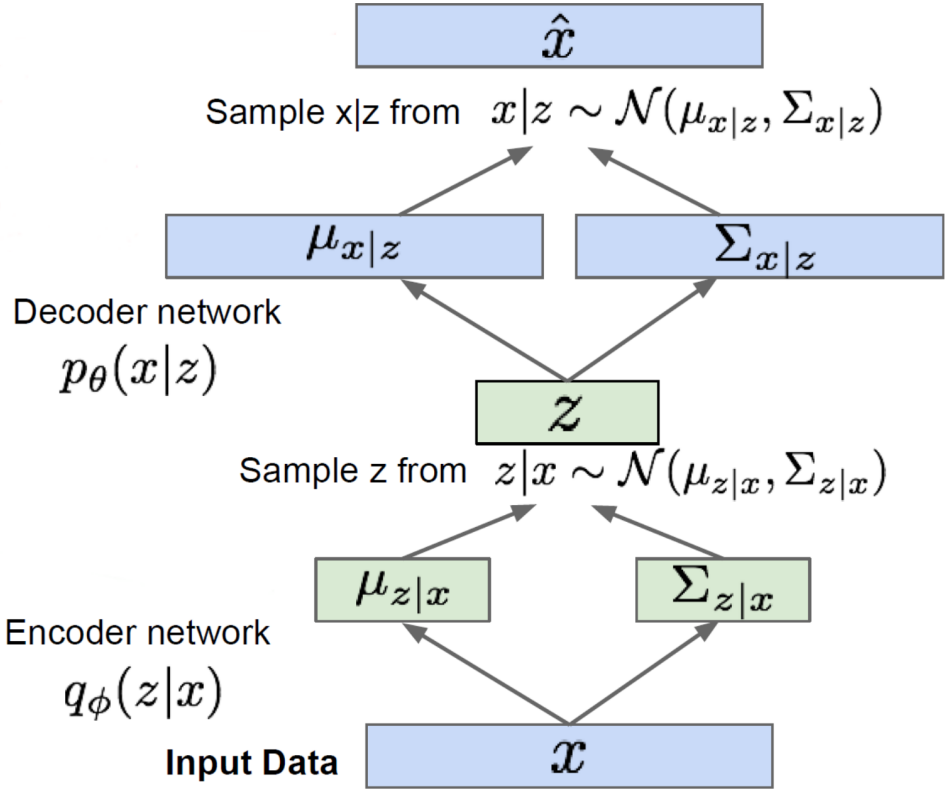
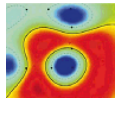
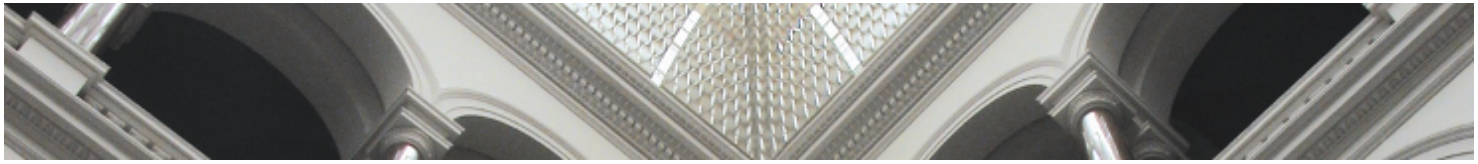


Figure: Variational autoencoder [LY17].





Technische Universität Berlin

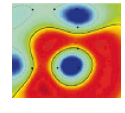
Optimization - ELBO

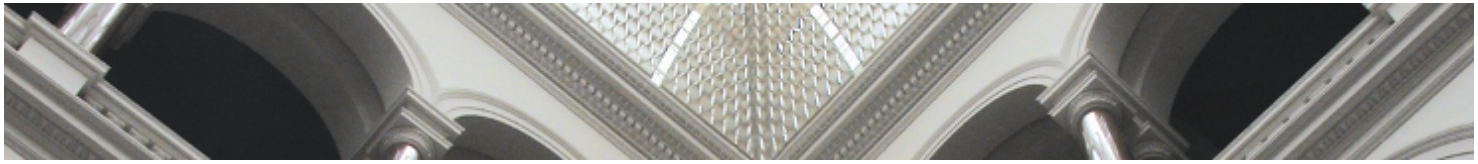
Reformulation of the data log likelihood:

$$
\begin{aligned}
\log p_\theta(x^i) &= \mathbb{E}_{z\sim q_\theta(z|x^i)}\left[\log p_\theta(x^i)\right] = \\
&= \mathbb{E}_z\left[\log \frac{p_\theta(x^i|z)p_\theta(z)}{p_\theta(z|x^i)}\right] = \\
&= \mathbb{E}_z\left[\log \frac{p_\theta(x^i|z)p_\theta(z)}{p_\theta(z|x^i)}\frac{q_\phi(z|x^i)}{q_\phi(z|x^i)}\right] = \\
&= \mathbb{E}_z\left[\log p_\theta(x^i|z)\right] - \mathbb{E}_z\left[\log \frac{q_\phi(z|x^i)}{p_\theta(z)}\right] + \mathbb{E}_z\left[\log \frac{q_\phi(z|x^i)}{p_\theta(z|x^i)}\right] = \\
&= \underbrace{\mathbb{E}_z\left[\log p_\theta(x^i|z)\right] - D_{KL}\left(q_\phi(z|x^i)||p_\theta(z)\right)}_{\mathcal{L}(x^i,\theta,\phi)} + \underbrace{D_{KL}\left(q_\phi(z|x^i)||p_\theta(z|x^i)\right)}_{\text{Intractable but } D_{KL}\geq 0}
\end{aligned}
$$

Maximize Variational Lower Bound:

$$
\phi^*, \theta^* = \arg\max_{\phi,\theta} \sum_i^N \mathcal{L}(x^i, \phi, \theta)
$$





Technische Universität Berlin

ELBO Components

$\mathbb{E}_z \left[\log p_\theta(x^i|z)\right]$

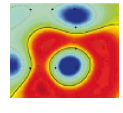
Reconstruction of data from latent space (decoder).
Estimated by sampling.
Differentiable via “reparametrization trick”.

$D_{KL}\left(q_\phi(z|x^i)||p_\theta(z)\right)$

Output distribution of encoder should be close to prior distribution.
Can be computed analytically for given prior distribution and encoder.
Acts as a regularizer.

$D_{KL}\left(q_\phi(z|x^i)||p_\theta(z|x^i)\right)$

Approximation error of encoder to latent distribution.
$p_\theta(z|x)$ is intractable and not optimizable but $D_{KL} \geq 0$.



Learned Manifolds

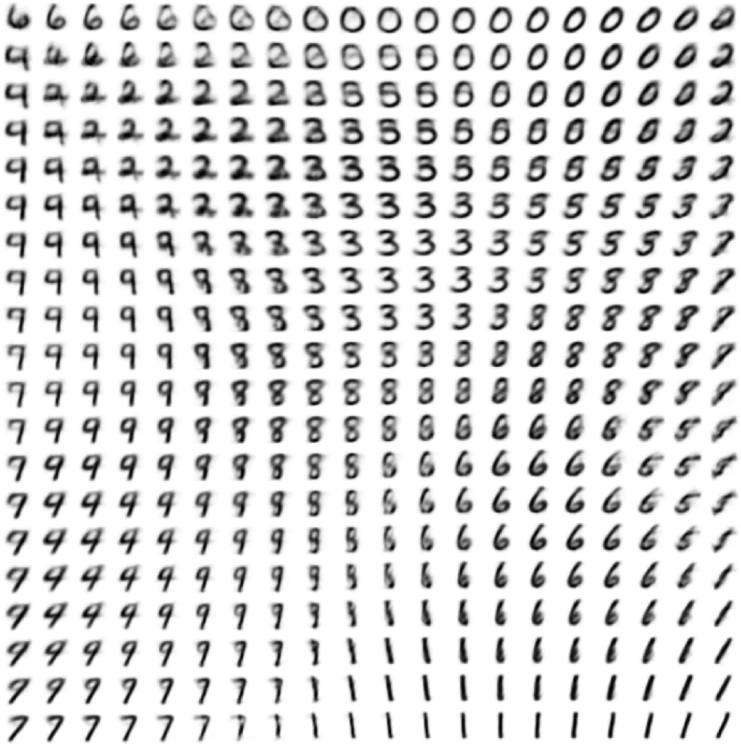
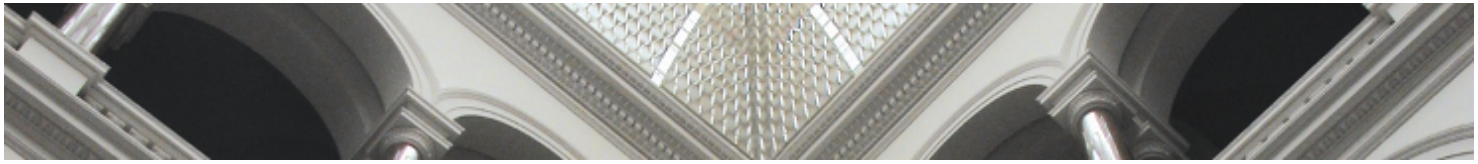


Figure: Learned Frey Faces manifold [KW13]

Figure: Learned MNIST manifold [KW13]



Technische Universität Berlin

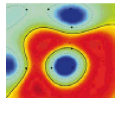
Strengths and Weaknesses

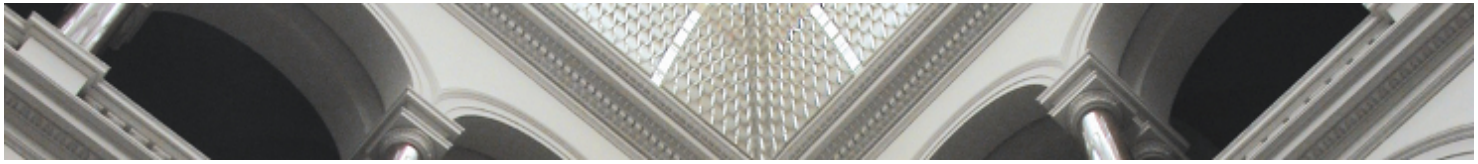
Strengths:

- Density model for $q(z|x)$ and $p(x|z)$. Prior knowledge can be embedded if class of the PDF is known.
- Relatively easy to train compared to GAN
- Sampling is straight forward

Weaknesses:

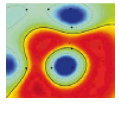
- Samples are blurrier and lower quality compared to GAN
- Maximizes a lower bound of likelihood which is only an approximation

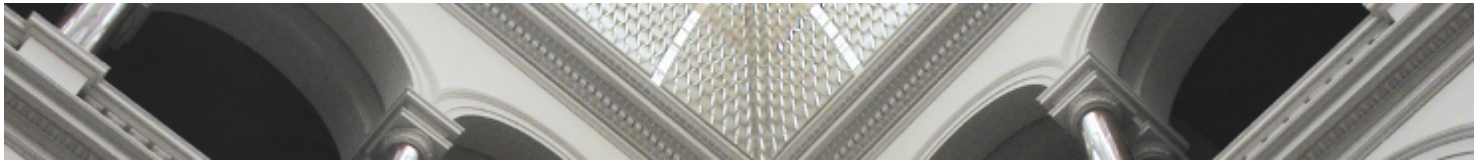




Technische Universität Berlin

Agenda

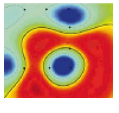


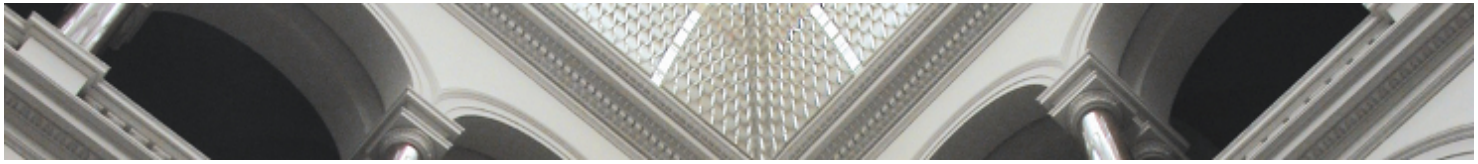


Technische Universität Berlin

Autoregression

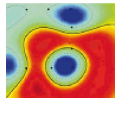
- Autoregressive model: $X_t = f\left(X_{t-1}, \ldots, X_{t-T}\right)$
- Model discrete probability of pixel/audio value
- Joint distribution is product of conditional distributions $\rightarrow$ sequence problem
- Optimize log-likelihood of data w. r. t. parameters

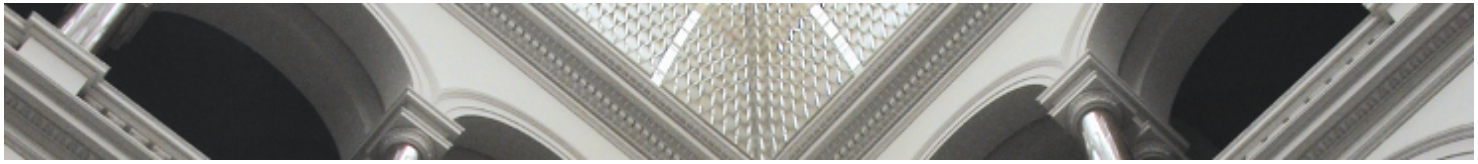




Technische Universität Berlin

Agenda





Technische Universität Berlin

PixelRNN and PixelCNN

Van den Oord et al. (2016) [OKK16]

- Masked convolution $\rightarrow$ causality (context: left and above pixels)
- Joint distribution of all pixels in an image $\vec{x}$:

$$p(\vec{x}) = p(x_1, \ldots, x_{n^2}) = \prod_{i=1}^{n^2} p\left(x_i | x_1, \ldots, x_{i-1}\right) \tag{16}$$

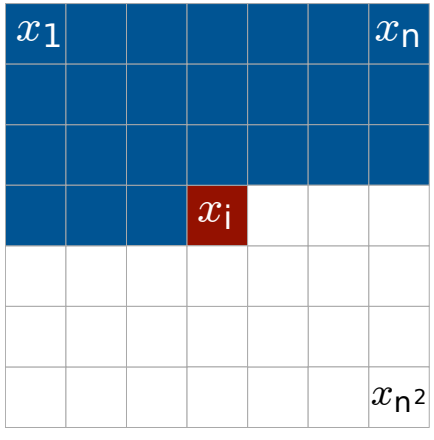
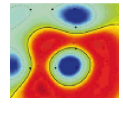
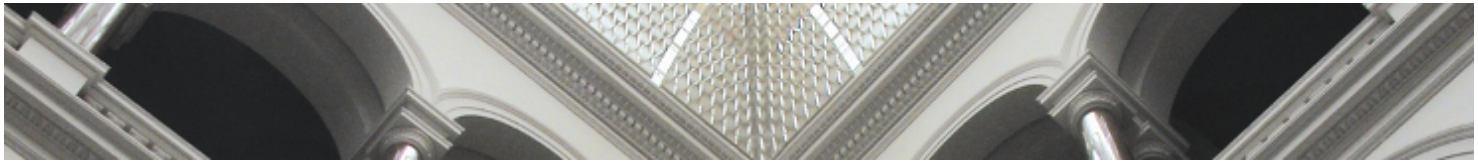


Figure: Context (see Eq. 16) of an autoregressive model for images [OKK16].

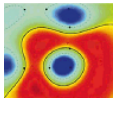


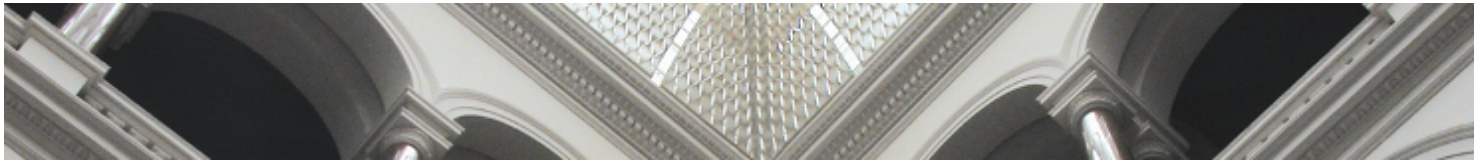


Technische Universität Berlin

Examples

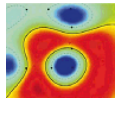
Figure: Image completions from a PixelRNN [OKK16].

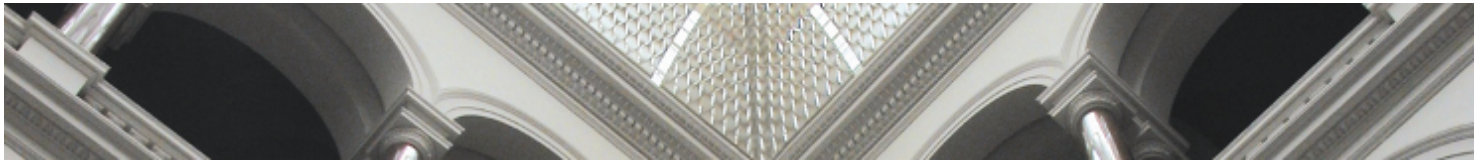




Technische Universität Berlin

Agenda

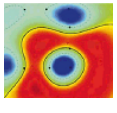


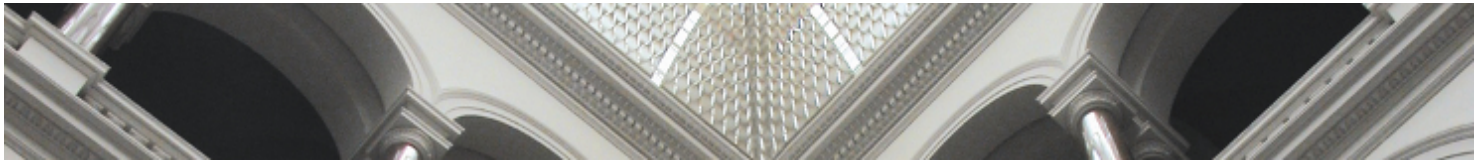


Technische Universität Berlin

WaveNet

- PixelRNN in 1D
- Model in native space $\rightarrow$ model 16k samples per second
- Trained on speech and music





Technische Universität Berlin

À-trous Convolution

Long-range dependencies (exponentially increasing receptive field)

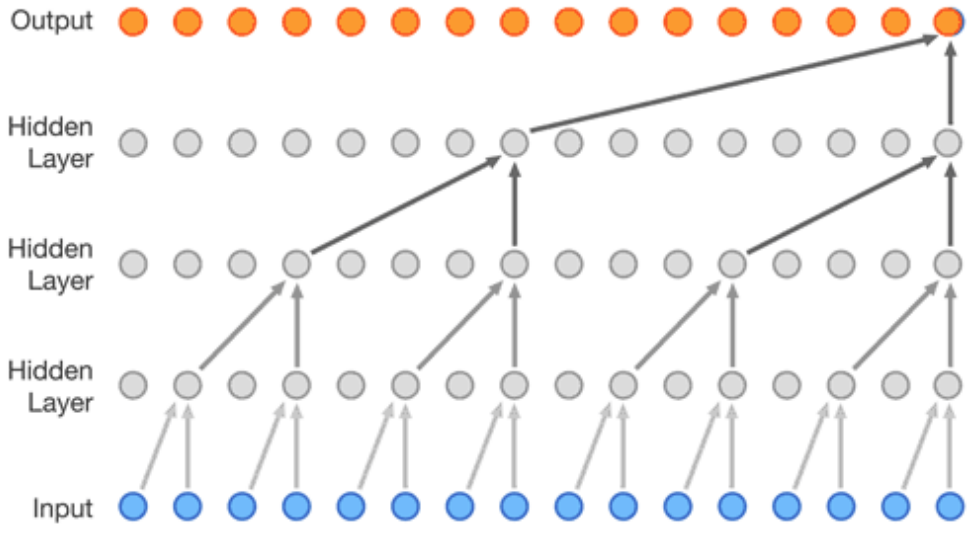
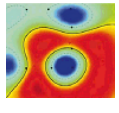
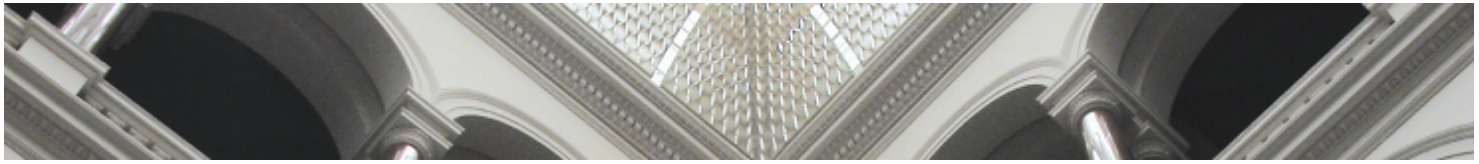


Figure: À trous convolution as used in WaveNet [OKK16].





Technische Universität Berlin

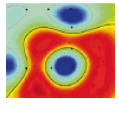
Conditioning

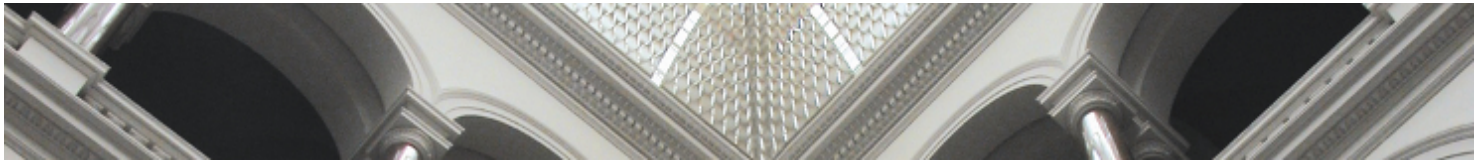
– Enforce characteristics using a condition $\vec{h}$:

$$p(\vec{x}|\vec{h}) = \prod_{i=1}^{n^2} p\left(x_i | x_1, \ldots, x_{i-1}, \vec{h}\right) \tag{17}$$

– Examples for conditions:

- Speaker identity
- Text to speech
- Linguistic features, fundamental frequency

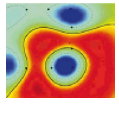


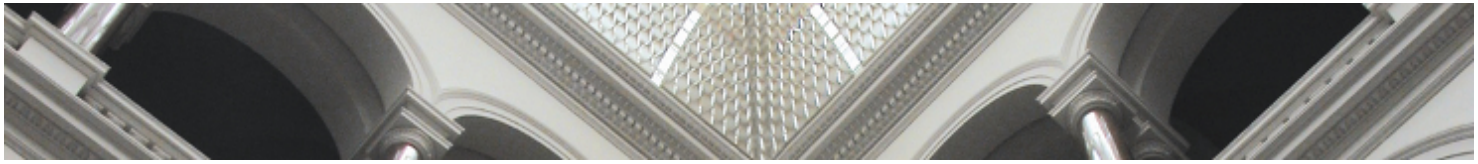


Technische Universität Berlin

Examples

See: `https://deepmind.com/blog/wavenet-generative-model-raw-audio/`





Technische Universität Berlin

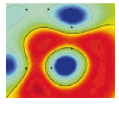
Strengths and Weaknesses

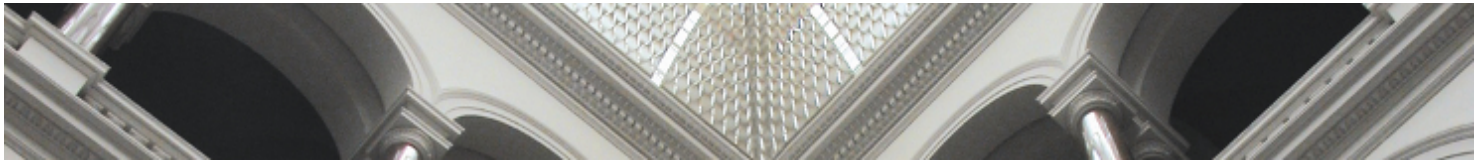
Strengths:

- Estimate log-likelihood
- High quality, sharp output
- Convolutions for an RNN task (more efficient)

Weaknesses:

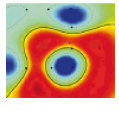
- Slow training and generation (hard to parallelize)

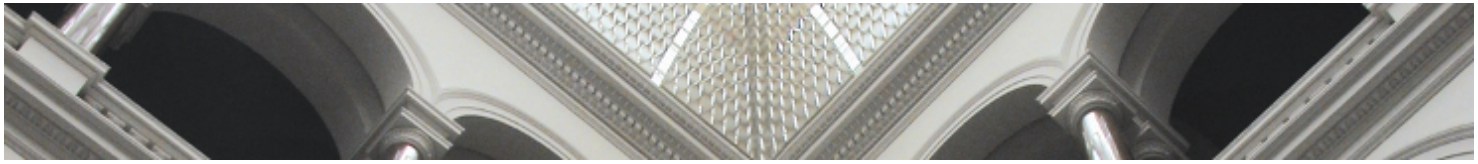




Technische Universität Berlin

Agenda





Technische Universität Berlin

Comparison

RBMs and DBNs:

- Generative pretraining
- Hard to sample (Markov chain)

GAN:

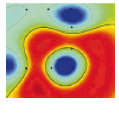
- Realistic output
- Hard to train
- Implicit density model
- Efficient sampling

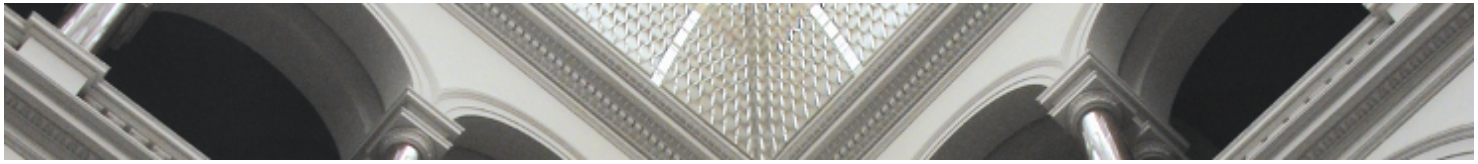
VAE:

- Blurry output
- Explicit density model
- Efficienct sampling

AR models:

- Realistic output
- Hard to sample





Technische Universität Berlin

References I

Martin Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein GAN”. In: arXiv preprint arXiv:1701.07875 (2017).

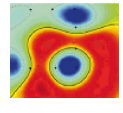
Christopher M Bishop. Pattern recognition and machine learning. springer, 2006.

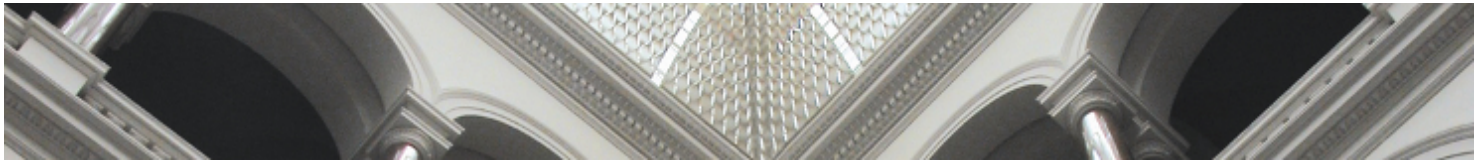
David Berthelot, Tom Schumm, and Luke Metz. “Began: Boundary equilibrium generative adversarial networks”. In: arXiv preprint arXiv:1703.10717 (2017).

Dumitru Erhan et al. “Why does unsupervised pre-training help deep learning?” In: Journal of Machine Learning Research 11.Feb (2010), pp. 625–660.

Jon Gauthier. “Conditional generative adversarial nets for convolutional face generation”. In: Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester (2014).

Ian Goodfellow et al. “Generative adversarial nets”. In: Advances in neural information processing systems. 2014, pp. 2672–2680.





Technische Universität Berlin

References II

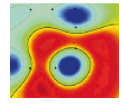
Geoffrey Hinton. Lecture Neural Networks for Machine Learning. 2012. URL: https://www.coursera.org/learn/neural-networks.

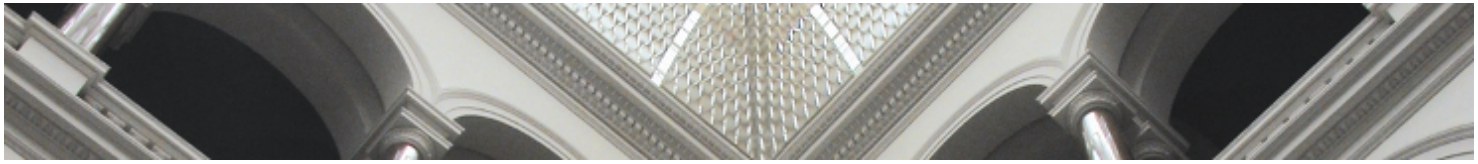
Tero Karras et al. “Progressive growing of gans for improved quality, stability, and variation”. In: arXiv preprint arXiv:1710.10196 (2017).

D. P Kingma and M. Welling. “Auto-Encoding Variational Bayes”. In: ArXiv e-prints (2013). arXiv: 1312.6114.

Fei-Fei Li and Serena Yeung. Lecture Generative Models. 2017. URL: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. “Pixel recurrent neural networks”. In: arXiv preprint arXiv:1601.06759 (2016).



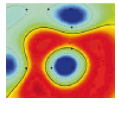


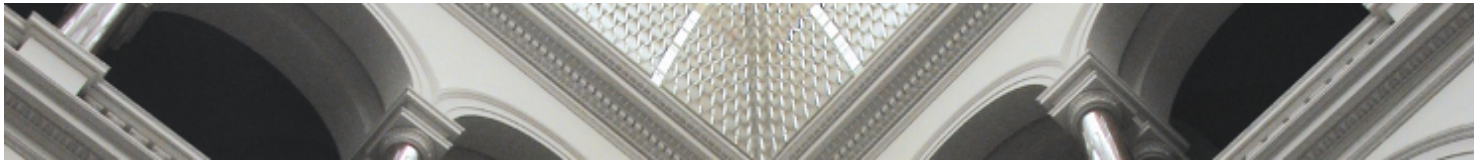
Technische Universität Berlin

References III

Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: arXiv preprint arXiv:1511.06434 (2015).

Tim Salimans et al. “Improved techniques for training gans”. In: Advances in Neural Information Processing Systems. 2016, pp. 2234–2242.





Technische Universität Berlin

Open Source Code

- DCGAN: `https://github.com/Newmu/dcgan_code`
- Wasserstein GAN: `https://github.com/martinarjovsky/WassersteinGAN`
- Cycle GAN and Pix2Pix:
  `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`
- Variational Autoencoder:
  `https://github.com/kvfrans/variational-autoencoder`