

Foundations of Deep Learning, Winter Term 2021/22

Week 12: Variational Autoencoders and Generative Adversarial Networks

Variational Autoencoders and Generative Adversarial Networks

Frank Hutter Abhinav Valada

University of Freiburg



Overview of Week 12

- 1 Introduction to Autoencoders
- 2 Sparse and Denoising Autoencoders
- 3 Generative Models and the Variational Autoencoder
- 4 Generative Adversarial Networks
- 5 Summary, Further Reading, References

Lecture Overview

- 1 Introduction to Autoencoders
- 2 Sparse and Denoising Autoencoders
- 3 Generative Models and the Variational Autoencoder
- 4 Generative Adversarial Networks
- 5 Summary, Further Reading, References

Foundations of Deep Learning, Winter Term 2021/22

Week 12: Variational Autoencoders and Generative Adversarial Networks

Introduction to Autoencoders

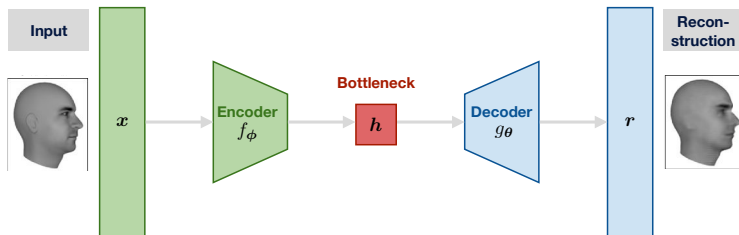
Frank Hutter Abhinav Valada

University of Freiburg



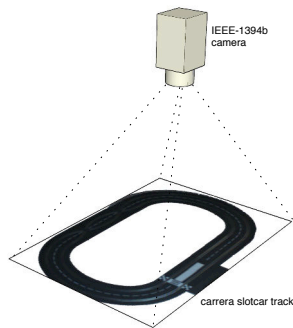
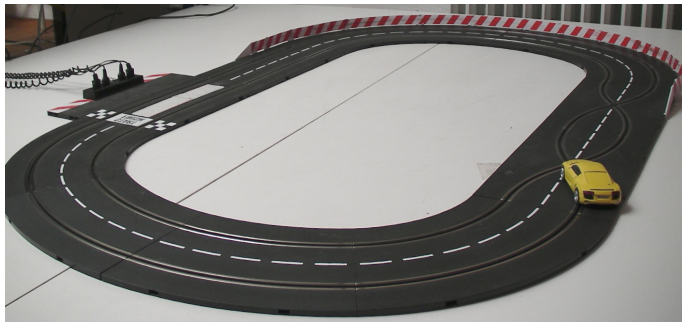
Autoencoders

- Neural networks trained to **reconstruct their input**.



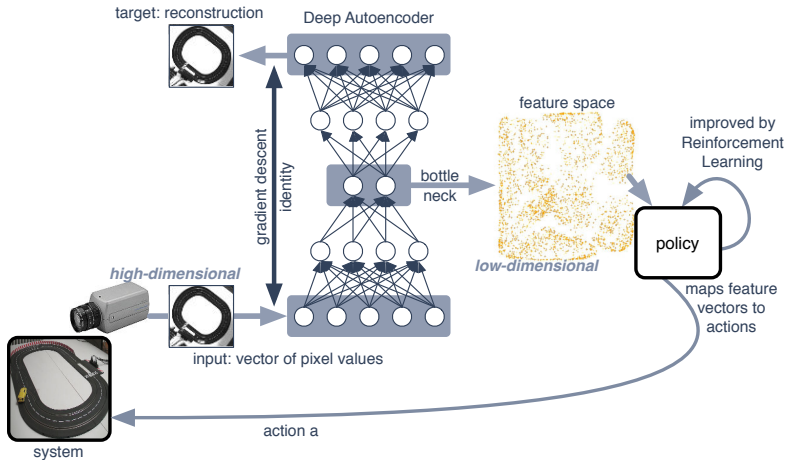
- Architecture** and/or **regularization** is chosen to **prevent** simple copying. → Forces the learning process to **prioritize** what information to keep and what to throw out.
- Hidden layer h describes a compressed **code** to represent the input.
- Two parts: **encoder** function $h = f_\phi(x)$, **decoder** function $r = g_\theta(h)$
- Learning process: minimize $L(x, g_\theta(f_\phi(x)))$

Example Application: Learning to Control a Slotcar from Pixel Input



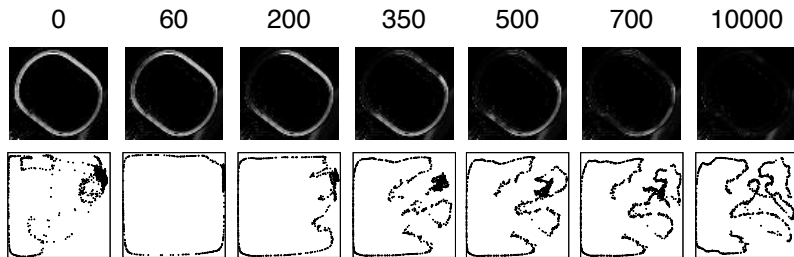
[Image source: Lange et al., 2012]

Overall Setup



[Image source: Lange et al., 2012]

Evolution of the Latent Space



[Image source: Lange et al., 2012]

Questions to Answer for Yourself / Discuss with Friends

- Repetition: How do you train an autoencoder and why is it useful?
- Application of what you just learned: Can you think of other applications where using an autoencoder would make sense?

Lecture Overview

- 1 Introduction to Autoencoders
- 2 Sparse and Denoising Autoencoders**
- 3 Generative Models and the Variational Autoencoder
- 4 Generative Adversarial Networks
- 5 Summary, Further Reading, References

Foundations of Deep Learning, Winter Term 2021/22

Week 12: Variational Autoencoders and Generative Adversarial Networks

Sparse and Denoising Autoencoders

Frank Hutter Abhinav Valada

University of Freiburg

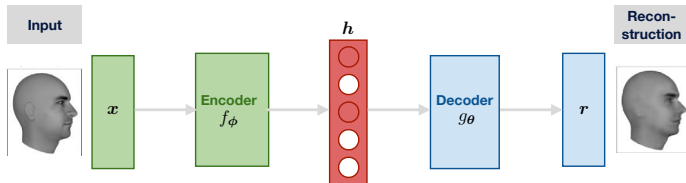


Regularized Autoencoders

- Allow for dimension of $\mathbf{h} \geq$ dimension of \mathbf{x} . \rightarrow **overcomplete** representation
- Model capacity can be chosen based on the complexity of the data distribution.
- Prevent simple copying of the input (and therefore pressure to learn sth. useful about the input data), e.g., by:
 - adding **sparsity** penalty on the hidden layer activations.
 - changing the reconstruction cost to a **denoising** objective.
 - penalizing derivatives of hidden layer activations w.r.t input.
 \rightarrow **contractive autoencoder** (see chapter 14.7 [Goodfellow et al., 2016] for more details)

Sparse Autoencoders

- Add a **sparsity penalty** $\Omega(\mathbf{h})$ on the code \mathbf{h} in the hidden layer to the loss function:
 $L(\mathbf{x}, g_{\theta}(f_{\phi}(\mathbf{x}))) + \Omega(\mathbf{h})$
- Often used as an unsupervised pre-processing step for downstream supervised processing, e.g., classification.
- Sparsity as regularization to avoid overfitting and increase generalization.
- Can be interpreted as **approximating maximum likelihood training** of a **generative model** with visible variables \mathbf{x} and latent variables \mathbf{h} .
- Joint distribution: $p_{model}(\mathbf{x}, \mathbf{h}) = p_{model}(\mathbf{h})p_{model}(\mathbf{x}|\mathbf{h})$

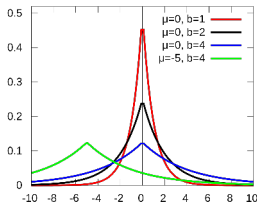


Sparse Autoencoders (cont.)

- Log likelihood can be decomposed as:

$$\log p_{\text{model}}(\mathbf{x}) = \log \sum_{\mathbf{h}} p_{\text{model}}(\mathbf{h}, \mathbf{x})$$

- Sparse autoencoder will approximate this sum with a point estimate of one highly likely value for \mathbf{h} .
- We maximize: $\log p_{\text{model}}(\mathbf{h}, \mathbf{x}) = \log p_{\text{model}}(\mathbf{h}) + \log p_{\text{model}}(\mathbf{x}|\mathbf{h})$
- Choosing a Laplace prior implements sparsity: $p_{\text{model}}(h_i) = \frac{\lambda}{2} \exp^{-\lambda|h_i|}$



[Image source: https://commons.wikimedia.org/wiki/File:Laplace_pdf_mod.svg]

Sparse Autoencoders (cont.)

The (negative) log-prior of

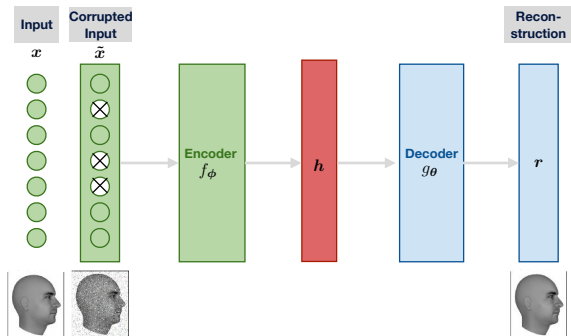
$$p_{model}(h_i) = \frac{\lambda}{2} \exp^{-\lambda|h_i|}$$

corresponds to an absolute value penalty $\Omega(\mathbf{h}) = \lambda \sum_i |h_i|$:

$$-\log p_{model}(\mathbf{h}) = \sum_i (\lambda|h_i| - \log \frac{\lambda}{2}) = \Omega(\mathbf{h}) + const$$

Denoising Autoencoders

- Rather than adding an explicit regularization term, denoising autoencoders (DAE) learn to **undo the effect of noise corruption** when learning to reconstruct the input.
- The DAE minimizes the loss function $L(x, g_{\theta}(f_{\phi}(\tilde{x})))$ with \tilde{x} being a corrupted version of input x .
- To implement this, we introduce a corruption process $C(\tilde{x}|x)$.

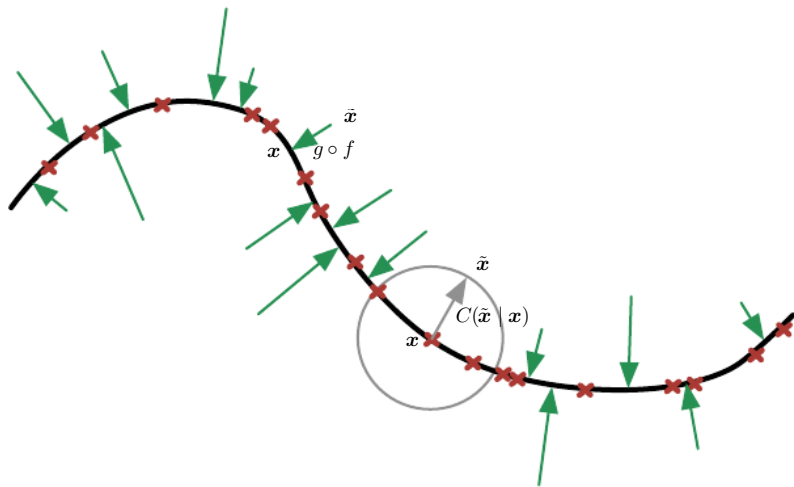


Denoising Autoencoders (cont.)

- The DAE learns a **reconstruction distribution** $p_{\text{reconstruct}}(\mathbf{x}|\tilde{\mathbf{x}})$ from training pairs $(\mathbf{x}, \tilde{\mathbf{x}})$ as follows:
 - ① Sample a training example \mathbf{x} from the training data.
 - ② Sample a corrupted version $\tilde{\mathbf{x}}$ from $C(\tilde{\mathbf{x}}|\mathbf{x} = \mathbf{x})$.
 - ③ Use $(\mathbf{x}, \tilde{\mathbf{x}})$ as a training example for estimating the autoencoder reconstruction distribution $p_{\text{reconstruct}}(\mathbf{x}|\tilde{\mathbf{x}}) = p_{\text{decoder}}(\mathbf{x}|\mathbf{h})$ with \mathbf{h} the output of encoder $f_{\phi}(\tilde{\mathbf{x}})$ and p_{decoder} typically defined by a decoder $g_{\theta}(\mathbf{h})$.
- Perform gradient-based approximate minimization on the negative log-likelihood $-\log p_{\text{decoder}}(\mathbf{x}|\mathbf{h})$ (same techniques as for other types of FFN, as long as encoder is deterministic).
- The DAE therefore performs stochastic gradient descent on:

$$-\mathbb{E}_{\mathbf{x} \sim \hat{p}_{\text{data}}(\mathbf{x})} \mathbb{E}_{\tilde{\mathbf{x}} \sim C(\tilde{\mathbf{x}}|\mathbf{x})} \log p_{\text{decoder}}(\mathbf{x}|\mathbf{h} = f(\tilde{\mathbf{x}}))$$

Denoising Autoencoders (cont.)



[Image source: Goodfellow et al., 2016, p. 512, Fig. 14.4]

Questions to Answer for Yourself / Discuss with Friends

- Repetition: Why do we need regularization in overcomplete autoencoders?
- Repetition: What is the difference between a sparse and a denoising autoencoder?

Lecture Overview

- 1 Introduction to Autoencoders
- 2 Sparse and Denoising Autoencoders
- 3 Generative Models and the Variational Autoencoder**
- 4 Generative Adversarial Networks
- 5 Summary, Further Reading, References

Foundations of Deep Learning, Winter Term 2021/22

Week 12: Variational Autoencoders and Generative Adversarial Networks

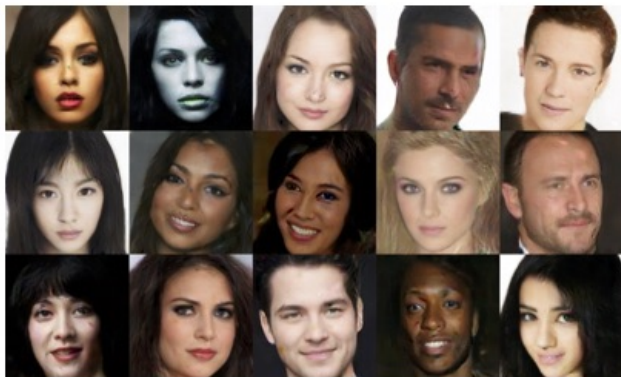
Generative Models and the Variational Autoencoder

Frank Hutter Abhinav Valada

University of Freiburg



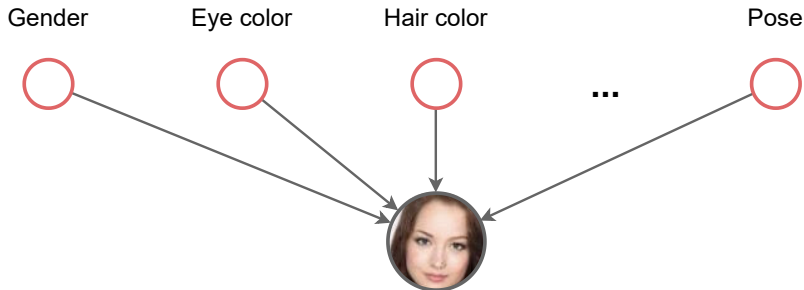
Illustration: Factors of Variation in Images



Factors that affect the **variability** in images of faces x are, e.g., gender, eye color, hair color, pose, etc. All we can observe in the image are the pixels though, the factors of variation are not explicitly available, i.e., they are **latent**.

[Image source: Berthelot et al., 2017]

Illustration: Factors of Variation in Images (cont.)



- **Idea:** explicitly model these factors using latent variables z
- If z reflects the **causal factors** that determine the appearance of the pixels in x , then $p(x|z)$ should be of an easier form of trying to model all the possible sources of variation in $p(x)$ directly.

[Image (face) source: Berthelot et al., 2017]

Latent Variable Generative Models

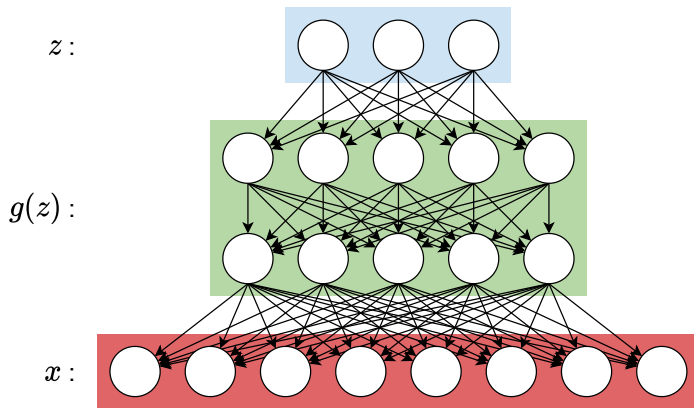
- Assuming set of **visible** variables \mathbf{x} and **latent** variables \mathbf{z} our goal is to model the distribution of the training data :

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

- This breaks down the modeling problem in a hierarchical way: a context / explanatory factors $p(\mathbf{z})$ of our data, and a generating distribution $p(\mathbf{x}|\mathbf{z})$ which is conditioned on them.
- This is used in a variety of probabilistic models, e.g., **Gaussian mixture models**, **topic models**, etc.
- Once we have a good approximation of $p(\mathbf{x})$, we can **sample** from it and generate new data that is **very similar, but different** from our original training data.

Variational Autoencoder (VAE) Approach

Use neural networks as **deterministic parts** within a graphical model to be able to express complex relationship between latent and visible variables.



The Problem: How to Do Inference?

- How do we get z values that are related to our input data?
- This is the problem of inferring the **posterior** distribution $p(z|\mathbf{x})$.
- Bayes formula says:

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

- Unfortunately, the integral in the denominator is intractable!
→ Have to resort to **approximate inference** methods

Variational Inference in the VAE

- **Idea**: Approximate the posterior with a simpler distribution $q(\mathbf{z}|\mathbf{x})$ with its own set of parameters and **optimize** them to get as close as possible to $p(\mathbf{z}|\mathbf{x})$.
- The difference between the approximate and the real posterior can be measured by the KL-divergence:

$$\begin{aligned} KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \left[\log \frac{q(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \left[\log q(\mathbf{z}|\mathbf{x}) - \log \frac{p(\mathbf{z}, \mathbf{x})}{p(\mathbf{x})} \right] \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} [\log q(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})] + \log p(\mathbf{x}) \end{aligned}$$

- We cannot compute this KL-divergence directly because of the intractable integration needed to get $p(\mathbf{x})$. We can, however, manipulate q .

Variational Inference in the VAE (cont.)

- Observe that

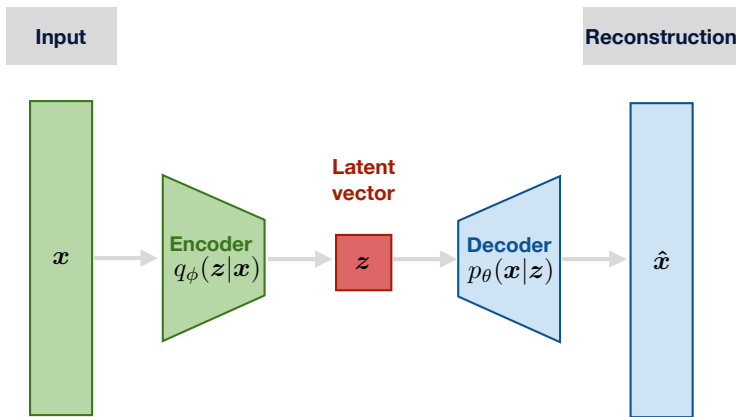
$$\log p(\mathbf{x}) = -\mathbb{E}_{q(\mathbf{z}|\mathbf{x})} [\log q(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})] + KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$$

and that the KL-divergence has a minimum of zero.

- This means that $\mathcal{L}(\mathbf{x}, q) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{z}, \mathbf{x}) - \log q(\mathbf{z}|\mathbf{x})]$ is a lower bound on $\log p(\mathbf{x})$.
- It is called the **evidence lower bound** (ELBO) or the **variational lower bound**.
- For an appropriate choice of q , \mathcal{L} is tractable to compute. We maximize \mathcal{L} to make the bound as tight as possible (KL-divergence will be zero at the maximum).

Variational Autoencoder

The **variational autoencoder** (VAE) uses two neural networks: an **encoder** network with parameters ϕ that learns to approximate the posterior, and a **decoder** network with parameters θ that learns to reconstruct the input.



Variational Autoencoder (cont.)

- With this parametrization, the ELBO for variational inference is:

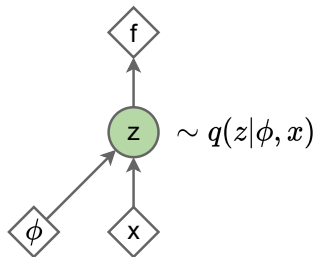
$$\begin{aligned}\mathcal{L}(\theta, \phi, \mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{z}, \mathbf{x}) - \log q_\phi(\mathbf{z} | \mathbf{x})] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z}) + \log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z} | \mathbf{x})] \\ &= \underbrace{-KL(\log q_\phi(\mathbf{z}|\mathbf{x}) || \log p_\theta(\mathbf{z}))}_{\text{regularization term}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z} | \mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})]}_{\text{reconstruction term}}\end{aligned}$$

- We approximate the expectation in the ELBO by sampling from posterior over \mathbf{z} , but sampling is not a continuous operation – how can we backprop through this step?
- Idea:** Reparametrize to express our random variable \mathbf{z} as a deterministic variable and an external, static noise source expressed in random variable ϵ .

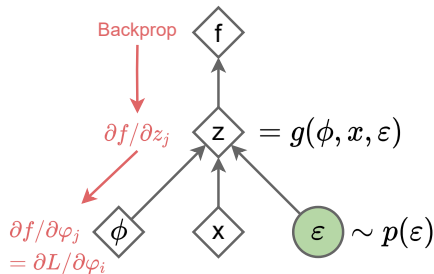
Reparametrization Trick

Let $\mathbf{z} \in \mathbb{R}$ and $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\mathbf{z}(\mathbf{x}), \boldsymbol{\sigma}_\mathbf{z}(\mathbf{x})^2 \odot \mathbf{I})$

Original form:



Reparameterized form:

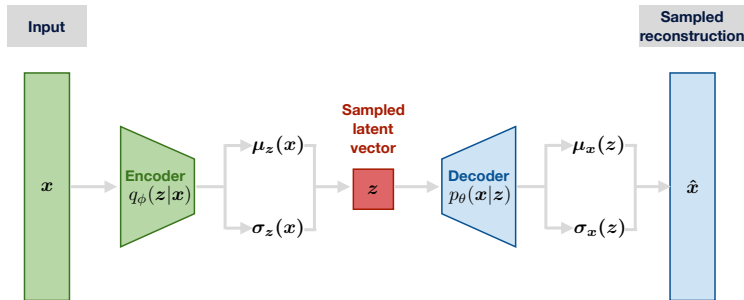


- Use parametrization $\mathbf{z} = \boldsymbol{\mu}_\mathbf{z}(\mathbf{x}) + \boldsymbol{\sigma}_\mathbf{z}(\mathbf{x}) \odot \epsilon_\mathbf{z}$ where $\epsilon_\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
- Optional: use parametrization $\mathbf{x} = \boldsymbol{\mu}_\mathbf{x}(\mathbf{z}) + \boldsymbol{\sigma}_\mathbf{x}(\mathbf{z}) \odot \epsilon_\mathbf{x}$ where $\epsilon_\mathbf{x} \sim \mathcal{N}(0, \mathbf{I})$

Training with Backpropagation

Now, with this [reparametrization](#), both the [generative](#) model $p_{\theta}(x|z)$ and the [inference](#) model $q_{\phi}(z|x)$ can be trained together using [backpropagation](#) to optimize the ELBO as objective function:

$$\mathcal{L}(\theta, \phi, x) = -KL(\log q_{\phi}(z|x) || \log p_{\theta}(z)) + \mathbb{E}_{q_{\phi}(z|x)} [\log p_{\theta}(x|z)]$$



Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\theta, \phi \leftarrow$ Initialize parameters

repeat

$\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)

$\epsilon \leftarrow$ Random samples from noise distribution $p(\epsilon)$

$\mathbf{g} \leftarrow \nabla_{\theta, \phi} \tilde{\mathcal{L}}^M(\theta, \phi; \mathbf{X}^M, \epsilon)$ (Gradients of minibatch estimator (8))

$\theta, \phi \leftarrow$ Update parameters using gradients \mathbf{g} (e.g. SGD or Adagrad [DHS10])

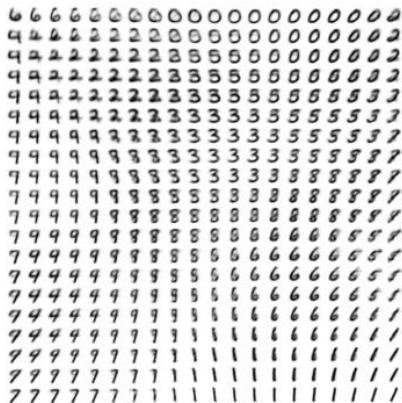
until convergence of parameters (θ, ϕ)

return θ, ϕ

[Algorithm source: Kingma and Welling, 2013]

Learning Smooth Latent Spaces

Training a VAE with a two-dimensional latent code on MNIST (left) and the Frey Face dataset, we get the following two latent spaces by varying z on a uniform grid.



[Image source: Kingma and Welling, 2013]

Questions to Answer for Yourself / Discuss with Friends

- Repetition: What is the idea behind generative models?
- Repetition: What is the reparametrization trick?
- Repetition: How do you train a VAE?

Lecture Overview

- 1 Introduction to Autoencoders
- 2 Sparse and Denoising Autoencoders
- 3 Generative Models and the Variational Autoencoder
- 4 Generative Adversarial Networks**
- 5 Summary, Further Reading, References

Foundations of Deep Learning, Winter Term 2021/22

Week 12: Variational Autoencoders and Generative Adversarial Networks

Generative Adversarial Networks

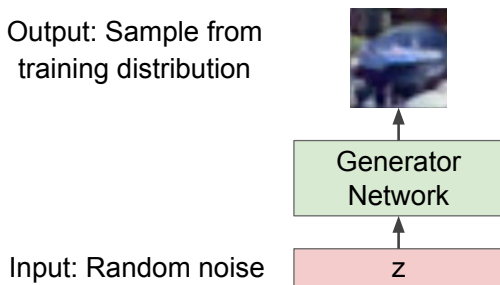
Frank Hutter Abhinav Valada

University of Freiburg



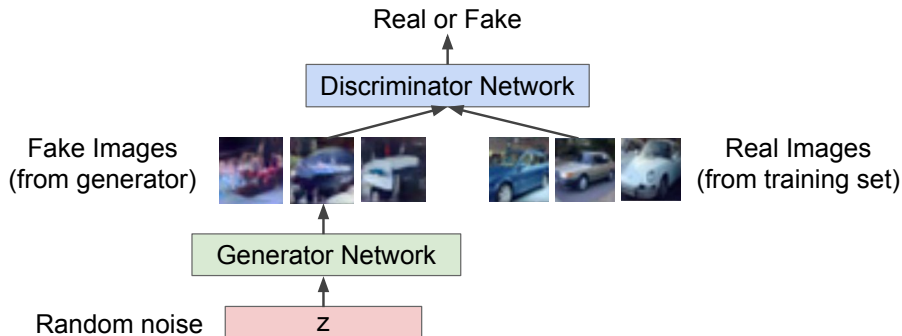
Generative Adversarial Networks

- **Motivation:** GANs are generative models, similar to VAEs but drop the assumption of optimizing the lower variational bound. Hence they often produce better results.
- Want to sample from complex, high-dimensional training distribution. No direct way to do this!
- Sample from a simple distribution, e.g., random noise. Learn transformation to training distribution.



Training GANs: Two-Player Game

- Generator network: try to fool the discriminator by generating real-looking images.
- Discriminator network: try to distinguish between real and fake images.



Training GANs: Two-Player Game (cont.)

- Train jointly in minimax game with the objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

- Discriminator outputs likelihood in (0,1) of real image.
- Discriminator (θ_d) wants to maximize objective such that $D(x)$ is close to 1 (real) and $D(G(z))$ is close to 0 (fake).
- Generator (θ_g) wants to minimize objective such that $D(G(z))$ is close to 1 (discriminator is fooled into thinking generated $G(z)$ is real).

Training GANs: Two-Player Game (cont.)

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. Gradient ascent on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. Gradient descent on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

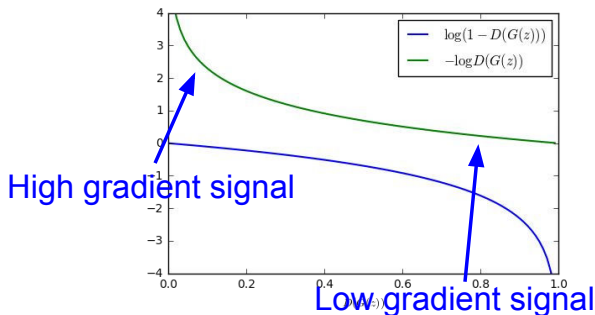
A Better Cost Function

Original minmax cost:

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Modified generator cost:

$$\max_{\theta_d} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$



GAN Training Algorithm

for number of training iterations **do**

for k steps **do**

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

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

end for

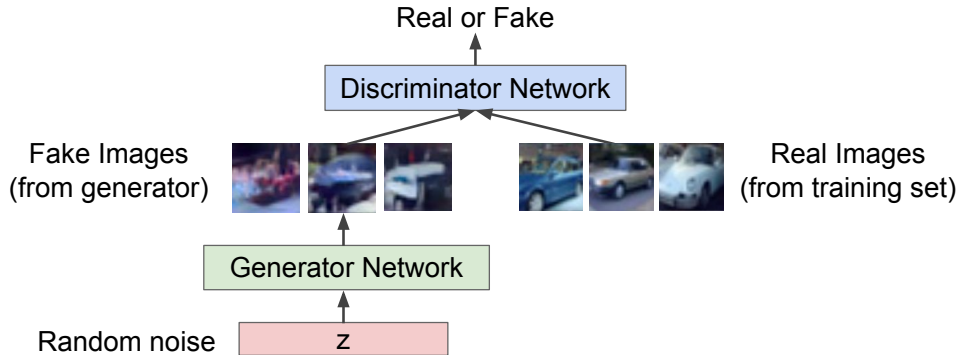
- Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Update the generator by ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

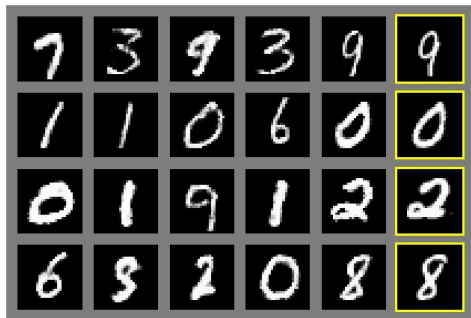
Training GANs: Two-Player Game (cont.)

- Generator network: try to fool the discriminator by generating real-looking images.
- Discriminator network: try to distinguish between real and fake images.
- After training, use generator network to generate new images.



Generative Adversarial Nets

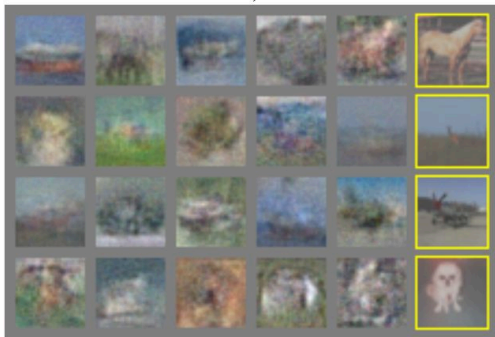
Generated Samples



[Image source: Goodfellow et al., 2014]

Generative Adversarial Nets (cont.)

Generated Samples (CIFAR-10)



[Image source: Goodfellow et al., 2014]

Generative Adversarial Nets – Convolutional Architectures

Architecture guidelines for stable deep convolutional GANs:

- Generator is an upsampling network with fractionally-strided convolutions.
- Discriminator is a convolutional network.
- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses tanh.
- Use LeakyReLU activation in the discriminator for all layers.

[Items 3 to 7 are from: Radford et al., 2016]

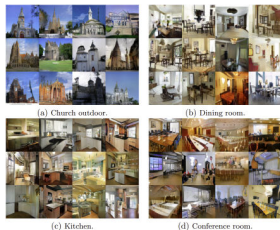
Generative Adversarial Nets – Convolutional Architectures (cont.)



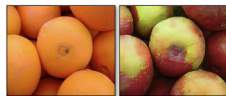
[Image source: Radford et al., 2016]

2017: Explosion of GANs

Better training and generation



Source → target domain transfer



CycleGAN [Zhu et al., 2017]

Text → image synthesis

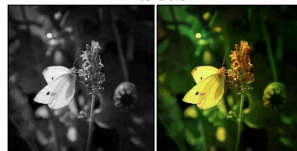
this small bird has a pink breast and crown, and black primaries and secondaries. this magnificent fellow is almost all black with a red crest, and white cheek patch.



[Zhu et al., 2017]

Many GAN applications

BW to Color



Labels to Street Scene



Pix2Pix [Isola et al., 2017]

2019: BigGAN



[Images source: Brock et al., 2019]

“The GAN Zoo”

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorical GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

[Source: <https://github.com/hindupuravinash/the-gan-zoo>]

Questions to Answer for Yourself / Discuss with Friends

- Repetition: While training a GAN, do we train the generator first or the discriminator first, and why not the other way around?
- Application of what you just learned: Can you think of other application domains, where GANs can be helpful, other than in computer vision?

Lecture Overview

- 1 Introduction to Autoencoders
- 2 Sparse and Denoising Autoencoders
- 3 Generative Models and the Variational Autoencoder
- 4 Generative Adversarial Networks
- 5 Summary, Further Reading, References**

Foundations of Deep Learning, Winter Term 2021/22

Week 12: Variational Autoencoders and Generative Adversarial Networks

Summary, Further Reading, References

Frank Hutter Abhinav Valada

University of Freiburg



Summary by Learning Goals

Having heard this lecture, you can now . . .

- explain the working mechanism behind autoencoders.
- describe regularization techniques for autoencoders such as sparsity and denoising.
- explain how to perform inference with variational autoencoders.
- elaborate on the reparametrization trick.
- describe the training mechanism of GANs.
- provide an overview of possible applications of GANs.

Further Reading

- Resources for this lecture: Deep Learning Book [Goodfellow et al., 2016, chapter 14]

References

- Berthelot, D., Schumm, T., Metz, Luke (2017)
BEGAN: Boundary equilibrium generative adversarial networks
arXiv preprint arXiv:1703.10717
<https://arxiv.org/pdf/1703.10717.pdf>
- Brock, A., Donahue, J., Simonyan, K. (2019)
Large Scale GAN Training for High Fidelity Natural Image Synthesis
International Conference on Learning Representations
<https://arxiv.org/pdf/1809.11096.pdf>
- Goodfellow, I., Bengio, Y., Courville, A. (2016)
Deep Learning
MIT Press
<https://www.deeplearningbook.org>
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2014)
Generative adversarial nets
Advances in neural information processing systems, 27
<https://arxiv.org/pdf/1406.2661.pdf>
- Isola, P., Zhu, J.-Y., Zhou, T., Efros, A. (2017)
Image-to-image translation with conditional adversarial networks
Proceedings of the IEEE conference on computer vision and pattern recognition, 1125–1134
<https://arxiv.org/pdf/1611.07004.pdf>
- Lange, S., Riedmiller, M., Voigtländer, A. (2012)
Autonomous reinforcement learning on raw visual input data in a real world application
The 2012 international joint conference on neural networks (IJCNN), 1–8
https://ml.informatik.uni-freiburg.de/former/_media/publications/rieijcnn12.pdf
- Kingma, D., Welling, M. (2013)
Auto-encoding variational bayes
arXiv preprint arXiv:1312.6114
<https://arxiv.org/pdf/1312.6114.pdf>
- Mao, X., Li, Q., Xie, H., Lau, R., Wang, Z., Paul Smolley, S. (2017)
Least squares generative adversarial networks
Proceedings of the IEEE international conference on computer vision, 2794–2802
<https://arxiv.org/pdf/1611.04076.pdf>
- Radford, A., Metz, L., Chintala, S. (2016)
Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
4th International Conference on Learning Representations ICLR (poster)
<https://arxiv.org/pdf/1511.06434.pdf>
- Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., Lee, H. (2016)
Generative adversarial text to image synthesis
International Conference on Machine Learning, 1060–1069
<https://arxiv.org/pdf/1605.05396.pdf>

References

Zhu, J.-Y., Park, T., Isola, P., Efros, A. (2017)

Unpaired image-to-image translation using cycle-consistent adversarial networks

Proceedings of the IEEE international conference on computer vision, 2223–2232

<https://arxiv.org/pdf/1703.10593.pdf>