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

Lecture Overview

Foundations of Deep Learning, Winter Term 2021/22

Week 12: Variational Autoencoders and Generative Adversarial Networks

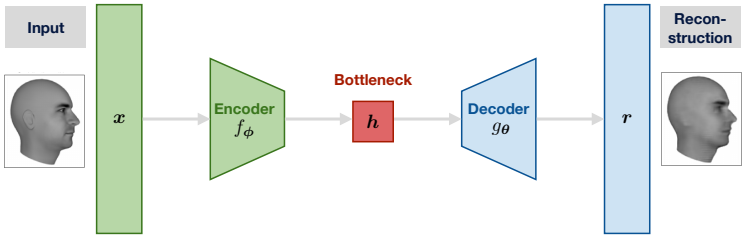
Introduction to Autoencoders

Frank Hutter Abhinav Valada

University of Freiburg

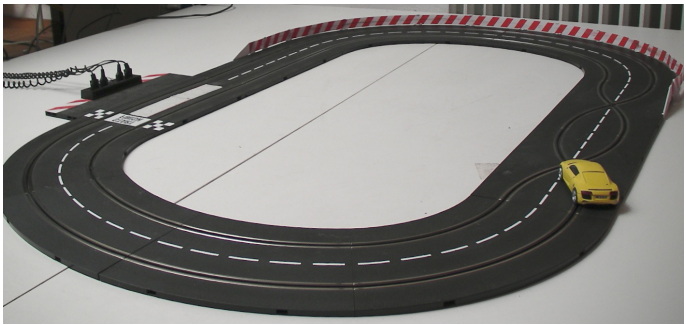
Autoencoders

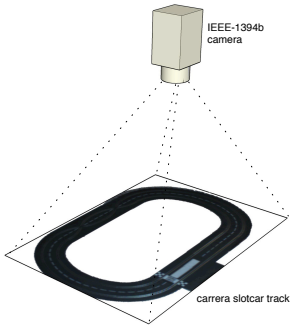
- Neural networks trained to reconstruct their input.



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

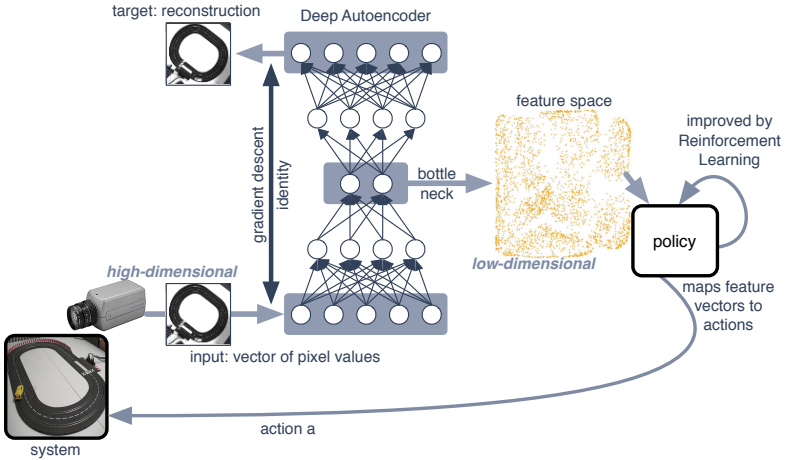
Example Application: Learning to Control a Slotcar from Pixel Input





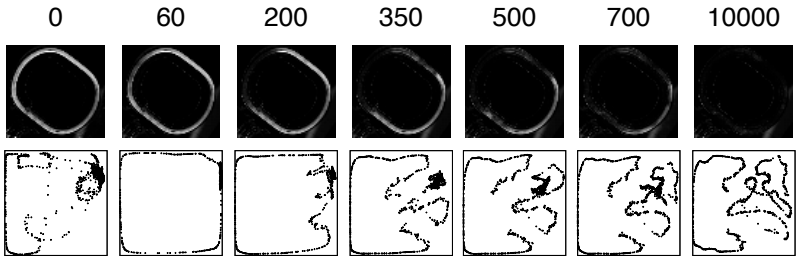
[Image source: Lange et al., 2012]

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

Foundations of Deep Learning, Winter Term 2021/22

Week 12: Variational Autoencoders and Generative Adversarial Networks

Sparse and Denoising Autoencoders

Frank Hutter Abhinav Valada

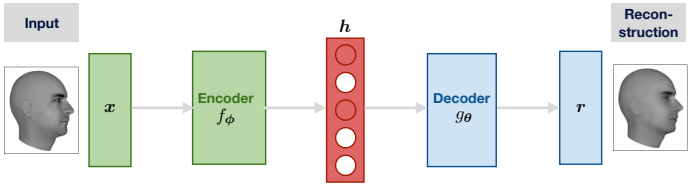
University of Freiburg

Regularized Autoencoders

- Allow for dimension of $\boldsymbol{h} \geq$ dimension of $\boldsymbol{x}$. $\rightarrow$ 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 $\Omega(\boldsymbol{h})$ on the code $\boldsymbol{h}$ in the hidden layer to the loss function: $L(\boldsymbol{x}, g_\theta(f_\phi(\boldsymbol{x}))) + \Omega(\boldsymbol{h})$
- 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 $\boldsymbol{x}$ and latent variables $\boldsymbol{h}$.
- Joint distribution: $p_{model}(\boldsymbol{x}, \boldsymbol{h}) = p_{model}(\boldsymbol{h}) p_{model}(\boldsymbol{x}|\boldsymbol{h})$

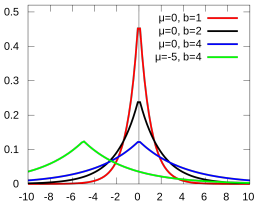


Sparse Autoencoders (cont.)

- Log likelihood can be decomposed as:

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

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



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

Sparse Autoencoders (cont.)

The (negative) log-prior of

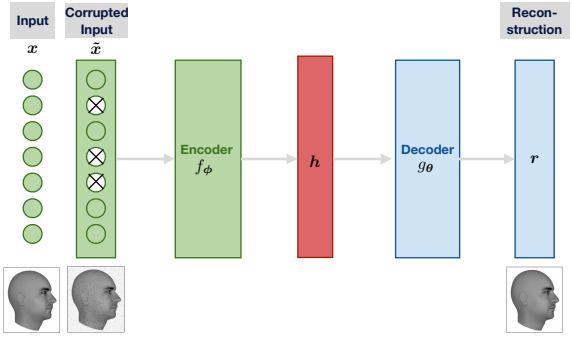
$$p_{model}(h_i) = \frac{\lambda}{2} \exp^{-\lambda|h_i|}$$

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

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

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 $L(\boldsymbol{x}, g_\theta(f_\phi(\tilde{\boldsymbol{x}})))$ with $\tilde{\boldsymbol{x}}$ being a corrupted version of input $\boldsymbol{x}$.
- To implement this, we introduce a corruption process $C(\tilde{\boldsymbol{x}}|\boldsymbol{x})$.

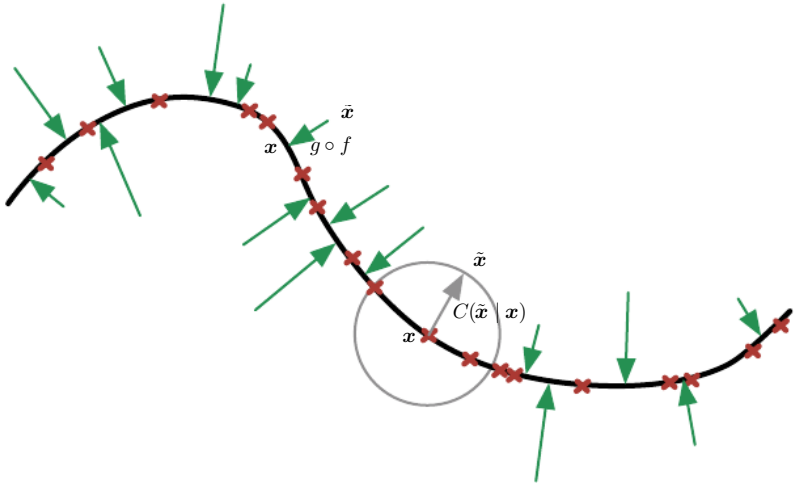


Denoising Autoencoders (cont.)

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

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

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

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

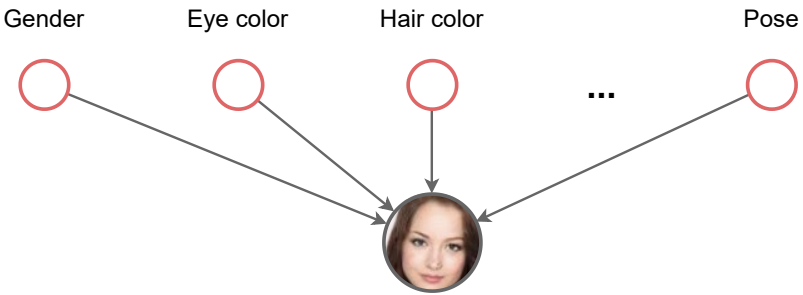
University of Freiburg

Illustration: Factors of Variation in Images

Factors that affect the variability in images of faces $\boldsymbol{x}$ 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 $\boldsymbol{z}$
- If $\boldsymbol{z}$ reflects the causal factors that determine the appearance of the pixels in $\boldsymbol{x}$, then $p(\boldsymbol{x}|\boldsymbol{z})$ should be of an easier form of trying to model all the possible sources of variation in $p(\boldsymbol{x})$ directly.

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

Latent Variable Generative Models

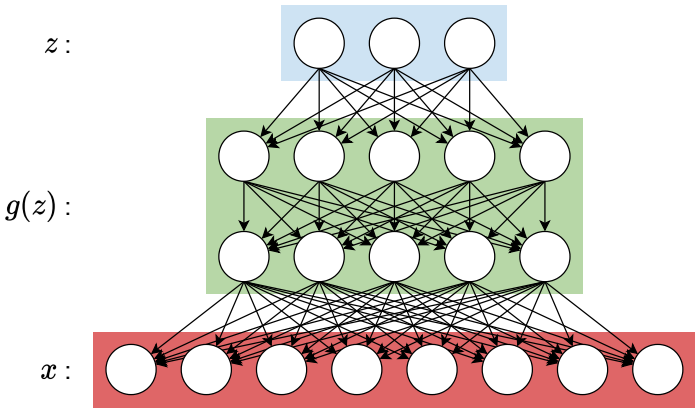
- Assuming set of visible variables $\boldsymbol{x}$ and latent variables $\boldsymbol{z}$ our goal is to model the distribution of the training data :

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

- This breaks down the modeling problem in a hierarchical way: a context / explanatory factors $p(\boldsymbol{z})$ of our data, and a generating distribution $p(\boldsymbol{x}|\boldsymbol{z})$ 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 $p(\boldsymbol{x})$, 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 $p(\boldsymbol{z}|\boldsymbol{x})$.
- Bayes formula says:

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

- Unfortunately, the integral in the denominator is intractable!
  $\rightarrow$ Have to resort to approximate inference methods

Variational Inference in the VAE

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

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

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

Variational Inference in the VAE (cont.)

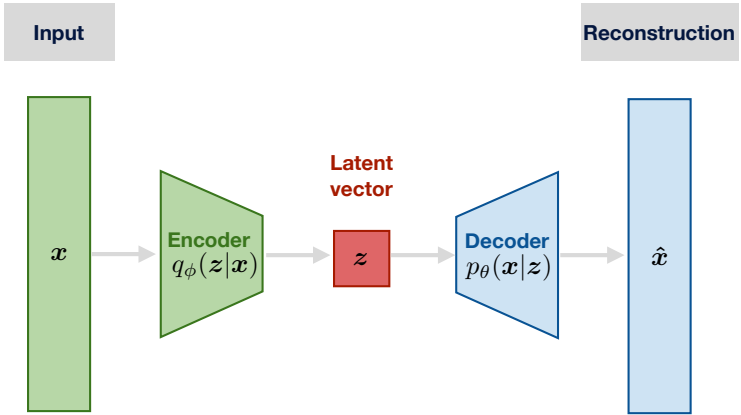
- Observe that

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

  and that the KL-divergence has a minimum of zero.
- This means that $\mathcal{L}(\boldsymbol{x}, q) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\boldsymbol{z}, \boldsymbol{x}) - \log q(\boldsymbol{z}|\boldsymbol{x})\right]$ is a lower bound on $\log p(\boldsymbol{x})$.
- 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:

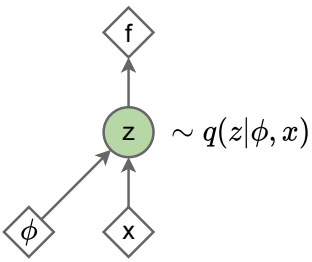
$$\begin{aligned}\mathcal{L}(\theta, \phi, \boldsymbol{x}) &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{z}, \boldsymbol{x}) - \log q_\phi(\boldsymbol{z} \mid \boldsymbol{x})\right] \\ &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})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

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

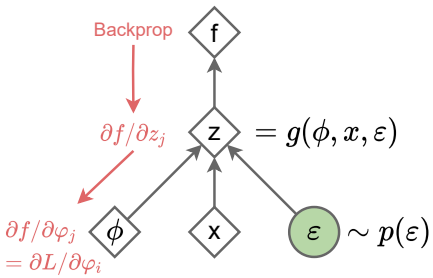
Reparametrization Trick

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

Original form:



Reparameterized form:

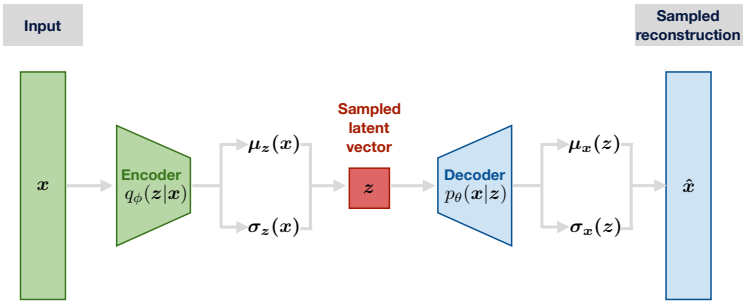


- Use parametrization $\boldsymbol{z} = \boldsymbol{\mu_z}(\boldsymbol{x}) + \boldsymbol{\sigma_z}(\boldsymbol{x}) \odot \epsilon_{\boldsymbol{z}}$ where $\epsilon_{\boldsymbol{z}} \sim \mathcal{N}(0, \boldsymbol{I})$
- Optional: use parametrization $\boldsymbol{x} = \boldsymbol{\mu_x}(\boldsymbol{z}) + \boldsymbol{\sigma_x}(\boldsymbol{z}) \odot \epsilon_{\boldsymbol{x}}$ where $\epsilon_{\boldsymbol{x}} \sim \mathcal{N}(0, \boldsymbol{I})$

Training with Backpropagation

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

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



VAE Training Pseudocode

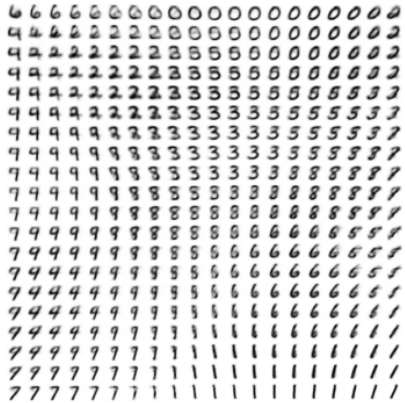
Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

[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 (right) 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

Foundations of Deep Learning, Winter Term 2021/22

Week 12: Variational Autoencoders and Generative Adversarial Networks

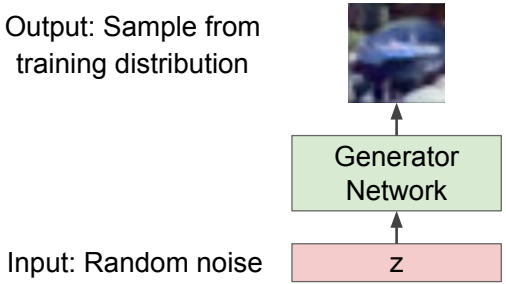
Generative Adversarial Networks

Frank Hutter Abhinav Valada

University of Freiburg

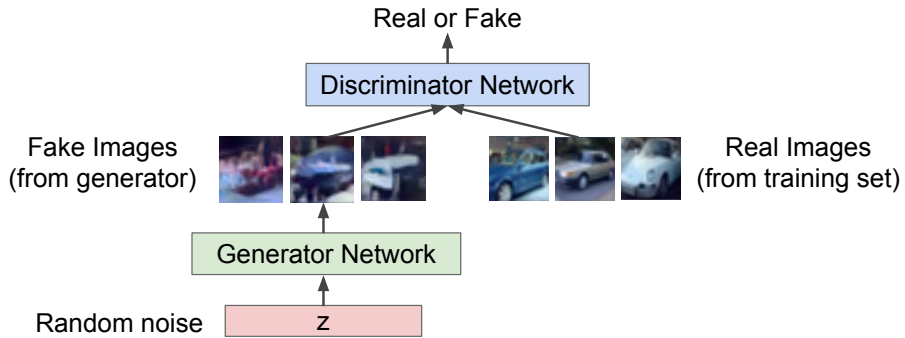
Generative Adversarial Networks

- 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_d) 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

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

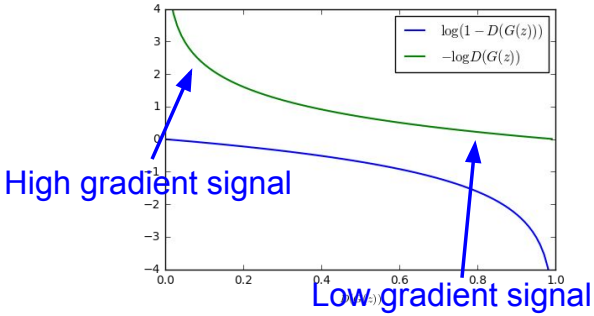
A Better Cost Function

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 $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ 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:

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

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 ascending its stochastic gradient (improved objective):

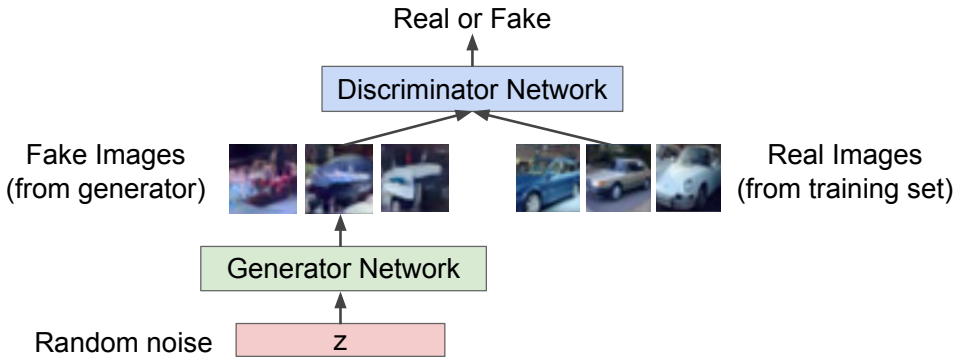
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

[Algorithm source: Goodfellow et al., 2014]

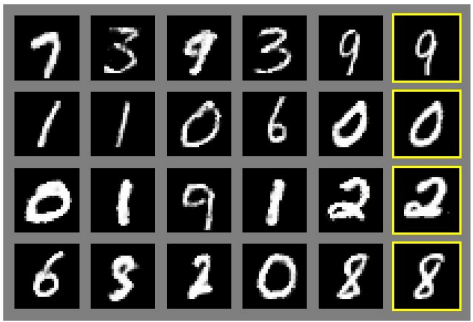
Training GANs: Two-Player Game (cont.)

- 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

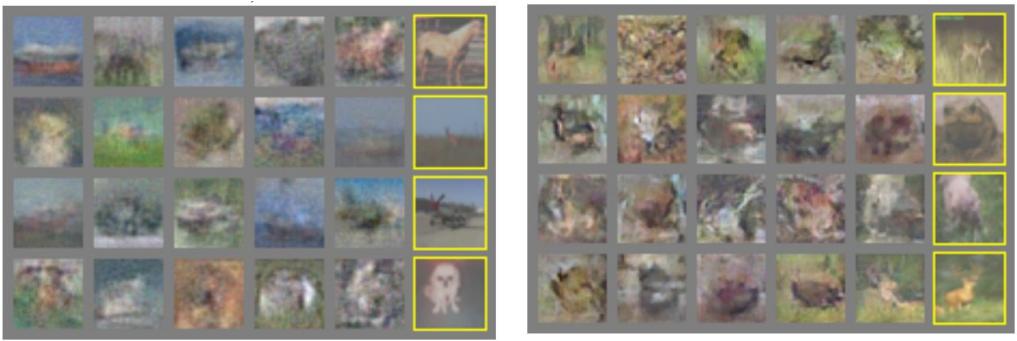
Generated Samples



[Image source: Goodfellow et al., 2014]

Generative Adversarial Nets (cont.)

Generated Samples (CIFAR-10)



[Image source: Goodfellow et al., 2014]

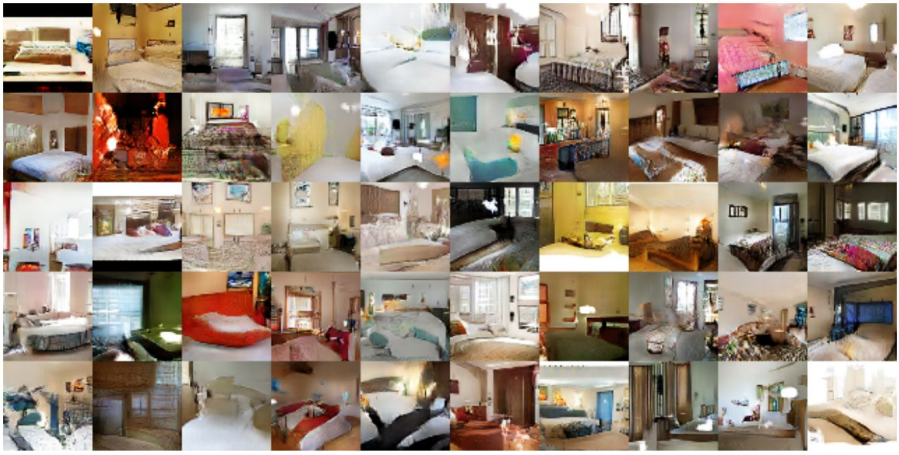
Generative Adversarial Nets – Convolutional Architectures

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.)

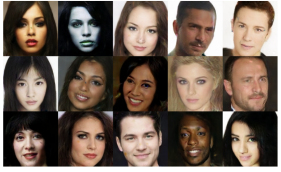


[Image source: Radford et al., 2016]

2017: Explosion of GANs

Better training and generation

LSGAN [Mao et al., 2017]



BEGAN [Berthelot et al., 2017]

Source → target domain transfer

CycleGAN [Zhu et al., 2017]

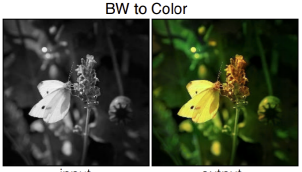
Text → image synthesis

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



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 Categorial 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

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