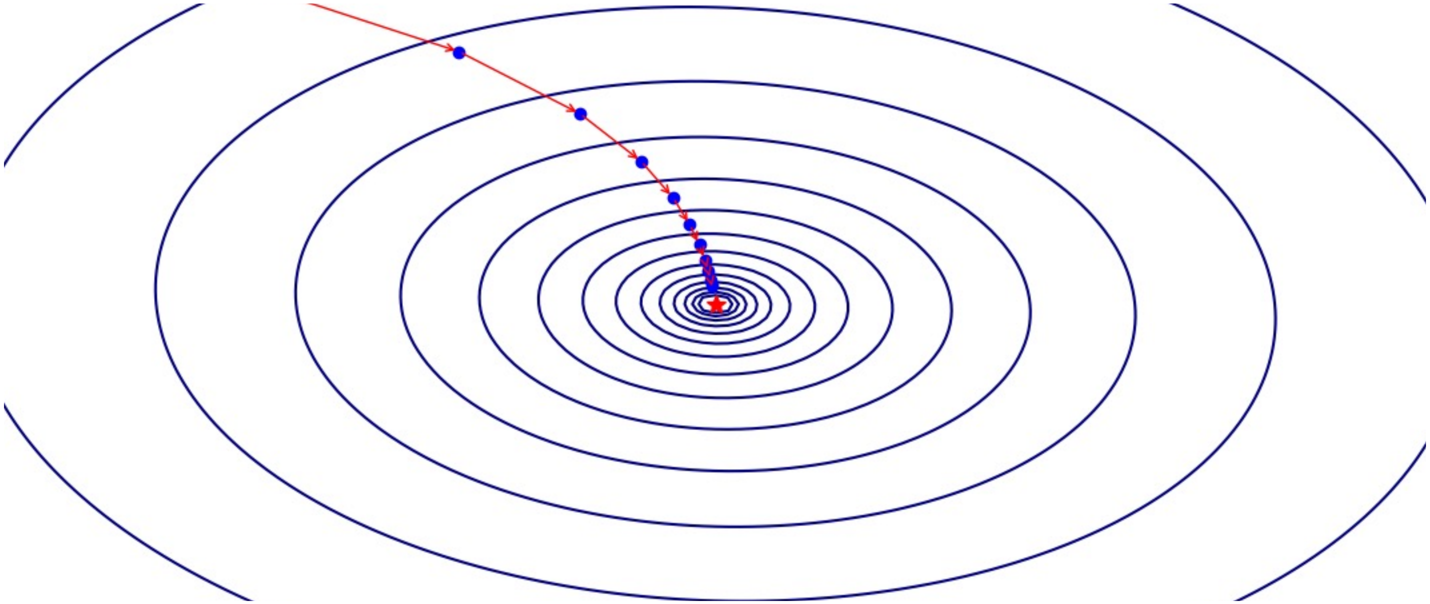


Advanced Topics in Deep Learning

Summer Semester 2024

6. Generative Models

27.05.2024

Prof. Dr. Vasileios Belagiannis

Chair of Multimedia Communications and Signal Processing

Friedrich-Alexander-Universität
Technische Fakultät

Course Topics

1. Interpretability.
2. Attention and Transformers.
3. Self-supervised Learning I.
4. Self-supervised Learning II.
5. Similarity Learning.
6. **Generative Models.**
7. Model Compression.
8. Transfer learning, domain adaptation, few-shot learning.
9. Uncertainty Estimation.
10. Geometric Deep Learning.
11. Recap and Q&A.

- The exam will be written.
- We will have an exam preparation test.

Acknowledgements

- Special thanks Arij Bouazizi, Julia Hornauer, Julian Wiederer, Adrian Holzbock and Youssef Dawoud for contributing to the lecture preparation.

Recap

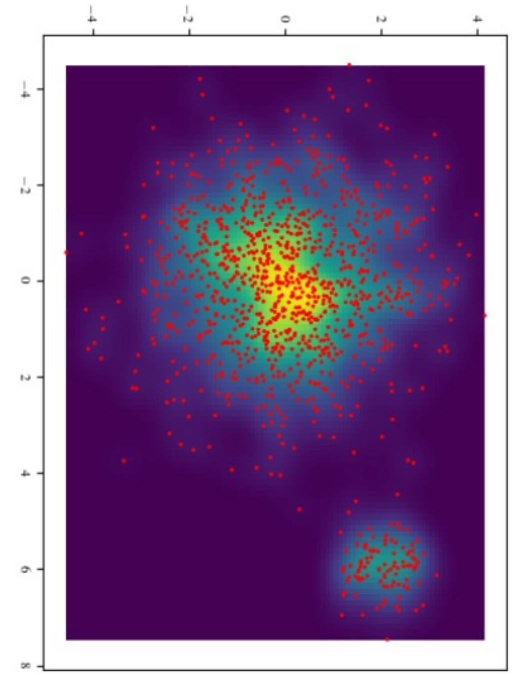
- Similarity learning.
- Deep metric learning.
- Triplet loss.
- Quadruplet loss.
- Hierarchical Triplet loss.

Today’s Agenda and Objectives

- Generative models.
- Generative Adversarial Networks.
- Auto-Encoders.
- Variational Auto-Encoders.

Generative Models

- A <u>generative</u> model is a statistical model of the joint distribution of observed (input) and target (output) variables.
  - $p\,(x, y)$.
- Unlike, a <u>discriminative</u> model is a statistical model of the conditional distribution of the target (output) variables, given the observed (input) variables.
  - $p\,(x|y)$.
- We can use the Bayes rule to transform transform a generative model to a conditional output.
  - $p\,(x|y) = \frac{p\,(x,y)}{p\,(y)}$.



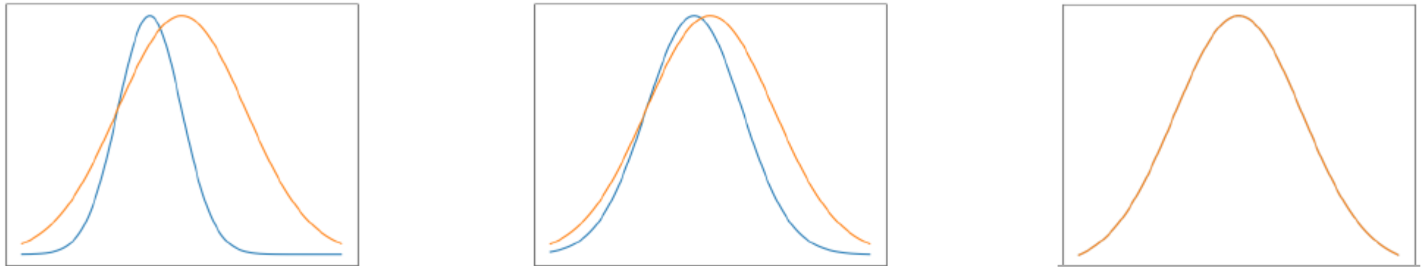
Generative Adversarial Networks

- Generative models are popular for <u>sampling</u> data, such as images. Generative Adversarial Networks (GANs) are popular for image generation*.
- GANs align the data and model distributions. The main idea is the following:
  - Draw samples (x) from the data distribution p_{data}, i.e. real images.
  - Draw samples from the model distribution (p_g) based on the set of latent variables (x) drawn from a prior distribution (p_z), i.e. generated (synthetic) images.
  - Detect samples coming from the data distribution (real images) with the discriminator network.

*Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

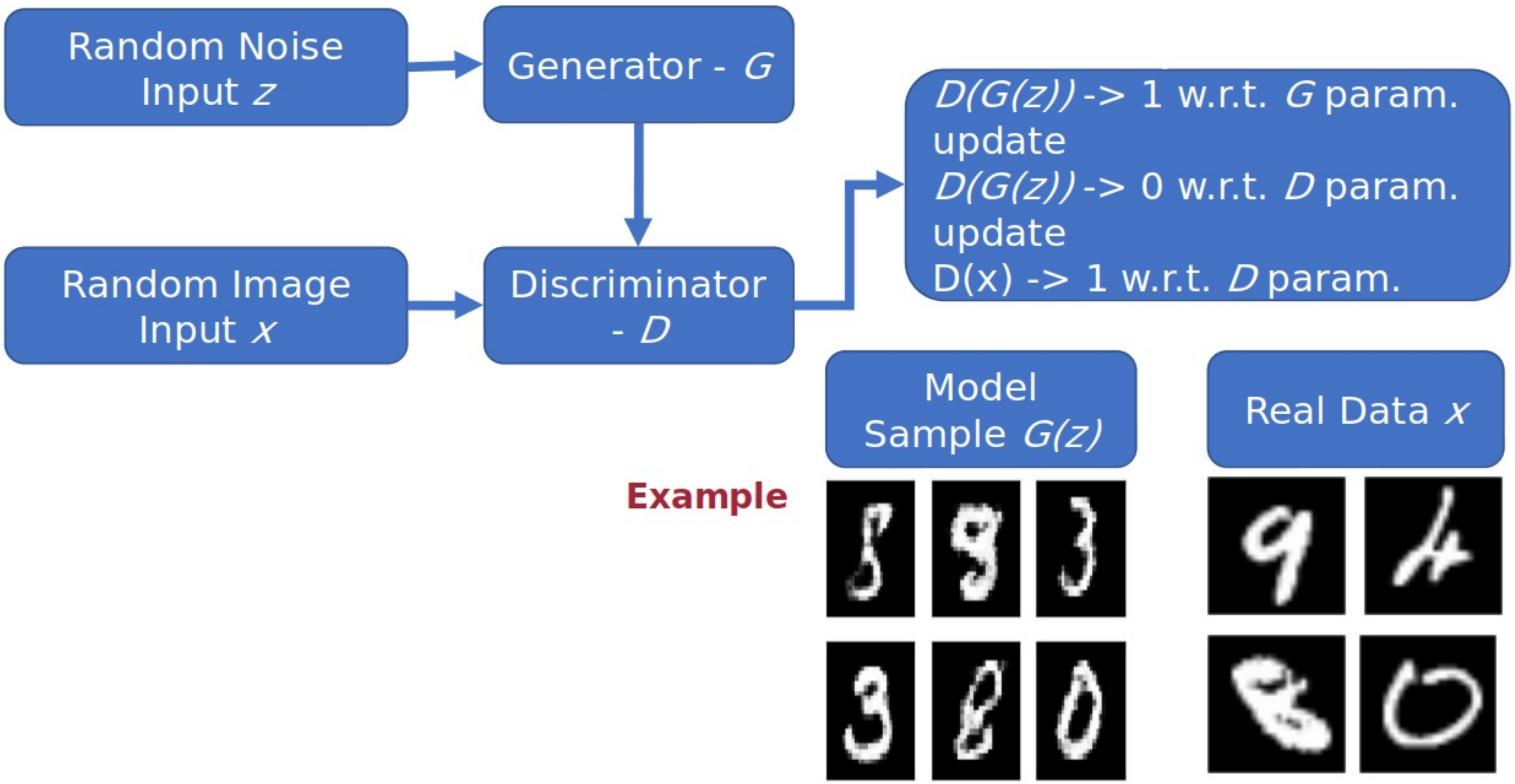
Generative Adversarial Networks (Cont.)

- The learning algorithm aims to train both the discriminator ($\mathcal{D}$)and generator ($\mathcal{G}$) at the same time. The idea of having two networks that compete with each other is called <u>adversarial training</u>.
  - The model $\mathcal{G}$ generates images from input noise.
  - The model $\mathcal{G}$ classifies images into real or fake/generated.
  - The model $\mathcal{G}$ tries to fool model $\mathcal{D}$.
  - The model $\mathcal{D}$ acts as supervision on model $\mathcal{G}$. As a result, we do not need labels to the train the GAN.
  - At the end of training, we usually need to keep the generator $\mathcal{G}$.

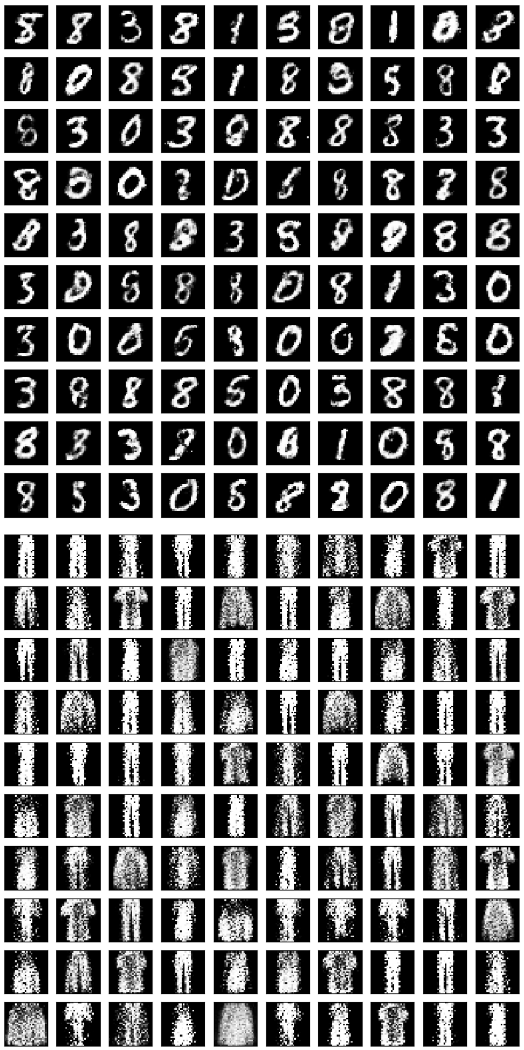


Adversarial Training

- We need a real and a synthetic (or two) samples for a training iteration.



Adversarial Training (Cont.)



- We can summarize the training process of the GAN in K training iterations as:
  - Sample N noise samples z from prior $p_g(\mathrm{z})$.
  - Sample N samples z from data distribution.
  - Update the discriminator D (gradient-based).
  - Sample N noise samples z from prior $p_g(\mathrm{z})$.
  - Update the generator G (gradient-based).

Adversarial Training Objective

- The idea of GAN training is motivated by the minimax game. We aim to minimize the image reconstruction loss in the generator, while maximizing the performance of the discriminator to detect real and fake images.
- The objective can be summarized as:
  - $min_G max_D \ \mathbb{E}_{x\sim p_{data}(x)}[Log\ (D(x))] + \mathbb{E}_{z\sim p_z(z)}[Log\left(1 - D\left(G(z)\right)\right)]$
  - $p_{data}(x)$: Data distribution over image x.
  - $\mathrm{D}(x)$: Discriminator.
  - $p_z(z)$: Data distribution over random noise z.
  - $\mathrm{G}(z)$: Generator.

Adversarial Training Objective (Cont.)

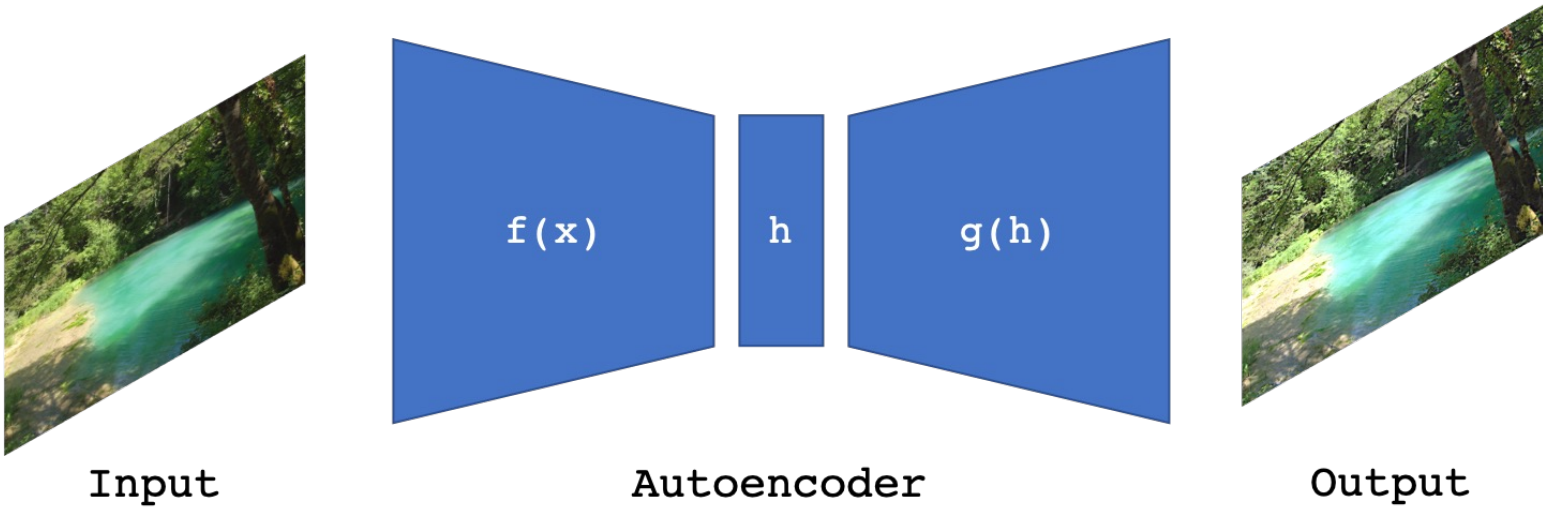
- The original GAN-formulation has some advantages and disadvantages compared to other generative models. In detail:
  - We do not need a sampling method. Using the generator, we can infinitely sample images.
  - We do not need to model the inferences. It's a feed-forward step.
  - We can approximate a wide range of functions. Generating videos, text, audio and other type of data is possible.
  - We do not need labels for training a GAN. By relying only on the real / fake objective, we can learn to generate images.

Autoencoder

- An autoencoder is a neural network that is trained to reconstruct its input. It is composed of the encoder, latent code and the decoder. The encoder $f(\cdot)$ transforms the input x to the latent code h, given by $\mathrm{h} = f(x)$. The decoder g$(\cdot)$ reconstructs the input x from the latent code h, given by $\hat{x} = g(h)$.
- In general, the autoencoder tries to copy the input. Since the latent code has usually smaller dimensions than the input, it can capture useful information in the latent code to reconstruct the input data.

Autoencoder Illustration

- An autoencoder has three major components.
- It can be trained as following:
  - Input: x
  - Output: $\hat{x}$
  - Goal: $x \approx \hat{x}$
  - Loss: $\mathcal{L} = \|x - \hat{x}\|^2$ where $\hat{x} = g(f(x))$.



Autoencoder Motivation

- The idea of building a neural network that learns a compressed representation of the input signal is relatively old.
  - Learning to copy the input*, known as recirculation, was a step towards autoencoders.
  - The motivation for the development of the autoencoder was to reduce the data dimensions**.
- Autoencoder learns in an unsupervised manner.
- Unlike PCA, autoencoder is a non-linear approach towards dimensionality reduction.

*Geoffrey E Hinton and James L McClelland. Learning representations by recirculation. In Neural information processing systems, pages 358–366, 1988.
**Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. science, 313(5786):504–507, 2006.

Under-Complete Autoencoder

- When the latent code has <u>smaller</u> dimensions than the input data of an autoencoder, then the autoencoder is called under-complete.
  - There is the possibility to build the opposite constellation, i.e. an over-complete autoencoder.
- Learning a linear decoder is similar to spanning the subspace, similar to PCA. However, a non-linear decoder and encoder with the right capacity can be much more powerful then PCA.
- The problem with the autoencoder is <u>overfitting</u>. Increasing the capacity can result in memorizing the data and not actually learning a useful latent code.
- To avoid over-fitting and constrain the latent code to learn useful information, different autoencoder models, training protocols and in general <u>regularization</u> approaches have been proposed.

Regularized Autoencoders

- The idea of a regularized encoder is to impose constraints during learning for the extraction of useful information from the input to the latent code.
- The constraints can be defined as:
  - Weight decay regularization.
  - Noise on the input to become more robust → Denoising autoencoder.
  - Small gradients during parameter update → Contractive autoencoder.
  - Sparsity of the latent code → Sparse autoencoder.
- The autoencoder can be formed with a MLP or Convolutional neural network.

Regularized Autoencoders: Weight Decay

- This is the standard L2 regularization which we already discssed in the past. It is given by:
  - $\mathcal{L}_{RAE} = \mathcal{L} + \beta||\boldsymbol{w}||^2$.
  - β controls the influence of the regularizer. This is the easiest way to regularize an autoencoder.
  - $\mathcal{L}$ is the standard mean-squared-error.

Denoising Autoencoders

- An autoencoder learns eventually the identity transformation, which can easily lead to overfitting. Another approach to avoid over-fitting are denoising autoencoders.
- Motivation: Humans can recognize an image even if it's corrupted by noise. The same functionality should be possible for an autoencoder as well.
- Input: It is partially corrupted by noise. The noise comes from some prior distribution. This is a stochastic mapping of the clean signal x to a noisy signal $\hat{x}$, represented by:
  - $\hat{x} \sim q_D(\hat{x}|x)$.
  - The stochastic mapping $q_D(\cdot)$ can combine multiple types of prior distributions. In the simple case, it can set a number of elements to zero.
- Output: Given the noisy signal $\hat{x}$ as input, the output will be the clean signal x.

Denoising Autoencoders (Cont.)

- The loss function for n training samples is defined as:
  - $\mathcal{L}_{DAE} = \frac{1}{n}\sum_{i=1}^{n}(x^i - g(f(\hat{x}^i)^2.$
  - By minimizing the loss, we force the encoder to extract features that contain useful information to reconstruct the clean signal.
- The original model* was a shallow deep neural network. It used just a single hidden layer.
- The denoising autoencoder can be interpreted as manifold learning where the low-dimensional manifold is the latent code.
- *How can we evaluate for the quality of the latent code?*

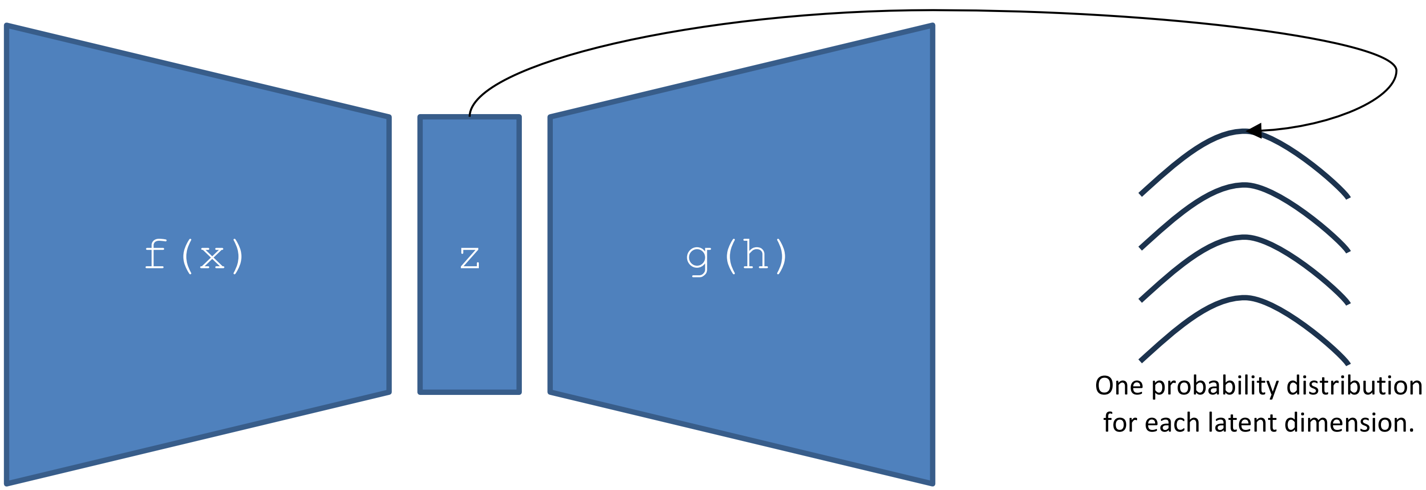
*Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th international conference on Machine learning, pages 1096–1103. ACM, 2008.

Stacked Denoising Autoencoders

- Stacked denoising autoencoders extend the original model to multiple hidden layers. This model is closer to deep learning.
- Learning is performed in an incremental way:
  - Train the autoencoder with the single hidden layer. Corrupt the input with noise.
  - Add a second hidden layer.
  - Train the second hidden layer (only) by corrupting the first hidden layer's output (only). This means that we pass a clean signal from the input and corrupt it after passing it through the first hidden layer.
  - Add a third hidden layer.
  - Train the third hidden layer (only) by corrupting the second hidden layer's output (only).
  - …

Variational Autoencoder

- A variational autoencoder (VAE) is a generative model to learn a latent (low-dimensional) representation which is described in a probabilistic way.
  - The standard auto-encoder learns a latent representation that is fixed for an input.
  - The variational autoencoder learn learns a probability distribution for each value of the latent representation.
- Similarly to an AE, the VAE also consists of two parts, the encoder and decoder neural network.



VAE: Marginal likelihood

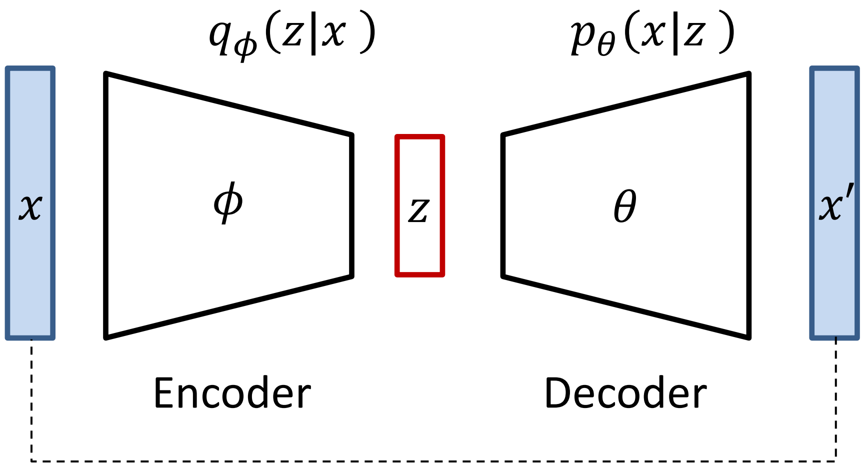
- Instead of mapping the input into a fixed vector, we aim to map it into the probability distribution p_θ, parameterized by θ.
- Our goal is to maximize the likelihood of the data x by the probability distribution $p_\theta(x) = p(x|\theta)$.
- Next to the input x, there is also the latent variable to learn z.
- To find $p_\theta(x)$, we can marginalize over z to obtain:
  - $p_\theta(x) = \int_z p_\theta(x, z)\, dz = \int_z p_\theta(x|z) p_\theta(z)\, dz.$
  - $p_\theta(x, z)$ is the joint distribution of the inputs x and latent representation z.
  - $p_\theta(x|z)$ is the likelihood.
  - $p_\theta(z)$ is a prior over the latent representation.

VAE: Marginal likelihood (Cont.)

- The integral of the marginal likelihood $p_\theta(x)$ is intractable.
  - We cannot evaluate or differentiate $p_\theta(x)$.
- The true posterior distribution $p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)}$ is also intractable.
  - We cannot thus use the EM algorithm or another related one to compute it.
- To approximate the posterior distribution $p_\theta(z|x)$, we introduce the following function:
  - $q_\phi(z|x) \approx p_\theta(z|x)$.
  - q is parametrized by ϕ.
- In the context of auto-encoders, we seek for an encoder to compute the approximated posterior distribution $q_\phi(z|x)$. Also, we seek for a decoder to compute the conditional likelihood distribution $p_\theta(x|z)$.

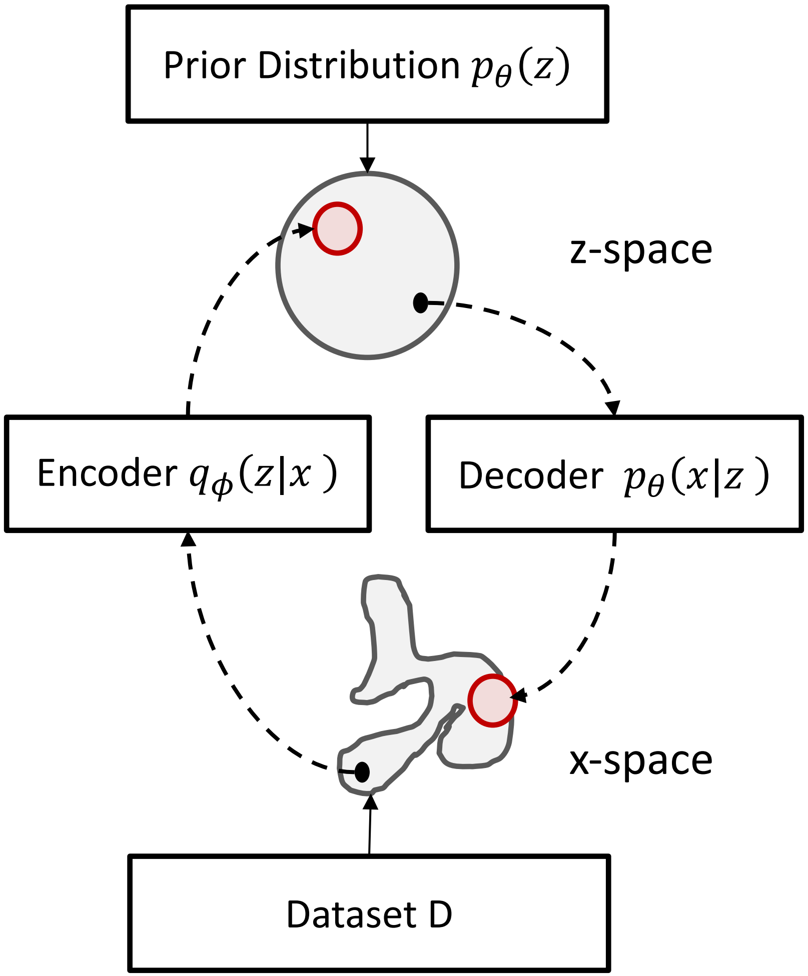
VAE: Encoder - Decoder

- The VAE learns stochastic mappings between the observed x-space and the latent space z-space.
- The empirical distribution $q_D(x)$ of the x-space is complex.
- The learned distribution of the latent z-space can be relatively simple.



VAE: Encoder – Decoder (Cont.)

- The model learns the <u>joint distribution</u> $p_\theta(x,z)$that is often factorized as $p_\theta(x,z) = p_\theta(z)p_\theta(x|z)$, with a prior distribution over the latent space $p_\theta(z)$, and the stochastic decoder $p_\theta(x|z)$.
- The stochastic encoder $q_\phi(z|x)$, also called <u>inference</u> model, approximates the true but intractable posterior $p_\theta(z|x)$ of the model.



Variational Bound

- Given a training set of N samples, the marginal likelihood $p_\theta(x)$ can be written as:
  - $log p_\theta(x^1, \dots, x^N) = \sum_{i=1}^{N} log p_\theta(x^i)$.
- The above expression can be written as:
  - $log\, p_\theta(x) = \mathbb{E}_{q_\phi(z|x)}[log\, p_\theta(x)] = \mathbb{E}_{q_\phi(z|x)}\left[log\left[\frac{p_\theta(x,z)}{p_\theta(z|x)}\right]\right] = \mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{p_\theta(x,z)q_\phi(z|x)}{q_\phi(z|x)p_\theta(z|x)}\right]\right] = \mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{p_\theta(x,z)}{q_\phi(z|x)}\right]\right] + \mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{q_\phi(z|x)}{p_\theta(z|x)}\right]\right]$.
  - $\mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{p_\theta(x,z)}{q_\phi(z|x)}\right]\right]$, the first term is called the variational lower bound.
  - $\mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{q_\phi(z|x)}{p_\theta(z|x)}\right]\right]$, the second term is the KL divergence which non-negative.

Variational Bound (Cont.)

- We can re-write the marginal likelihood $p_\theta(x)$ for the sample x^i as:
  - $log p_\theta\left(x^i\right) = D_{KL}\left(q_\phi\left(z|x^i\right)||p_\theta\left(z|x^i\right)\right) + \mathcal{L}(\theta, \varphi; x^i).$
  - $D_{KL}\left(q_\phi\left(z|x^i\right)||p_\theta\left(z|x^i\right)\right)$ is the KL divergence of the approximated posterior distribution from the true posterior distribution.
  - $\mathcal{L}\left(\theta, \varphi; x^i\right)$ is the variational lower bound on the marginal likelihood of the sample x^i. It is also called evidence lower bound (ELBO).
  - Due to the non-negativity of the KL divergence, the ELBO is a lower bound on the log-likelihood of the data.
- The marginal likelihood of the sample x^i can be then written as:
  - $log p_\theta\left(x^i\right) \geq \mathcal{L}(\theta, \varphi; x^i).$
- The variational lower bound can be reformulated as:
  - $\mathcal{L}(\theta, \varphi; x) = \mathbb{E}_{q_\phi(z|x)}\left[log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right] = \left[-log q_\phi(z|x) + log p_\theta(x, z)\right].$
- For the sample x^i, it can be also written as:
  - $\mathcal{L}\left(\theta, \varphi; x^i\right) = -D_{KL}\left(q_\phi\left(z|x^i\right)||p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x^i)}\left[log p_\theta\left(x^i|z\right)\right]$

Variational Bound (Cont.)

Not for sharing (LMS, Friedrich-Alexander-Universität Erlangen-Nürnberg)

- We want to use the variational lower bound on the marginal likelihood of each sample x^i as the loss function to train the VAE.
  - $\mathcal{L}(\theta, \varphi; x^i) = -D_{KL}\left(q_\phi(z|x^i)||p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x^i)}\left[logp_\theta(x^i|z)\right].$
- $logp_\theta(x^i|z)$ can be implemented as the reconstruction loss, given by:
  - $logp_\theta(x^i|z) = -\frac{1}{2}\|x - x'\|_2^2.$
  - x' is the output of the decoder.
  - The distribution x conditioned on z is modelled to be Gaussian centred on the decoder output.
- $q_\phi(z|x^i)$ and $p_\theta(z)$ are also chosen to be Gaussians.

VAE Computational Flow

- The inference model corresponds to the encoder and the generative model to the decoder.

Inference Model
$q_\phi(z|x\ )$

Sample
z

Generative Model
$p_\theta(x, z)$

$$\mathcal{L}(\theta, \varphi; x^i) = -D_{KL}\left(q_\phi(z|x^i)||p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x^i)}[logp_\theta(x^i|z)].$$

ELBO Optimisation

- We want optimise w.r.t the encoder, namely the ϕ variational parameters, and the decoder, namely the θ generative parameters using gradient descent.
- Given the training set dataset, the ELBO objective is the sum (or average) of individual-datapoint:
  - $\mathcal{L}_{\theta,\phi}(\mathcal{D}) = \sum_{x \in \mathcal{D}} \mathcal{L}_{\theta,\phi}(x)$
- The gradients of the ELBO w.r.t. θ are simple to obtain.
  - $\nabla_\theta \mathcal{L}_{\theta,\phi}(x) = \nabla_\theta \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x,z) - \log q_\phi(z|x)\right] = \mathbb{E}_{q_\phi(z|x)}\left[\nabla_\theta (\log p_\theta(x,z) - \log q_\phi(z|x))\right] \approx \nabla_\theta (\log p_\theta(x,z) - \log q_\phi(z|x)) = \nabla_\theta (\log p_\theta(x,z))$.
- The gradients w.r.t. to ϕ are difficult to obtain, since the ELBO's expectation is taken w.r.t. $q_\phi(z|x)$, which is a function of ϕ.
  - $\nabla_\phi \mathcal{L}_{\theta,\phi}(x) = \nabla_\phi \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x,z) - \log q_\phi(z|x)\right] \neq \mathbb{E}_{q_\phi(z|x)}\left[\nabla_\phi (\log p_\theta(x,z) - \log q_\phi(z|x))\right]$.
  - We <u>cannot</u> put ∇_ϕ inside the expectation. ϕ is on the probability itself.
  - We have an <u>undifferentiable</u> expectation.

Differentiable & Undifferentiable Expectation

- This is a <u>general</u> example that also applies to VAE.
- Consider the following expectation:
  - $\mathbb{E}_{p(z)}[f_\theta(z)]$.
  - $p(\cdot)$ is a density function.
  - $f_\theta(\cdot)$ is differentiable.
- We can easily compute the gradient ∇_θ :
  - $\nabla_\theta \mathbb{E}_{p(z)}[f_\theta(z)] = \nabla_\theta \left[\int_z p(z)\, f_\theta(z)\, dz\right] = \int_z p(z)\, [\nabla_\theta f_\theta(z)] dz = \mathbb{E}_{p(z)}[\nabla_\theta f_\theta(z)]$.
- We conclude that the gradient of the expectation $\nabla_\theta \mathbb{E}_{p(z)}[f_\theta(z)]$ is equal to the expectation of the gradient $\mathbb{E}_{p(z)}[\nabla_\theta f_\theta(z)]$.

Differentiable & Undifferentiable Expectation (Cont.)

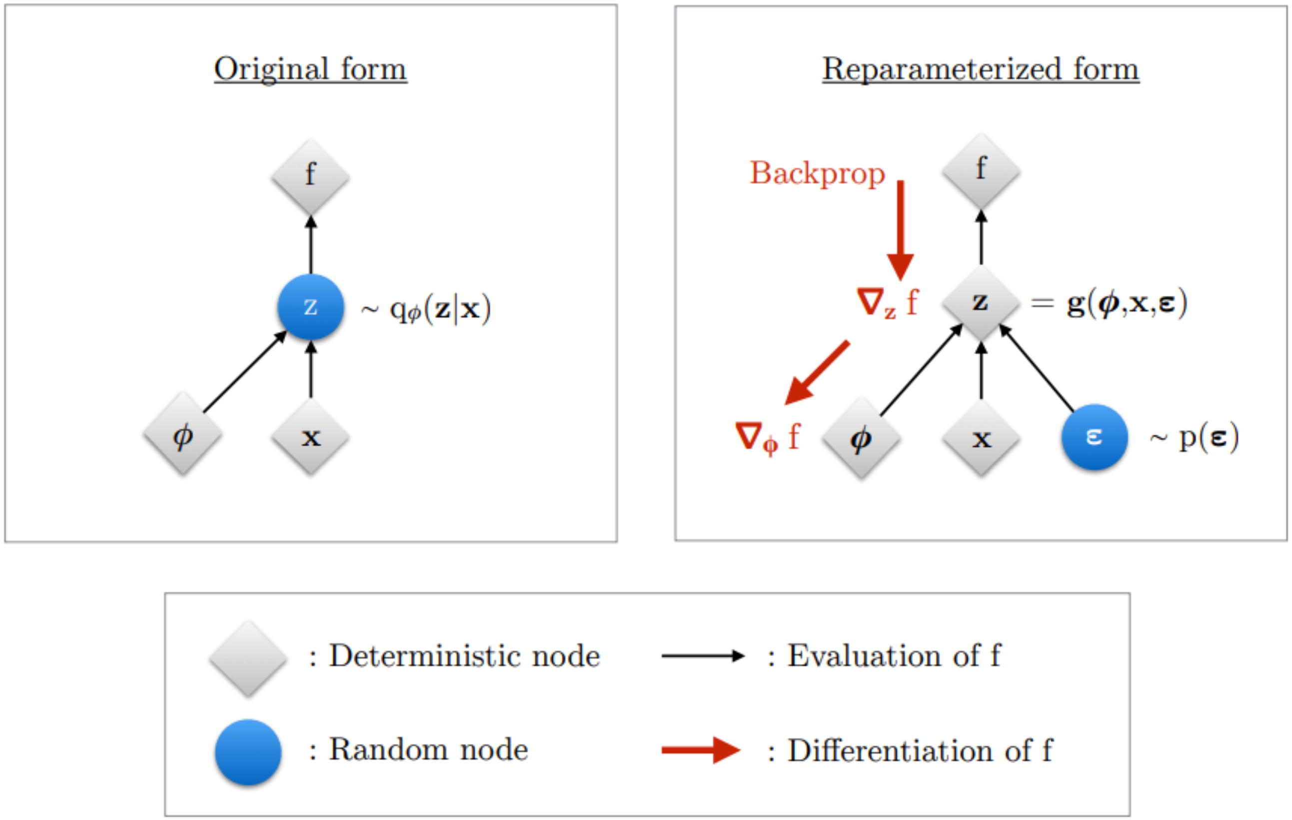
- Now consider the following expectation:
  - $\mathbb{E}_{p_\theta(z)}[f_\theta(z)]$.
  - $p_\theta(\cdot)$ is a density function that is also parametrized by θ.
- We can try to compute the gradient:
  - $\nabla_\theta \mathbb{E}_{p_\theta(z)}[f_\theta(z)] = \nabla_\theta \left[\int_z p_\theta(z)\, f_\theta(z)\, dz\right] = \int_z \nabla_\theta\, [p_\theta(z)\, f_\theta(z)\, dz] = \int_z \nabla_\theta\, p_\theta(z) f_\theta(z) dz + \int_z p_\theta(z)\, \nabla_\theta\, f_\theta(z)\, dz = \int_z \nabla_\theta\, p_\theta(z) f_\theta(z) dz + \mathbb{E}_{p_\theta(z)}[\nabla_\theta\, f_\theta(z)]$.
  - The term $\int_z \nabla_\theta\, p_\theta(z) f_\theta(z) dz$ is problematic because we cannot take the gradient.
  - The <u>reparameterization trick</u> helps to obtain gradients.

Reparameterization Trick

- The ELBO can be easily differentiated with respect to both ϕ and θ by changing the variables, also called the reparameterization trick.
- We express the random variable $z \sim q_\phi(z|x)$ as a a differentiable transformation of another random variable ε, given z and ϕ.
  - $z = g\ (\varepsilon, \phi, x)$.
  - $\varepsilon \sim p(\varepsilon)$ is Gaussian distribution with zero mean and variance one.
  - ε is independent of x and ϕ.
- Consider as $f(\cdot)$ the overall objective, then the expectation can be written as:
  - $\mathbb{E}_{q_\phi(z|x)}[f(z)] = \mathbb{E}_{p(\varepsilon)}[f(g\ (\varepsilon, \phi, x))]$
- We can take the gradient as:
  - $\nabla_\varphi \mathbb{E}_{q_\phi(z|x)}[f(z)] = \nabla_\varphi \mathbb{E}_{p(\varepsilon)}[f(g\ (\varepsilon, \phi, x))] = \mathbb{E}_{p(\varepsilon)}\left[\nabla_\varphi f(g\ (\varepsilon, \phi, x))\right] = \nabla_\varphi f(g\ (\varepsilon, \phi, x))$

Reparameterization Trick (Cont.)

- In the original form, we cannot backpropagate through the random variable z. We reparametrize z as a deterministic and differentiable function of ε, ϕ, x.



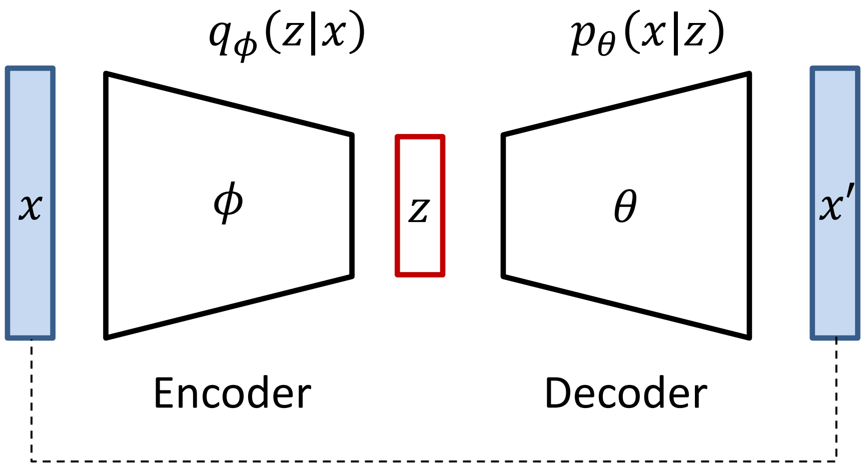
Source: https://arxiv.org/pdf/1906.02691.pdf

Reparameterization Trick (Cont.)

- After reparameterization, we can replace $q_\phi(z|x)$ with $p(\varepsilon)$ and re-write ELBO as:
  - $\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x)}\left[log\, p_\theta(x,z) \;-log\, q_\phi(z|x)\right] = \mathbb{E}_{p(\varepsilon)}\left[log\, p_\theta(x,z) \;-log\, q_\phi(z|x)\right]$.
  - With $z = g\,(\varepsilon, \phi, x)$.
- Given that $g(\cdot)$ is <u>differentiable</u>, we can use a Monte Carlo estimator to estimate the gradients.

Implementation

- The encoder is corresponding to the approximated posterior distribution $q_\phi(z|x)$.
- The encoder is corresponding to the conditional likelihood distribution $p_\theta(x|z)$.
- Recall the original loss function:
  - $\mathcal{L}(\theta, \varphi; x^i) = -D_{KL}\left(q_\phi(z|x^i)||p_\theta(z)\right) + \mathbb{E}_{q_\phi(z|x^i)}\left[log p_\theta(x^i|z)\right].$
- $p_\theta(z)$: is fixed and modelled as a Gaussian. We make the same assumption for the output of the encoder $q_\phi(z|x)$.
- Using the reparameterization trick, we can sample z.
- Finally, we condition z to generate x.



Implementation: Training

- Provide x to the encoder to obtain μ_x and σ_x.
- Sample Gaussian noise $\varepsilon \sim p(\varepsilon)$ with zero mean and variance one.
- Reparametrize as $z = \mu_x + \varepsilon\sigma_x$.
- Provide z to the decoder to obtain the generated sample x'.
- Compute loss functions:
  - Reconstruction loss $\frac{1}{2}\|x - x'\|_2^2$.
  - Variational loss $-D_{KL}(\mathcal{N}(\mu_x, \sigma_x)||\mathcal{N}(0, I))$.
  - Total loss $\frac{1}{2}\|x - x'\|_2^2 - D_{KL}\,(\mathcal{N}(\mu_x, \sigma_x)||\mathcal{N}(0, I))$.
- Compute gradients, backpropagate and update the parameters of the encoder and decoder.

Implementation: Inference

- The trained VAE can used to sample x.
- Given the trained model, one can needs only the learned μ_x and σ_x and the decoder to generate samples.

Source: https://arxiv.org/pdf/1312.6114.pdf

VAE Advantages and Disadvantages

- The main benefit of a variational autoencoder is the learning *smooth* latent state representations of the input data.
- VAEs can be used, not only for data generation but also for dimensionality reduction.
- The latent space visualization can be beneficial for other tasks.
- Blurry and unrealistic outputs: GAN produces sharper and more realistic images.

Study Material

- Chapter 2, Kingma, Diederik P., and Max Welling. "An introduction to variational autoencoders." *Foundations and Trends® in Machine Learning* 12.4 (2019): 307-392.
- Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." *arXiv preprint arXiv:1312.6114* (2013).
- Blog: https://gregorygundersen.com/blog/2018/04/29/reparameterization/

Next Lecture

Non-graded Test