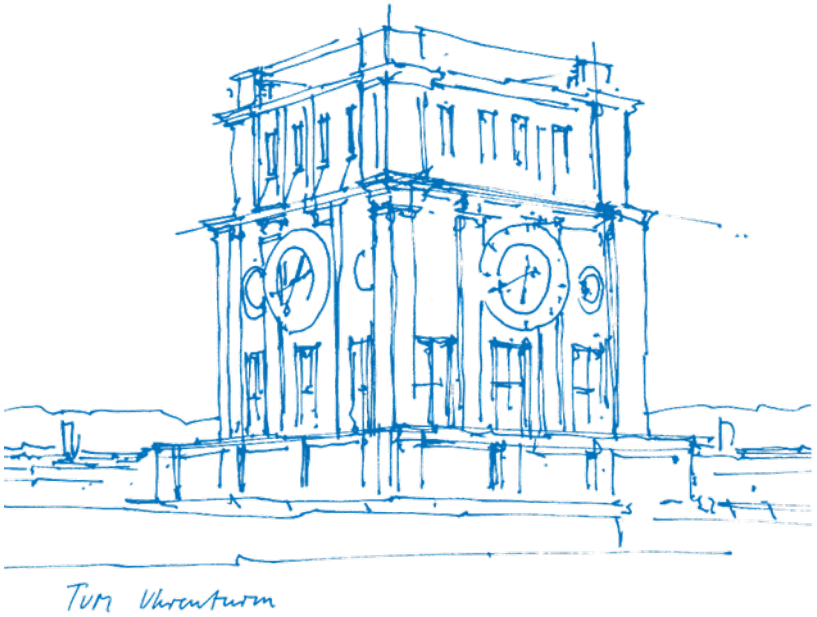
Variational Autoencoders

Robin Mittas
Technische Universität München
Computation, Information and Technology (MA)
Applied Mathematics
Munich, 25th of January 2023



Agenda

1. Motivation
2. Small Recap
3. Variational Bayesian
4. The Variational Bound
5. Gaussian VAE
6. β-VAE
7. σ-VAE
8. Own Coding Work
9. Limitations and Challenges
10. State of the Art
11. Appendix

Motivation

VAEs are used for:

- Anomaly Detection.
- Generating new Data.
- Dimensionality Reduction: For huge Datasets we can train Autoencoder and use encoded Data for further Processes. Often used in Finance or Bio-Processing
- Useful for data representation tasks. Useful in computer vision applications such as image denoising, inpainting and super-resolution.
- Predicting future frames for a video sequence.
- Image Segmentation.
- Music, Speech Generation ...

Recap: Dimensionality Reduction

PCA: Principal Component Analysis

Suppose we have a Data Matrix $A \in \mathbb{R}^{m \times n}$, then the k-th Principal Component is

$$v_k = \underset{\|v\|_2=1 \& v \perp v_1,\ldots,v_{k-1}}{\operatorname{argmax}} \|Av\|_2.$$

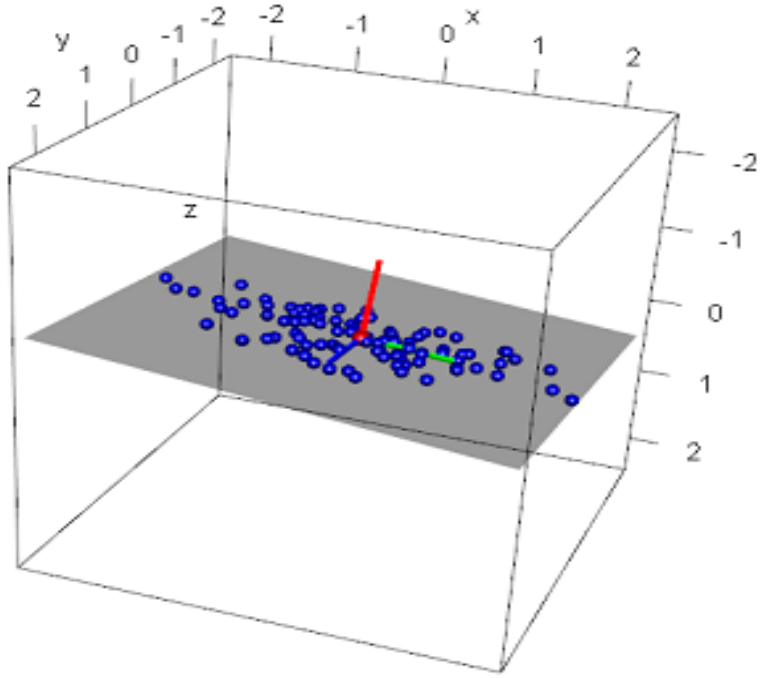


Figure: v_1, v_2, v_3, $span(v_1, v_2)$ [11]

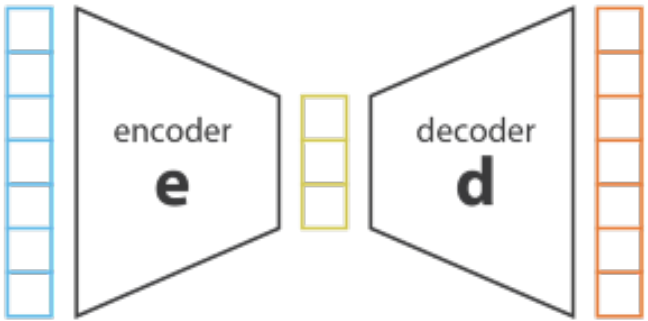
Autoencoder: Encoder-Decoder Structure

Encoder = Dimensionality Reduction with latent space dimension $n \in \mathbb{N}$: Returns latent vector $x \in \mathbb{R}^n$. This is done by e.g. Convolutional or Linear layers.

Decoder = Decode Encoded Data back into initial Space. This is done by Transposed Convolutions or Linear/ Upsampling Layers into higher dimensions.

Question: What happens if we plug a random sample from some distribution into the Decoder?

Motivation: Learn a latent distribution from which we can sample from e.g. to generate new data.



Variational Autoencoder

Encoder returns two vectors for latent dimension $n \in \mathbb{N}$: Mean $\mu \in \mathbb{R}^n$ and Variance $\sigma \in \mathbb{R}^n$, which define the parameters for a Normal distribution, e.g. the posterior $p(z|x)$.

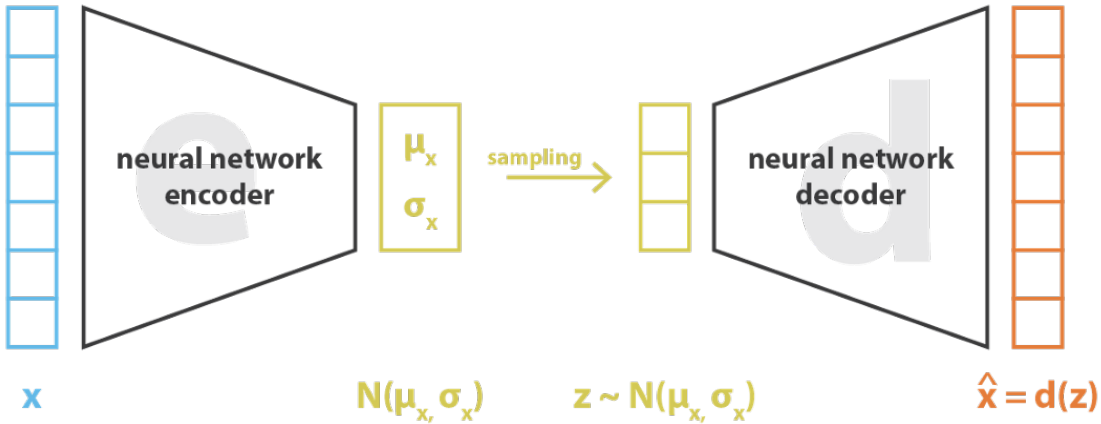


Figure: Variational Autoencoder Structure [13]

Variational Bayesian

Setting:

- Data generated by random process, involving an unobserved continuous random variable z (the latent variable). [6]
- A Value $z^{(i)}$ is generated from the prior distribution $p_{\theta^*}(z)$ and $x^{(i)}$ is generated from conditional distribution $p_{\theta^*}(x|z)$ (likelihood) where θ^* are the true/ unknown parameters. [6]
- Dataset $X = \{x_i\}_{i=1}^N$ of N i.i.d. samples of continuous or discrete variable x. Assume we have given a large dataset, e.g. we use batch-optimization to learn the distributions. [6]
- Intractability: The Integral of the marginal likelihood

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

  is intractable. This is equivalent to the intractability of the true posterior $p_\theta(z|x) = p_\theta(z)p_\theta(x|z)/p_\theta(x)$. This is very common in Neural Networks due to None-Linearities. [6]

Recognition Model

- Recognition model $q_\phi(z|x)$ as an approximation of the intractable true posterior $p_\theta(z|x)$. [6]
- Recognition model $q_\phi(z|x)$ as a **probabilistic encoder**, given a datapoint x it produces a distribution (e.g. a Gaussian) over the possible values of the latent random variable z (or: „code“) from which the datapoint x could have been generated.
- Similarly $p_\theta(x|z)$ as a **probabilistic decoder**, since given a code z it produces a distribution over the possible corresponding values of x.
- We refer to ϕ and θ as parameters from the probabilistic encoder and decoder respectively. [6]

The Variational Lower Bound

The ELBO (evidence lower bound) is defined as lower bound for the Log-marginal likelihood and satisfies following inequality (see Appendix for derivation)

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

In general we want to maximize the (log-) marginal likelihood thus we can also maximize this Lower Bound.

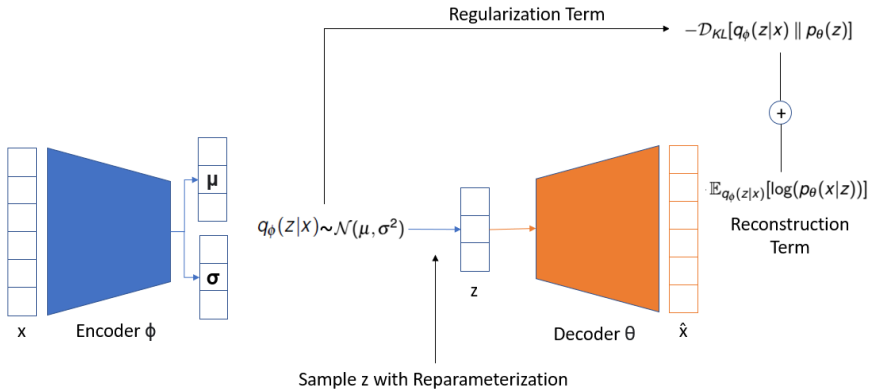


Figure: Structure of VAE with Loss Terms (own Figure)

Sample from Latent Space, Re-parametrization

- Encoder returns two vectors $\mu, \sigma \in \mathbb{R}^n$ with latent dimension $n \in \mathbb{N}$.
- μ, σ are the parameters for a Normal Distribution, e.g. mean and variance vectors.
- In order to be able to backpropagate through the network w.r.t. μ and σ, sample $\varepsilon \sim \mathcal{N}(0, I)$ where $0 \in \mathbb{R}^n$ and $I \in \mathbb{R}^{n \times n}$ the identity matrix. Then reparametrize z as

$$z = \mu + \sigma \odot \varepsilon,$$

where $\odot$ refers to the element-wise multiplication.

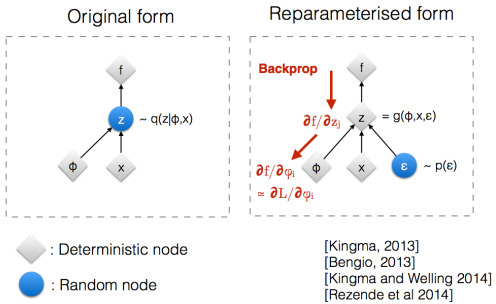


Figure: Illustration of the reparamtrization trick together with the gradient flow in order to be able to backpropagate through the Network [2]

Bayesian VAE

- Use a neural network to find the parameters of the probabilistic encoder and decoder for an n-dimensional latent space, e.g. we aim to find the approximation to the posterior $q_\phi(z|x)$.
- Assume the prior $p_\theta(z) = \mathcal{N}(z; 0, I)$ is a centered multivariate n-dimensional standard Gaussian.
- Let the Likelihood $p_\theta(x|z)$ be a multivariate Gaussian (in case of real-valued data) or Bernoulli (in case of binary data) whose distribution parameters are computed from z with a Neural Network including None-Linearities.
- Assume the true posterior takes on an approximate Gaussian form with an approximately diagonal covariance. Thus the latent dimensions are not correlated with each other. Now we can let the variational approximate posterior $q_\theta(z|x)$ be a multivariate Gaussian with a diagonal covariance structure. Let the encoder output two vectors $\mu, \sigma \in \mathbb{R}^n$ and by e.g. Linear Layers.

Then it holds for a single data point $x^{(i)}$ [6]

$$\log(q_\theta(z|x^{(i)})) = \log(\mathcal{N}(z; \mu^{(i)}, (\sigma^{(i)})^2)).$$

Bayesian VAE

Now we sample from the approximate posterior $z^{(i,l)} \sim q_\theta(z|x^{(i)})$ using

$$z^{(i,l)} = g_\phi(x^{(i)}, \varepsilon^{(l)}) = \mu^{(i)} + \sigma^{(i)} \odot \varepsilon^{(l)}$$

where $\varepsilon^{(l)} \sim \mathcal{N}(0, I)$. Then with the Appendix the KL Divergence term between the prior (multivariate standard gaussian) and the posterior (multivariate gaussian) can be computed and the Loss function for the datapoint $x^{(i)}$ is defined as

$$\mathcal{L}(\theta, \phi; x^{(i)}) = \underbrace{\frac{1}{2}\sum_{j=1}^{n}\left(1 + \log((\sigma_j^{(i)})^2) - (\mu^{(i)})^2 - (\sigma_j^{(i)})^2\right)}_{-\mathcal{D}_{KL}[p_\theta(z)\|q_\phi(z|x)]} + \underbrace{\frac{1}{L}\sum_{l=1}^{L}\log(p_\theta(x^{(i)}|z^{(i,l)}))}_{\mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x|z))]},$$

where L corresponds to the Number of Samples of $\varepsilon^{(l)} \sim p(\varepsilon) = \mathcal{N}(0, I)$ (in General we use $L = 1$) to calculate the Expectation via Monte Carlo estimates with g_ϕ the reparametrization function, f the decoding function and $n \in \mathbb{N}$ the latent dimension as

$$\mathbb{E}_{q_\phi(z|x^{(i)})}[f(z)] = \mathbb{E}_{p(\varepsilon)}[f(g_\phi(\varepsilon, x^{(i)}))] \simeq \frac{1}{L}\sum_{l=1}^{L} f(g_\phi(\varepsilon^{(l)}, x^{(i)})).$$

Bayesian VAE

Now with our posterior $q_\phi(z|x) = \mathcal{N}(\mu, \sigma^2)$ and the given reparametrization $z = \mu + \varepsilon \odot \sigma$ with $\varepsilon \sim \mathcal{N}(0, I)$ we can write the Expectation as

$$\mathbb{E}_{\mathcal{N}(z|\mu,\sigma^2)}[f(z)] = \mathbb{E}_{\mathcal{N}(\varepsilon|0,I)}[f(\mu + \varepsilon \odot \sigma)] \tag{1}$$

$$\simeq \frac{1}{L} \sum_{l=1}^{L} f(\mu + \varepsilon^{(l)} \odot \sigma) \tag{2}$$

$$= \frac{1}{L} \sum_{l=1}^{L} \log(p_\theta(x|z^{(l)})) \tag{3}$$

Note that maximizing the log-likelihood is equivalent to minimizing the negative log-likelihood.

Classic-Gaussian-VAE

- Assume that the prior $p_\theta(z) = \mathcal{N}(0, I)$ is an n-dimensional Multivariate Standard Gaussian, with $I \in \mathbb{R}^{n \times n}$ and $n \in \mathbb{N}$ the latent dimension.
- Assume that the likelihood $p_\theta(x|z) = \mathcal{N}(\hat{x}, I)$, where $\hat{x} \in \mathbb{R}^D$ is the output of the decoding function.
- We can rewrite equation (3) with $L = 1$ and one single datapoint $x \in \mathbb{R}^D$ and a Constant $C \in \mathbb{R}$, where each pixel x_j is conditionally independent of the others given z (but they become dependent if we marginalize out z), as

$$-\log(p_\theta(x|z)) = -\log(\mathcal{N}(\mu, I)) = -\prod_{i=1}^{D} \log(\mathcal{N}(\mu_i, 1)) \tag{4}$$

$$= \sum_{i=1}^{D} \frac{1}{2}\log(2\pi) - \log(\exp(-\frac{1}{2}(x_i - \hat{x}_i)^2)) \tag{5}$$

$$= \sum_{i=1}^{D} C + \frac{1}{2}(x_i - \hat{x}_i)^2 = DC + \frac{1}{2}\|x - \hat{x}\|_2^2 \tag{6}$$

$$= DC + \frac{1}{2}\mathrm{MSE}(\hat{x}, x) \tag{7}$$

Classic-Gaussian-VAE

MSE Loss as our Reconstruction term between x and $\hat{x}$, the output of our VAE!
If we had assume $p_\theta(x|z) = \text{Bernouli}(\lambda)$ we would end up with the BCE Loss. In Practice we use MSE and BCE Loss as Reconstruction Error Term. These terms are reduced by the sum over all Pixels (for image data).

$$\text{MSE}(\hat{x}, x)_n = (\hat{x}_n - x_n)^2$$
$$\text{BCE}(\hat{x}, x)_n = \hat{x}_n \log(x_n) + (1 - \hat{x}_n) \log(1 - x_n)$$

The complete objective Loss function with the Standard Normal Prior is

$$\begin{aligned}\mathcal{L} &= \text{Reconstruction}(\hat{x}, x) + \mathcal{D}_{KL}[q_\phi(z|x) \parallel p_\theta(z)] \\ &= \text{Reconstruction}(\hat{x}, x) - 0.5 \sum_{i=0}^{n} (1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2)\end{aligned}$$

Problem: We assumed Unit Variance for $p(x|z)$ and the loss might be dominated by either KLD Loss term or reconstruction loss term. In practice one chooses the losses that suits, BCE is also often used for none-Binary data.

β-VAE/ Disentangled VAE

Introduction of weight $\beta \in \mathbb{R}$ for KL-Divergence Term. The objective Loss function is

$$\mathcal{L}_\beta(\hat{x}, x) = \text{Reconstruction}(\hat{x}, x) + \beta \mathcal{D}_{KL}[q(z|x) \,\|\, p(z)].$$

The goal is for some $\beta >> 1$ to have a disentangled representation, such that each latent dimension learns its own features as we force the posterior to be closer to the prior. [8]

Problem: The weight β of the KLD Loss is a hyperparameter which needs to be tuned.

Calibrated Decoder: σ-VAE

- Assume $p_\theta(x|z) = \mathcal{N}(\mu(z), \sigma(z)^2)$, instead of Unit variance.
- This parameterization of the decoding distribution outputs one variance value for each pixel and channel.
- While powerful, it led to suboptimal performance and numerical instability. [14].
- Instead use a simpler parameterization, in which the covariance matrix is specified with a single shared parameter $\sigma \in \mathbb{R}$.
- This σ can either be trainable or batch dependent. For first case the Neural Network learns the variance of the decoder as another parameter. [14]
- As in equation (7) we end up with

$$\begin{aligned} -\log(p_\theta(x|z)) &= \frac{D}{2\sigma^2}\text{MSE}(\hat{x}, x) + D\log(\sigma 2\pi) \\ &= \frac{D}{2\sigma^2}\text{MSE}(\hat{x}, x) + D\log(\sigma) + C \end{aligned}$$

Calibrated Decoder: σ-VAE

- The proposed optimal σ is calculataed per batch (e.g. per training step). For a batch B of size $k \in \mathbb{N}$ with $B = \{x_1, ..., x_k\}$ [14]

$$\sigma^* = \log\left(\sqrt{\frac{1}{k}\sum_{i=1}^{k}(\text{MSE}(\hat{x}_i, x_i, \textit{reduction} = \textit{mean}))}\right).$$

- The Loss function for the specified σ^* is

$$\mathcal{L}(\hat{x}, x) = \frac{D}{2}\frac{\text{MSE}(\hat{x}, x)}{exp((\sigma^*)^2)} + D\sigma^* + \mathcal{D}_{KL}[q(z|x) \parallel p(z)],$$

  where D is the dimension of the input x.
- This effectively scales the original reconstruction loss term and we do not require the hyperparameter β to balance out the KLD and the reconstruction loss terms. [14]
- Parameter σ will not get too large due to the additional logarithmic Penalty in Loss Function. [14]

Own Coding Work I

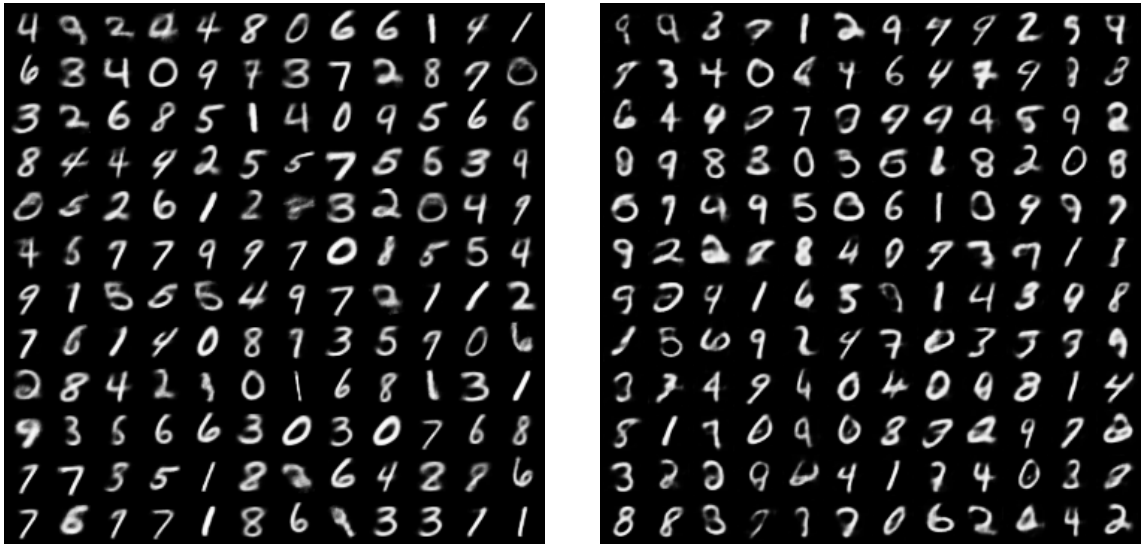


Figure: Samples from Linear VAE on the left and Convolutional VAE on the right with Latent Dimension 8 on MNIST Dataset. [9]

Own Coding Work II

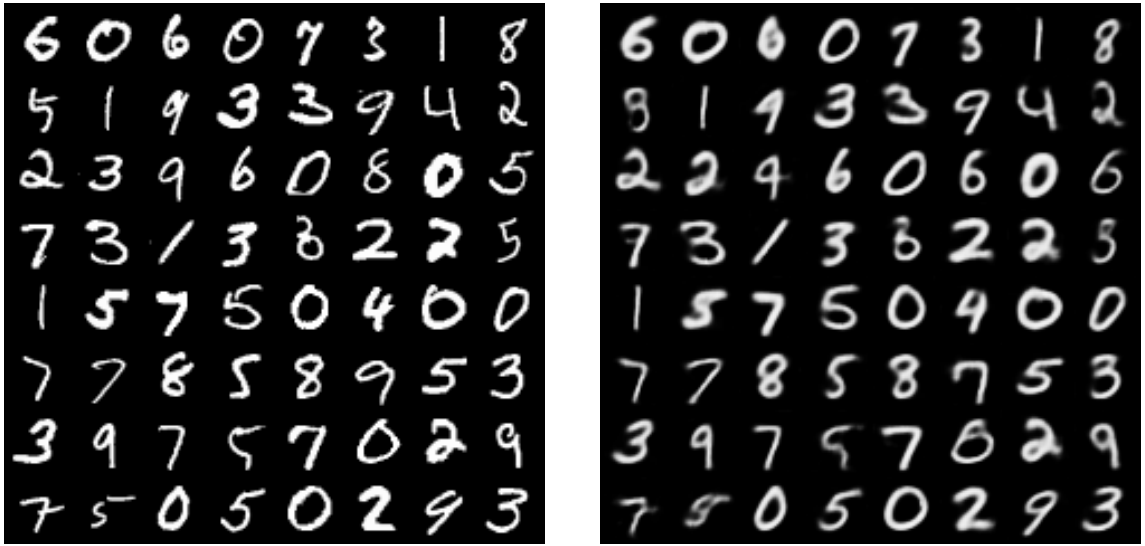
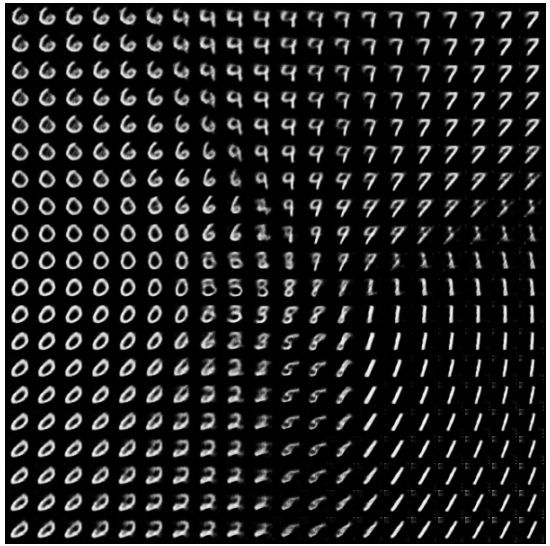


Figure: Reconstructions from Convolutional VAE with 8 latent dimensions. [9]

Own Coding Work III: Latent Space Representation



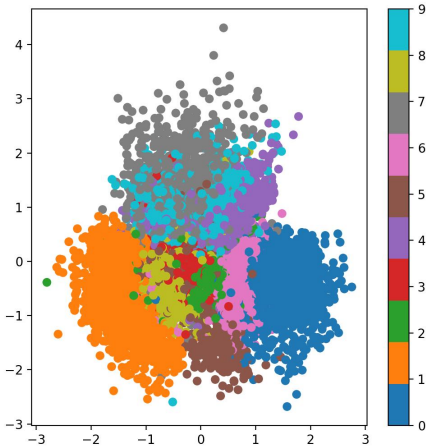


Figure: Left: Illustration of the Latent Space distribution for a σ-VAE with $2-$dimensional latent space. The image on the top right is a decoded Image of the vector $(-1,1)$, the top right displays the decoded Image of the latent vector $(1,1)$. Right: Class-dependent Latent Space Representation of MNIST Data. [9]

Own Coding Work IV

Figure: Samples from β-VAE with 10 latent dimensions. [9]

Own Coding Work V

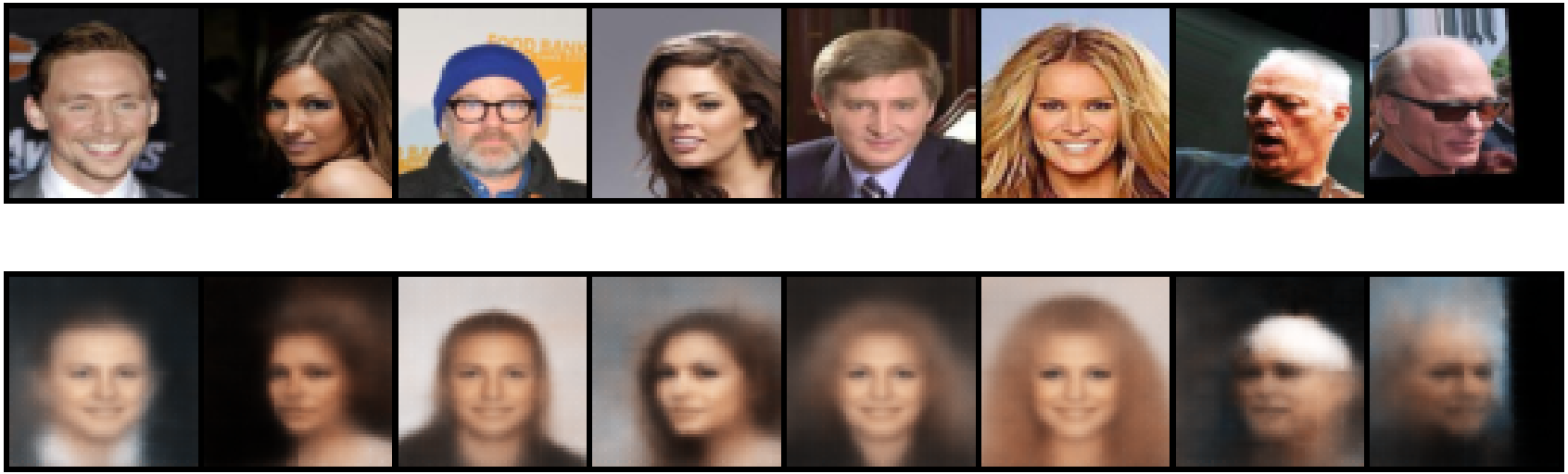


Figure: Reconstructions from β-VAE with 10 latent dimensions. [9]

Own Coding Work VI: β-VAE/ Disentangled VAE

Figure: Latent Traversals from β-VAE with 10 latent dimensions. Each image represents one decoded Vector $z \in \mathbb{R}^{10}$ where each entry is standard normal, $z_i \sim \mathcal{N}(0,1)\ \forall i \in \{1,...10\}$. Each row represents one latent dimension $i \in \{1,...10\}$ when adding small deltas to one latent dimension at a time. [9] Ideas from [5].

Own Coding Work VII

Figure: Samples from β-VAE with 128 latent dimensions. [9]

Own Coding Work VIII

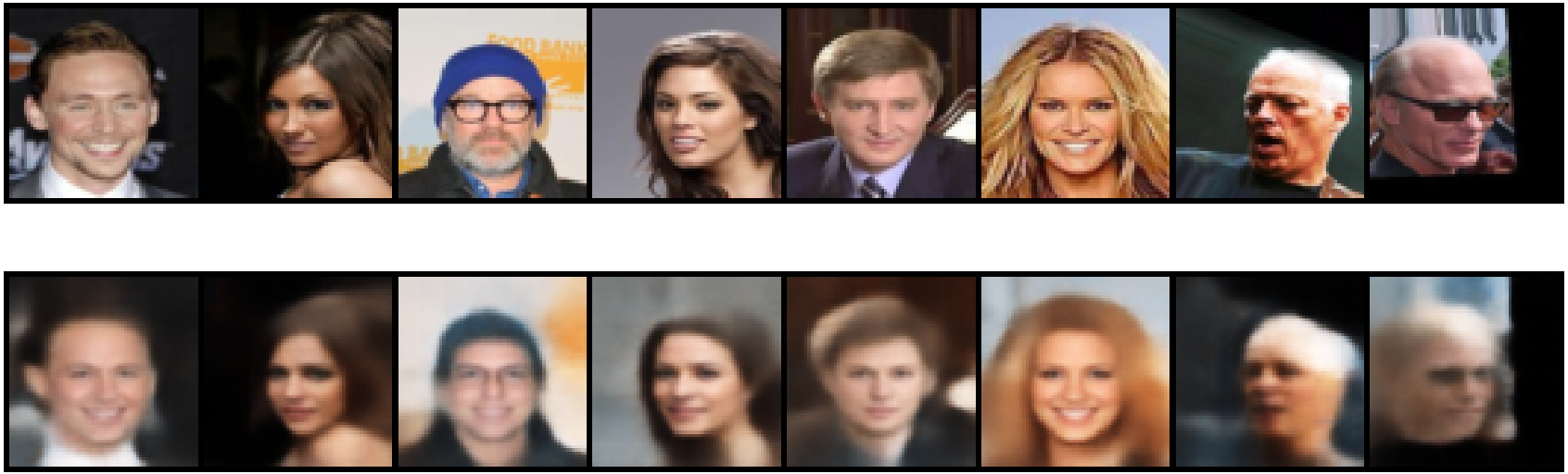


Figure: Reconstructions from β-VAE with 128 latent dimensions. [9]

Own Coding Work IX: β-VAE/ Disentangled VAE

Figure: Latent Traversals from β-VAE with 128 latent dimensions. [9]

Own Coding Work X: Question

What was the approximate latent dimension here? Also have a look at the next Slide!

Figure: Samples. [9]

Own Coding Work XI: Question

What was the approximate latent dimension here?

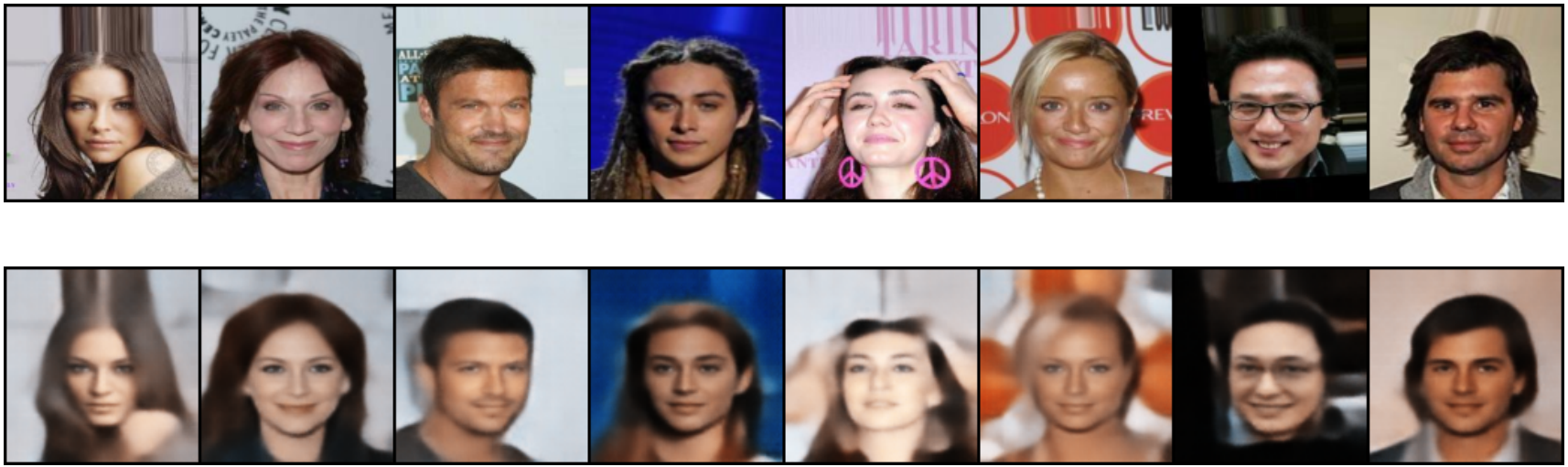


Figure: Reconstructions. [9]

Limitations and Challenges

- The dimension of the Latent Space is a hyperparameter: Can lead to over-/ underfitting
- Tendency to generate blurry, unrealistic outputs.
- KLD Loss Term and Reconstruction loss Term on different Scales and the Neural Network thus just optimizes one of them
- Assumption that the variational posterior distribution $q_\theta(z|x)$ follows an isotropic Gaussian. VAE cannot guarantee that the inference algorithm will converge onto the standard Gaussian prior if the k latent neurons $z = (z_1, ..., z_k)$ are in fact correlated.
- Assumption that prior is standard gaussian: We try to approximate the posterior to also be standard gaussian. Leads to potential limitations in the flexibility of our posterior approximation.
- There are tighter bounds then ELBO, see [10]
- Posterior collapse: Each datapoint has the exact same value in one latent dimension: $\exists i$ s.t. $\forall x : q_\phi(z_i|x) \approx p(z_i)$. Reduces capacity of generative model, can not use information of all latent dimensions.

Ladder VAE

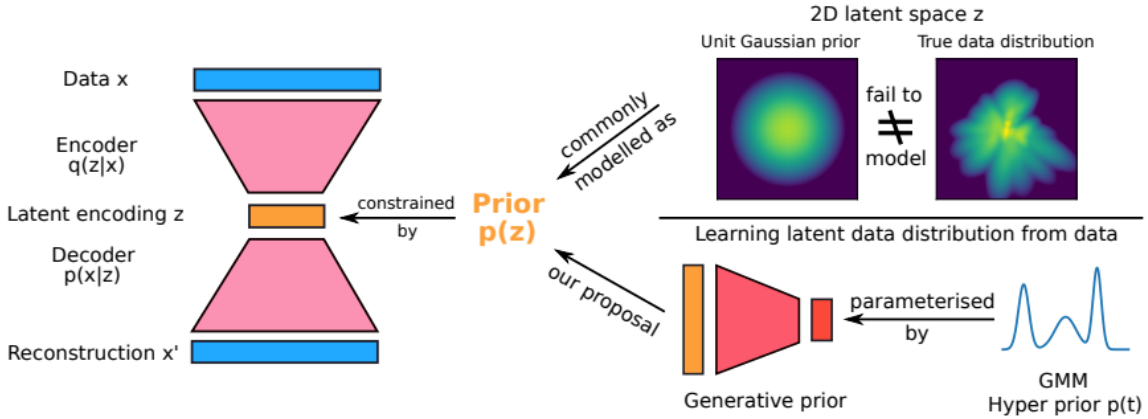


Figure: Ladder VAE: Modelling more complex Priors though multiple stochastic layers which are shared between Encoder and Decoder. [15]

Ladder VAE: Generative Model

Same Setting as in Variational Bayes: Train a generative model $p_\theta(x,z) = p_\theta(x|z)p_\theta(z)$ for some data x using latent variables z, and an inference model q_ϕ optimizing the ELBO.
In generative model p_θ we split the latent variable into L layers and we receive for $i = 1, ..., L$ [15]

$$
\begin{aligned}
p_\theta(z) &= p_\theta(z_L) \prod_{i=1}^{L-1} p_\theta(z_i|z_{i+1}) \\
p_\theta(z_i|z_{i+1}) &= \mathcal{N}(z_i|\mu_{p,i}(z_{i+1}), \sigma^2_{p,i}(z_{i+1})), \quad p_\theta(z_L) = \mathcal{N}(z_L|0, I) \\
p_\theta(x|z_1) &= \mathcal{N}(x|\mu_{p,0}(z_1), \sigma^2_{p,0}(z_1)),
\end{aligned}
$$

where we can also model $p_\theta(x|z_1)$ as Bernoulli in case of Binary data. We use p for the generative model, and q for the inference model. The hierarchical specification allows the lower layers of the latent variables to be highly correlated but still maintain the computational efficiency of fully factorized models. [15]
We just fix the prior of the top most stochastic layer z_L while the prior of all other stochastic layers are learned.

Ladder VAE: Inference Model

Define Layers with $d_0 = x$ for $i = 1, ..., L$ as

$$
\begin{aligned}
d_i &= \text{MLP}(d_{i-1}) \\
\hat{\mu}_{q,i} &= \text{Linear1}(d_i) \\
\hat{\sigma}^2_{q,i} &= \text{Softplus}(\text{Linear2}(d_i))
\end{aligned}
$$

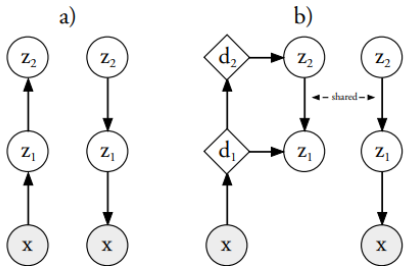


Figure: Inference (or encoder/recognition) and generative (or decoder) models for a) VAE and b) LVAE. Circles are stochastic variables and diamonds are deterministic variables. [15] In LVAE we first go from x to d_L and z_L. Then we can have a none-linear mapping from z_L (standard normal) to z_{L-1} (normal with mean and covariance matrix learned). [15]

Ladder VAE: Inference Model

VAE inference models are parameterized as a bottom-up process. Each stochastic layer is defined as a fully factorized gaussian distribution [15]

$$q_\phi(z|x) = q_\phi(z_L|x) \prod_{i=1}^{L-1} q_\phi(z_i|z_{i+1})$$

$$\sigma_{q,i} = \frac{1}{\hat{\sigma}_{q,i}^{-2} + \sigma_{p,i}^{-2}}$$

$$\mu_{q,i} = \frac{\hat{\mu}_{q,i}\hat{\sigma}_{q,i}^{-2} + \mu_{p,i}\sigma_{p,i}^{-2}}{\hat{\sigma_{q,i}}^{-2} + \sigma_{p,i}^{-2}}$$

$$q_\phi(z_i|\cdot) = \mathcal{N}(z_i|\mu_{q,i}, \sigma_{q,i}^2),$$

The inference model is a precision-weighted combination of $\hat{\mu}_q$ and $\hat{\sigma}_q^2$ carrying bottom-up information and μ_p and σ_p^2 from the generative distribution carrying top-down prior information. [15]

We aim to optimize

$$\mathcal{L}(\theta, \phi, x) = \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x|z)] - \mathcal{D}_{KL}[p_\theta(z_L) \parallel q_\phi(z_L|x)] - \sum_{l=1}^{L-1} \mathbb{E}_{q_\phi(z_{l+1}|x)}[\mathcal{D}_{KL}[p_\theta(z_l|z_{l+1}) \parallel q_\phi(z_l|x, z_{l+1})]].$$

Ladder VAE

To generate new data we first sample from the prior $p_\theta(z_L) = \mathcal{N}(0, I)$ and then iterate through the Decoder Network to obtain the final vector in the lowest level latent space $p_\theta(z) = p_\theta(z_L) \prod_{i=1}^{L-1} p_\theta(z_i | z_{i+1})$. This of course takes longer as we need to iterate through L stochastic layers, but it results in less blurrier samples.

Figure: Conditional Samples from Prior $p_\theta(z)$. [3]

Outlook: NVAE

NVAE is hierarchical model which uses depthwise Convolutions, a residual paramerization of the approximate posterior, spectral regularization and reduces memory burden [16] - overall a lot of new techniques to improve the performance.
Downside: Training takes 92 hours on a machine with 8 16-GB V100 GPUs! [10]

Figure: Current Achievements: Sample from Latent Space with NVAE results in very clear and sharp images. [10]

Outlook: Further Approaches and Applications of VAEs

- BIVA: A Very Deep Hierarchy of Latent Variables for Generative Modeling
- Flow Based Algorithms
- Glow: Generative Flow with Invertible 1×1 Convolutions
- VAE for Reference based Image Super Resolution: Combination of VAEs and GANs: Transformation of Low Resolution Images to High Resolution
- Generating Diverse High-Fidelity Images with VQ-VAE-2

Thanks For your Attention! Questions?

References

[1] user3658307 (https://math.stackexchange.com/users/346641/user3658307). *marginal likelihood in variational bayes*. Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/3341523 (version: 2022-09-13). eprint: https://math.stackexchange.com/q/3341523. URL: https://math.stackexchange.com/q/3341523.

[2] David Dao (https://stats.stackexchange.com/users/98120/david-dao). *How does the reparameterization trick for VAEs work and why is it important?* Cross Validated. URL:https://stats.stackexchange.com/q/205336 (version: 2018-01-16). eprint: https://stats.stackexchange.com/q/205336. URL: https://stats.stackexchange.com/q/205336.

[3] ermongroup. *Variational-Ladder-Autoencoder*. https://github.com/ermongroup/Variational-Ladder-Autoencoder. 2017.

[4] *Gibbs' inequality*. https://myweb.uiowa.edu/pbreheny/7110/wiki/gibbs-inequality.html. 2022.

[5] Irina Higgins et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=Sy2fzU9gl.

[6] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2013. DOI: 10.48550/ARXIV.1312.6114. URL: https://arxiv.org/abs/1312.6114.

[7] Diederik P. Kingma, Tim Salimans, and Max Welling. "Improving Variational Inference with Inverse Autoregressive Flow". In: *CoRR* abs/1606.04934 (2016). arXiv: 1606.04934. URL: http://arxiv.org/abs/1606.04934.

[8] Emile Mathieu et al. *Disentangling Disentanglement in Variational Autoencoders*. 2018. DOI: 10.48550/ARXIV.1812.02833. URL: https://arxiv.org/abs/1812.02833.

[9] Robin Mittas. *variational-autoencoder*. https://github.com/robinmittas/variational-autoencoder. 2023. URL: https://github.com/robinmittas/variational-autoencoder.

[10] NVLABS. *NVAE*. https://github.com/NVlabs/NVAE. 2021.

[11] *PCA FOR 3-DIMENSIONAL POINT CLOUD*. URL:https://www.algosome.com/articles/pca-three-dimensions-point-cloud.html. URL: https://www.algosome.com/articles/pca-three-dimensions-point-cloud.html.

References

[12] Danilo Jimenez Rezende and Shakir Mohamed. *Variational Inference with Normalizing Flows*. 2015. DOI: 10.48550/ARXIV.1505.05770. URL: https://arxiv.org/abs/1505.05770.

[13] Joseph Rocca. *Understanding Variational Autoencoders (VAEs)*. URL:https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73. URL: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73.

[14] Oleh Rybkin, Kostas Daniilidis, and Sergey Levine. "Simple and Effective VAE Training with Calibrated Decoders". In: *CoRR* abs/2006.13202 (2020). arXiv: 2006.13202. URL: https://arxiv.org/abs/2006.13202.

[15] Casper Kaae Sønderby et al. *Ladder Variational Autoencoders*. 2016. DOI: 10.48550/ARXIV.1602.02282. URL: https://arxiv.org/abs/1602.02282.

[16] Arash Vahdat and Jan Kautz. *NVAE: A Deep Hierarchical Variational Autoencoder*. 2020. DOI: 10.48550/ARXIV.2007.03898. URL: https://arxiv.org/abs/2007.03898.

[17] Ritchie Vink. *Another normalizing flow: Inverse Autoregressive Flows*. URL:https://www.ritchievink.com/blog/2019/11/12/another-normalizing-flow-inverse-autoregressive-flows/. URL: https://www.ritchievink.com/blog/2019/11/12/another-normalizing-flow-inverse-autoregressive-flows/.

Appendix: Derivation of Variational Bound I [1]

Assume the true posterior $p_\theta(z|x)$ is too difficult to compute directly (i.e., via Bayes rule), so we instead approximate it with $q_\phi(z|x)$. We then optimize ϕ instead of working with probabilities directly. Mathematically we have that

$$p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)} = \frac{p_\theta(x|z)p_\theta(z)}{\int p_\theta(x|z)p(z)dz} \approx q_\phi(z|x).$$

The Log-marginal likelihood can then be expressed as

$$\log(p_\theta(x)) = \mathcal{D}_{KL}[p_\theta(z|x) \parallel q_\phi(z|x)] + \mathcal{L}(\theta, \phi|x), \tag{8}$$

where the first term is the non-negative KL-Divergence of the approximate to the true posterior. The second term is called the variational lower bound (or: evidence lower bound (ELBO)) on the marginal likelihood of datapoint i. We can rewrite this as

$$\begin{aligned}
\mathcal{L}(\theta, \phi|x) &= \mathbb{E}_{q_\phi(z|x)}[-\log(q_\phi(z|x)) + \log(p_\theta(x, z))] \\
&= \int q_\phi(z|x)[-\log(q_\phi(z|x)) + \log(p_\theta(x|z)) + \log(p_\theta(z))]dz \\
&= \int q_\phi(z|x)\log\Big(\frac{p_\theta(z)}{q_\phi(z|x)}\Big)dz + \int q_\phi(z|x)\log(p_\theta(x|z))dz \\
&= -\mathcal{D}_{KL}[p_\theta(z) \parallel q_\phi(z|x)] + \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x|z))]
\end{aligned} \tag{9}$$

Appendix: Derivation of Variational Bound II [1]

In the last step we have used the fact that the KL-Divergence loss for some probability distributions p_λ and p_τ is defined as

$$\mathcal{D}_{KL}[p_\lambda(y) \parallel p_\tau(y)] = \int \log\Big(\frac{p_\lambda(y)}{p_\tau(y)}\Big) p_\lambda(y) dy = \mathbb{E}_{p_\lambda(y)}\Big[\log\Big(\frac{p_\lambda(y)}{p_\tau(y)}\Big)\Big]. \tag{10}$$

Now let us derive equation (8):

$$\begin{aligned}
\log(p_\theta(x)) &= \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x))] \\
&= \mathbb{E}_{q_\phi(z|x)}\Big[\log\Big(\frac{p_\theta(x,z)}{p_\theta(z|x)}\frac{q_\phi(z|x)}{q_\phi(z|x)}\Big)\Big] \\
&= \mathbb{E}_{q_\phi(z|x)}\Big[\log\Big(\frac{q_\phi(z|x)}{p_\theta(z|x)}\Big)\Big] + \mathbb{E}_{q_\phi(z|x)}\Big[\log\Big(\frac{p_\theta(x,z)}{q_\phi(z|x)}\Big)\Big] \\
&= \mathcal{D}_{KL}[p_\theta(z|x) \parallel q_\phi(z|x)] + \mathcal{L}(\theta,\phi|x),
\end{aligned}$$

where in the first step, taking the expectation, we used the fact that there is no dependence on z in the marginal and the second used the identity $p_\theta(x,z) = p_\theta(z|x)p_\theta(x)$. Now using the fact that the KL-Divergence is none-negative [4] in combination with equation (9) we will recieve the following bound for the marginal likelihood

$$\log(p_\theta(x)) \geqslant \mathcal{L}(\theta,\phi|x) = -\mathcal{D}_{KL}[p_\theta(z) \parallel q_\phi(z|x)] + \mathbb{E}_{q_\phi(z|x)}[\log(p_\theta(x|z))]. \tag{11}$$

Appendix: Derivation of Variational Bound III [1]

We note that maximizing equation (9), with respect to θ will concurrently maximize the expectation (right term) and minimize the KL-Divergence Term (left term). In general we want to maximize the Log-marginal likelihood $\log(p_\theta(x))$ which is then equivalent to maximizing equation 11.

Appendix: KL-Divergence of two Gaussians

Let $n \in \mathbb{N}$ be the dimensionality of the latent space. Assume that the prior $p_\theta(z) \sim \mathcal{N}(0, I)$ and the posterior approximation $q_\phi(z|x) \sim \mathcal{N}(\mu, \sigma)$ are Gaussian, with μ and σ as the variational mean and standard deviation evaluated at datapoint k, and let μ_i and σ_i simply denote the i-th element of these vectors. Then with the definition of the KLD term given in Equation (10) we split the log into two parts and calculate two integrals:

$$\begin{aligned}\int q_\phi(z|x)\log(q_\phi(z|x))dz &= \int \mathcal{N}(\mu,\sigma)\log(\mathcal{N}(\mu,\sigma)) \\ &= -\frac{n}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{n}(1+\log(\sigma_j^2))\end{aligned}$$

For the second integral we get:

$$\begin{aligned}\int q_\phi(z|x)\log(p(z))dz &= \int \mathcal{N}(\mu,\sigma)\log(\mathcal{N}(0,I))dz \\ &= -\frac{n}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{n}(\mu_j^2+\sigma_j^2)\end{aligned}$$

Putting now both together, it follows

$$-\mathcal{D}_{KL}[q_\phi(z|x) \parallel p_\theta(z)] = \frac{1}{2}\sum_{j=1}^{n}(1+\log(\sigma_j^2) - \mu_j^2 - \sigma_j^2)$$

Appendix: Inverse Autoregressive Flow

Goal: Building flexible posterior distributions $q_\theta(z|x)$ through an iterative procedure. Think of the „simplest“ special case of IAF where we transform a Gaussian variable with diagonal covariance to one with linear dependencies, thus the Covariance between latent dimension is not necessarily 0 (see Appendix in [7]). Autoregressive means that each variable is dependent only on the previously indexed variables.

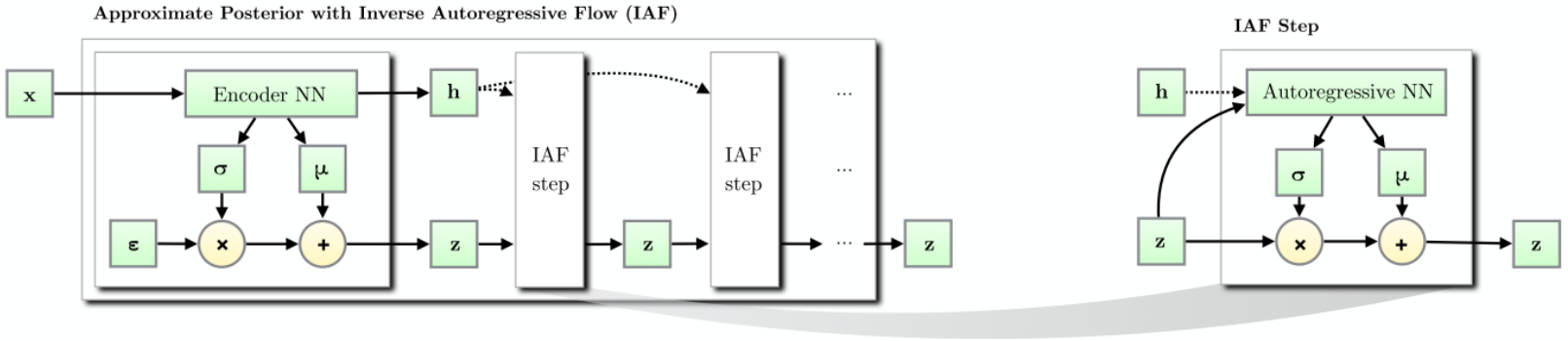


Figure: (Left) An Inverse Autoregressive Flow (IAF) transforming the basic posterior of a variational encoder to a more complex posterior through multiple IAF transforms. (Right) A single IAF transform. [7]

Appendix: Normalizing Flow

Idea: Start with initial random variable with a simple distribution with known density function and then apply a chain of invertible parameterized transformations f_t, where $t \in \mathbb{N}$ represents a time-step/ iteration, such that the last iterate z_T has a more flexible distribution. [7]

$$z_0 \sim q(z_0|x)$$
$$z_t = f_t(z_{t-1}, x) \quad \forall t = 1, ..., T$$

If Jacobian determinant of each of the transformation f_t can be computed, the probability density function of the last iterate can still be computed as [7]

$$\log q(z_T|x) = \log(q(z_0|x)) - \sum_{t=1}^{T} \log \det \left| \frac{dz_t}{dz_{t-1}} \right|. \tag{12}$$

A single IAF layer takes as input a point in the latent space and produces as output a transformed point in the latent space. Note that the term „Normalizing“ means that after each iterate t the probability distribution needs to fulfill

$$\int q(z_t|x) dz_t = 1.$$

Appendix: Inverse Autoregressive Flow

The Encoder Neural Network outputs μ_0, σ_0 as usual in addition to an extra output h. We sample $\varepsilon \sim \mathcal{N}(0, I)$ and initialize the chain of length T with z_0 and define the iterate z_t

$$z_0 = \mu_0 + \sigma_0 \odot \varepsilon$$
$$z_t = \mu_t + \sigma_t \odot z_{t-1},$$

where for every step $t \in \{1, ..., T\}$ we use a different Autoregressive Neural Network with input z_{t-1}. The Jacobian $\frac{dz_t}{dz_{t-1}}$ is diagonal with σ_t on the diagonal (note that the Jacobian $d\mu_t/dz_{t-1}$ and $d\sigma_t/dz_{t-1}$ is triangular with 0 on the diagonal) and thus the determinant

$$\det \left| \frac{dz_t}{dz_{t-1}} \right| = \prod_{i=1}^{t} \sigma_{t,i}.$$

Using equation (12) following density for the last iterate can be derived (note that all exp terms are removed by log) [7]

$$\log q(z_T|x) = -\sum_{i=1}^{n} \left(\frac{1}{2}\varepsilon_i^2 + \frac{1}{2}\log(2\pi) + \sum_{t=0}^{T} \log \sigma_{t,i} \right). \tag{13}$$

Appendix: IAF numerically stable Transformation

A numerically more stable variant lets the Autoregressive Step outputs $[m_t, s_t]$ and and reparameterize z_t as

$$
\begin{aligned}
[m_t, s_t] &= \text{AutoregressiveNN}[t](z_t, h; \theta) \\
\sigma_t &= \text{sigmoid}(s_t) \\
z_t &= \sigma_t \odot z_{t-1} + (1 - \sigma_t) \odot m_t,
\end{aligned}
$$

where such parameterization is known as a forget gate bias in LSTMs.
The Authors propose to use following loss function with some proposed $\lambda \in [0.125, 2]$ [7]

$$
\mathcal{L}_\lambda(\theta, \phi | x) = \mathbb{E}_{q_\phi(z_T|x)}[\log(p_\theta(x|z_T))] - \sum_{j=1}^{T} \max\{\lambda, \mathcal{D}_{KL}[q_\phi(z_j|x) \parallel p_\theta(z_j)]\}.
$$

Summarising the IAF learns the posterior distribution $q_\phi(z|x)$ over the latent space using the training data. Generating new images by first sampling $\varepsilon \sim \mathcal{N}(0, I)$ and then iterating through all T trained IAF Layers, allows the samples to be much closer to the true posterior and the generated new samples to be more representative of the training data.

Appendix: Algorithm IAF

The flexibility of the distribution of the final iterate z_T , and its ability to closely fit to the true posterior, increases with the expressivity of the autoregressive models and the depth of the chain. [7]

Algorithm 1: Pseudo-code of an approximate posterior with Inverse Autoregressive Flow (IAF)

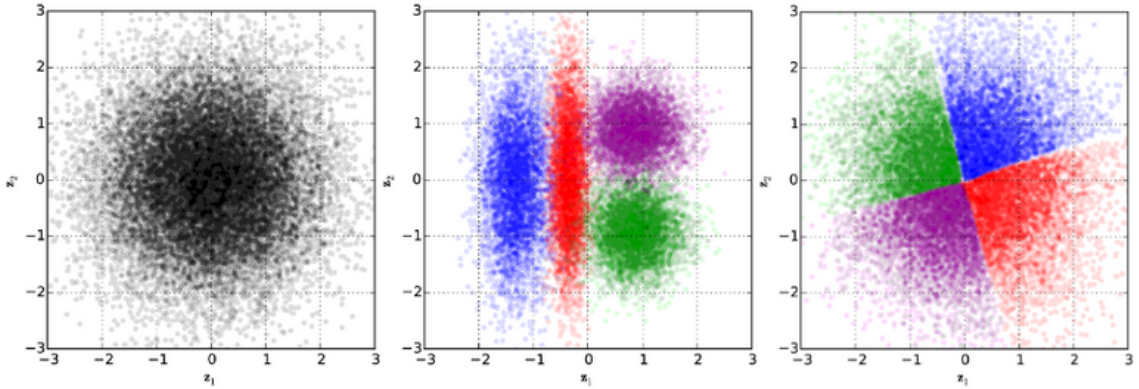
Data:
- $\mathbf{x}$: a datapoint, and optionally other conditioning information
- $\boldsymbol{\theta}$: neural network parameters
- $\texttt{EncoderNN}(\mathbf{x}; \boldsymbol{\theta})$: encoder neural network, with additional output $\mathbf{h}$
- $\texttt{AutoregressiveNN}[*](\mathbf{z}, \mathbf{h}; \boldsymbol{\theta})$: autoregressive neural networks, with additional input $\mathbf{h}$
- $\texttt{sum}(.)$: sum over vector elements
- $\texttt{sigmoid}(.)$: element-wise sigmoid function

Result:
- $\mathbf{z}$: a random sample from $q(\mathbf{z}|\mathbf{x})$, the approximate posterior distribution
- l: the scalar value of $\log q(\mathbf{z}|\mathbf{x})$, evaluated at sample '$\mathbf{z}$'

$[\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{h}] \leftarrow \texttt{EncoderNN}(\mathbf{x}; \boldsymbol{\theta})$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$
$\mathbf{z} \leftarrow \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} + \boldsymbol{\mu}$
$l \leftarrow -\texttt{sum}(\log \boldsymbol{\sigma} + \frac{1}{2}\boldsymbol{\epsilon}^2 + \frac{1}{2}\log(2\pi))$
for $t \leftarrow 1$ **to** T **do**
  $[\mathbf{m}, \mathbf{s}] \leftarrow \texttt{AutoregressiveNN}[t](\mathbf{z}, \mathbf{h}; \boldsymbol{\theta})$
  $\boldsymbol{\sigma} \leftarrow \texttt{sigmoid}(\mathbf{s})$
  $\mathbf{z} \leftarrow \boldsymbol{\sigma} \odot \mathbf{z} + (1 - \boldsymbol{\sigma}) \odot \mathbf{m}$
  $l \leftarrow l - \texttt{sum}(\log \boldsymbol{\sigma})$
end

Figure: Proposed Pseudo-Algorithm [7]

Appendix: Posterior IAF



(a) Prior distribution (b) Posteriors in standard VAE (c) Posteriors in VAE with IAF

Figure: Fitted VAE with a spherical Gaussian prior, and with factorized Gaussian (class distributions are independent - each class has its own Covariance matrix) posteriors (b) or inverse autoregressive flow (IAF) posteriors (c) to a toy dataset with four classes. Each colored cluster corresponds to the posterior distribution of one class. IAF greatly improves the flexibility of the posterior distributions, and allows for a much better fit between the posteriors and the prior. [7]

Appendix: Posterior IAF

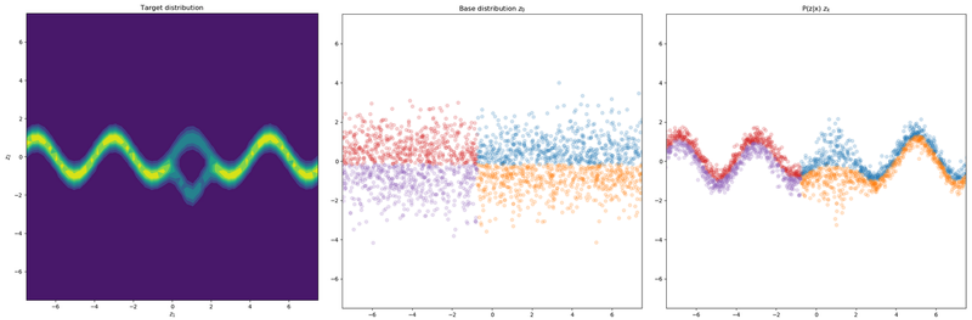


Figure: Approximate Posterior with 4 IAF Layers with Base Distribution $z_0 \sim \mathcal{N}(0, I)$. [17]

Advantage: Scales well to high-dimensional data, can be parallelized and we can model flexible posteriors. Note that in the paper [7] they also use a hierarchical structure, e.g. inference model has information of generative model.
Drawpack: Sampling and Generation from posterior/ prior takes much longer as we need to iterate through T flow layers, in comparison to just sample $\varepsilon \sim \mathcal{N}(0, I)$ and reparameterize in standard VAE. Not very straightforward to implement.

Appendix: IAF Transformation and Parallelizability

Consider some autoregressive Autoencoders such as MADE or PixelCNN where the basic idea is that each pixel is dependent on the previously indexed pixel - which is an autoregressive structure, e.g. for an $n \times n$ image we have

$$p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, ..., x_{i-1}).$$

Now let us consider some variable $y \in \mathbb{R}^d$ which can be ordered arbitrarily. We use $[\mu(y), \sigma(y)]$ to denote some function of the vector y to some vectors μ, σ. Now let us sample some $\varepsilon \sim \mathcal{N}(0, I)$ as usual and define the first entry of y as $y_0 = \mu_0 + \sigma_0 \varepsilon_0$ and for $i > 0$ we get $y_i = \mu(y_{1:i-1}) + \sigma(y_{1:i-1})\varepsilon_i$. So this computation is clearly proportional to the dimension d. However the inverse transformation is of interest as long as we have $\sigma_i > 0$ for all $i = \{0, ...d\}$ as we then have a one-to-one transformation

$$\varepsilon_i = \frac{y_i - \mu(y_{1:i-1})}{\sigma(y_{1:i-1})}.$$

Now one key observation is that the entries ε_i do not depend on each other and thus the transformation can be prallelized in case of autoregressive autoencoders.

Appendix: IAF Transformation of Random Variable

The basic rule for transformations of densities considers an invertible, smooth mapping $f : \mathbb{R}^n \to \mathbb{R}^n$ with inverse $f^{-1} = g$, s.t. for the composition $g(f(z)) = z$ holds. If we use this mapping to transform a random variable z with distribution $q(z)$, the resulting random variable $z^* = f(z)$ has following distribution [12]

$$q(z^*) = q(z)\left|\det \frac{\partial f^{-1}}{\partial z^*}\right| = q(z)\left|\det \frac{\partial f}{\partial z}\right|^{-1}.$$

Now with that we receive for $z_0 = \mu + \sigma \odot \varepsilon$ for some $\varepsilon \sim \mathcal{N}(0, I)$ and $\mu, \sigma \in \mathbb{R}^n$ the density

$$q_\phi(z_0|x) = \mathcal{N}(\varepsilon|0, I)\prod_{i=1}^{n} \sigma_i^{-1}$$

Appendix: Inverse Autoregressive Flow

Let a flow consitst of a chain of T transformations, where for every step t we use a different autoregressive neural network with inputs z_{t-1}, h and outputs μ_t, σ_t. The Neural Network is structured to be autoregressive w.r.t. z_{t-1} which can be expressed as

$$\begin{aligned} \mu_{t,i} &= f(z_{t-1,\{0:i-1\}}) \\ \sigma_{t,i} &= g(z_{t-1,\{0:i-1\}}) \\ z_t &= \mu_t + \sigma_t \odot z_{t-1}, \end{aligned}$$

such that for any choice of its parameters, the Jacobians $\frac{d\mu_t}{dz_{t-1}}, \frac{d\sigma_t}{dz_{t-1}}$ are triangular with 0 on the diagonal. [7]

Appendix: IAF Loss Function

With equation (9) we can derive following objective function with $n \in \mathbb{N}$ the latent dimension

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

where

$$\log(q_\phi(z_T|x) = -\sum_{i=0}^{n} \frac{\varepsilon_i}{2} + \frac{1}{2}\log(2\pi) - \sum_{t=0}^{T} \log(\sigma_{t,i})$$
$$\log(p(z_T)) = \log(\mathcal{N}(0, I)).$$