University of Bergen
Department of Informatics

Multitask variational autoencoders

Author: Edvards Zakovskis
Supervisors: Nello Blaser and Kristian Gundersen

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

January, 2024

Abstract

Lorem ipsum dolor sit amet, his veri singulis necessitatibus ad. Nec insolens periculis ex. Te pro purto eros error, nec alia graeci placerat cu. Hinc volutpat similique no qui, ad labitur mentitum democritum sea. Sale inimicus te eum.

No eros nemore impedit his, per at salutandi eloquentiam, ea semper euismod meliore sea. Mutat scaevola cotidieque cu mel. Eum an convenire tractatos, ei duo nulla molestie, quis hendrerit et vix. In aliquam intellegam philosophia sea. At quo bonorum adipisci. Eros labitur deleniti ius in, sonet congue ius at, pro suas meis habeo no.

Acknowledgements

Est suavitate gubergren referrentur an, ex mea dolor eloquentiam, novum ludus suscipit in nec. Ea mea essent prompta constituam, has ut novum prodesset vulputate. Ad noster electram pri, nec sint accusamus dissentias at. Est ad laoreet fierent invidunt, ut per assueverit conclusionemque. An electram efficiendi mea.

`Your name`

Monday 29th January, 2024

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

In recent years, Variational Autoencoders (VAEs) have emerged as a powerful tool in the field of deep learning and generative modeling. VAEs offer a principled framework for capturing complex data distributions and have found applications in diverse domains, including image generation, anomaly detection, and natural language processing. However, like any other machine learning model, VAEs have their own set of limitations and challenges.

One of the key challenges in training VAEs is achieving a balance between the reconstruction accuracy and the effectiveness of the learned latent representations. Traditional VAEs are typically trained to optimize a single objective, which often results in suboptimal performance on complex datasets with multiple sources of variation. In real-world scenarios, data often exhibits multiple underlying structures and tasks, and capturing these structures with a single VAE can be a challenging endeavor.

Multitask learning, on the other hand, is a paradigm that aims to improve model generalization and performance by simultaneously learning multiple related tasks. The idea of combining VAEs with multitask learning has recently gained attention in the machine learning community. Multitask Variational Autoencoders (MT-VAEs) extend the VAE framework to enable the joint learning of multiple latent variable spaces, each dedicated to a specific task or source of variation. This approach holds the promise of not only improving the reconstruction accuracy of VAEs but also enhancing their interpretability and generalization capabilities.

This research endeavors to investigate the effectiveness of MT-VAEs and their potential to outperform traditional VAEs in terms of data generation, representation learning,

and task-specific performance. By conducting a comprehensive analysis and experimentation, we aim to shed light on the advantages and limitations of MT-VAEs and provide insights into the conditions under which they excel.

1.1 Research Question

The central research question addressed in this thesis is as follows:

> *Can Multitask Variational Autoencoders (MT-VAEs) enhance the performance and versatility of Variational Autoencoders (VAEs) by jointly optimizing multiple tasks, and under what conditions do MT-VAEs demonstrate superior performance in comparison to traditional VAEs?*

Chapter 2

Background

In this chapter I am going to introduce the reader to the concepts that are necessary to understand the research presented in this thesis. The chapter is divided into three sections. The first section provides an overview of Variational Autoencoders (VAEs) and their applications. The second section introduces Vector Quantized VAEs (VQ-VAEs). The third section introduces Multitask VAEs (MT-VAEs), which are the main focus of this thesis.

2.1 VAEs

Variational Autoencoders (VAEs), first introduced in 2013 by Kingma and Welling[1], have become a prominent class of generative models in the field of machine learning. At their core, VAEs consist of an encoder network with parameters ϕ that maps data points x into a latent space z and a decoder network with parameters θ that generates data $\hat{x}$ from latent representations[2].

The key innovation that makes VAEs work is the introduction of a probabilistic interpretation of the latent space. More specifically, VAEs assume that the latent space z is a random variable that follows a certain prior distribution $p(z)$, which is typically a Gaussian distribution and that the mapping from the latent space to the data space is also probabilistic[1].

The optimization target for VAEs is the evidence lower bound (ELBO) which is:

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

where $q_\phi(z|x)$ is the encoder distribution, $p_\theta(z,x)$ is the decoder distribution and $p(z)$ is the prior distribution.

The ELBO can be also written as a sum of two terms:

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

And if we look at the Monte Carlo estimate of the ELBO, we can see that the first term is the regularization term and the second term is the reconstruction term:

$$L_{\theta,\phi}(x) = -D_{KL}(q_\phi(z|x)||p(z)) + \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x|z^{(l)})$$

where $z^{(l)} \sim q_\phi(z|x)$. The regularization term is the Kullback-Leibler divergence between the encoder distribution and the prior distribution. The regularization term encourages the encoder distribution to be close to the prior distribution. The reconstruction term is the reconstruction error of the decoder. The reconstruction term encourages the decoder to reconstruct the input data as accurately as possible.

The individual datapoint ELBO and it's gradient in general is intractable to compute. However, unbiased estimates of the ELBO and its gradients can be obtained using the reparametrization trick, which is described in the next section[2].

2.1.1 Reparametrization Trick

The Reparametrization trick also is a crucial component of VAEs. It is used to make the ELBO differentiable w.r.t the parameters of the encoder ϕ and decoder θ. The notion is based on the fact we can define a random variable $z \sim q_\phi(z|x)$ as a deterministic, differentiable function of a random variable ϵ and the parameters ϕ such that $z = g_\phi(\epsilon, x)$, where the distribution of ϵ is independent of ϕ and x: $\epsilon \sim p(\epsilon)$. With this parameterization, the expectation w.r.t $q_\phi(z|x)$ can be rewritten as an expectation w.r.t. $p(\epsilon)$:

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

$$= \mathbb{E}_{p(\epsilon)}[\log p_\theta(x, g_\phi(\epsilon,x)) - \log q_\phi(g_\phi(\epsilon,x)|x)]$$

which is differentiable w.r.t ϕ and θ.

2.1.2 Gaussian VAEs

Altough Gaussian VAEs are just a special case of VAEs, they are the most common type of VAEs. Gaussian VAEs assume that the prior distribution $p(z)$ is a centered Gaussian distribution $p(z) = \mathcal{N}(0, I)$. They also assume that the decoder distribution $p_\theta(x|z)$ is a Gaussian distribution whose distribution parameters are computed from z with a single fully connected layer:

$$p_\theta(x|z) = \mathcal{N}(f_\theta(z), \sigma_\theta(z))$$

where $f_\theta(z)$ is the mean and $\sigma_\theta(z)$ is the standard deviation of the Gaussian distribution. Whilst there is a lot of freedom in the form $q_\phi(z|x)$ can take, Gaussian VAEs assume that $q_\phi(z|x)$ is also a Gaussian distribution with an approximately diagonal covariance matrix:

$$q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$$

where $\mu_\phi(x)$ and $\sigma_\phi(x)$ are the mean and standard deviation of the Gaussian distribution which are computed by the encoder network.

To sample z from $q_\phi(z|x)$, we can use the reparametrization trick described in the previous section:

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon$$

where $\epsilon \sim \mathcal{N}(0, I)$ is a random variable sampled from a standard Gaussian distribution.

When applying these assumptions to the ELBO, we get the following expression:

$$L_{\theta,\phi}(x) = -D_{KL}(q_\phi(z|x)||p(z)) + \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x|z^{(l)})$$

$$= -D_{KL}(\mathcal{N}(\mu_\phi(x), \sigma_\phi(x))||\mathcal{N}(0, I)) + \frac{1}{L}\sum_{l=1}^{L} \log \mathcal{N}(x|f_\theta(z^{(l)}), \sigma_\theta(z^{(l)}))$$

where $f_\theta(z^{(l)}) = f_\theta(\mu_\phi(x) + \sigma_\phi(x) \odot \epsilon^{(l)})$ and $\sigma_\theta(z^{(l)}) = \sigma_\theta(\mu_\phi(x) + \sigma_\phi(x) \odot \epsilon^{(l)})$ and $\epsilon^{(l)} \sim \mathcal{N}(0, I)$.

However, the loss function to be minimized for VAE's usually used in practise is quite different from the ELBO negative. The function that is used in practice consists of: Mean Squared Error (MSE) reconstruction loss, KL divergence regularization loss and a constant β that controls the importance of the regularization term:

$$L = \frac{1}{n}\sum_{i=1}^{n} ||x_i - f_\theta(z^{(i)})||^2 + D_{KL}(\mathcal{N}(\mu_\phi(x), \sigma_\phi(x))||\mathcal{N}(0, I))$$

where $f_\theta(z^{(i)}) = f_\theta(\mu_\phi(x_i) + \sigma_\phi(x_i) \odot \epsilon^{(i)})$ and $\epsilon^{(i)} \sim \mathcal{N}(0, I)$. The MSE reconstruction loss is used because maximizing the Gaussian likelihood is approximately equivalent to minimizing the MSE reconstruction loss:

$$
\begin{aligned}
p(x|z) &= \mathcal{N}(x|f(z), \sigma(z)) \\
\log p(x|z) &\sim \log \mathcal{N}(x|f(z), \sigma(z)) \\
\log p(x|z) &\sim \log \frac{1}{\sqrt{2\pi\sigma(z)^2}} \exp(-\frac{(x - f(z))^2}{2\sigma(z)^2}) \\
\log p(x|z) &\sim \log \exp(-\frac{(x - f(z))^2}{2\sigma(z)^2}) \\
\log p(x|z) &\sim -\frac{1}{2\sigma(z)^2} * (x - f(z))^2 \\
\log p(x|z) &\sim -(x - f(z))^2
\end{aligned}
\tag{2.1}
$$

In the figure belowe 2.1 there is a visualization of the architecture of Gaussian VAEs.

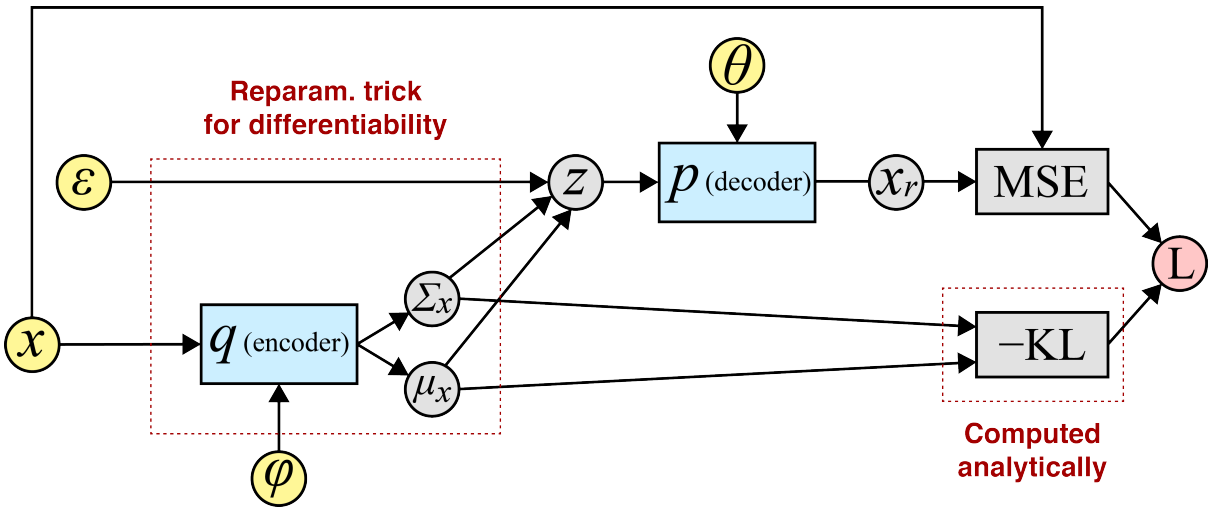

Figure 2.1: The architecture of VAEs.

Credit: Aäron van den Oord et al.

2.2 Vector Quantized VAEs

Vector Quantized VAEs (VQ-VAEs) are a variant of VAEs that were introduced in 2017 by Aäron van den Oord et al[3]. VQ-VAEs use discrete latent variables with a new way of training, which was inspired by vector quantization (VQ). Both posterior and prior distributions are categorical, and the samples are drawn from these distributions index an embedding table[3]. These embeddings are then used to reconstruct the input data. The architecture of VQ-VAEs is shown in figure 2.2.

2.2.1 Discrete Latent Variables

VQ-VAEs define a latent embedding space $e \in \mathbb{R}^{K \times D}$, where K is the number of embeddings and D is the dimension of each latent embedding vector. The model takes an input x, which is passed through the encoder producing output $z_e(x)$, as shown in figure 2.2. The discrete latent variables z are then calculated by nearest neighbour lookup in the embedding space:

$$z = \arg\min_k ||z_e(x) - e_k||^2$$

where e_k is the k-th embedding vector in the embedding space. The decoder then takes the discrete latent variables z and produces the output $\hat{x}$.

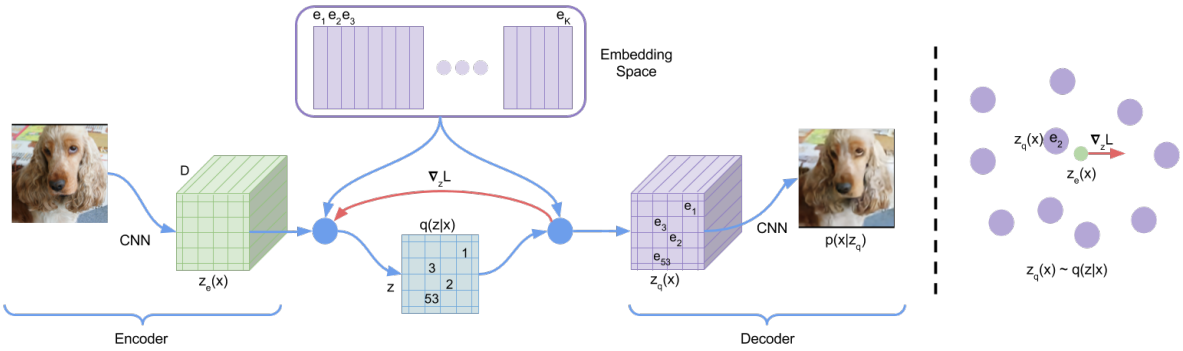

Figure 2.2: On the left side there is a figure describing the VQ-VAE architecture. On the right side there is visualization of the latent space whilst training. The figure is taken from [3].

Credit: Aäron van den Oord et al.

Chapter 3

Methodology

Chapter 4

Results

Chapter 5

Discussion

Bibliography

[1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[2] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019. ISSN 1935-8245. doi: 10.1561/2200000056.
URL: http://dx.doi.org/10.1561/2200000056.

[3] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *CoRR*, abs/1711.00937, 2017.
URL: http://arxiv.org/abs/1711.00937.

Appendix A

Generated code from Protocol buffers

Listing A.1: Source code of something

```
1 System.out.println("Hello Mars");
```