University of Bergen
Department of Informatics

Multitask variational autoencoders

Author: Edvards Zakovskis
Supervisors: Associate professor Nello Blaser, Postdoktor Kristian Gundersen

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

February, 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 12th February, 2024

Contents

List of Figures

Chapter 1

Introduction

In this chapter, I will explain the reasoning and motivation behind this research. In the second section, I will talk about the objective of the thesis and the research questions that I will attempt to answer. Finally, I will provide a brief outline for the rest of the thesis

1.1 Motivation

Over the past 10 years, Variational Autoencoders (VAEs) have become valuable assets in the field of deep learning and generative modeling. At the most basic level, they are used as tools for data generation and compression by learning the underlying patterns and structures present in a given dataset. VAEs have shown their versatility and have found applications in multiple domains, including image generation, anomaly detection, natural language processing, and speech synthesis. However, VAEs, given a limited set of data and compute power, have limitations and challenges.[7, 8, 15, 10]

One of the key challenges traditional VAE's face is that they tend to produce blurry, over-smoothed samples. More recently Vector Quantized Variational Autoencoders (VQ-VAEs) have been introduced to address VAE's limitation, which combines the strenghts of VAEs and discrete vector quantization.[10] The VQ-VAE architecture solves the problem of capturing fine-grained details in the data, whilst maintaning interpretability of latent representations and generational properties of VAEs. However, both VAEs and VQ-VAEs have the same challange of striking a balance between the reconstruction accuracy and

the effectiveness of the learned latent representations and when the data has multiple sources of variation it can be difficult to capture all of that in a single model.[8, 6, 15]

Multitask learning, on the contrary, is a paradigm that aims to improve model generalization and performance by simultaneously learning multiple related tasks. The concept of combinging VAEs with multitask learning has undergone some experimental exploration in the research community, altough it has not been thoroughly investigated.[9] Multitask Variational Autoencoders (MT-VAEs) extends the VAE model to take advantage the extra information or tasks thata are available in the data. This approach holds the promise of improving the reconstruction accuracy and generalization capabilities of VAEs, as well as enhancing their interpretability.[2]

This research aims to investigate the effectiveness and potential of combining semi-conditioned and conditioned VAEs in one model which will be explored in the context of standard VAEs and as well as VQ-VAEs. This research proposes 2 methods of combining semi-conditioned and conditioned VAEs, which will be referred to as SCVAE1D, SCVAE2D, SCVQVAE1D and SCVQVAE2D. SCVAE2D and SCVQVAE2D is an approach to combine semi-conditioned and non-conditioned VAEs with 2 decoders, where the first decoder is conditioned and the second decoder is non-conditioned. SCVAE1D and SCVQVAE1D is an approach to combine semi-conditioned and non-conditioned VAEs with 1 decoder, where the decoder can be conditioned or non-conditioned. The main goal of this research is to investigate the potential of these methods to improve the reconstruction accuracy and generalization capabilities of VAEs and VQ-VAES and by conducting a comprehensive analysis and experimentation, the aim is to shed light on the advantages and limitations of these methods.

1.2 Objective

In this thesis, I aim to investigate and explore the integration of SCVAEs can be combined with standard non-conditioned VAEs through multi-task learning. My objective is to determine whether leveraging the multitask property enhances the performance of VAEs compared to traditional ones, and if so, under what conditions. Additionally, I will explore the applicability of this multitask property to VQ-VAEs.

More specifically, I will attempt to answer the following questions:

- Can semi-conditional variational autencoders by combined with standard VAEs through multi-task learning?
- Can semi-conditional variational autoencoders can also be combined with the VQ-VAEs with multi task learning?
- .....

1.3 Thesis Outline

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) and PixelCNN. The third section introduces the concept of semi-conditioned VAEs. The chapter is concluded with a section that introduces the concept of multitask learning.

2.1 VAEs

Variational Autoencoders (VAEs), first introduced in 2013 by Kingma and Welling[7], 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[8].

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[7].

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, $p_\theta(x, z)$ is the decoder distribution.

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

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

where $D_{KL}(q_\phi(z|x)||p(z))$ is the Kullback-Leibler divergence between the encoder distribution $q_\phi(z|x)$ and the prior distribution $p(z)$ and $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$ is the reconstruction term. The Kullback-Leibler divergence term encourages the encoder distribution to be close to the prior distribution, while the reconstruction term encourages the reconstruction to be as accurate as possible. Whilst this represantion of the ELBO may look different from the original, it is equivalent definition using different terms.[8].

Taking random samples from from $q_\phi(z|x)$, Monte Carlo estimate of ELBO can be written of this as

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

where $z^{(l)} \sim q_\phi(z|x)$ and L is the number of samples and first term is the regularization term and the second term is the reconstruction term[8].

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[8].

2.1.1 Reparametrization Trick

The Reparametrization trick also is a crucial component of VAEs. It is used to make the ELBO differentiable with respect to the parameters of the encoder ϕ and decoder θ through a change of variables.[8]

Change of variables

The notion is based on the fact that it is possible to express the random variable $z \sim q_\phi(z|x)$ as a differentiable function of a random variable ϵ and the parameters ϕ such that $z = g_\phi(\epsilon, x)$, where epsilon is a random variable that is independent of ϕ and x and $\epsilon \sim p(\epsilon)$. Given this change of variables, the expection with respect to $q_\phi(z|x)$ can be rewritten as an expectation with respect to $p(\epsilon)$

$$E_{q_\phi(z|x)}[f(z)] = E_{p(\epsilon)}[f(g_\phi(\epsilon, x))],$$

where f is an arbitrary function.[8] As a result, the gradients the expectation and gradient operators become cummutive, and there can be formed a Monte Carlo estimate of the gradients

$$\begin{aligned}\nabla_\phi E_{q_\phi(z|x)}[f(z)] &= \nabla_\phi E_{p(\epsilon)}[f(g_\phi(\epsilon, x))] \\ &= E_{p(\epsilon)}[\nabla_\phi f(g_\phi(\epsilon, x))] \\ &\simeq \frac{1}{L}\sum_{l=1}^{L} \nabla_\phi f(g_\phi(\epsilon^{(l)}, x))\end{aligned}$$

where $\epsilon^{(l)} \sim p(\epsilon)$ and L is the number of samples. This is the reparametrization trick, which is further explained and illustrated in the figure 2.1.

Gradients of the ELBO

When applying the reparametrization trick to the ELBO it becomes differentiable with respect to both ϕ and θ and it is possible to form a Monte Carlo estimate of the gradients

$$\begin{aligned}\nabla_{\phi,\theta} L_{\theta,\phi}(x) &= \nabla_{\phi,\theta} E_{q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z|x)] \\ &= E_{p(\epsilon)}[\nabla_{\phi,\theta}[\log p_\theta(x, g_\phi(\epsilon, x)) - \log q_\phi(g_\phi(\epsilon, x)|x)]] \\ &\simeq \frac{1}{L}\sum_{l=1}^{L} \nabla_{\phi,\theta}[\log p_\theta(x, g_\phi(\epsilon^{(l)}, x)) - \log q_\phi(g_\phi(\epsilon^{(l)}, x)|x)]\end{aligned}$$

where $\epsilon^{(l)} \sim p(\epsilon)$ and L is the number of samples.

This is the key to training VAEs using stochastic gradient descent. The resulting Monte Carlo gradient estimate is used to update the parameters of the encoder and decoder networks.[8]

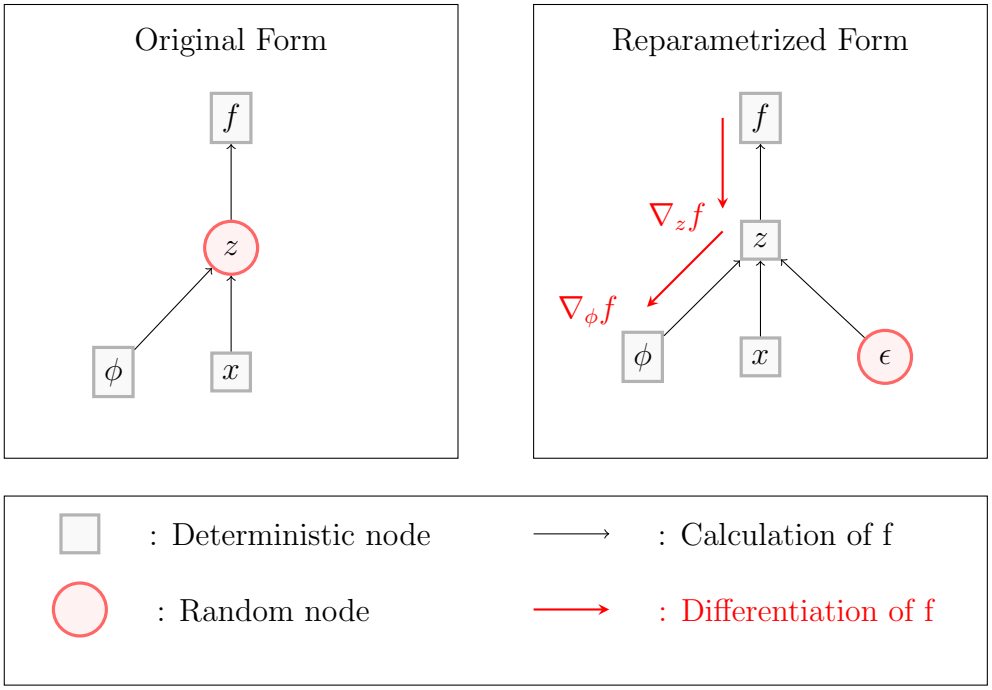

Figure 2.1: The illustration diagram of the reparametrization trick. The input of a function f is x. The parameters θ affect the objective of the function f through a random variable z. In the original form we can not compute the gradients $\nabla_\phi f$, because a direct backpropagation is not possible through a random variable. In the reparameterized form, the randomness is seperated from the parameters ϕ, which enables the gradients to be computed. This is done by reparameterizing the random variable z as a deterministic function and differentiable function of ϕ, x and a new random variable ϵ.[8]

Credit: Adapted from Kingma and Welling[8]

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 by the decoder network. The decoder distribution is given by

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

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 and $\odot$ denotes element-wise multiplication.

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

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

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

where $f_\theta(z^{(l)}) = f_\theta(\mu_\phi(x) + \sigma_\phi(x) \odot \epsilon^{(l)})$ 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

$$L = \frac{1}{D}\sum_{i=1}^{D} ||x_i - \hat{x}||^2 + \beta\frac{1}{2}\bigg(-\log\sigma_\phi^2(x) - 1 + \mu_\phi^2(x) + \sigma_\phi^2(x)\bigg),$$

where $\hat{x} = f_\theta(\mu_\phi(x_i) + \sigma_\phi(x_i) \odot \epsilon^{(i)})$ and $\epsilon^{(i)} \sim \mathcal{N}(0, I)$, D is the dimension of the input data. The second term in the function is derived from simplifing the KL divergence term in the ELBO, which is shown in the equation 2.1. The first term in the function is the MSE reconstruction loss because maximizing the Gaussian likelihood is approximately equivalent to minimizing the MSE reconstruction loss. This is shown in the equation 2.2.

$$\begin{aligned}
D_{KL}(q_\phi(z|x)\|p_\theta(z)) &= \int q_\phi(z|x)\bigg[\ \log q_\phi(z|x) - \log p_\theta(z)\ \bigg]dz \\
&= \int q_\phi(z|x)\bigg[\ -\frac{1}{2}\log(2\pi\sigma_\phi^2(x)) - \frac{(z-\mu_\phi(x))^2}{2\sigma_\phi^2(x)} \\
&\qquad\qquad - \bigg(-\frac{1}{2}\log 2\pi - \frac{z^2}{2}\bigg)\ \bigg] \\
&= \frac{1}{2}\int q_\phi(z|x)\bigg[-\log\sigma_\phi^2(x) - \frac{(z-\mu_\phi(x))^2}{\sigma_\phi^2(x)} + z^2\bigg] \\
&= \frac{1}{2}\bigg(-\log\sigma_\phi^2(x) - 1 + \mu_\phi^2(x) + \sigma_\phi^2(x)\bigg)
\end{aligned} \tag{2.1}$$

$$\begin{aligned}
\arg\max_\theta \log\mathcal{N}(x|f_\theta(z), I) &= \arg\max_\theta \log\bigg[\frac{1}{\sigma\sqrt{2\pi}}\exp\bigg(-\frac{1}{2\sigma^2}(x - f_\theta(z))^2\bigg)\bigg] \\
&= \arg\max_\theta \bigg[\log\frac{1}{\sigma\sqrt{2\pi}} - \frac{1}{2\sigma^2}(x - f_\theta(z))^2\bigg] \\
&= \arg\max_\theta -\frac{1}{2}(x - f_\theta(z))^2
\end{aligned} \tag{2.2}$$

In the figure belowe 2.2 there is a visualization of the architecture of Gaussian VAEs.

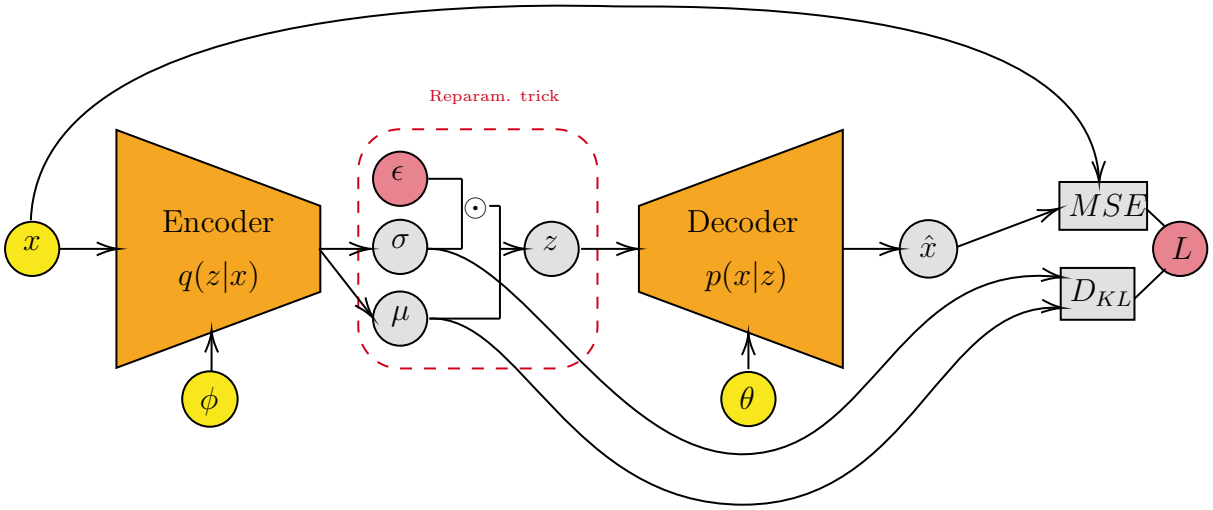

Figure 2.2: The architecture of Gaussian VAEs. The input x is passed through the encoder with parameters ϕ producing the mean μ and the standard deviation σ of the Gaussian distribution. The random variable ϵ is sampled from a standard Gaussian distribution and is used to sample $z = \mu + \sigma \odot \epsilon$. The sampled z is then passed through the decoder with parameters θ producing the output $\hat{x}$. The loss function to be minimized is the sum of the MSE reconstruction loss and the KL divergence regularization loss.

Credit: Adapted from Kingma and Welling[8]

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[15]. The VQ-VAEs have shown various improvements over the standard VAEs, such as higher quality of the generated sampled, better disentanglement of the latent space and better generalization to unseen data. The VQ-VAEs have been used in various applications, such as image generation, speech synthesis and music generation, text to image generation.[11, 3, 5, 10]

The VQVAE fundamentally differs in two key ways from a VAEs. Firstly, the latent representation is discrete instead of continuous. Secondly, the prior distribution is learnt rather than being fixed. The posterior and prior distributions are categorical and the samples taken from these distributions are the indices of the embeddings in the embedding space. These matched indices are then used to look up the embeddings in the embedding space and then used as input to the decoder.[15]. VQ-VAEs learning process consists of two stages. In the first stage, the encoder and the decoder is trained. In the second stage prior over these discrete latent variables is trained.[15]

2.2.1 Discrete Latent Variables

VQ-VAEs focus on discrete latent variables, which is a more natural fit for many types of data. Language and speech naturaly is stream of discrete units, such as words or phonemes. Images can be often well described by lanuage, which can the discrete representations well suiteed for images as well. Morever, discrete representations work very well with complex reasoning, decision making.[15]

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 2.3. 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 $\hat{x}$. One can see this forward propagation as a regular autoencoder with a quantization step in the middle.[15]

The posterior categorical distribution $q_\phi(z|x)$ is defined as follows:

$$q(z = k|x) = \begin{cases} 1 & \text{if } k = \arg\min_k ||z_e(x) - e_k||^2 \\ 0 & \text{otherwise} \end{cases}, \tag{2.3}$$

where $z_e(x)$ is the output of the encoder network and e_k is the k-th vector in the embedding table. The discrete latent variable z is then used to look up the corresponding embedding vector e_k in the embedding space, which is then used as input to the decoder network. The decoder network then produces the output $\hat{x}$.[15] The decoder distribution $p_\theta(x|z)$ is assumed to be a Gaussian distribution.

2.2.2 Learning

As mentioned earlier, the VQ-VAEs introduce learning the prior distribution separately from the posterior distribution. The prior distribution is defined as a categorical distribution $p_\omega(z)$, where z is a discrete latent variable.[15]

Since the proposed posterior distribution $q_\phi(z|x)$ is a deterministic by applying it to ELBO objective, we get the following expression:

$$\begin{aligned} L_{\theta,\phi,\omega}(x) &= -D_{KL}(q_\phi(z=k|x)||p_\omega(z)) + \mathbb{E}_{q_\phi(z=k|x)}[\log p_\theta(x|z=k)], \\ &= -\mathbb{E}_{q_\phi(z=k|x)}[\log \frac{q_\phi(z=k|x)}{p_\omega(z)}] + \mathbb{E}_{q_\phi(z=k|x)}[\log p_\theta(x|z=k)], \\ &= -\log \frac{1}{p_\omega(z)} + \log p_\theta(x|z=k), \\ &= \log p_\omega(z) + \log p_\theta(x|z=k), \end{aligned} \tag{2.4}$$

The VQVAe learning process is then divided into two stages, where in the first stage the first term is ignored and the second term is maximized. In the second stage, the prior distribution is trained. In the next 2 sections, I will describe both stages in more detail.

First stage

In the first stage the log likelihood of posterior distribution is **maximized**, which means the encoder and the decoder is trained with the prior distribution being arbitrary. The training objective in first stages reduces to

$$L_{\theta,\phi}(x) = \log p_\theta(x|z=k),$$

where k is the index of the nearest embedding vector in the embedding space, which is defined in equation 2.3. We can look at the first stage as training a regular autoencoder with a quantization step in the middle, which inherently makes the latent space distribution categorical.[15]

However, the expression $k = \arg\min_k ||z_e(x) - e_k||^2$ is not differentiable with respect to the parameters of the network. To make the training process differentiable, the authors of the VQ-VAEs propose to use the straight-through estimator, which is a way of estimating the gradients of the non-differntiable function, and copy the gradietns of $z_q(x)$ to $z_e(x)$.[15] The straight through estimator only works if the difference between $z_e(x)$ and e_k is small, which can be achieved by adding extra loss terms to the training objective.wd[16]

This is where the VQ objective comes in. The VQ objective uses the second term of equation 2.5 to encourage the encoder to produce representations that are close to the embedding vectors in the embedding space, which is called the commitement loss.[15]

However, since the embedding space can be arbitrary large the embedding vectors can be arbitrary far from the encoder output. To prevent this, the authors of the VQ-VAEs propose to add another term to the training objective, which is called the codebook loss. The codebook loss encourages the embedding vectors to be close to the encoder output. The codebook loss has β as a hyperparameter, which controls the importance of the codebook loss.[15]

Thus, the resulting training objective becomes

$$L = \log p_\theta(x|z=k) - \Big(||sg(z_e(x)) - e_k||^2 + \beta||z_q(x) - sg(e_k)||^2\Big), \tag{2.5}$$

where sg is the stop gradient operation, which is is defined as identity function, but with the gradients of the output set to zero.

The loss function to be **minimized** for VQ-VAE's usually used in practice is the sum of the VQ objective and the MSE reconstruction loss. The first term in the function is the MSE reconstruction loss because maximizing the Gaussian likelihood is approximately equivalent to minimizing the MSE reconstruction loss. This is shown in the equation 2.2. Thus the resulting training objective to be minimized becomes

$$L = \frac{1}{D}\sum_{i=1}^{D} ||x_i - \hat{x}||^2 + ||sg(z_e(x)) - e_k||^2 + \beta||z_q(x) - sg(e_k)||^2,$$

where $\hat{x} = f_\theta(z_q(x))$, where function f_θ is the decoder network, D is the dimension of the input data and k is the index of the nearest embedding vector in the embedding space, which is defined in equation 2.3.

In the figure below 2.3 there is a visualization of the architecture of VQ-VAEs.

Second stage

The second stage objective is to train the prior distribution $p_\omega(z)$ over the discrete latent variables. The latent variables z are sampled from the posterior distribution $q_\phi(z|x)$, which is defined in equation 2.3. The prior distribution is categorical and can be made autoregressive by depending on other latent variables z.[15]

The prior distribution $p_\omega(z)$ is then trained to match the distribution of the latent variables z sampled from the posterior distribution $q_\phi(z|x)$. To achieve this the authors

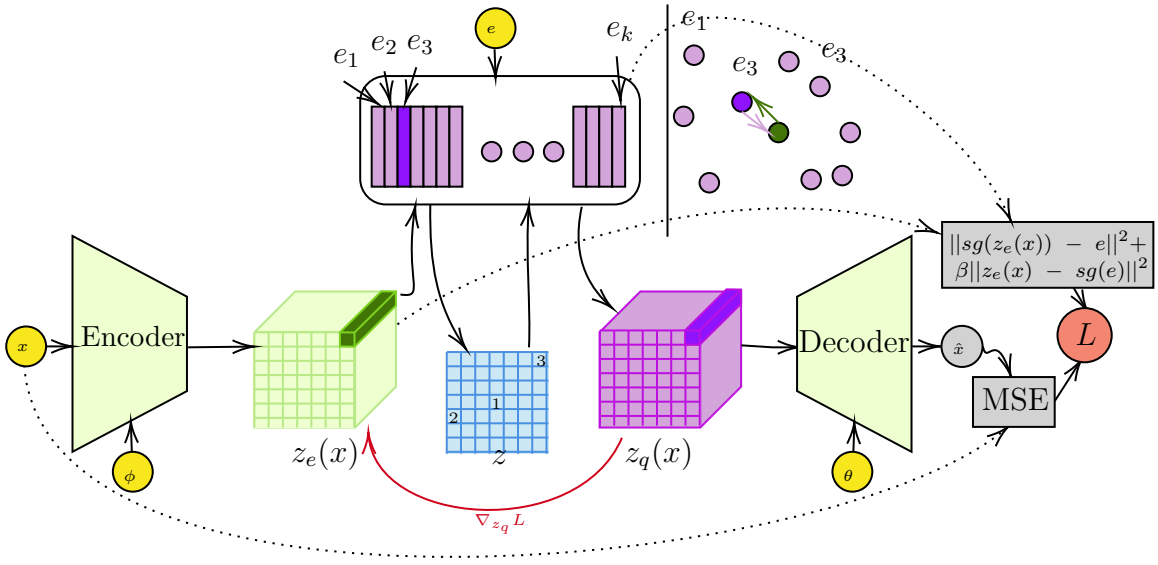

Figure 2.3: The architecture of VQ-VAEs. The input x is passed through the encoder convolutional neural network producing the output $z_e(x)$. For each output vector in $z_e(x)$, the nearest embedding vector in the embedding table e is found. The indices of the nearest embedding vectors are then used as the discrete latent variables z. The discrete latent variables z are then used to look up and retrieve the corresponding embedding vectors. The retrieved embedding vectors are then used as input to the decoder convolutional neural network producing the output $\hat{x}$. During the backward pass the gradients of the gradients of $z_q(x)$ are copied to $z_e(x)$ using the straight-through estimator, which is illustrated with a red arrow. Upper Left: The visualization of the embedding space during training. The encoder output vector is shown as a dark green dot and the nearest embedding vector is shown as a dark purple dot. The commitement and codebook loss encourage the both the encoder output vector and the nearest embedding vector to be close to each other.[15]

Credit: Adapted from Aäron van den Oord et al.[15]

of the VQ-VAEs use an autoregressive model to model the prior distribution. The autoregressive model authors used is a Gated PixelCNN, which is a variant of PixelCNN, which is described in the next section.[15]

2.2.3 PixelCNNs

The PixelCNNs are a prominent autoregressive architecture used in the field of pixel-level prediction. These models operate on images at the level of each individual pixels, learning to generate images or predict missing pixels one at a time. Deep autoregressive models have been shown to be very effective at modeling the full distribution and generating relatively low-resolution images. Generating high-resolution images with merely autoregressive models is challenging, because the size of the network increases rapidly with the size of the image.[14, 13]

Auto regressive models treat and image as a sequence of pixels and the goal is to model the conditional distribution of each pixel given the previous pixels. Image x is represented as a one-dimensional sequence of pixels $x = (x_1, x_2, \ldots, x_N)$, where x_i is the i-th pixel in the image and N is the number of pixels in the image. The estimate of the joint distribution of the pixels over an image x is given by the product of the conditional distributions of each pixel given the previous pixels

$$p(x) = \prod_{i=1}^{N} p(x_i | x_1, x_2, \ldots, x_{i-1}),$$

where $p(x_i|x_1, x_2, \ldots, x_{i-1})$ is the conditional distribution of the i-th pixel given the previous pixels. The generational process of the image is then done by sampling each pixel sequantially from the conditional distribution of the pixel given the previous pixels, which is shown in the figure 2.4.[14]

The PixelCNN in the original architecture is a stack of fifteen fully convolutional network layers with masked convolutions. The masked convolutions are used to ensure that the model can only access the previous pixels, which is crucial for model to be autoregressive. The model is trained to minimize the negative log likelihood of the training data. The PixelCNNs have been shown to be very effective at capturing the distribution of the data. Most current state of the art models use the PixelCNNs as a building block for example PixelCNN++ and PixelSNAIL.[14, 12, 1]

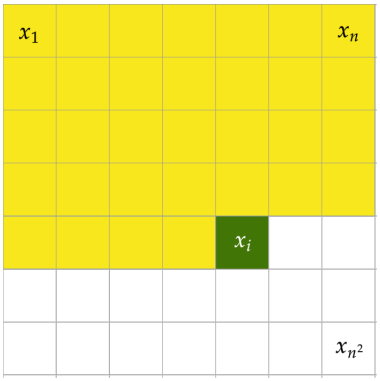

Figure 2.4: The conditional generation in autoregressive models. The model generates the pixels of the image one at a time, conditioning on the previous pixels. The model is autoregressive, because the distribution of each pixel is conditioned on the previous pixels.

Credit: Adapted from the original PixelCNN paper.[14]

Application in VQ-VAEs

The PixelCNNs are used in the VQ-VAEs to model the prior distribution $p_\omega(z)$ over the discrete latent variables. The prior distribution is trained to match the distribution of the latent variables z sampled from the posterior distribution $q_\phi(z|x)$. The PixelCNN in combination with the VQ-VAEs has been shown to be exceptional at capturing the distribution of the latent variables and generating high resolution samples.[15]

2.3 Semi-Conditioned VAEs

Semi-conditional VAEs were first introduced in 2020.[4] Semi-conditional VAEs are a variant of VAEs that first designed for the reconstruction of non-linear dynamic processes based on sparse observations. The semi-conditional VAEs extend the standard VAEs by adding an additional input m to the decoder network $p_\theta(x|z, m)$, which is used to condition the decoder on the additional input.

The semi-conditional VAEs showed to be effective in making use of the additional information to improve the reconstruction of the data and the generalization to unseen data.[4] This shows that the similar approach could be used to improve the generalization of the VQ-VAEs to unseen data.

2.4 Multitask Learning

Multitask learning is a machine learning paradigm where multiple tasks are learned at the same time, which has the aim of leveraging the shared information between the tasks to improve the performance of the individual tasks. Unlike the traditional single task learning, where each task is learned independently, multitask learning allows to take advantage of task relationships and to learn a shared representation that is useful for all tasks.[2]

The notion of using multitask learning comes from the observation that the tasks are often related and depend on the same underlying features. This can be beneficial is scenarious where the data is limited, which is often the case for medical data.

The multitask learning can be also applied to VAEs and VQ-VAEs.

Chapter 3

Methodology

Chapter 4

Results

Chapter 5

Discussion

Bibliography

[1] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model, 2017.

[2] Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *CoRR*, abs/2009.09796, 2020.
URL: https://arxiv.org/abs/2009.09796.

[3] Shuvayanti Das, Jennifer Williams, and Catherine Lai. Analysis of voice conversion and code-switching synthesis using vq-vae, 2022.

[4] Kristian Gundersen, Anna Oleynik, Nello Blaser, and Guttorm Alendal. Semi-conditional variational auto-encoder for flow reconstruction and uncertainty quantification from limited observations. *Physics of Fluids*, 33(1), January 2021. ISSN 1089-7666. doi: 10.1063/5.0025779.
URL: http://dx.doi.org/10.1063/5.0025779.

[5] Sangjun Han, Hyeongrae Ihm, and Woohyung Lim. Symbolic music loop generation with vq-vae, 2021.

[6] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 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.

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

[8] 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.

[9] Weizhu Qian, Bowei Chen, and Franck Gechter. Multi-task variational information bottleneck. *CoRR*, abs/2007.00339, 2020.
URL: `https://arxiv.org/abs/2007.00339`.

[10] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *CoRR*, abs/2102.12092, 2021.
URL: `https://arxiv.org/abs/2102.12092`.

[11] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2, 2019.

[12] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications, 2017.

[13] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks, 2016.

[14] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders, 2016.

[15] 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`.

[16] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets, 2019.

Appendix A

Generated code from Protocol buffers

Listing A.1: Source code of something

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